Leveled Software Implementation of Polka and Comparison with Uniformly Masked Kyber

Thibaud Schoenauen[1], Clément Hoffmann[1], Charles Momin[1], Thomas Peters[1], and François-Xavier Standaert[1]

[1] Crypto Group, ICTEAM Institute, UCLouvain, Louvain-la-Neuve, Belgium

Abstract. Polka is a post-quantum public-key encryption scheme from PKC 2023, designed in order to be efficiently protected against side-channel attacks. Its motivation arises from the acknowledged difficulty of protecting Kyber against such attacks. Concretely, the structure of Polka aims to allow so-called leveled implementations, so that protecting its long-term key requires strong and expensive countermeasures (like masking) for a part of its operations only. This contrasts with Kyber, for which preventing side-channel attacks requires to uniformly protect all its operations. The good leakage-resilience features of Polka nevertheless come with performance overheads in an unprotected implementation context. Since no concrete implementations of Polka were proposed so far, it left the question of the number of shares for which it can become an interesting alternative to Kyber open. We bridge this gap by proposing a leveled software implementation of Polka and show that, already for two shares, it leads to significant performance gains over the state-of-the-art uniformly masked implementations of Kyber (Bos et al., TCHES 2021, Bronchain and Cassiers, TCHES 2022).

1 Introduction

CRYSTALS-Kyber (short: Kyber) is a new standard for key encapsulation and public-key encryption [1]. While it is expected to provide strong security even in the presence of quantum computers, securing its implementations against side-channel attacks has been shown to be very challenging. Informally, one of the main reasons of this difficulty is that the security of Kyber against Chosen-Ciphertext Attacks (CCA) relies on the so-called Fujisaki-Okamoto (FO) transform [17], which essentially checks that ciphertexts are well formatted by decrypting and re-encrypting them. As a result, a side-channel adversary can perform a chosen-ciphertext attack against the part of Kyber that is only secure against Chosen-Plaintext Attacks (CPA), by observing the leakage of a decryption before the re-encryption step [27,31,32,35,36,38]. This is a severe weakness since exploiting it only requires to target the confidentiality of the re-encrypted message with a distinguishing attack, a task that is considerably easier than performing a key-recovery [34]. Informally, this is because message distinguishing attacks can directly exploit the leakage of all the target operations of a

leaking implementation, without key guessing. As a result, the traditional intuitions prevailing when analyzing the implementation of symmetric cryptographic primitives against leakage may not extend to the post-quantum asymmetric setting. For example, the aforementioned message distinguishing attack is a Simple Power Analysis (SPA) exploiting the measurement of a single manipulation of an ephemeral secret. Yet, it frequently turns out to be more powerful (hence, harder to prevent) than a Differential Power Analysis (DPA) attacks against Kyber's decryption, which exploits multiple manipulations of the long-term secret [2].

Given the anticipated cost to secure Kyber with standard countermeasures like masking [6,7,11,13,14], Polka was introduced as an alternative CCA-secure post-quantum public-key encryption scheme [20]. It embeds a number of features aimed at simplifying secure implementation. First, Polka is designed in such a way that CCA security is obtained without relying on the FO transform. Second, its decryption process is randomized in order to remove adversarial control on intermediate computations that can facilitate side-channel attacks. Third, it leverages key-homomorphic computations that are easy to mask. Finally, it can rely on hard physical learning problems to argue about the security of some (unmasked) operations [16,21]. As a result, Polka is expected to enable leveled implementations, as popular in symmetric cryptography [5]. That is, it aims to be implemented securely without uniformly masking all its operations.

It is important to emphasize that a leveled implementation of Polka and a uniformly masked implementation of Kyber lead to different security guarantees. Borrowing the terminology used in symmetric cryptography [19], a leveled implementation of Polka is *leakage-resilient*: its ephemeral secrets may be compromised in the presence of leakage, but security is restored once leakage is removed from the adversary's view. In other words, a leveled implementation of Polka only protects its long-term secret key. By contrast, a uniformly protected implementation of Kyber is *leakage-resistant* and also protects its ephemeral secrets. In this work, we are therefore concerned with the situation where implementers want to efficiently ensure leakage-resilience. This is a natural first step since, for example, all the aforementioned attacks against Kyber's FO-transform target its long-term secret key. But if implementers want leakage-resistance guarantees, the interest of Polka over Kyber currently vanishes.

Given this cautionary note, the interesting consequence of Polka's design, that we want to investigate, is that the asymptotic performances of its leveled implementation scale linearly in the number of shares used for its masked computations. By contrast, the performances of a uniformly masked Kyber scale quadratically in the number of shares. Since this trend comes at the cost of some overheads for Polka's unprotected implementation, it raises the question of *when (i.e., for which number of shares) can Polka become an interesting alternative to Kyber (assuming that only leakage-resilience is required)?*

In order to answer this question, we describe a leveled software implementation of Polka taking advantage of standard optimization techniques for post-quantum cryptography. Doing so, we also instantiate the few components that its

authors left unspecified and slightly tweak the proposal of [20] in order to further reduce the side-channel attack surface. As a result, we put forward that despite the overheads of its unprotected implementation, a leveled implementation of Polka leads to better performances than a uniformly masked implementation of Kyber on an ARM Cortex-M4 device, already with $d = 2$ shares.

We then wrap up the paper by discussing the remaining challenges in order to turn this improved but qualitative "performances vs. number of shares tradeoff" into an improved quantitative "performances vs. side-channel security tradeoff". Conjecturing that key recovery attacks against Polka should be significantly more difficult to perform than message distinguishing attacks against Kyber, we conclude that Polka, or more generally alternative encryption schemes tailored for improved side-channel security, have a strong potential to considerably improve this tradeoff for post-quantum public-key encryption schemes. We complete this discussion with a number of interesting open problems.

2 Background

We start with some preliminaries needed for the understanding of the paper. Namely, we give more details about the Polka algorithm that we will fully instantiate and implement in subsection 2.1, and about the NTT that will be the main tool for the polynomial arithmetic operations of our implementation in subsection 2.2. We do not provide details about Kyber that we do not re-implement and for which various comprehensive descriptions can be found online. The most relevant references for our purposes are the masked implementations in [6, 7], of which we will extract performance tables for our comparisons.

2.1 Polka

Polka is a CCA-secure post-quantum public-key encryption scheme tailored for efficiency when implemented with side-channel security guarantees [20]. Its black-box security relies on the Ring Learning With Errors (RLWE) assumption [24] in the Quantum Random Oracle Model (QROM), while its side-channel efficiency stems from heuristic design tweaks aimed at lowering the overheads of countermeasures. The scheme comes with two noise distribution variants over the coefficients of the polynomials: either centered Gaussian or centered binomial distributions. Here, we focus on the latter version, like Kyber.

While the structure of Polka follows the KEM-DEM paradigm by deriving an ephemeral secret key from an asymmetric key encapsulation part (KEM) to encrypt the message with a symmetric deterministic encryption (DEM), the way its decryption rejects ill-formed ciphertexts differs from most post-quantum public-key encryption schemes designed for practical use. Polka rejects ciphertexts through norm checks on re-extracted noises to confer a rigidity property without relying on the FO transform [17]. This eliminates Kyber's critical re-encryption step and the associated attacks listed in introduction.

Polka also includes additional tricks in its black-box design to ease side-channel countermeasures. For example, its KEM part leverages the bounded additive homomorphy of RLWE ciphertexts to randomize the intermediate values in the decryption step, and ensures that the long-term key is only used in linear operations that can be masked with linear overheads. Besides, Polka can leverage a structured version of the Learning With Physical Rounding (LWPR) problem [16, 21], a physical learning problem of which the heuristic hardness is used to further increase its opportunities of leveled implementation.

Algorithmic description. We define a noise distribution, to be used over integer vector coefficients, as $\{\sum_{i=1}^{2}(u_i - v_i) \bmod 3 \mid u_i, v_i \leftarrow \{0,1\}\}$, where mod 3 means the integer representative over $\{-1, 0, 1\}$. The resulting ternary distribution over $\mathcal{R} = \mathbb{Z}_q[X]/(X^n + 1)$ is denoted as $\mathcal{D}_{n,B}^{\mathsf{coeff}}$, where $B = 1$. For any $r \leftarrow \mathcal{D}_{n,B}^{\mathsf{coeff}}$, we have $\|r\| \leq B$ for the infinity norm over the vector of coefficients. We refer to [20] for the key generation, which produces $b = p(as + e) \in \mathcal{R}^*$ from $s, e \leftarrow \mathcal{D}_{n,B}^{\mathsf{coeff}}$ and $p \geq 2B + 1 \in \mathbb{Z}$, and defines the secret key as s. Next, we describe the encryption and decryption steps of Polka:

Encrypt: Given a public key (n, q, p, a, b) and a message $M \in \{0,1\}^{\ell_m}$:

1. Sample $r, e_1, e_2 \leftarrow \mathcal{D}_{n,B}^{\mathsf{coeff}}$ and compute the KEM part
$$c_1 = a \cdot r + e_1 \in \mathcal{R}, \qquad c_2 = b \cdot r + e_2 \in \mathcal{R}$$
together with $K = \mathsf{H}(r, e_1, e_2) \in \{0,1\}^{\kappa}$.
2. Compute $c_0 = \mathsf{E}_K(M)$ as the DEM part.

Output the ciphertext $C = (c_0, c_1, c_2)$.

Decrypt: Given a secret key $s \in \mathcal{R}$ and $C = (c_0, c_1, c_2)$, do the following:

1. Sample $r', e_1', e_2' \leftarrow \mathcal{D}_{n,B}^{\mathsf{coeff}}$ and compute $c_1' = a \cdot r' + e_1'$ and $c_2' = b \cdot r' + e_2'$.
2. Compute $\bar{c}_1 = c_1 + c_1'$ and $\bar{c}_2 = c_2 + c_2'$.
3. Compute $\bar{\mu} = \bar{c}_2 - p \cdot \bar{c}_1 \cdot s$ over $\mathcal{R}$.
4. Compute $\bar{e}_2 = \bar{\mu} \bmod p \in \{-\frac{p-1}{2}, \ldots, \frac{p-1}{2}\}$. If $\|\bar{e}_2\| > 2B$, return $\perp$.
5. Compute $\bar{r} = (\bar{c}_2 - \bar{e}_2) \cdot b^{-1} \in \mathcal{R}$. If $\|\bar{r}\| > 2B$, return $\perp$.
6. Compute $\bar{e}_1 = \bar{c}_1 - a \cdot \bar{r} \in \mathcal{R}$. If $\|\bar{e}_1\| > 2B$, return $\perp$.
7. Compute $r = \bar{r} - r'$, $e_1 = \bar{e}_1 - e_1'$ and $e_2 = \bar{e}_2 - e_2'$. If $\|r\| > B$, or $\|e_1\| > B$, or $\|e_2\| > B$, then return $\perp$.
8. Compute $K = \mathsf{H}(r, e_1, e_2) \in \{0,1\}^{\kappa}$ and return
$$M = \mathsf{D}_K(c_0) \in \{0,1\}^{\ell_m} \cup \{\perp\}.$$

As suggested in [20], we choose to pre-compute the inverse of b and store it in the public key to avoid computing it on the decryption step.

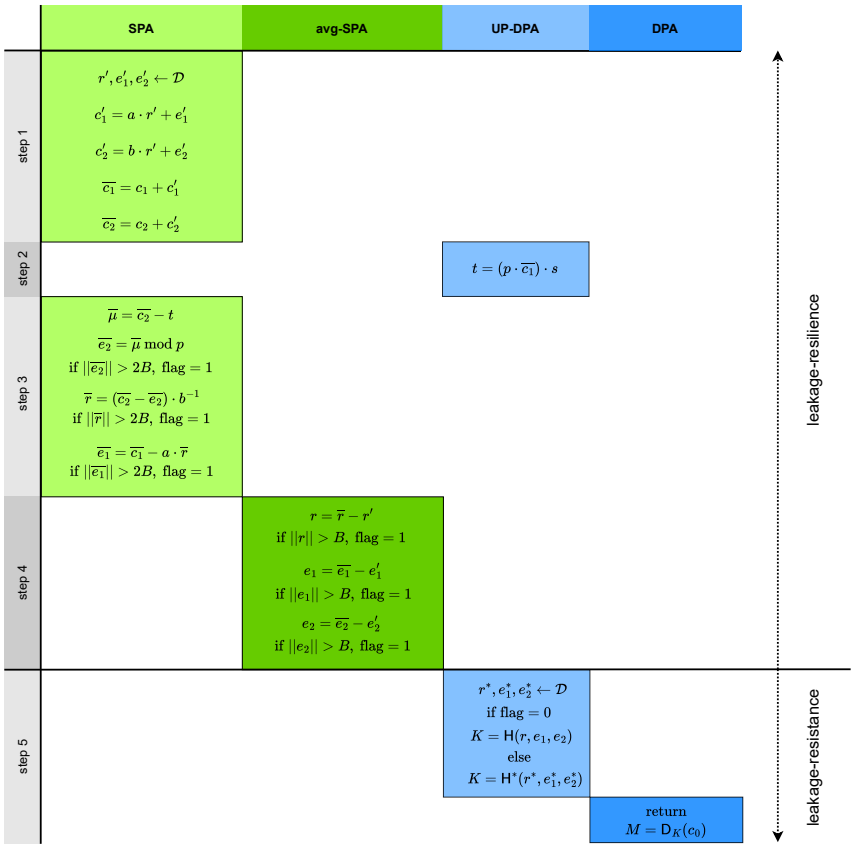

Fig. 1: Leveled implementation of Polka, from [20].

Leveled implementation. We now describe the protected implementation of Polka's decryption proposed in [20], for which we re-use the visual description of Figure 1. The color codes indicate the type of side-channel attacks that must be prevented by implementers. Light green means SPA security for randomized values, dark green means "SPA with repetition" security for non-randomized values (i.e., SPA with repeated measurements of the same value), light blue means DPA security for key-homomorphic operations manipulating unknown (randomized) inputs and dark blue is for standard DPA security. As discussed in introduction, our focus in this paper is on the leakage-resilience guarantees of Polka (i.e., the upper part of the figure, where the long-term secret is manipulated).

The general structure of Polka, that rejects ill-formed ciphertexts without FO transform thanks to norm checks, directly removes one critical attack vector

that severely affects the side-channel security of KYBER. We next describe the other design features that support the leveled implementation of POLKA.

Dummy ciphertexts. When many traces of the same operation with identical inputs are collected, adversaries can combine them in order to reduce the noise in the leakage measurements (e.g., thanks to averaging). In POLKA, intermediate values are expressed as RLWE samples of the form $c = a \cdot r + e$, where a is public, and r and e are small secret values that the adversary aims to retrieve. To prevent such an averaging, POLKA introduces randomized dummy ciphertexts of the form $c' = a \cdot r' + e'$, where r' and e' are small random polynomial, and combines them with the original ciphertext as a pseudorandom $\bar{c} = c + c'$, which is unknown to the adversary. The security proofs are then adjusted in order to account for the new norms of these randomized intermediate values. While this approach incurs some mild overheads for generating the dummy ciphertexts, it effectively transforms a SPA with repetition attack path into a regular SPA attack path, significantly simplifying the task of protecting the operations concerned.

Key-homomorphic operations and (Ring-)LWPR. Despite having a DPA attack path seems unavoidable in any decryption scheme that must manipulate a long-term secret key, POLKA is designed in such a way that the main operation manipulating this long-term secret (i.e., s in step 2 of Figure 1) is key-homomorphic. As a result, it can be efficiently performed share by share (i.e., with overheads that scale linearly in the number of shares). Furthermore, under the LWPR assumption, it is possible to unmask the value of t so that all the norm checks of steps 3 and 4 can be performed on unshared values. The LWPR assumption is a physical learning problem that has been heuristically shown to be computationally hard [16, 21]. Standard LWPR samples have the form $(r, \mathsf{L}(r \cdot s))$, where r is a random public vector, s is a secret vector, and the leakage function L acts as an injective rounding function. In POLKA, the polynomial product involving the long-term secret naturally suggests a ring variant of the LWPR problem.

2.2 NTT

Let Φ_N be the N-th cyclotomic polynomial and q be a prime number. Since $\Phi(X)$ has integer coefficients, we can consider the polynomial ring $\mathcal{R} = \mathbb{Z}_q[X]/(\Phi(X))$, where $\mathbb{Z}_q$ is the field of integers with addition and multiplication modulo q, i.e., $\mathbb{Z}/(q)$. As $\Phi(X)$ factors into coprime irreducible polynomials $f_i(X)$ of degree δ in $\mathcal{R}$ if the order of q modulo N is δ, the well-known Chinese Reminder Theorem ensures that we have the ring isomorphism $\mathcal{R} \approx \mathcal{R}_1 \times \cdots \times \mathcal{R}_{\varphi(N)/\delta}$, where $\mathcal{R}_i = \mathbb{Z}_q[X]/(f_i(X))$ with $\mathcal{R} \to \mathcal{R}_i; a \to a \bmod f_i(X)$. The naive multiplication in $\mathcal{R}$ has an asymptotic cost of n^2 multiplications in $\mathbb{Z}_q$, where $n = \varphi(N)$ is the degree of Φ_N. Assuming that q is chosen given N so that δ is constant, the corresponding multiplication can be carried out using only around δ^2 multiplications modulo q in each $\mathcal{R}_i$, leading to $\mathcal{O}(n)$ such multiplications. For well-chosen N values, typically a power of 2, the Number Theoretic Transform (NTT) then allows

applying the above isomorphism back and forth in $\mathcal{O}(n \log n)$ operations in $\mathbb{Z}_q$, with the same complexity to perform multiplications over $\mathcal{R}$.

For Kyber, we have $N = 512$ and the prime $q = 3329$ of order 2 modulo N. For Polka, we have $N = 2048$ and the prime $q = 59393$ of order 1, thus a fully splitting ring $\mathcal{R}$. In both cases, we have $2n = N$ and $\Phi_N(X) = X^n + 1$.

3 Instantiation and tweaks

In this section, we first complete the specifications of Polka by instantiating the few components that were left open by its authors in subsection 3.1. We next bring relevant clarifications to the leveled implementation of Figure 1 and slightly tweak it in order to further reduce the side-channel attack surface.

3.1 Choice of symmetric primitives

Step 5 of Polka requires a hash function. As standard in post-quantum cryptography, we use SHA-3 for this purpose.[1] As a new standard, it benefits from various open implementations. It is also natural given our goal to compare the performances of Polka with the ones of Kyber (which relies on SHA-3).

Besides, Step 5 of Polka requires an Authenticated Encryption (AE) scheme. The authors of [20] proposed an instantiation based on a key-homomorphic MAC, which may be interesting in order to upgrade the leakage-resilience guarantees of Polka towards leakage-resistance ones. Yet, as is, this solution is specialized to fixed-length messages which makes Polka less comparable to Kyber. It is also not sufficient to make Polka leakage-resistant (since DPA can anyway target its hash function). Given our restricted goal of leakage-resilient implementation, we therefore stepped back to a more standard choice, which is to select an AE from the NIST lightweight cryptography competition. Among the candidates, Saturnin appeared as a natural choice, due to its focus on post-quantum security which we also target [9]. Our implementations rely on the (unprotected) `C` implementation made available with its submission to the NIST.

3.2 Clarifications and improvements

Despite Figure 1 highlights the security requirements of a leveled implementation of Polka, it does not directly clarify which parts of the computations are masked, nor which operations are performed in the NTT domain. The updated Figure 2 provides such clarifications. First, we use the "widehat" notation for all the operations that are performed in the NTT domain. Changes from hatted to non-hatted variables (and vice-versa) therefore indicate where NTTs and inverse NTTs must be computed. Second, we use subscripts in Step 2 of the figure in order to clarify the operations that are computed per share due to masking.

[1] `https://csrc.nist.gov/pubs/fips/202/final`.

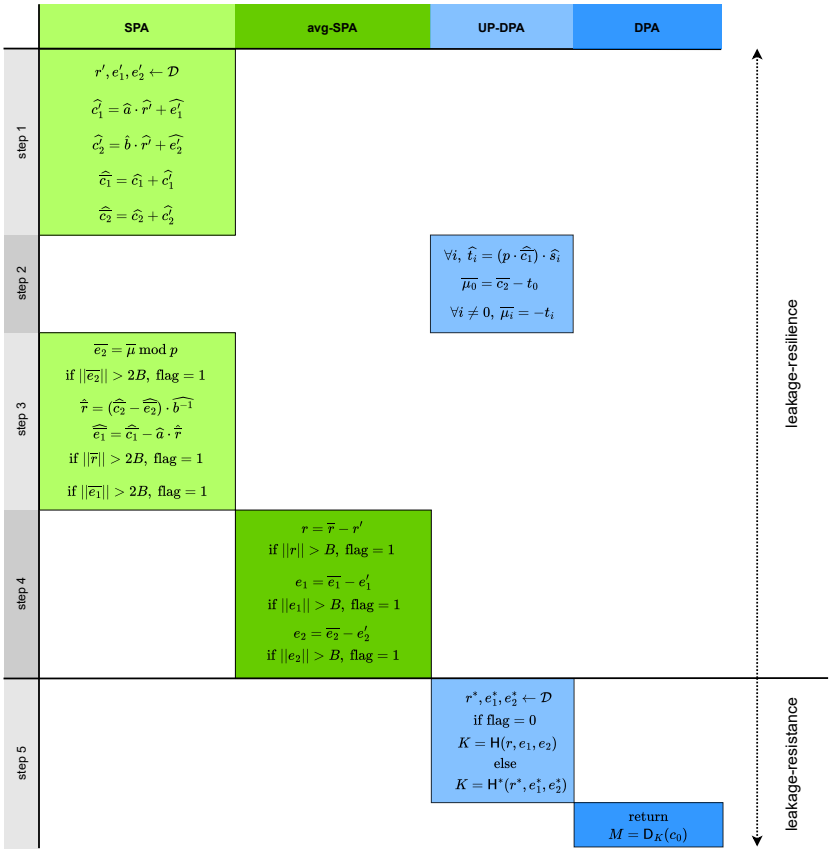

Fig. 2: Tweaked leveled implementation of Polka.

We note that while the impact of masking and unmasking on side-channel vulnerability is evident, NTT conversions can also have an impact in this respect. Specifically, the result of the product in Step 2 must not remain in its NTT representation when unmasked. The hardness analysis of LWPR indeed relies on classical polynomial multiplication [16, 21], as the algebraic structure of point-wise multiplication in the NTT domain offers weaker key mixing. For efficiency, the product can still be computed in the NTT domain. However, the result must undergo a masked inverse NTT before unmasking. In Polka, this constraint aligns naturally with the subsequent operation (i.e., a modular reduction) which cannot be performed in the NTT domain anyway. Interestingly, this observation suggests a slight improvement of Polka's leveled implementation that is also reflected in Figure 2. Namely, since we need to exit the NTT domain before unmasking, we can perform the subsequent subtraction operation in the masked

domain. Such a change comes at no additional cost (since the subtraction applies to a single share) and further minimizes the exposition of sensitive values.

4 Implementation choices

In this section, we outline the implementation choices made for Polka, focusing on polynomial multiplication and modular arithmetic techniques.

4.1 Polynomial multiplication

As with all RLWE-based schemes, Polka's computations rely heavily on polynomial multiplications in rings. This operation constitutes a significant portion of the overall computation cost. When performed using a naive method, the complexity is in $\mathcal{O}(n^2)$, where n is the polynomial degree. Fortunately, Polka's ring structure is chosen to allow the use of the NTT (described in subsection 2.2). The NTT converts polynomials to their image in the NTT domain. The naive implementation of the NTT runs in $\mathcal{O}(n^2)$ but using the Cooley-Tukey butterfly algorithm [10] (and the Gentleman-Sande one for the inverse NTT [18]) this complexity becomes $\mathcal{O}(n \log n)$. Once in the NTT domain, polynomial multiplications can be performed point-wise, reducing the complexity to $\mathcal{O}(n)$.

4.2 Switches between representations

The polynomial operations in Polka include additions, multiplications, norm checks, and small modular reductions. Additions can be performed in either representation, as long as all terms share the same domain. However, for efficiency reasons, multiplications must be carried out in the NTT representation, whereas norm checks and modular reductions require the polynomials to be in their natural representation. In order to minimize the number of representation switches, the polynomials in the public and secret keys are stored directly in their NTT representations. From Figure 2, we can then directly count that Polka's decryption requires 4 NTT and $3 + d$ inverse NTT operations, where d represents the number of shares used for the masked computations of Step 2.

4.3 Modular arithmetic

Most of Polka's operations ultimately reduce to perform modular arithmetic over the integers. However, the default implementation of modular operations in C neither runs in constant time nor achieves optimal performance. In order to address these inefficiencies, we employ Montgomery's and Barrett's reduction algorithms, both widely used in cryptographic implementations.

Montgomery's reduction. Montgomery's reduction is an efficient algorithm for modular multiplication [26]. Its key insight is that performing divisions and modular operations with powers of two is significantly faster than with arbitrary

values. To leverage this, we first compute the number m of multiples of q that need to be added to an input T to make it divisible by $R = 2^{32}$. We then efficiently divide $(T + m \cdot q)$ by R to obtain a result in the range $[0, 2q)$. The original algorithm includes a final step to subtract q if necessary, ensuring that the output strictly lies within $[0, q)$. However, depending on subsequent operations in Polka, this step can often be skipped, allowing the result to remain in $[0, 2q)$. This procedure yields the value $TR^{-1} \pmod q$ instead of $T \pmod q$. To recover the desired result, we pre-multiply T by $R \pmod q$, which comes for free if T arises from a product between a value and a constant, as in the NTT.

Barrett's reduction. While Montgomery reduction is highly efficient, it has two drawbacks: it performs poorly when reducing overflowed values (e.g., sums), and its output can exceed q. In cases where these issues are problematic, we use Barrett's reduction [4] instead. Barrett's reduction replaces division by the modulus q with a cheaper multiplication and bit-shift operation. Although it is slower than Montgomery's reduction, it guarantees the output remains in the range $[0, q)$, making it preferable when strict modular bounds are required.

Both for Montgomery's and Barret's reductions, we actually rely on improved versions that are able to deal with signed inputs, proposed by Seiler [33].

4.4 Small hash function input

A naive approach to organize the hash function input in Step 5 of Figure 2 would be to concatenate the coefficients of r, $e1$ and $e2$ in a big array. However, this approach would make the input of this hash function quite large (about 100 kilobits) and therefore long to process. A better way to organize this input is to stack several coefficients per register. Indeed, r, $e1$ and $e2$ are guaranteed to be small polynomials with coefficients among three values. Therefore, it is possible to represent every of those coefficients with only two bits and to construct our hash input by storing chunks of coefficients in single variables. This technique allows us to reduce the size of the hash input to roughly 6 kilobits.

5 Results and discussions

In this section, we finally benchmark both Polka and Kyber, analyze Polka 's performances per decryption step and operation and compare the performances of the two algorithms, in the unprotected and protected settings.

5.1 Experimental setup

In order to perform our measurements, we ran our code on the NUCLEO-L4R5ZI board with an ARM Cortex-M4 32-bit micro-controller as required by the framework `pqm4` [23], and more specifically the fork `pqm4_masked` [7]. The measurements of Kyber were done using the code from [7] for Kyber768. The `pqm4_masked` framework allows defining personalized bench cases. In addition to

the bench cases related to `Encrypt` and `Decrypt`, we defined one bench case per decryption step, one per polynomial operator (addition, substraction, multiplication, scalar product, randomization, NTTs, inverse NTTs, norm computation, mask refreshing, modular reduction), one for Saturnin and one for Keccak.

5.2 Analysis of Polka' performances

We first focus on Polka's decryption step, which is the most sensitive from the side-channel analysis viewpoint. We therefore observe the cycle counts and their evolution as the number of shares increases, first per step in `Decrypt` and then per type of operation in `Decrypt`, in Figure 3 and Figure 4, respectively.[2]

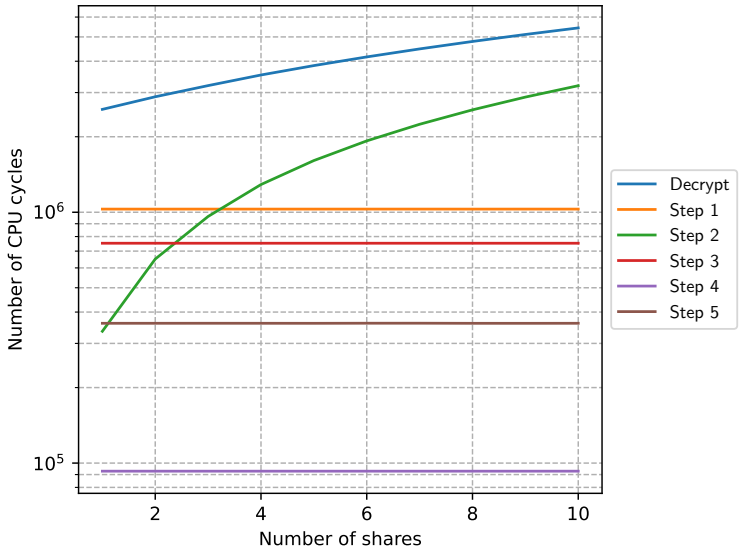

Fig. 3: Cycle counts for each decryption step in Polka

Figure 3 leads to the expected observation that the performances of a leveled implementation of Polka's decryption scale linearly with the number of shares used to protect its long-term key, which is a direct outcome of key-homomorphic computations. For the rest, we note that when the number of shares is low ($d < 4$), most of the computation time is spent in the first and third steps,

[2] Our implementations are not perfectly constant time, mostly due to the randomness generation and the implementation of Saturnin not being constant time. Yet, the standard deviations observed are three orders of magnitude smaller than the mean values we report, so they do not affect our conclusions regarding the performance trends of Polka implementations when their number of shares increases.

which are the steps dealing with dummy ciphers. We observe that the cost of those steps is constant, since they are performed on unmasked data. Beyond 4 shares, the time spent in the second step (i.e., the masked multiplication) starts to dominate as it is the only step performed on masked data.

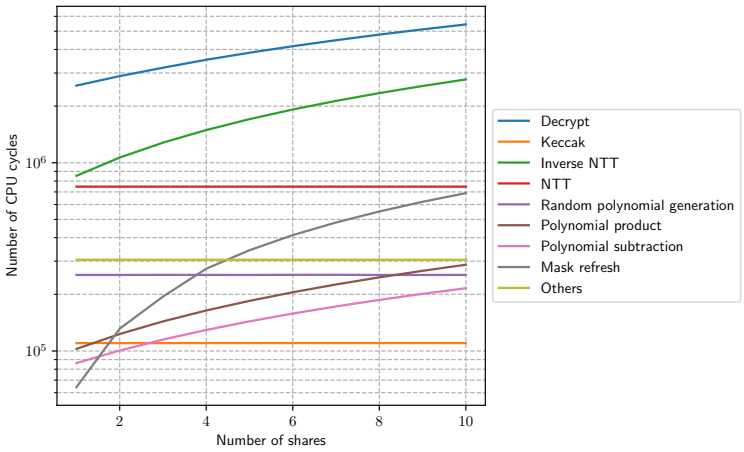

Fig. 4: Cycle counts for the main operations in Polka's decryption.

Figure 4 offers a complementary view. It highlights that most of the computation time is spent on the NTTs and inverse NTTs. It also shows that only the cycles spent to perform the inverse NTT, refreshing, multiplication and subtraction operations (i.e., the operations of Step 2 in Figure 2) increase linearly with the number of shares. We note that the refresh step is increasing faster than the other routines. This is due to the fact that refreshing a mask requires $d-1$ polynomial randomizations, $d-1$ additions and $d-1$ subtractions.

5.3 Comparison with Kyber

We finally compare the performances of Polka and Kyber, first for unprotected implementations in Table 1, then for their protected decryption in function of the number of shares, in Figure 5. We recall that the physical security guarantees targeted by both implementations are different: leakage-resistance for the uniformly masked Kyber, leakage-resilience for the leveled Polka.

Table 1 shows that, as expected, the leakage-resilience features of Polka come with overheads in the unprotected setting: its decryption is about four times slower than the one of Kyber. We note that the Polka's encryption is slightly faster, which we assume is due to Kyber's compression step.

Figure 5 shows that these overheads are rapidly compensated when side-channel protections are needed. Already for $d = 2$ shares, Polka is twice faster

	Kyber	Polka
`Encrypt`	957,176	880,201
`Decrypt`	707,827	2,571,178

Table 1: Cycle counts for Kyber and Polka's unprotected implementation.

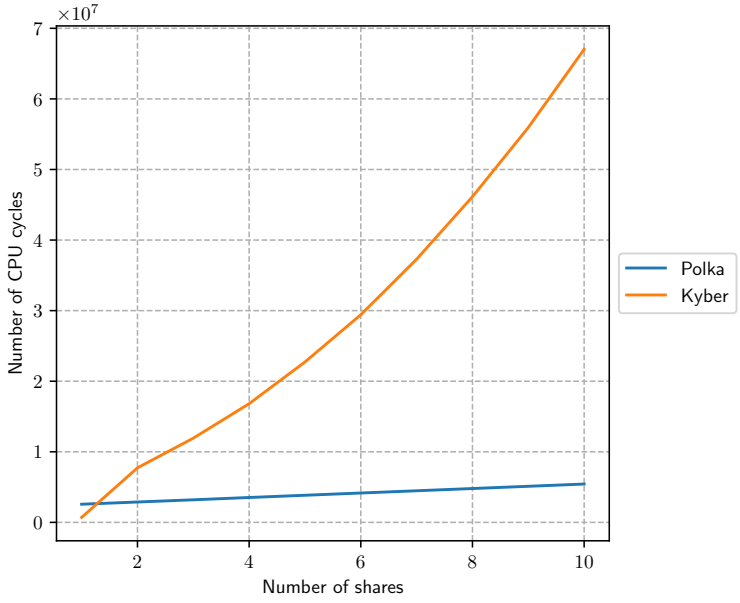

Fig. 5: Cycle counts for Kyber and Polka's protected decryption.

than Kyber. This factor grows with d and improves to 8 for 8 shares. We discuss the expected quantitative impact of these figures in the next conclusions.

6 Conclusion and open problems

Our results essentially confirm the conjecture from [20] that leveled implementations of schemes like Polka can rapidly lead to better performances than a uniformly masked implementation of Kyber. It turns out this is already true for $d = 2$ shares. It also raises interesting open problems that we now detail.

First, both the uniformly masked implementations of Kyber in [6,7] and the leveled implementation of Polka in this paper focus on masking the necessary operations. While this is an important first step, it remains that SPA attack paths need to be prevented (at the share level for Kyber, both at the share level and after Step 2 for Polka). Such SPA protections are expected to be cheap. We nevertheless list the operations that should require special care.

Starting with KYBER, it is for example well-known that single-trace (SPA) attacks against the (share by share implementation of the) NTT may be a threat [29, 30]. If successful, such attacks indeed cancel the impact of masking. A natural option to prevent this issue is to rely on a hardware coprocessor. Alternatively, shuffling can be used as a surrogate to emulate parallelism [37].

A similar issue pops up in POLKA. In particular, it is important that the inverse NTT operations in Step 2 of Figure 2 are secure against SPA (which, again, is calling for parallelism or shuffling). Besides, the security of the LWPR assumption that we leverage in order to unmask the computations after this Step 2 also requires that the operations in Step 3 do not leak "too much" about t [16,21]. Once more, parallelism and shuffling appear as natural options. Yet, it is also worth noticing that POLKA embeds design features to minimize this leakage. Namely, the LWPR assumption was so far studied for a public vector multiplied with a secret key (as it is usually the case for hard learning problems). But the leveled implementation of Figure 2 multiplies a secret (randomized) vector, of which the adversary only sees the leakage with a shared key. Hence, studying how much this hardens the problem is an interesting open problem. Second, Step 3 in the same figure starts with a modular reduction which turns 16-bit values into 3-bit values, substantially reducing the leakage on t that subsequent operations can provide. Finally, LWPR was so far studied in a conservative setting where leakages are assumed to be noise-free. So it is another interesting open problem to find out whether there is a gap to exploit between the LWPR assumption and a more realistic "learning with physical rounding and noise" assumption.

Next, the "security vs. performance" tradeoff that we study is so far qualitative. Again, this appears to be a necessary first step and, for example, a similar tradeoff was also first studied for masked implementations of KYBER. But it also suggests investigating how this qualitative analysis can be translated into a quantitative one as an important extension. This will likely be a non-trivial task, since it implies understanding in depth how to assess the worst-case (quantitative) security guarantees of KYBER and POLKA, which both have multiple (SPA and DPA) attack paths. There are nevertheless reasons to believe that the turn to a quantitative analysis will further amplify the impact of POLKA. First, and as already mentioned, the absence of FO transform removes the possibility of a message distinguishing attack which is, in many cases, the most critical one for KYBER [2]. Concretely, it leads implementations with large number of shares to be breakable [15]. Second, it is already well documented that key-homomorphic primitives bring significant advantages in terms of dealing with physical defaults [8, 16]. So it is expected that translating the number of shares into a "statistical security order" will be easier for POLKA than KYBER: both in software, due to transitions [3, 12], and in hardware due to glitches [25, 28].

In order to stimulate research in these directions, we make our baseline implementation of POLKA available at the following address:

`https://github.com/uclcrypto/pqm4_polka`.

While such an implementation is not expected to directly satisfy the aforementioned security requirements, we believe it is relevant to assess the severity of different attack paths given the randomized nature of Polka's decryption.

As mentioned in the introduction, the leveled implementation of Polka in this paper provides security for its long-term secret key (i.e., leakage-resilience). While this is a practically-relevant guarantee given that the most critical side-channel attacks against Kyber target its long-term key, finding out whether leakage-resistance could be obtained in a more efficient manner than by masking Step 5 in Figure 2 instantiated with standard hash functions and authenticated encryption schemes is a natural next step. The original description of [20] suggests that key-homomorphic primitives could be used for a part of this final step. Yet, for now, it does not remove the need to securely mask a cryptographic hash function, which would incur quadratic overheads. So it is yet another promising scope for further research efforts to improve Polka on this front.

Besides, the recent work of Hövelmanns et al. introduced an alternative framework to proving the security of post-quantum encryption schemes ensuring rigidity via norm checks [22]. It would be interesting to study whether it applies to Polka (or variations thereof), and whether it can lead to prove the security of its KEM component rather than encryption scheme as a whole.

Eventually, the physical security guarantees of Polka remain heuristic so far. So a comprehensive proof of leakage-resilience or leakage-resistance, under weak and falsifiable physical assumptions is a challenging long-term goal.

Acknowledgments. Thibaud Schoenauen is a PhD student funded by the FSR project PQShield. Thomas Peters and François-Xavier Standaert are respectively senior research associate and research director of the Belgian Fund for Scientific Research (F.R.S.-FNRS). This work has been funded in part by the European Research Council (ERC) Advanced Grant BRIDGE (number 101096871) and by the Walloon Region through the project CyberExcellence (convention number 2110186). Views and opinions expressed are those of the authors and do not necessarily reflect those of the European Union or the ERC. Neither the European Union nor the granting authority can be held responsible for them.

References

1. Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Kyber: Algorithm specifications and supporting documentation. *NIST Post-Quantum Cryptography Standard*, 2022.
2. Melissa Azouaoui, Olivier Bronchain, Clément Hoffmann, Yulia Kuzovkova, Tobias Schneider, and François-Xavier Standaert. Systematic study of decryption and re-encryption leakage: The case of kyber. In *COSADE*, volume 13211 of *Lecture Notes in Computer Science*, pages 236–256. Springer, 2022.

3. Josep Balasch, Benedikt Gierlichs, Vincent Grosso, Oscar Reparaz, and François-Xavier Standaert. On the cost of lazy engineering for masked software implementations. In *CARDIS*, volume 8968 of *Lecture Notes in Computer Science*, pages 64–81. Springer, 2014.
4. Paul Barrett. Implementing the rivest shamir and adleman public key encryption algorithm on a standard digital signal processor. In *CRYPTO*, volume 263 of *Lecture Notes in Computer Science*, pages 311–323. Springer, 1986.
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 *CRYPTO (1)*, volume 12170 of *Lecture Notes in Computer Science*, pages 369–400. Springer, 2020.
6. 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.
7. Olivier Bronchain and Gaëtan Cassiers. Bitslicing arithmetic/boolean masking conversions for fun and profit with application to lattice-based kems. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(4):553–588, 2022.
8. Olivier Bronchain, Tobias Schneider, and François-Xavier Standaert. Reducing risks through simplicity: high side-channel security for lazy engineers. *J. Cryptogr. Eng.*, 11(1):39–55, 2021.
9. Anne Canteaut, Sébastien Duval, Gaëtan Leurent, María Naya-Plasencia, Léo Perrin, Thomas Pornin, and André Schrottenloher. Saturnin: a suite of lightweight symmetric algorithms for post-quantum security. *IACR Trans. Symmetric Cryptol.*, 2020(S1):160–207, 2020.
10. James Cooley and John Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301, 1965.
11. Jean-Sébastien Coron, François Gérard, Simon Montoya, and Rina Zeitoun. High-order polynomial comparison and masking lattice-based encryption. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(1):153–192, 2023.
12. Jean-Sébastien Coron, Christophe Giraud, Emmanuel Prouff, Soline Renner, Matthieu Rivain, and Praveen Kumar Vadnala. Conversion of security proofs from one leakage model to another: A new issue. In *COSADE*, volume 7275 of *Lecture Notes in Computer Science*, pages 69–81. Springer, 2012.
13. Jan-Pieter D'Anvers. One-hot conversion: Towards faster table-based A2B conversion. In *EUROCRYPT (4)*, volume 14007 of *Lecture Notes in Computer Science*, pages 628–657. Springer, 2023.
14. Jan-Pieter D'Anvers, Daniel Heinz, Peter Pessl, Michiel Van Beirendonck, and Ingrid Verbauwhede. Higher-order masked ciphertext comparison for lattice-based cryptography. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(2):115–139, 2022.
15. Elena Dubrova, Kalle Ngo, Joel Gärtner, and Ruize Wang. Breaking a fifth-order masked implementation of crystals-kyber by copy-paste. In *APKCAsiaCCS*, pages 10–20. ACM, 2023.
16. Sébastien Duval, Pierrick Méaux, Charles Momin, and François-Xavier Standaert. Exploring crypto-physical dark matter and learning with physical rounding towards secure and efficient fresh re-keying. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(1):373–401, 2021.
17. 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.

18. W. Morven Gentleman and G. Sande. Fast fourier transforms: for fun and profit. In *AFIPS Fall Joint Computing Conference*, volume 29 of *AFIPS Conference Proceedings*, pages 563–578. AFIPS / ACM / Spartan Books, Washington D.C., 1966.
19. Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Authenticated encryption with nonce misuse and physical leakage: Definitions, separation results and first construction - (extended abstract). In *LATINCRYPT*, volume 11774 of *Lecture Notes in Computer Science*, pages 150–172. Springer, 2019.
20. Clément Hoffmann, Benoît Libert, Charles Momin, Thomas Peters, and François-Xavier Standaert. POLKA: towards leakage-resistant post-quantum cca-secure public key encryption. In *Public Key Cryptography (1)*, volume 13940 of *Lecture Notes in Computer Science*, pages 114–144. Springer, 2023.
21. Clément Hoffmann, Pierrick Méaux, Charles Momin, Yann Rotella, François-Xavier Standaert, and Balazs Udvarhelyi. Learning with physical rounding for linear and quadratic leakage functions. In *CRYPTO (3)*, volume 14083 of *Lecture Notes in Computer Science*, pages 410–439. Springer, 2023.
22. Kathrin Hövelmanns, Andreas Hülsing, Christian Majenz, and Fabrizio Sisinni. (un)breakable curses - re-encryption in the fujisaki-okamoto transform. In *EUROCRYPT (2)*, volume 15602 of *Lecture Notes in Computer Science*, pages 245–274. Springer, 2025.
23. Matthias J. Kannwischer, Richard Petri, Joost Rijneveld, Peter Schwabe, and Ko Stoffelen. PQM4: Post-quantum crypto library for the ARM Cortex-M4. `https://github.com/mupq/pqm4`.
24. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 2010.
25. Stefan Mangard, Norbert Pramstaller, and Elisabeth Oswald. Successfully attacking masked AES hardware implementations. In *CHES*, volume 3659 of *Lecture Notes in Computer Science*, pages 157–171. Springer, 2005.
26. Peter L Montgomery. Modular multiplication without trial division. *Mathematics of computation*, 44(170):519–521, 1985.
27. 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.
28. Svetla Nikova, Vincent Rijmen, and Martin Schläffer. Secure hardware implementation of nonlinear functions in the presence of glitches. *J. Cryptol.*, 24(2):292–321, 2011.
29. 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.
30. 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.
31. Gokulnath Rajendran, Prasanna Ravi, Jan-Pieter D'Anvers, Shivam Bhasin, and Anupam Chattopadhyay. Pushing the limits of generic side-channel attacks on lwe-based kems - parallel PC oracle attacks on kyber KEM and beyond. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(2):418–446, 2023.
32. 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.
33. Gregor Seiler. Faster AVX2 optimized NTT multiplication for ring-lwe lattice cryptography. *IACR Cryptol. ePrint Arch.*, page 39, 2018.

34. 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.
35. Yutaro Tanaka, Rei Ueno, Keita Xagawa, Akira Ito, Junko Takahashi, and Naofumi Homma. Multiple-valued plaintext-checking side-channel attacks on post-quantum kems. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(3):473–503, 2023.
36. 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 Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(1):296–322, 2022.
37. Nicolas Veyrat-Charvillon, Marcel Medwed, Stéphanie Kerckhof, and François-Xavier Standaert. Shuffling against side-channel attacks: A comprehensive study with cautionary note. In *ASIACRYPT*, volume 7658 of *Lecture Notes in Computer Science*, pages 740–757. Springer, 2012.
38. Zhuang Xu, Owen Pemberton, Sujoy Sinha Roy, David F. Oswald, Wang Yao, and Zhiming Zheng. Magnifying side-channel leakage of lattice-based cryptosystems with chosen ciphertexts: The case study of kyber. *IEEE Trans. Computers*, 71(9):2163–2176, 2022.