Systematic Study of Decryption and Re-Encryption Leakage: the Case of Kyber

Melissa Azouaoui[1], Olivier Bronchain[2], Clément Hoffmann[2], Yulia Kuzovkova[1], Tobias Schneider[1], François-Xavier Standaert[2]

[1] NXP Semiconductors
[2] UCLouvain, ICTEAM/ELEN/Crypto Group

Keywords: Side-Channel Attacks · Post-Quantum Cryptography · Key Encapsulation Mechanism · Fujisaki-Okamoto Transformation · Masking

Abstract. The side-channel cryptanalysis of Post-Quantum (PQ) key encapsulation schemes has been a topic of intense activity over the last years. Many attacks have been put forward: Simple Power Analysis (SPAs) against the re-encryption of schemes using the Fujisaki-Okamoto (FO) transform are known to be very powerful; Differential Power Analysis (DPAs) against the decryption are also possible. Yet, to the best of our knowledge, a systematic and quantitative investigation of their impact for designers is still missing. In this paper, we propose to capture these attacks with shortcut formulas in order to compare their respective strength in function of the noise level. Taking the case of Kyber for illustration, we then evaluate the (high) cost of preventing them with masking and the extent to which different parts of an implementation could benefit from varying security levels. We finally discuss tweaks to improve the situation and enable a better leveling of the countermeasures. Our conclusions confirm that current solutions for side-channel secure PQ key encapsulation schemes like Kyber are unlikely to be efficient in low-noise settings without (design or countermeasures) improvements.

1 Introduction

Many Post-Quantum (PQ) Key Encapsulation Mechanisms (KEMs), including third-round finalists of the NIST post-quantum standardization effort, rely on the Fujisaki-Okamoto (FO) transform [15]. It allows building a Chosen-Ciphertext (CCA) secure scheme from a Chosen-Plaintext (CPA) secure Public-Key Encryption (PKE) scheme. This transform first decrypts the ciphertext c with the underlying CPA-secure PKE to retrieve the message m. Then, it re-encrypts (in a deterministic manner) m to obtain a ciphertext c'. By construction, any ciphertext c that has not been generated by the CPA-secure encryption scheme will result in a case where $c' \neq c$ (up to a negligible probability). In such a case, the CCA-secure KEM returns a random message which cannot be exploited by the adversary. Yet, while the FO transform is well suited to reach mathematical security, several recent works showed that the situation strongly differs when physical attacks are considered [29, 34, 32, 25]. In such a context, leakage about intermediate computations makes it possible to circumvent mathematical security guarantees. Roughly, an adversary can then use a chosen-ciphertext attack against the part of the scheme that is only CPA secure. For example in [29], the adversary carefully crafts ciphertexts such that the decrypted message m leaks a bit of a secret key coefficient. Since this message m is used as input for the deterministic re-encryption, the adversary then only has to distinguish between an encryption of 0 or 1 given leakage of the computation. To do so, she can target all the intermediate computations within the (long) deterministic re-encryption jointly, which can include hundreds, thousands or even millions of intermediate bytes/words. Furthermore, this leakage is easy to exploit, whether being via standard techniques (e.g., template attacks with dimensionality reduction) or machine learning based cryptanalysis.

Echoing the situation in symmetric cryptography, such an attack actually corresponds to the strongest (state comparison) one in the taxonomy of [30, Slide 1.7]. In terms of design, [30] also recalls that symmetric decryption ensuring CCA security with leakage requires a two-pass design where the validity of the ciphertexts is verified before being decrypted (e.g., thanks to a MAC).

In addition to these attacks targeting the re-encryption, Differential Power Analysis attacks (DPAs) that target the leakage in the first (guessable) parts of a secret computation are also possible. In the case of PQ KEMs, such attacks naturally apply to the part of the decryption that takes place before the re-encryption. As usual, they can be extended from a standard DPA to analytical attacks exploiting even the hard-to-guess parts of the computation thanks to belief propagation, leading to strong key recoveries [27, 26].

In parallel to the efforts regarding the identification of attack vectors, countermeasures against side-channel attacks have also been adapted to the PQ setting. As one of the leading protections, masking has received particular attention and, for example, first- and higher-order masked implementations of Saber and Kyber have been proposed [4, 7, 19, 14]. To the best of our knowledge, masked implementations of PQ schemes so far mostly considered an uniform protection level, where all the parts of the computations embed the same number of shares. Again echoing the situation in symmetric cryptography, these works naturally question the possibility to consider so-called leveled implementations, where different parts of the computations have different security levels, for example based on the number of operations exploitable via side-channel leakage [5].

Based on this state-of-the-art, the objectives of the paper are threefold:

First, we propose a model for both (i.e., SPA and DPA) attack paths. For each of them, we derive a shortcut formula (i.e., a generic expression for the minimal number of traces needed for a successful attack) that takes as parameters the number of shares in the masking scheme, the level of noise in the leakage (measured as the mutual information between the shares and the leakage λ) and the (cipher-specific) amount of operations for which the leakage is exploitable. These shortcut formulas enable a first (simplified) approximation of attack complexities to compare side-channel attacks according to the above parameters.

Second, we illustrate these formulas at the example of CRYSTALS-Kyber [2]. This specific choice of KEM is motivated by the already large literature dedicated to its side-channel analysis and countermeasures. We derive our model parameters based on state-of-the-art implementation results from [7] and, in order to enable a comparative study of the attack paths, additionally express the cost needed to protect the different subparts of the Kyber computations.

Third, we use our results to discuss masked implementations of Kyber and the possibility to leverage the leveled implementation concept for PQ KEMs. We show that for unmasked implementations, the re-encryption becomes the preferred attack vector as the noise level increases (since it enables very strong horizontal attacks). But somewhat surprisingly, we also show that as the number of shares (hence, the target security level) increases, the impact of the re-encryption in the overall security vs. efficiency tradeoff tends to vanish, which significantly limits the interest of leveled implementations.

As a conclusion, we first recall that getting rid of the FO-transform in a PQ KEM such as Kyber would require a way to identify “well structured ciphertexts” without re-encryption, as performed in the symmetric cryptographic setting thanks to a leakage-resilient MAC. We note that this is a hard problem in itself. At this stage, it is not clear whether simple filtering heuristics can be sufficient [34], and the more formal solution of relying in a zero-knowledge proof (which we briefly discuss in Section 6) is quite expensive. In this respect, our results show that the performance margin available to (heuristically or formally) get rid of the FO-transform is quite limited. This conclusion derives from the observation that as the number of shares increases, the cost of protecting the (CPA-secure) decryption and the re-encryption becomes increasingly balanced in Kyber. In other words, improving the efficiency of side-channel secure Kyber implementations would not only require to get rid of the FO-transform, but also to increase the performance gap between its protected (CPA-secure) decryption and re-encryption. Candidates for this purpose include improving the efficiency of this decryption (especially the compression part), reducing the complexity of its masking and taking advantage of hardware tweaks (e.g., exploiting different noise levels in the decryption and re-encryption). We believe a similar challenge appears in related PQ KEMs relying on the FO-transform (e.g., Saber [3]).

2 Background

We next recall the necessary background for the rest of the paper. We first describe shortcut formulas for side-channel attacks based on Information Theoretic (IT) metrics. We continue with a short description of CRYSTALS-Kyber [2].

2.1 Information Theory for Side-Channel Attacks

Side-channel attacks allow recovering cryptographic secrets by observing the leakages $\boldsymbol{L}$ from an implementation. All side-channel attacks (whether based on a divide-and-conquer or analytical strategy) include an extraction phase where they collect information about guessable intermediate computations. A standard strategy to estimate the number of traces required to recover leaking target intermediate values with confidence is to use information theoretic metrics such as the Mutual Information (MI) [31, 13, 11]. Next, we recall the simple relations that we are going to leverage in order to derive shortcut formulas. We start with unprotected variables, continue with masked variables and finally discuss the impact of attacks exploiting multiple intermediate computations.

Unprotected variables. In order to evaluate the number of independent leakages N required to recover a (sub)-secret X, one can use the relation:

$$N \approx \frac{c}{\mathrm{MI}(X;\boldsymbol{L})}\,, \tag{1}$$

where c is a small constant that depends on the size of the intermediate variable and the target success rate (we will set it to the Shannon's entropy such that $c = \mathrm{H}(X)$ for simplicity), and $\mathrm{MI}(X;\boldsymbol{L})$ is the Mutual Information between the target variable X and the leakage vector $\boldsymbol{L}$.

Roughly, the attack complexity is inversely proportional to the MI, which is itself inversely proportional to the noise variance of the leakages [24]. In the following, we will use the MI between target intermediate computations and their leakage as a parameter of our evaluations, and denote it as λ.

Masked variables. In order to protect implementations against SCA, masking is a countermeasure that has been extensively studied. It consists in representing a sensitive variable X as d shares $(X^0, X^1, \ldots, X^{d-1})$ such that any set of $d-1$ shares remains independent of X. The operations are then applied to the shares of X instead of on X directly. When implemented securely (i.e., under some noise and independence conditions), it guarantees an exponential security increase at the cost of quadratic performance overheads [9, 22, 28, 12, 13]. Concretely, this security amplification is reflected by a reduction of the MI:

$$\mathrm{MI}(X;\boldsymbol{L}) \approx \prod_{i=0}^{d-1} \mathrm{MI}(X^i;\boldsymbol{L}) = \lambda^d, \tag{2}$$

leading to an increased attack data complexity captured by:

$$N \approx \frac{c}{\lambda^d}\,\cdot \tag{3}$$

Targeting multiple (independent) operations. Side-channel attacks are not limited to the exploitation of a single intermediate computation and its corresponding leakage. Multiple operations can be exploited jointly. The simplest way to do so is when multiple intermediate computations relate to the same guessable secret. In this case, a natural abstraction is to model their leakages as providing independent information on the secret, as captured by the Independent Operations' Leakages (IOL) proposed in [17]. It leads to the approximation:

$$\mathrm{MI}(X;\boldsymbol{L}) \approx \sum_j \mathrm{MI}(var_j(X);\boldsymbol{L}) \approx \#var \cdot \lambda^d, \tag{4}$$

where $var_j(X)$ is one of the $\#var$ variables depending only on X.

For example in the case of block-ciphers, an attack can be performed by exploiting leakage (e.g., with Gaussian templates) at the input of the Sbox and at its output and combining information from both. This will double the information about X (hence $\#var = 2$) compared to an attack exploiting only its output (with $\#var = 1$). It turns out this simple setting will be quite frequently observed (and generalized) in our following investigations.

Cautionary notes. The above formulas are admittedly simplified and may ignore a part of the leakages. First, it is also possible to exploit the leakages of operations that are not exploitable via a divide-and-conquer DPA, for example using analytical strategies [33], as considered by [27, 26] in the PQ setting. Second, masked multiplications with d shares imply quadratic overheads, e.g., they require computing d^2 partial products which may leak as well. Yet, as modeled in [18], the leakage of intermediate values that do not relate to the same guessable secret does not simply add up as in the previous IOL case and rather implies a constant gain in the attacks; and the leakage of partial products in masked multiplication is dominated by the leakage of the tuples as the noise level of the implementation increases. So simply stated, while the following shortcut formulas could be refined to take into account advanced attacks, they are a convenient first step to study the large design space of masked PQ KEMs up to small constants.

2.2 CRYSTALS-Kyber

Arithmetic & notations. We denote the polynomial ring $\mathbb{Z}_q[X]/(X^n + 1)$ as $\mathbb{R}_q$. We denote polynomials with lower case such that $f \in \mathbb{R}_q$. As a result, the polynomials are of the form:

$$f = f_0 + f_1 \cdot X^1 + \ldots + f_{n-1} \cdot f^{n-1}, \tag{5}$$

where $f_i \in \mathbb{Z}_q$ is the i-th coefficient of the polynomial. We denote vectors and matrices with bold letters. For example, $\boldsymbol{v} \in \mathbb{R}_q^k$ is a vector of k polynomials such that $\boldsymbol{v}[i] \in \mathbb{R}_q$ for all $0 \leq i < k$. Hence, $\boldsymbol{v}[i]_j$ refers to the j-th coefficient of the i-th polynomial in $\boldsymbol{v}$. Additionally, $\boldsymbol{A} \in \mathbb{R}_q^{k\times k}$ is a matrix of polynomials with size $k \times k$. For CRYSTALS-Kyber, the prime q is chosen such that polynomial multiplications can be performed very efficiently via the Number-Theoretic Transform (NTT). Concretely, the NTT is first applied to the polynomials such that:

$$\hat{f} = \mathrm{NTT}(f) = \hat{f}_0 + \hat{f}_1 \cdot X^1 + \ldots + \hat{f}_{n-1} \cdot X^{n-1}, \tag{6}$$

where $\hat{f}$ (resp., $\hat{f}_0$) denotes the NTT domain representation of f (resp., f_0). NTT$(\cdot)$ is efficiently applied with a butterfly algorithm [10, 16]. The multiplication between two polynomials can then leverage their representations in the NTT domain such that:

$$c = a \cdot b = \mathrm{NTT}^{-1}(\hat{a} \circ \hat{b}), \tag{7}$$

where $\circ$ is the coefficient-wise multiplication.

CRYSTALS-Kyber PKE. We first detail the underlying CPA-secure PKE scheme denoted as Kyber.CPAPKE, which relies on the hardness of Module-LWE [23]. Kyber.CPAPKE encryption and decryption are recalled in Algorithm 1 and Algorithm 2, respectively. There, $\hat{\boldsymbol{s}}$ denotes a secret key, pk a public key containing a vector of polynomials $\hat{\boldsymbol{t}}$ and a random matrix of polynomials $\hat{\boldsymbol{A}}$, m the 256-bit message, σ a 256-bit random seed used to derive deterministically the randomness from a pseudo-random generator and CBD_η denotes the sampling, from a uniform random string, of a centered binomial distribution with parameter η. Comp_q and Decomp_q are respectively compression and decompression functions such that $\mathrm{Decomp}_q(\mathrm{Comp}_q(x, d_x), d_x) \approx x$. Both PRF, KDF, G and H are hash functions with various output lengths based on the Keccak permutation. For more details about these algorithms, we refer to their specifications in [2].

Algorithm 1 Kyber.CPAPKE.Enc(pk,m,σ)

Input: Public key $pk := (\hat{\boldsymbol{t}}, \hat{\boldsymbol{A}})$ with $\hat{\boldsymbol{t}} \in \mathbb{R}_q^k$ and $\hat{\boldsymbol{A}} \in \mathbb{R}_q^{k \times k}$, message $m \in \{0,1\}^n$, random coin $\sigma \in \{0,1\}^{256}$.
Output: Ciphertext $c := (c_1, c_2)$.

```
for $i$ in $[0, \dots, k-1]$ do
    $\boldsymbol{r}[i] := \mathrm{CBD}_{\eta_1}(\mathrm{PRF}(\sigma, i))$
    $\boldsymbol{e}_1[i] := \mathrm{CBD}_{\eta_2}(\mathrm{PRF}(\sigma, i+k))$
$e_2 := \mathrm{CBD}_{\eta_2}(\mathrm{PRF}(\sigma, 2 \cdot k))$
$\hat{\boldsymbol{r}} := \mathrm{NTT}(\boldsymbol{r})$
$\boldsymbol{u} := \mathrm{NTT}^{-1}(\hat{\boldsymbol{A}}^T \circ \hat{\boldsymbol{r}}) + \boldsymbol{e}_1$
$v := \mathrm{NTT}^{-1}(\hat{\boldsymbol{t}}^T \circ \hat{\boldsymbol{r}}) + e_2 + \mathrm{Decomp}_q(m, 1)$
$c_1 := \mathrm{Comp}_q(\boldsymbol{u}, d_u)$
$c_2 := \mathrm{Comp}_q(v, d_v)$
return $(c_1, c_2)$
```

Algorithm 2 Kyber.CPAPKE.Dec($\hat{\boldsymbol{s}}$,c)

Input: Secret key $\hat{\boldsymbol{s}} \in \mathbb{R}_q^k$, ciphertext $c := (c_1, c_2)$.
Output: Message m.

```
$\boldsymbol{u} := \mathrm{Decomp}_q(c_1, d_u)$
$v := \mathrm{Decomp}_q(c_2, d_v)$
$\hat{z} = \hat{\boldsymbol{s}}^T \circ \mathrm{NTT}(\boldsymbol{u})$
$w := v - \mathrm{NTT}^{-1}(\hat{z})$
$m := \mathrm{Comp}_q(w, 1)$
return $m$
```

CRYSTALS-Kyber KEM & FO-transform. CRYSTALS-Kyber leverages the FO-transform [15, 20] to build a CCA-secure KEM from a CPA-secure PKE. The resulting Kyber.CCAKEM encapsulation and decapsulation algorithms are described respectively in Algorithm 3 and Algorithm 4. More precisely, Kyber.CCA KEM.Dec first retrieves a message candidate m' by decrypting the ciphertext c thanks to Kyber.CPAPKE.Dec and the secret key.

Then, it computes c' by re-encrypting m' using the encryption $c' =$ Kyber.CPAPKE.Enc(pk, m', σ'). It then only outputs the correct symmetric key material K if c is equal to c'. As a result, the resulting scheme is protected against chosen-ciphertext attacks: as soon as an adversary attempts to use a forged ciphertext, the resulting re-encrypted c' will differ from c (up to a negligible probability).

We note that for such a construction to be correct, the randomness used during the (re)-encryption should be deterministically generated from a random coin associated with the plaintext. Otherwise, the ciphertexts c and c' could differ even though they are both correct ciphertexts of the same message m.

Algorithm 3 Kyber.CCAKEM.Enc(pk)

```
Input: Public key $pk := (\hat{\boldsymbol{t}}, \hat{\boldsymbol{A}})$
Output: Ciphertext $c$, encap. secret $K$.
$m \leftarrow \{0,1\}^{256}$
$m := \mathrm{H}(m)$
$(\bar{K}, \sigma) := \mathrm{G}(m||\mathrm{H}(pk))$
$c :=$ Kyber.CPAPKE.Enc$(pk, m, \sigma)$
$K := \mathrm{KDF}(\bar{K}||\mathrm{H}(c))$
return $(c, K)$
```

Algorithm 4 Kyber.CCAKEM.Dec(c, sk)

```
Input: Ciphertext $c := (c_1, c_2)$, secret key
    $sk := (\hat{\boldsymbol{s}}, pk, \mathrm{H}(pk), z)$.
Output: Decap. secret $K$.
$m' :=$ Kyber.CPAPKE.Dec$(\hat{\boldsymbol{s}}, c)$
$(\bar{K}', \sigma') := \mathrm{G}(m'||\mathrm{H}(pk))$
$c' :=$ Kyber.CPAPKE.Enc$(pk, m', \sigma')$
if $c = c'$ then
    return $K := \mathrm{KDF}(\bar{K}'||\mathrm{H}(c))$
else
    return $K := \mathrm{KDF}(z||\mathrm{H}(c))$
```

Concrete parameters. Finally, we give the parameters used by CRYSTALS-Kyber for the third round of the NIST PQC standardization process [2] in Table 1, for the different versions corresponding to different security levels.[3]

	n	k	q	η_1	η_2	d_u	d_v
Kyber512	256	2	3329	3	2	10	4
Kyber768	256	3	3329	2	2	10	4
Kyber1024	256	4	3329	2	2	11	5

Table 1: Summary of Kyber parameters (from [2]).

3 Shortcut formulas for SPA and DPA

As stated in the introduction, there exists two main attack paths against PQ KEMs based on the FO-transform. In this section, we derive shortcut formulas for their data complexity. For this purpose, we use the illustration of Figure 1, which is a simplified graphical representation of Algorithm 4.

[3] Namely, NIST security levels 1, 3 and 5 for which any attack requires comparable computational resources to attacks against AES-128,AES-196 and AES-256 [1, 2].

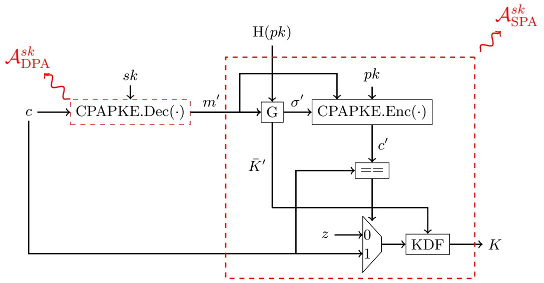

Fig. 1: Attack paths against Kyber.CCAKEM.Dec(c, sk) (Algorithm 4).

3.1 $\mathcal{A}^{sk}_{\text{DPA}}$: DPA against CPAPKE.Dec

Description. A DPA attack against PQ KEMs can be performed in a similar way as classical (divide-and-conquer) DPA against block-ciphers. In this case, $\mathcal{A}^{sk}_{\text{DPA}}$ targets directly each of the secret key coefficients $\hat{s}_i$ independently. To do so, she asks for the decryption of legit ciphertexts c's and records the corresponding traces. She then exploits every operations that directly involve $\hat{s}_i$, as she would target the Sbox's input and output in the case of a block-cipher.

For illustration, in the case of CRYSTALS-Kyber, this adversary targets each $\hat{\boldsymbol{s}}_i$ independently by observing the leakage of $\hat{\boldsymbol{s}}^T \circ \text{NTT}(\boldsymbol{u})$ (see Algorithm 2, line 3) where $\boldsymbol{u}$ is known and random. Indeed, she obtains leakage of the coefficient-wise operations $\hat{\boldsymbol{s}}_i \circ \hat{\boldsymbol{u}}_i$ as detailed in Equation 7.

Shortcut formula. Because this attack targets each of the coefficients independently, the resulting shortcut formula can leverage the approximations given in Subsection 2.1. Namely, we can estimate the data complexity of $\mathcal{A}^{sk}_{\text{DPA}}$ with:

$$N_{Dec} \approx \frac{\alpha_{Dec}}{\lambda^{d_{Dec}}}, \tag{8}$$

where d_{Dec} is the amount (#) of shares used to protect the CPAPKE.Dec and $\alpha_{Dec} = \frac{c}{\#var}$ is the cipher-dependent constant reflecting the # of operations that can be exploited via DPA (and the constant c), discussed for Kyber in Section 5.

Other DPA attacks. As already mentioned in Subsection 2.1, advanced adversaries could additionally exploit the leakages in NTT^{-1} thanks to belief-propagation [27, 26]. By exploiting the leakage of hard to guess intermediate variables, such advanced attacks can reduce the data complexity by additional constant factors, which could be reflected by adapting α_{Dec} [18].

3.2 $\mathcal{A}^{sk}_{\text{SPA}}$: SPA against re-encryption

Description. The SPA adversary $\mathcal{A}^{sk}_{\text{SPA}}$ targets leakages from all operations that directly depend on the decrypted message m' [29, 34, 32, 25]. In short, these attacks are chosen-ciphertext attacks against the part of the KEMs that are only CPA-secure. This is made possible despite the FO-transform by exploiting the leakage of m' as a plaintext-checking oracle. This can be done simply by observing whether m' is equal or not equal to a reference message. In order to recover the full secret key sk, multiple forged ciphertexts (and corresponding oracle accesses) are required. We denote their number as $\#\mathcal{O}$.

Shortcut formula. In order to predict the data complexity of such an SPA adversary, we first observe that by targeting the leakage of the re-encryption, it is possible to exploit multiple independent variables that directly depend on m'. We denote the number of these operations as $\#var$. Again leveraging the approximations of Subsection 2.1, the number of traces required to succeed in detecting whether m' is equal to the reference is then estimated as:

$$N_{Enc} \approx \frac{\alpha_{Enc}}{\lambda^{d_{Enc}}}\,, \tag{9}$$

with $\alpha_{Enc} = \frac{c' \cdot \#\mathcal{O}}{\#var}$. Here again, the number of exploitable operations $\#var$ is cipher-specific. The same holds for the number of required oracle accesses $\#\mathcal{O}$. Those values will be specified for the case of Kyber in the next section.

4 Generic intuitions

In this section, we discuss the case of masked implementations of PQ KEMs. We assume a designer aiming at an implementation that can resist attacks of complexity γ (which therefore corresponds to the security level) on a platform with physical security parameter λ (which, as outlined in Section Subsection 2.1, can be viewed as a measure of the implementation's noise level). More precisely, we first study the case where a designer selects the same number of shares to protect the decryption and the re-encryption, and show that such a strategy requires a large number of shares. Then, we quantify the effect of using different number of shares for decryption and re-encryption and discuss its impact regarding the relative cost of the re-encryption in the overall implementation.

4.1 Masking can be (very) expensive

Next, we estimate the number of shares required to protect against both decryption and re-encryption attacks such that $d = d_{Enc} = d_{Dec}$. To do so, the target security level γ should be smaller than the attack complexities N derived in Section 3 such that:

$$\gamma \leq \alpha \cdot \frac{1}{\lambda^{d}}\,, \tag{10}$$

with $\alpha = \min(\alpha_{Enc}, \alpha_{Dec})$. As a result, the required number of shares to protect the entire implementation is defined by:

$$d \geq \log_\lambda \left(\frac{\alpha}{\gamma} \right) \tag{11}$$

$$\geq \frac{\log_{10}(\alpha) - \log_{10}(\gamma)}{\log_{10}(\lambda)} \,, \tag{12}$$

up to $\lceil \cdot \rceil$ to ensure that d is an integer (for readability, we omit the rounding for the rest of the section). Based on this equation, we report the number of shares for various settings in Figure 2. As expected from Equation 12, the parameter that has the strongest influence on the number of shares is the noise λ, which corresponds to the slope of the curves. For example, moving from $\lambda = 0.1$ to $\lambda = 0.01$ reduces the number of shares by a factor 2 since $|\log_{10}(\lambda)|$ is increased similarly. The number of shares also depends on the attack parameter α, but to a lower extent (since this factor is constant in d). For example, reducing α by a factor 10 implies the need for $1/\log_{10}(\lambda)$ additional shares to compensate.

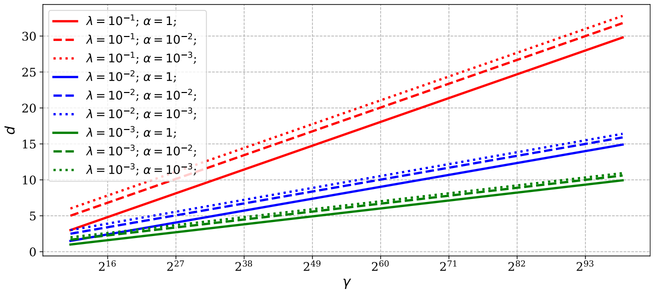

Fig. 2: Impact of the attack parameters α and the noise level λ on the number of shares d in function of the target security level γ.

The combination of these two effects is at the basis of the somewhat paradoxical conclusions in the following sections. On the one hand, large number of operations (which are observed for PQ KEMs) make side-channel attacks very strong in the absence of countermeasures. On the other hand, the impact of reducing this number of leaking operations vanishes as the target security level (and number of shares) increases, since it is then increasingly dominated by the amplified noise. For the rest, the figure also recalls that with low-noise devices, the number of shares needed to reach high security against side-channel attacks, and therefore the implementation overheads, can be prohibitive [8].

4.2 Leveling moderately helps

State-of-the-art analyses of PQ KEMs indicate that SPA attacks targeting re-encryption are very powerful. This is for exampled witnessed by the "curse of re-encryption" discussed in [32]. Concretely, the origin of this curse mainly lies in the different number of exploitable operations that decryption and re-encryption may have. A natural target to mitigate this issue is to use a different number of shares for these two parts of a KEM. By doing so, we could expect to reduce the overall cost of an implementation by protecting less its stronger (i.e., CPA decryption) part. In order to model such a leveling, we assume that masking leads to overheads that are quadratic in d and express the cost of protecting one component as $\zeta = \beta \cdot d^2$, where β is the cycle count of an unprotected implementation. Note that the quadratic cost overheads are naturally dependent on the fraction of linear and non-linear operations in the algorithm to protect. But concrete results recalled in Appendix A show that this trend is observed even for low number of shares in the case of Kyber.[4] More precisely, we are interested in the proportion ζ_{Enc}/ζ_{Dec} of time that is spent to protect re-encryption and decryption. A large ratio means that the re-encryption is the dominating the overall cost. Hence, removing re-encryption would lead to significant performance improvements in this case. A small ratio means the opposite. Next, we study this ratio according to attack parameter α, target security level γ as well as the implementation complexity β.[5] This proportion can be given by:

$$\begin{aligned}\zeta_{Enc}/\zeta_{Dec} &= \frac{\beta_{Enc}}{\beta_{Dec}} \cdot \frac{d_{Enc}^2}{d_{Dec}^2} && (13)\\ &= \frac{\beta_{Enc}}{\beta_{Dec}} \cdot \frac{(\log_{10}(\alpha_{Enc}) - \log_{10}(\gamma))^2}{(\log_{10}(\alpha_{Dec}) - \log_{10}(\gamma))^2}\,, && (14)\end{aligned}$$

thanks to the expression of the number of shares in Equation 12. For increasing security target γ, this ratio converges such that:

$$\lim_{\gamma\to\infty} \zeta_{Enc}/\zeta_{Dec} = \frac{\beta_{Enc}}{\beta_{Dec}}\,. \tag{15}$$

We observe that as γ increases, the proportion of time spent between the two blocks converges towards the ratio of their performances when implemented without side-channel protections. In other words, the gap between the number of shares needed to compensate different number of exploitable operations vanishes with the security level. This equation is illustrated in Figure 3. For low security, the re-encryption dominates the overall cost of the secure implementation. By contrast, it is no more the case for large security levels.

Interestingly, even if the idea of leveling we analyze in this work looks quite similar to the one leveraged in symmetric cryptography [5], its impact is much

[4] For both CPAPKE.Enc and CPAPKE.Dec, lattice arithmetic can be efficiently masked with linear complexity. However, both include polynomial comparison, polynomial compression and binomial noise sampling which require boolean-to-arithmetic and/or arithmetic-to-boolean masking conversions implying quadratic overheads. In the case of Kyber, these conversions dominate the overall cost of protected implementations even for small masking order.

[5] β is strongly connected to α in the case of PQ KEMs where the leakage of most operations can be exploited. Yet, it could be more different in other cases.

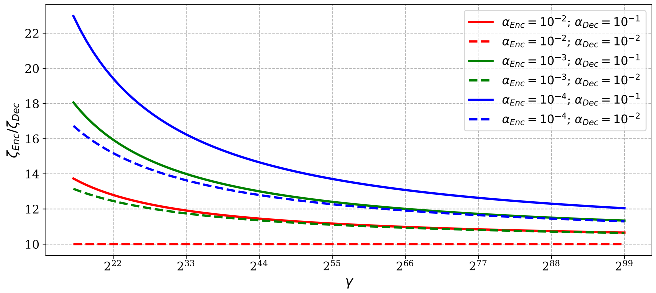

Fig. 3: Ratio between the cost of decryption and re-encryption ($\beta_{Enc}/\beta_{Dec} = 10$).

lower. The best factor that a designer could hope to gain in the case of a PQ KEM that is strongly protected against side-channel attacks is $\approx 1+\beta_{Enc}/\beta_{Dec}$. But in fact, the explanation is also quite natural. The main factor making the leveling process very effective in the symmetric case is the possibility to amortize the cost of a highly masked implementation for long messages. In the case of a PQ KEM, the size of the messages is fixed. Admittedly again, these conclusions are based on generic formulas that may be over simplifying and their concrete impact depends on the exact complexity of concrete PQ KEMs. We next show that a specialization to Kyber leads to these conclusions as well.

5 Applications to CRYSTALS-Kyber

We now aim to characterize the parameters of the SPA and DPA attack paths more concretely. For this purpose, we first introduce a slightly finer-grain analysis than the shortcut formulas in the previous section. Its goal is to enable a discussion of how stable our conclusions are in front of technology-dependent variations. We next quantify the operation counts in the specific case study of a software CRYSTALS-Kyber implementation from [7]. We finally use these concrete values to revisit the generic intuitions of the previous section.

Preliminaries. Before these discussions, we mention two preliminary steps towards specializing our shortcut formulas to Kyber. The first one is to observe that the c constant in Section 3 will be different for the two attack paths. In the $\mathcal{A}^{sk}_{\text{DPA}}$ case, one aims to recover 12-bit values (key coefficients in NTT domain), leading to $c = 12$. In the $\mathcal{A}^{sk}_{\text{SPA}}$ case, one only wants to recover a 1-bit value (i.e., to distinguish the leakage of a message from the leakage of another message), leading to $c' = 1$. The second step is to determine the number of oracle accesses $\#\mathcal{O}$ required for the SPA to succeed. For Kyber, this number is worth $\#\mathcal{O} = k \cdot n \cdot 3$, as given in Table 3 of [32], based on the analysis in [21].[6]

[6] The basic $\mathcal{A}^{sk}_{\text{SPA}}$ aims to recover a single coefficient of $\boldsymbol{s}$ per oracle access. But it can be extended to target b coefficients of the secret at once, which essentially works by targeting b bits of the decrypted message rather than a single one.

5.1 Finer grain analysis

Equations 8 and 9 in the previous section implicitly assume that all the intermediate variables they target leak the same amount of information λ. However, it may not be the case in practice, for different (implementation-specific) reasons. The simplest one, that we will characterize next, is that the operations may not all manipulate the same amount of bits per cycle. For example, in a software implementation of Kyber, an efficient implementation of the hash function Keccak will typically be obtained by bitslicing. In this case 32 bits would be manipulated per cycle. By contrast, the 12-bit arithmetic operations will generally lead to 12 bits manipulated by cycle. Assuming a Hamming weight leakage model, where one can roughly approximate $\mathrm{MI}(X;\boldsymbol{L})$ by $\log_2(n)$ when X is an n-bit value, it means that the leakage of 32-bit operations will be $f = \frac{\log_2(32)}{\log_2(12)}$ time larger than the one of 12-bit operations. A completely accurate fine-grain characterization of how the different operations of an implementation of Kyber leak is outside the scope of this work (and would go against the simplicity goal of shortcut formulas). Yet, in order to assess the potential impact of such a characterization, we will next consider two leakage parameters: λ_{32} for 32-bit operations and λ_{12} for 12-bit operations. By default, we will further assume that $\lambda_{32} = f \cdot \lambda_{12}$ with $f = \frac{\log_2(32)}{\log_2(12)}$. As a result, Equations 8 and 9 will be updated as:

$$N_{Dec} \approx \frac{12}{\#var_{Dec} \cdot \lambda_{12}^{d_{Dec}}}\,, \tag{16}$$

since there are no Keccak instances in Kyber.CPAKEM.Dec, and:

$$N_{Enc} \approx \frac{3 \cdot k \cdot n}{\#var_{Enc}^{12} \cdot \lambda_{12}^{d_{Enc}^{12}} + \#var_{Enc}^{32} \cdot \lambda_{32}^{d_{Enc}^{32}}}\,. \tag{17}$$

In the next subsection, we therefore focus on evaluating the number of 12-bit and 32-bit operations that can be exploited in the different sub-computations that are found in the Kyber decryption and re-encryption algorithms.

5.2 Concrete attack parameters

DPA against decryption. The $\mathcal{A}_{\mathrm{DPA}}^{sk}$ adversary exploits the leakage generated by the first few operations in the decryption involving the private key $\hat{\boldsymbol{s}}$ (Algorithm 2 - Line 3). It aims to recover the NTT representation of the private key $\boldsymbol{s}$. More precisely, $\mathcal{A}_{\mathrm{DPA}}^{sk}$ exploits leakage from the product between the two vectors of polynomials $\hat{\boldsymbol{u}} \in \mathbb{R}_q^k$ and $\hat{\boldsymbol{s}} \in \mathbb{R}_q^k$ where $\hat{\boldsymbol{u}}$ is known by the adversary (since derived from the ciphertext). In the specific case of Kyber, the polynomial-wise base multiplications $\hat{\boldsymbol{u}}[i] \circ \hat{\boldsymbol{s}}[i]$ is performed by computing coefficient-wise operations $\hat{\boldsymbol{u}}[i]_{2j} \cdot \hat{\boldsymbol{s}}[i]_{2j}$, $\hat{\boldsymbol{u}}[i]_{2j} \cdot \hat{\boldsymbol{s}}[i]_{2j+1}$, $\hat{\boldsymbol{u}}[i]_{2j+1} \cdot \hat{\boldsymbol{s}}[i]_{2j}$ and $\hat{\boldsymbol{u}}[i]_{2j+1} \cdot \hat{\boldsymbol{s}}[i]_{2j+1}$ for $0 \leq i < k$ and $0 \leq j < n/2$. As a result by requesting one single decryption, $\mathcal{A}_{\mathrm{DPA}}^{sk}$ can directly exploit, thanks to dedicated attacks such as [8], the leakage from the multiplication of every coefficient in $\hat{s}$ with two know coefficients in $\hat{\boldsymbol{u}}$. Hence in this case $\#var_{Dec} = 2$.

SPA against re-encryption. As mentioned above, the $\mathcal{A}^{sk}_{\mathrm{SPA}}$ adversary exploits the leakage of the deterministic re-encryption to gain knowledge on the message m' decrypted by Kyber.CPAPKE.Enc [29]. Next, we evaluate the number of operations in Kyber of which the leakage can be jointly exploited to gain information on m' during the re-encryption. We first estimate the number of operations performed on 32-bit words (i.e., for hash functions). Then, we estimate the number of operations on 12-bit words (i.e., arithmetic operations).

- $\#var^{32}_{Enc}$. The re-encryption involves many calls to hash functions depending directly on the decrypted message m'. The first call to a hash function depending on m' is G in $\mathrm{G}(m'||\mathrm{H}(pk))$ (see Algorithm 4 - Line 2). The others are made within the re-encryption with Kyber.CPAKEM.Enc (Algorithm 1). There, $(1 + 2 \cdot k)$ calls to $\mathrm{PRF}(\cdot, \cdot)$ are performed in order to sample $\boldsymbol{e}_1$, $\boldsymbol{r}$ and e_2 with $\mathrm{CBD}_\eta(\cdot)$. As a result, $(2 + 2 \cdot k)$ calls to hash functions can be exploited by the $\mathcal{A}^{sk}_{\mathrm{SPA}}$ adversary. All of them are based on the Keccak[1600] permutation [6].[7] For each of the calls to the hash functions, because of the input and output length, one single call to Keccak[1600] is performed.[8] Keccak[1600] requires 24 rounds alternating between linear and Sbox layers. Each rounds implies about 320 operations on 32-bit words. As a result, $\mathcal{A}^{sk}_{\mathrm{SPA}}$ exploits jointly the leakage of $\#var^{32}_{Enc} \approx (2 + 2 \cdot k) \cdot 24 \cdot 320$.
- $\#var^{12}_{Enc}$. To model the information that can be extracted from the NTT and NTT^{-1} operations, we assume that each of the intermediate stages in the butterfly algorithm provides independent leakages. As a result, the adversary can exploit the leakage of $n \cdot \log_2 n$ operations for each of the calls to NTT or its inverse. Kyber.CPAPKE.Enc includes $2 + k$ NTT calls, leading to a total of $n \cdot \log_2 n \cdot (2 + k)$ operations on 12-bits from the NTT.

  Besides, Kyber.CPAPKE.Enc also uses 12-bit element-wise operations such as the base multiplication ($\circ$) and additions. The base multiplication implies $2 \cdot n$ operations and the polynomial addition only n. In Algorithm 1, the variable $\boldsymbol{u}$ is computed thanks to k^2 base multiplications and k^2 vector additions since $\hat{\boldsymbol{A}} \in \mathbb{R}_q^{k\times k}$ and $\hat{\boldsymbol{r}} \in \mathbb{R}_q^k$. Similarly, k base multiplications and $k + 2$ additions are needed to derive $\boldsymbol{v}$. Kyber.CPAPKE.Enc also includes $(2k + 1) \cdot n$ calls to CBD, $(k + 1) \cdot n$ calls to Comp. The final comparison consists in $(k + 1) \cdot n$ coefficient-wise operations. Overall, the adversary can exploit a total of $(3k^2 + 8k + 5) \cdot n$ leaking element-wise operations.

  Hence in the following, we take $\#var^{12}_{Enc} \approx n \cdot (3k^2 + 8k + 5 + \log_2 n \cdot (2 + k))$.

5.3 A look at unprotected implementations

In Figure 4, we report the data complexity of attacks against an unprotected ($d = 1$) implementation of Kyber768. The x-axis corresponds to physical noise

[7] Expect for the Kyber 90s versions.

[8] The only exception is for its call in $\mathrm{CBD}_\eta(\mathrm{PRF}(\cdot, \cdot))$ of Algorithm 1 if $\eta = 3$ (for Kyber512). Indeed, in such a case, $\mathrm{CBD}_3(\cdot)$ requires 1526 random bits. This requires two executions of Keccak[1600] since 1526 is larger than the rate of $\mathrm{PRF}(\cdot, \cdot)$.

parameter λ. The y-axis reports the data complexity N which is derived from Equation 8 for the $\mathcal{A}^{sk}_{\text{DPA}}$ adversary and from Equation 9 for the $\mathcal{A}^{sk}_{\text{SPA}}$ adversary, with the concrete parameters estimated above. We can observe that for low noise levels (i.e., large λ values), the DPA attack path leads to stronger attacks. But as the noise increases, the SPA becomes the preferred attack path. Its complexity remains constant for a wide range of noise levels (up to λ values of 10^{-5} per leakage sample), as reflected by the plateau region of the curve. This is because a single oracle access is then sufficient to recover the key, leading to a successful attack in $\#\mathcal{O} = 3 \cdot n \cdot k$ traces. Concretely, for small λ values, the $\mathcal{A}^{sk}_{\text{SPA}}$ requires $(\#var^{32}_{Enc} \cdot f + \#var^{12}_{Enc}) \cdot 12/(\#var_{Dec} \cdot \#\mathcal{O}) \approx 2.8 \cdot 10^2$ less traces than the $\mathcal{A}^{sk}_{\text{DPA}}$, providing a quantitative view on the "curse of re-encryption".

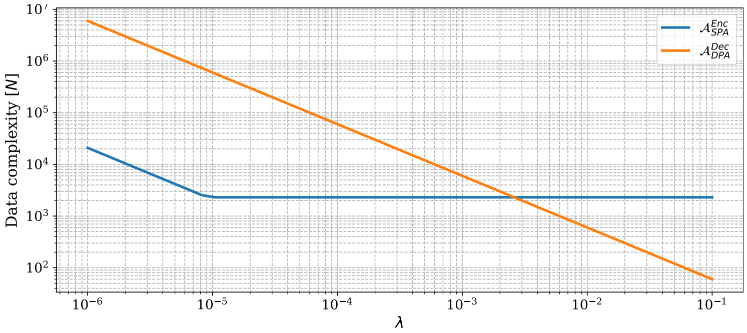

Fig. 4: Data complexity of attacks against unprotected Kyber768.

5.4 Generic intuition revisited

In this subsection, we finally revisit the intuitions put forward in Section 4 with concrete parameters corresponding to a software implementation of Kyber. We aim to derive the implementation cost needed to reach a given security level γ. Concretely, this requires to optimize the number of shares d_{Dec}, d^{12}_{Enc} and d^{32}_{Enc} in Equations 16 and 17 based on the numbers given in Appendix A. We used a grid search for this purpose, considering only integer number of shares.

Masking is (very) expensive. The cycle counts of a protected implementation of Kyber768 is reported in Figure 5. It confirms the large overheads needed to reach high physical security, especially in case of low noise levels. For example, 10^9 cycles are required to reach a 2^{32} security for a noise $\lambda = 0.1$. With a larger noise $\lambda = 0.01$, similar overheads would lead to a security level $\gamma = 2^{74}$. Our results consolidate and quantify the intuition that masking PQ KEMs like Kyber can only be done at realistic cost under sufficient noise levels. Those may not be directly available in low-end embedded microcontrollers (where

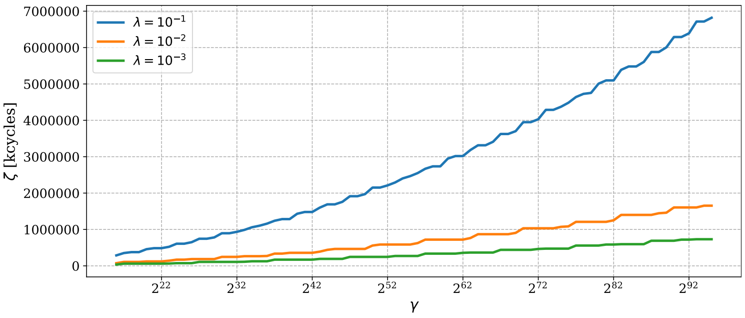

Fig. 5: Cost (kcycles) of Kyber768 for noise parameter λ and security level γ.

$\lambda = 0.1$ is a typical value, see [8] Fig.10), and therefore suggest relying on secure microcontrollers with noise engines or hardware implementations for this purpose.

Levelling still moderately helps. Figure 6 contains the estimated ratio between the cost of encryption and decryption for the minimal d_{Enc} and d_{Dec} leading to a given security level γ. The steps on the curves are due to the grid search on d_{Enc} and d_{Dec} which only considers integer number of shares. Overall Figure 6 also confirms that the impact of the re-encryption in the overall implementation cost decreases as the security level increases. For example, for high security levels, the protected re-encryption is about 12 times slower than the protected decryption for Kyber 768.

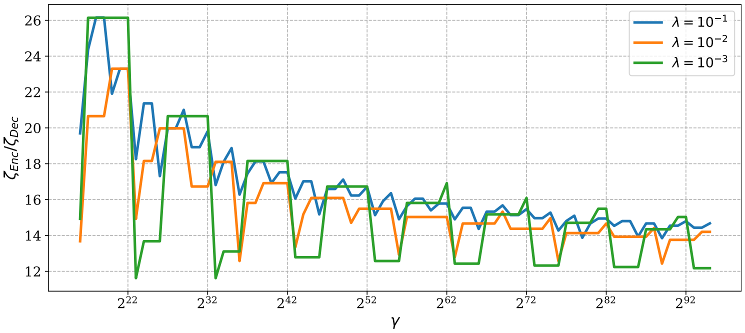

Fig. 6: Ratio between the cost of decryption and re-encryption for Kyber768.

6 Discussion & challenges

The generic analysis in this work enables clarifying the challenges for improving the side-channel security of PQ KEMs. It first confirms that the re-encryption used in the FO transform is an important and hard-to-contain source of leakage. For an unprotected implementation, it gives rise to a very large amount of easy-to-exploit leakage samples. As a result, an SPA against this part of a PQ KEM becomes the best attack as soon as the noise in the implementation increases. Our results also show that the direct application of higher-order masking to prevent such attacks can lead to very expensive implementations. We are unaware of a CPA to CCA FO-like transform that would avoid this (or a similar) issue.

An intuitive design goal following these observations would be to limit the use or even to get rid of the FO transform. Heuristic solutions for this purpose, such as rejecting certain easy-to-exploit ciphertext structures (e.g., with low Hamming weight) or adding redundancy in the plaintext or the key are unlikely to solve the problem in a generic manner [34]. The standard solution used to obtain CCA security with leakage in symmetric cryptography (i.e., to MAC the ciphertext with a careful verification mechanism [5]) is not applicable in the public-key setting. A generic replacement of this solution would be to rely on zero knowledge proof techniques on top of the CPA encryption, in order to avoid using the secret key before verifying the validity of the resulting ciphertext during its decapsulation. By quantifying the performance gap between the decryption and re-encryption parts of a masked KEM, our results clarify the limited budget that such a zero-knowledge proof can use to beat the security vs. performance tradeoff of a uniformly masked implementation. For Kyber, it roughly corresponds to a factor 10 in clock cycles (which ignores the increase of ciphertext size).

Overall, we therefore put forward the paradoxical observation that the FO transform leads to very strong attack vectors agains unprotected PQ KEMs, but that the impact of getting rid of this transform in order to leverage a leveled implementation vanishes as the security level increases. The latter is mostly due to the eventually comparable complexities of a protected decryption and re-encryption, which cannot be amplified by an amortization over long messages like in the symmetric encryption setting. Hence, improving the side-channel security vs. efficiency tradeoff of PQ KEMs like Kyber will not only require to mitigate the leakages due to re-encryption, but also to find a way to make the protected decryption significantly more efficient. Our shortcut formulas directly indicate optimizations that could play a role in this direction, for example reducing the number of exploitable operations reflected by the α parameters in our formulas or leveraging different noise levels in decryption and re-encryption (i.e., playing with the f parameters of Subsection 5.1). Alternatively, finding solutions to optimize the masking complexity of the decryption would also be beneficial. In absence of such advances, ensuring a sufficient noise level in order to limit the number of shares in PQ KEMs appears as the only practical option.

Eventually, while the interest of shortcut formulas lies in their ability to identify design trends, the counterpart of this genericity naturally lies in the limited details they provide about fine-grain implementation-specific issues. Evaluating the impact of the previous optimizations based on concrete case studies and experiments is therefore an interesting scope for further investigations.

Acknowledgments. François-Xavier Standaert is a senior research associate of the Belgian Fund for Scientific Research (F.R.S.-FNRS). This work has been funded in parts by the European Union through the ERC consolidator grant SWORD (project 724725) and by the Walloon Region through the FEDER project USERMedia (convention number 501907-379156).

References

[1] Gorjan Alagic, Gorjan Alagic, Jacob Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Dang, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, et al. *Status report on the first round of the NIST post-quantum cryptography standardization process.* National Institute of Standards and Technology, 2019.

[2] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-kyber algorithm specifications and supporting documentation. *NIST PQC Round*, 3:4, 2019.

[3] Andrea Basso, Jose Maria Bermudo Mera, Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, Michiel Van Beirendonck, and Frederik Vercauteren. Saber algorithm specifications and supporting documentation. *NIST PQC Round*, 3:44, 2019.

[4] Michiel Van Beirendonck, Jan-Pieter D'Anvers, Angshuman Karmakar, Josep Balasch, and Ingrid Verbauwhede. A side-channel-resistant implementation of SABER. *ACM J. Emerg. Technol. Comput. Syst.*, 17(2):10:1–10:26, 2021.

[5] Davide Bellizia, Olivier Bronchain, Gaëtan Cassiers, Vincent Grosso, Chun Guo, Charles Momin, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Mode-level vs. implementation-level physical security in symmetric cryptography - A practical guide through the leakage-resistance jungle. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 369–400. Springer, 2020.

[6] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. The keccak reference, 2011.

[7] Joppe W. Bos, Marc Gourjon, Joost Renes, Tobias Schneider, and Christine van Vredendaal. Masking kyber: First- and higher-order implementations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(4):173–214, 2021.

[8] Olivier Bronchain and François-Xavier Standaert. Breaking masked implementations with many shares on 32-bit software platforms or when the security order does not matter. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(3):202–234, 2021.

[9] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.

[10] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19:297–301, 1965.

[11] Eloi de Chérisey, Sylvain Guilley, Olivier Rioul, and Pablo Piantanida. Best information is most successful mutual information and success rate in side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(2):49–79, 2019.

[12] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. In *EUROCRYPT*, volume 8441 of *Lecture Notes in Computer Science*, pages 423–440. Springer, 2014.

[13] Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. Making masking security proofs concrete - or how to evaluate the security of any leaking device. In *EUROCRYPT (1)*, volume 9056 of *Lecture Notes in Computer Science*, pages 401–429. Springer, 2015.

[14] Tim Fritzmann, Michiel Van Beirendonck, Debapriya Basu Roy, Patrick Karl, Thomas Schamberger, Ingrid Verbauwhede, and Georg Sigl. Masked accelerators and instruction set extensions for post-quantum cryptography. *IACR Cryptol. ePrint Arch.*, page 479, 2021.

[15] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer, 1999.

[16] W. Morven Gentleman and G. Sande. Fast fourier transforms: for fun and profit. In *American Federation of Information Processing Societies: Proceedings of the AFIPS '66 Fall Joint Computer Conference, November 7-10, 1966, San Francisco, California, USA*, volume 29 of *AFIPS Conference Proceedings*, pages 563–578. AFIPS / ACM / Spartan Books, Washington D.C., 1966.

[17] Vincent Grosso and François-Xavier Standaert. Masking proofs are tight and how to exploit it in security evaluations. In *EUROCRYPT (2)*, volume 10821 of *Lecture Notes in Computer Science*, pages 385–412. Springer, 2018.

[18] Qian Guo, Vincent Grosso, François-Xavier Standaert, and Olivier Bronchain. Modeling soft analytical side-channel attacks from a coding theory viewpoint. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(4):209–238, 2020.

[19] Daniel Heinz, Matthias J. Kannwischer, Georg Land, Thomas Pöppelmann, Peter Schwabe, and Daan Sprenkels. First-order masked kyber on arm cortex-m4 (work in progress), 2021.

[20] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the fujisaki-okamoto transformation. In *TCC (1)*, volume 10677 of *Lecture Notes in Computer Science*, pages 341–371. Springer, 2017.

[21] Loïs Huguenin-Dumittan and Serge Vaudenay. Classical misuse attacks on NIST round 2 PQC - the power of rank-based schemes. In Mauro Conti, Jianying Zhou, Emiliano Casalicchio, and Angelo Spognardi, editors, *Applied Cryptography and Network Security - 18th International Conference, ACNS 2020, Rome, Italy, October 19-22, 2020, Proceedings, Part I*, volume 12146 of *Lecture Notes in Computer Science*, pages 208–227. Springer, 2020.

[22] Yuval Ishai, Amit Sahai, and David A. Wagner. Private circuits: Securing hardware against probing attacks. In *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.

[23] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Des. Codes Cryptogr.*, 75(3):565–599, 2015.

[24] Stefan Mangard, Elisabeth Oswald, and François-Xavier Standaert. One for all - all for one: unifying standard differential power analysis attacks. *IET Inf. Secur.*, 5(2):100–110, 2011.

[25] Kalle Ngo, Elena Dubrova, Qian Guo, and Thomas Johansson. A side-channel attack on a masked IND-CCA secure saber KEM implementation. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(4):676–707, 2021.

[26] Peter Pessl and Robert Primas. More practical single-trace attacks on the number theoretic transform. In *LATINCRYPT*, volume 11774 of *Lecture Notes in Computer Science*, pages 130–149. Springer, 2019.

[27] Robert Primas, Peter Pessl, and Stefan Mangard. Single-trace side-channel attacks on masked lattice-based encryption. In *CHES*, volume 10529 of *Lecture Notes in Computer Science*, pages 513–533. Springer, 2017.

[28] Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In *EUROCRYPT*, volume 7881 of *Lecture Notes in Computer Science*, pages 142–159. Springer, 2013.

[29] Prasanna Ravi, Sujoy Sinha Roy, Anupam Chattopadhyay, and Shivam Bhasin. Generic side-channel attacks on cca-secure lattice-based PKE and kems. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(3):307–335, 2020.

[30] François-Xavier Standaert. Towards and Open Approach to Secure Cryptographic Implementations (Invited Talk). In *EUROCRYPT I*, volume 11476 of *LNCS*, pages xv, `https://www.youtube.com/watch?v=KdhrsuJT1sE`, 2019.

[31] François-Xavier Standaert, Tal Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In *EUROCRYPT*, volume 5479 of *Lecture Notes in Computer Science*, pages 443–461. Springer, 2009.

[32] Rei Ueno, Keita Xagawa, Yutaro Tanaka, Akira Ito, Junko Takahashi, and Naofumi Homma. Curse of re-encryption: A generic power/em analysis on post-quantum kems. *IACR Cryptol. ePrint Arch.*, page 849, 2021.

[33] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Soft analytical side-channel attacks. In *ASIACRYPT (1)*, volume 8873 of *Lecture Notes in Computer Science*, pages 282–296. Springer, 2014.

[34] Zhuang Xu, Owen Pemberton, Sujoy Sinha Roy, and David F. Oswald. Magnifying side-channel leakage of lattice-based cryptosystems with chosen ciphertexts: The case study of kyber. *IACR Cryptol. ePrint Arch.*, 2020:912, 2020.

A Masked Kyber.CCAKEM.Dec

In Table 2, we show the performance values for masked Kyber.CCAKEM.Dec for different masking orders based on [7] but with an optimized comparison. These values are provided for the STM32F407G (ARM Cortex-M4) MCU and up to order 6, since the maximum stack size on the target MCU is 112 KiB (for SRAM1). Starting from *2nd* order, the implementation requires more than 13.5 KiB of stack size for each subsequent order. The *7th* order masked code requires at least 114 KiB of stack.

Operation	**Order**					
	1st	*2nd*	*3rd*	*4th*	*5th*	*6th*
`crypto_kem_dec`	3 178	57 141	97 294	174 220	258 437	350 529
`indcpa_dec`	200	4 203	7 047	13 542	20 323	27 230
`unpack`	24	30	36	43	50	56
`poly_arith`	89	119	148	598	713	790
`compression`	87	4 054	6 863	12 901	19 561	26 384
`indcpa_enc`	2 024	18 879	32 594	53 298	75 692	104 191
`decompression`	118	291	537	889	1 267	1 745
`gen_at`	391	391	391	391	391	391
`poly_getnoise`	1 217	17 727	31 069	49 390	70 856	98 435
`prf`	706	11 483	19 577	30 318	43 541	60 640
`cbd`	510	6,243	11 492	19 071	27 314	37 794
`poly_arith`	302	453	603	2 627	3 172	3 643
`ntt`	66	99	131	585	670	817
`invntt`	70	105	140	1 008	1 274	1 410
`comparison`	693	32 293	54 725	102 922	156 075	210 518
`hashg (SHA3-512)`	98	1 639	2 801	4 489	6 456	8 794
`hashh (SHA3-256)`	113	113	113	113	113	113
`kdf`	13	13	13	13	13	13

Table 2: Performance numbers for masked Kyber.CCAKEM.Dec and its subroutines (values are in kCycles).