IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2022, No. 4, pp. 1–24. DOI:10.46586/tches.v2022.i4.1-24

When Bad News Become Good News

Towards Usable Instances of Learning with Physical Errors

Davide Bellizia[1], Clément Hoffmann[1], Dina Kamel[1], Pierrick Méaux[1] and François-Xavier Standaert[1]

[1] UCLouvain, ICTEAM, Crypto Group, Louvain-la-Neuve, Belgium
firstname.lastname@uclouvain.be
[2] Luxembourg University, SnT, Luxembourg
pierrick.meaux@uni.lu

Abstract. Hard physical learning problems have been introduced as an alternative option to implement cryptosystems based on hard learning problems. Their high-level idea is to use inexact computing to generate erroneous computations directly, rather than to first compute correctly and add errors afterwards. Previous works focused on the applicability of this idea to the Learning Parity with Noise (LPN) problem as a first step, and formalized it as Learning Parity with Physical Noise (LPPN). In this work, we generalize it to the Learning With Errors (LWE) problem, formalized as Learning With Physical Errors (LWPE). We first show that the direct application of the design ideas used for LPPN prototypes leads to a new source of (mathematical) data dependencies in the error distributions that can reduce the security of the underlying problem. We then show that design tweaks can be used to avoid this issue, making LWPE samples natively robust against such data dependencies. We additionally put forward that these ideas open a quite wide design space that could make hard physical learning problems relevant in various applications. And we conclude by presenting a first prototype FPGA design confirming our claims.

Keywords: Learning With Errors · Physical Assumptions · FPGA Implementations

1 Introduction

The Learning With Errors (LWE) problem is an important computational assumption which is the basis of many modern cryptographic constructions [Reg09]. It has in particular been used to design several Fully Homomorphic Encryption (FHE) schemes [BV11, BGV12, GSW13] and is the core of many Post-Quantum (PQ) algorithms such as Frodo [ABD+21a], New Hope [AAB+20], CRYSTALS-Kyber [ABD+21b] or CRYSTALS-Dilithium [DLL+21]. In a standard implementation of LWE-based cryptographic primitives, one typically generates errors following a given distribution explicitly, which are then added to some correct computations. Yet, it has been shown that the cost of generating the random errors with a TRNG or with cryptographically strong PRNG can become a performance bottleneck, especially if side-channel attacks are a concern [SPOG19, BGR+21].

In parallel, a line of works initiated by Kamel et al. has shown that leveraging inexact computing in order to directly compute erroneous computations following the appropriate distribution can lead to conceptual advantages for the implementation of primitives based on the Learning Parity with Noise (LPN) problem [KSD+20]. The LPN problem is a special case of the LWE problem where computations are performed in GF(2). Its physical variant has been formalized as the Learning Parity with Physical Noise (LPPN) problem. It has been exhibited that it is possible to control the error distribution of LPPN prototypes with a sufficient accuracy [KBS+18] and that the resulting (key-homomorphic) implementations have interesting features for side-channel security via masking [KBBS20].

Received: 2022-04-15 Accepted: 2022-06-15 Published: 2022-08-31

More recently, Bellizia et al. investigated the (lack of) gap between the (mathematical) LPN problem and the (physical) LPPN problem [BHK+21]. They first showed that LPPN implementations can suffer from output data dependencies, where the probability of the errors depends on the correct computations (while they are independent in LPN). They next showed that these dependencies can be reduced thanks to implementation tweaks, despite not being able to make them negligible. As a result, they finally gave a reduction from the standard LPN problem to a (new) LPN-OD problem with such output dependencies, allowing one to quantity the limited security loss that such dependencies imply.

In terms of applicability, the main open question raised by these different previous works is whether generalized hard physical learning problems could be instantiated for use in actual applications. Our contributions in this respect are threefold.

First, we define such a generalization, next coined Learning With Physical Errors (LWPE), and discuss the new implementation challenges it raises. In particular, we show that directly applying the techniques used in the LPPN context leads to a new type of (mathematical) dependencies in the error distributions (hence the bad news part of our title). In short, these dependencies are due to the fact that the errors introduced affect bits rather than field elements. Yet, we also show that it is possible to get rid of them by inserting the errors early in the inner product computations of a LWPE prototype.

Second, we discuss the LWPE design space and the extent to which LWPE could be used for different lattice structures, multiplication algorithms, modulus, and error distributions. We conclude that hard physical learning problems formalized by LWPE have a broad applicability while also being more challenging in certain specific contexts.

Eventually, we describe a first FPGA prototype of LWPE instance. It allows us to confirm that it is possible to perform erroneous computations that follow practically-relevant distributions (e.g., for PQ cryptography). It also leads us to observe that the design tweak used to mitigate mathematical error dependencies has for positive effect to mitigate physical error dependencies as well (hence the good news part of our title).[1]

Cautionary note. Leveraging the LWPE assumption is only possible for PQ schemes that do not require deterministic errors in their decryption which, for example, is not the case of Key Encapsulation Mechanisms (KEMs) based on the Fujisaki-Okamoto (FO) transform [FO99]. Yet, it is applicable in any context where LWE is used without such an additional requirement (e.g., IND-CPA public key encryption or signature schemes that do not need to be de-randomized). Besides, given that the FO transform is the root of important weaknesses against side-channel attacks [UXT+22] while LWPE actually gains interest when side-channel attacks are a concern, our results give further incentives for designing new PQ cryptographic schemes that do not rely on such transforms.

Related work. In a recent independent work, Kundi et al. leverage a different flavor of approximate computing to implement a Ring-LWE Co-Processor [KKB+22]. Their approximate computations are based on the truncation of the multiplicative operands. Hence, errors are still explicitly generated which cancels the advantages of LWPE for masking.

2 Background

In this section, we provide the necessary background on the hard learning problems needed in this work, discuss some constructions taking advantage of these problems and introduce statistical tests that we will need for our experimental investigations.

[1] While the focus of this paper is on output data dependencies, it has been shown in the literature that input data dependencies can also be observed. As discussed in [KSD+20], Section 7.1 and [KBS+18], Section 4.1, input dependencies are computationally harder to exploit. This is in contrast with output dependencies that more directly affect the LPPN or LWPE security, which explains our focus. Yet, better analyzing and formalizing security against input data dependencies is an interesting open problem.

2.1 The LWE problem

The standard "algebraically unstructured" LWE problem is specified as follows [Reg09]:

Definition 1 (LWE problem)**.** Let $n, q \in \mathbb{N}$, let $\langle ., . \rangle$ denote the inner product and χ be a distribution over $\mathbb{Z}_q$. For a secret $\boldsymbol{s} \in \mathbb{Z}_q^n$, the LWE distribution $L_{\boldsymbol{s},\chi}$ is defined as:

$$L_{\boldsymbol{s},\chi} =: \{(\boldsymbol{a}, \langle \boldsymbol{a}, \boldsymbol{s} \rangle + e) : \boldsymbol{a} \overset{\$}{\leftarrow} \mathbb{Z}_q^n; e \overset{\$}{\leftarrow} \chi\}.$$

Let $\mathcal{O}_{\boldsymbol{s},\chi}$ denote an oracle outputting independent samples according to the distribution $L_{\boldsymbol{s},\chi}$. The LWE$_{n,\chi}$ (search) problem is said to be (c, t, m, θ)-hard to solve if for any algorithm A,

$$\Pr[\boldsymbol{s} \overset{\$}{\leftarrow} \mathbb{Z}_q^n \ : A^{\mathcal{O}_{\boldsymbol{s},\chi}}(1^n) = \boldsymbol{s}] \leq \theta,$$

and A runs in time $< t$, with memory $< m$ and makes at most c queries to $\mathcal{O}_{\boldsymbol{s},\chi}$.

It is worth noting that the LPN problem is a special case of the LWE problem where the modulus q is equal to two 2 and therefore χ is a Bernouilli distribution.

2.2 The MLWE problem

The Module LWE problem generalizes both the LWE and Ring LWE [LPR12] problems.

Definition 2 (MLWE problem [BGV12, LS12])**.** Let $R_q = \mathbb{Z}_q[x]/f(x)$ be the ring of integer polynomials modulo both $f(x)$ and q, where $f(x) = x^d + 1 \in \mathbb{Z}[x]$, the dimension of the ring d is a power of 2 and $q \in \mathbb{N}$. Let χ be a distribution over R_q.For a secret $\boldsymbol{s} \in R_q^n$, where n is called the rank of the module the MLWE distribution $M_{\boldsymbol{s},\chi}$ is defined as:

$$M_{\boldsymbol{s},\chi} =: \left\{ \left(\boldsymbol{a}, \sum_{i=1}^{n} \boldsymbol{a}_i \cdot \boldsymbol{s}_i + e \right) : \boldsymbol{a} \overset{\$}{\leftarrow} R_q^n; e \overset{\$}{\leftarrow} \chi \right\},$$

where $\cdot$ denotes the multiplication in R_q. Let $\mathcal{O}_{\boldsymbol{s},\chi}$ denotes an oracle outputting independent samples according to the distribution $M_{\boldsymbol{s},\chi}$. The MLWE$_{d,n,\chi}$ problem is said to be (c, t, m, θ)-hard to solve if for any algorithm A,

$$\Pr[\boldsymbol{s} \leftarrow U(R_q^n) \ : A^{\mathcal{O}_{\boldsymbol{s},\chi}}(1^{nd}) = \boldsymbol{s}] \leq \theta,$$

and A runs in time $< t$, with memory $< m$ and makes at most c queries to $\mathcal{O}_{\boldsymbol{s},\chi}$.

Note that LWE is the particular case of MLWE where $d = 1$ and the ring version of LWE (coined RLWE) is the particular case of MLWE where $n = 1$.

2.3 The LWPE problem

Kamel et al. introduced LPPN as a variant of the LPN problem where a physical function directly computes incorrect inner products. Analogously, we introduce the LWPE problem. It requires the definition of physical function adapted from [AMS+11] recalled next.

Definition 3 (Physical function)**.** A physical function $\mathsf{PF}_{pd,\alpha}$ is a probabilistic procedure based on a physical device pd, which can be stimulated with an input $\boldsymbol{x} \in \mathbb{Z}_q^{n_i}$, making pd respond with a (probabilistic) output $\boldsymbol{y} \in \mathbb{Z}_q^{n_o}$, with α a set of parameters.

In the LWPE case, pd will be an implementation of an inexact inner product computations modulo q (implying $n_i = n$ represented in $\lceil log_2(q) \rceil$ bits and $n_o = 1$ represented in $\lceil log_2(q) \rceil$ bits) and α is the set of parameters determining its error distribution.

Definition 4 (Physical inner product)**.** Let $\boldsymbol{s} \in \mathbb{Z}_q^n$ be a uniform random secret stored in a keyed device $pd_{\boldsymbol{s}}$. A physical function $\mathsf{PF}_{pd_{\boldsymbol{s}},\alpha}$ is called a χ-Physical Inner Product (χ-PIP) if, on uniform public input $\boldsymbol{a} \in \mathbb{Z}_q^n$, it outputs $\langle \boldsymbol{a}, \boldsymbol{s} \rangle$ with an additional error following a distribution χ such that:

$$\forall i \in \mathbb{Z}_q, \hat{\Pr}[\mathsf{PF}_{pd_{\boldsymbol{s}},\alpha}(\boldsymbol{a}) - \langle \boldsymbol{a}, \boldsymbol{s} \rangle = i] = \Pr[X = i] \text{ where } X \xleftarrow{\$} \chi.$$

The LWPE problem can then be specified as follows:

Definition 5 (LWPE problem)**.** Let $\mathsf{PF}_{pd_{\boldsymbol{s}},\alpha}$ be a χ-PIP. The $\mathrm{LWPE}^{n,\chi}_{pd_{\boldsymbol{s}},\alpha}$ problem is said to be (c, t, m, θ)-hard to solve if for any algorithm A running in time $< t$, memory $< m$ and making at most c uniformly random queries to a χ-PIP, it holds that:

$$\Pr[\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n \ : A^{\mathsf{PF}_{pd_{\boldsymbol{s}},\alpha}}(1^n) = \boldsymbol{s}] \leq \theta.$$

2.4 Noise distributions

The noise distribution χ of the LWE/MLWE error/secret are sampled either from a discrete Gaussian distribution DGD_σ or from a centered binomial distribution CBD_η.

Discrete Gaussian distribution. The discrete Gaussian distribution DGD_σ parametrized by the Gaussian parameter $\sigma \in \mathbb{R}^+$ is defined by assigning a weight proportional to $exp(\frac{-x^2}{2\sigma^2})$ to all integers x. Note that it can be challenging to implement a discrete Gaussian sampler efficiently and protected against timing attacks [BCNS15, ADPS16].

Centered binomial distribution The binomial distribution with parameters n and p is the discrete probability distribution of the number of successes in a sequence of n independent experiments with individual success of probability p. Its probability mass function is defined as $\binom{n}{x} p^x (1-p)^{n-x}$ with mean $= np$ and variance $= np(1-p)$. The centered binomial distribution is obtained by subtracting the mean from this distribution. We next focus on a sub-family that has the particularity to give symmetric distributions. For η a non negative integer we denote CBD_η the centered binomial distribution with parameters $p = 1/2$ and $n = 2\eta$. CBD_η has mean 0 and variance equal to $\eta/2$, and its samples can be computed as $\Sigma_{i=0}^{\eta-1} a_i - b_i$ where $a_i, b_i \in \{0, 1\}$ are uniform independent bits. Therefore, sampling from the centered binomial distribution is easy and does not require high-precision computations or large tables like Gaussian sampling does.

2.5 Related constructions

We now provide a few illustrative use cases of LWE/MLWE/RLWE problems exploited in recent post-quantum cryptographic primitives submitted to NIST competition.

Frodo. The Frodo scheme [ABD+21a] is tightly related to the hardness of a corresponding LWE problem. Its security depends on the lattice dimension n. Frodo-n with n equal to 640, 976, and 1344 targets Level 1, 3 and 5 as specified in the NIST requirements. The main operations are simple matrix-vector products. The modulus q is always a power of two. This ensures that modular arithmetic operations do not require modulo reduction schemes such as Montgomery or Barrett. Instead, the reduction modulo q is computed by low-cost bit-masking or even at no cost if the data type of the target platform supports the number of bits required by the modulus. For level 1 security $q = 2^{15}$ and for level 3 and 5 security $q = 2^{16}$. Frodo requires to sample errors from an approximated discrete Gaussian distribution DGD_σ via lookup tables, where σ is the standard deviation (equal to 2.8, 2.3 and 1.4 for Level 1, Level 3 and Level 5). Frodo relies on the standard "algebraically unstructured" LWE problem and is therefore simple to implement. However, this simplicity comes at the cost of reduced efficiency (i.e., a high running time and large bandwidth).

NewHope. The NewHope cryptosystem [AAB+20] is based on the hardness of the RLWE problem, both in the Random Oracle Model (ROM) and the Quantum Random Oracle Model (QROM). Its security depends on the polynomial ring dimension d which has to be a power of two to maintain the security properties of RLWE and to support the efficient implementation of Number Theoretic Transform (NTT) algorithms. For bit security levels of 101 and 233, d is set to 512 and 1024, respectively. The modulus q is fixed to $12,289$ as it is the smallest prime for which it holds that $q \equiv 1 \mod 2d$ (so that the NTT can be performed efficiently). The noise in NewHope is sampled from a CBD_η with parameter $\eta = 8$, avoiding the aforementioned challenge to implement a discrete Gaussian sampler efficiently and protecting against timing attacks. In contrast to Frodo, NewHope can be very efficient, both in terms of speed and in terms of key and ciphertext sizes. This is because it relies on a highly structured ideal lattice. However, exploiting this algebraic structure requires implementing fast polynomial multiplication algorithms, which also results in more complicated implementations than those of Frodo.

CRYSTALS-Kyber. The security of CRYSTALS-Kyber [ABD+21b] is based on the hardness of solving the MLWE problem. In all Kyber variants, the polynomial ring dimension is fixed to $d = 256$ and the security depends on the lattice dimension n. Kyber-$(d \times n)$ with n equal to 2, 3 and 4 targets Level 1, 3 and 5. The modulus $q = 3,329$ is a prime satisfying $q \equiv 1 \mod d$.[2] Similarly to the NewHope cryptosystem, the Kyber scheme uses the centered binomial distribution CBD_η as the noise distribution, with parameter η varying between 1 and 2 depending on the target security level. Since Kyber relies on the MLWE problem, it has a reduced structure compared to (RLWE-based) NewHope. However, it maintains similar performances (in the specific case of the MLWE parameters used in Kyber). Also, it offers better scalability because its security can be adapted by increasing the module size while keeping the ring dimension constant. As for NewHope, the need to implement fast polynomial multiplication algorithms to exploit the algebraic structure results in more complicated implementations than those of Frodo.

CRYSTALS-Dilithium. CRYSTALS-Dilithium [DKL+18] is a PQ signature scheme based on the Fiat-Shamir with aborts approach [Lyu09]. As CRYSTALS-Kyber [ABD+21b], its security relies on the hardness of solving the MLWE problem. The signature scheme's implementation consists of symmetric primitives (SHAKE) and operations over a polynomial, of which the dimension is given by the modulus $q = 2^{23} - 2^{13} + 1$ and the degree $n = 256$. CRYSTALS-Dilithium is a 3rd-round NIST candidate. Its strengths are the small size of its signatures and public keys, and its simplicity of implementation. CRYSTAL-Dilithium uses a uniform noise in its instances of the MLWE problem in order to avoid the difficult implementation of Gaussian sampling, which leads to side-channel attacks if performed incorrectly [BHLY16, EFGT17, PBY17]. However, using a physical rounded-Gaussian sampler based on LWPE could be considered as an interesting variant.

2.6 Statistical tests

In this paper, we will need statistical tests to detect potential data dependencies in the LWPE samples distributions due to physical imperfections. In general, such tests use probability theory to reject (or not) a hypothesis, commonly called as the null hypothesis, thanks to a set of samples. Therefore, statistical tests can produce two types of errors:

- Type I errors (or false positives, or alpha risk). These errors capture the risk of rejecting the null hypothesis when it is true. Most statistical tests (and the ones we will use) quantify the likelihood of such errors and name it p-value.

[2] The use of NTT algorithms requires that $q \equiv 1 \, mod \, 2d$. However, to reduce the bandwidth (by decreasing the modulus), version 2.0 of Kyber preprocesses the polynomial before applying the NTT so that a polynomial with degree d is divided into two low-dimensional polynomials of degree $d/2$.

- Type II errors (or false negatives, or beta risk). These errors capture the risk of not rejecting the null hypothesis when it is false. They cannot be quantified in general. However, the bigger the set of samples used, the lower this error becomes.

Concretely, we used the Kruskal-Wallis test (or one-way ANOVA on ranks), which is a nonparametric test used for testing if independent samples originate from the same distribution [KW52]. It works under the null hypothesis that the empirical distributions are identical, which should be the case if there are no data-dependent errors.

3 Introducing LWPE with a design example

We now introduce the main design ideas needed to instantiate practically-relevant LWPE instances. For this purpose, and for the sake of simplicity, we will first focus on a small LWE modulus ($q = 2^8$) and a small lattice dimension $n = 128$. Based on these parameters, we will describe how to implement an LWPE processor whose errors' distributions approximate a centered binomial distribution (CBD_η) with parameter $\eta \in \{2, 3\}$. We investigate two approaches for this purpose, and provide rationale about the resulting error distribution and its security. In **Design A**, we directly extend the LPPN prototype in [KBS+18] and sample the two Least Significant Bits (LSBs) of the LWPE processor's output incorrectly, so that the combined errors approximately follow a CBD_3. Yet, we show that such an approach is not successful as it leads to (mathematical) data dependencies in the error distributions. Instead of sampling at the final stage just before outputting the result, **Design B** then generates the errors by sampling the LSB of η bytes at a designated internal stage. Interestingly, we will show that this approach cancels the previously mentioned mathematical data dependencies. Both designs are implemented in a 65-nm technology using a fully digital design approach. Pre-layout simulations are performed using ULTRASIM within the CADENCE environment. We used 2^{16} uniformly distributed random 128-byte inputs $\boldsymbol{a}$ and a single uniformly distributed random 128-byte secret key $\boldsymbol{s}$. The supply voltage is fixed at nominal 1.2V and the clock frequency is 500MHz.

Note that, as will be detailed in Section 3.3, the shape of the noise distribution is not a critical security requirement. So the selection of a CBD in our next designs is just used as an illustrative example that can be found in PQ cryptographic schemes.

3.1 Design A

We first present the architecture of the LWPE_A processor and demonstrate the feasibility of generating samples following an approximated CBD_η ($\eta = 3$) distribution.

3.1.1 LWPE_A implementation

Our first approach to implement a χ-PIP is illustrated in Figure 1. The inner product block takes 128-byte vectors of the secret $\boldsymbol{s}$ and the challenge $\boldsymbol{a}$ as inputs, and outputs a single byte y. Two specific sampling clocks *clkD0, clkD1* are generated by two variable delay lines (VDL0 and VDL1) such that the error probability of the sampled output bits provides a good approximation of the target error distribution.

The inner product block comprises 128 parallel 8-bit multipliers used to process the byte vectors $\boldsymbol{a}$ and $\boldsymbol{s}$. The sum of these products is implemented with a seven-stage tree of multiple-size parallel adders. Two bits (b_0 and b_1) of the inner product block output byte y are sampled by two flip-flops clocked thanks to delayed versions of the system clock, namely $clkD0$ and $clkD1$. These flip-flops are responsible for sampling the designated output bits during their glitchy period before they stabilize. The output of these two flip-flops along with the remaining bits of the inner product output y ($\langle b_7 : b_2 \rangle$) are sampled with the system clock clk to synchronize and produce the LWPE_A output y_{out}.

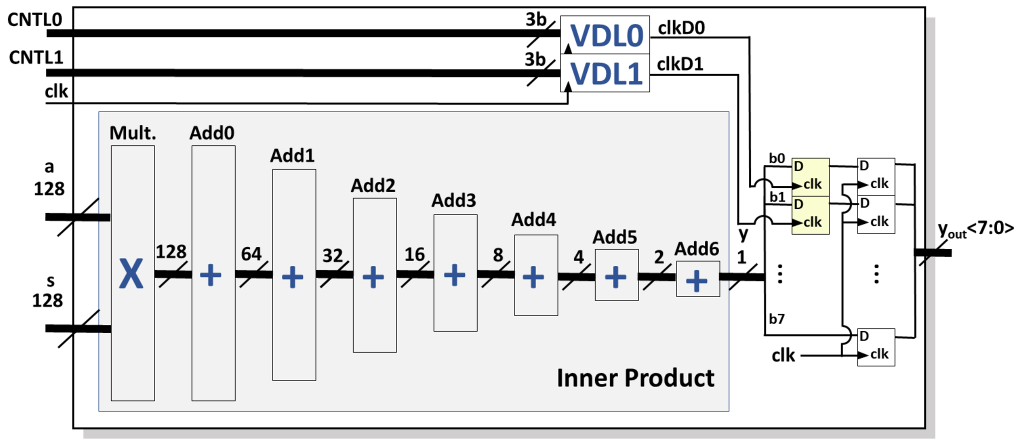

Figure 1: Architecture of the $\mathrm{LWPE_A}$ processor.

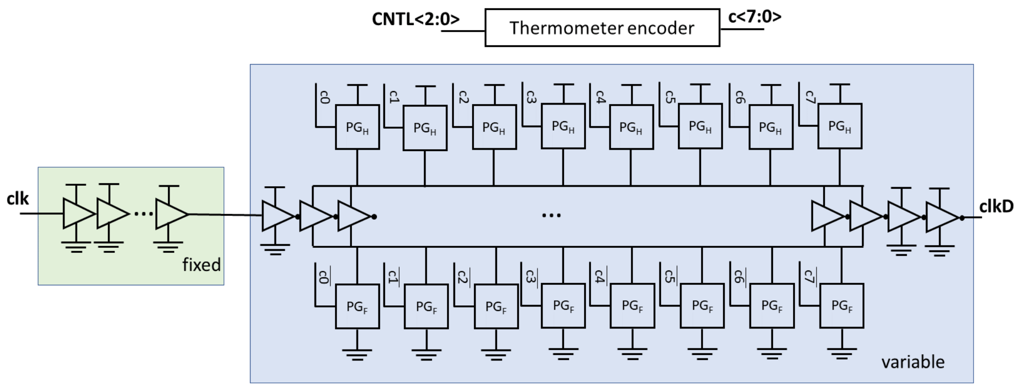

Figure 2: Block diagram of the Variable Delay Line (VDL).

To ensure that the generated errors follow a CBD_η distribution, the error probability on each bit needs to be bounded. As in LPPN prototypes, the key aspect to guarantee such bounds is to design a variable delay line and a control circuitry per bit, such that they can set the delay of the sampling clocks $clkD0$ and $clkD1$ to capture the non-stable state of the designated bits. Concretely, this typically works in two phases: calibration and generation [KBS+18]. Since the calibration we use is identical to the LPPN case, we do not describe it nor implement it for simplicity. Instead, the control signals $CNTL0$ and $CNTL1$ (which are 3-bits each) are entered as inputs to the $\mathrm{LWPE_A}$ module. Each variable delay line consists of two parts: a fixed delay to compensate for the computation of the inner product until the final adder stage, and a variable delay to sample incorrectly the relevant output bit. Figure 2 demonstrates the block diagram of the VDL. The fixed delay part uses multiple delay cells provided by the standard cell library of the foundry. The implementation of the variable delay part is based on the current starved inverter concept [MNS05]. It comprises a chain of inverters whose supply and drain currents are digitally controlled by a power gate header and footer cells (also provided by the standard cell library of the foundry). The number of stages in the variable delay block is adjusted such that both the tuning range and the delay steps provide reliable control over the required probabilities of error. Finally, an encoder is needed to convert the binary $CNTL\langle 2:0\rangle$ signals into its corresponding thermometer encoded $c\langle 7:0\rangle$ signal to control how much power gate cells are activated, and thus the delay.

In order for the error distribution of the proposed LWPE$_\mathrm{A}$ processor to approximate a CBD_η (with $\eta = 3$ in our example), we explored all the error probability combinations on b_0 and b_1 of the LWPE$_\mathrm{A}$ processor output, and computed the statistical distance ($\Delta = \Sigma_{j=-\eta}^{\eta} |CBD_\eta(j) - \chi(j)|$) between the CBD_3 distribution and the one of the LWPE$_\mathrm{A}$ error, with χ taken over multiple samples by computing the error $e = y - \langle \boldsymbol{a}.\boldsymbol{s} \rangle$. After mathematically searching all such possible combinations using a Python script, the minimum statistical distance (0.128) was achieved by setting the error probability on b_0 to 0.5 and on b_1 to 0.37. Figure 3 shows the resulting error distribution of the LWPE$_\mathrm{A}$ processor compared to the required CBD_3, highlighting a first limitation: the errors generated with the best fitted parameters do not exactly follow a CBD_3 distribution.[3]

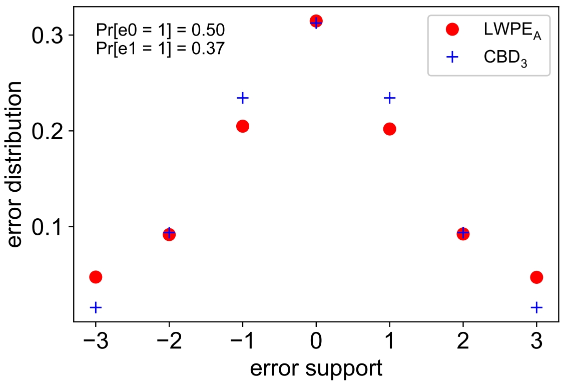

Figure 3: LWPE$_\mathrm{A}$: Error distribution approximating CBD_3.

3.1.2 Mathematical output data-dependencies of the errors

As previously mentioned, exactly approximating a CBD is not a critical security requirement for LWPE. By contrast, a more sensitive implementation challenge when instantiating hard physical learning problems is to limit the data dependencies of their error distribution. For example, it has already been put forward in [BHK+21] that the errors of an ϵ-PIP generating LPPN samples can exhibit significant output data dependencies due to physical imperfections. In this section, we highlight that the previous generalization of an LPPN design towards an LWPE one leads to another source of strong (mathematical) output data dependencies that neither LWE samples nor LPPN samples exhibit.

In summary, since the error is not added (in the sense that it is sampled from a distinct CBD_η distribution) as in a standard LWE sample $t = \langle \boldsymbol{a}.\boldsymbol{s} \rangle + e$, but generated by sampling incorrectly the last two LSBs of the LWPE$_\mathrm{A}$ output, this error does not propagate to higher-order bits. In fact, the incorrect sampling of the LSBs corresponds to an XOR operation between the output and a binary vector equivalent to 0, 1, 2, or 3 (since two bits of noise are considered in our example). As a result, the error range is not equal across different values of the correct LWPE output. This means that the error of an LWPE processor will depend on the correct value of the last two LSBs of the output. We plot in Figure 4 the error ranges as well as the output of both LWE and LWPE$_\mathrm{A}$ samples for our example where $q = 16$ and $\eta = 3$, where the mathematical dependencies are clear.

[3] Another limitation of this approach is the difficulty to approximate a CBD_η distribution with $\eta \neq 2^l - 1$. With $\eta = 3$, manipulating the last two LSBs of the LWPE output directly supports an error range $\{-3, \ldots, +3\}$. However, if $\eta = 2$, the required support range of the error distribution is $\{-2, \ldots, +2\}$ and to achieve this we need to reduce the probability of occurrence of ± 3 errors to a negligible level.

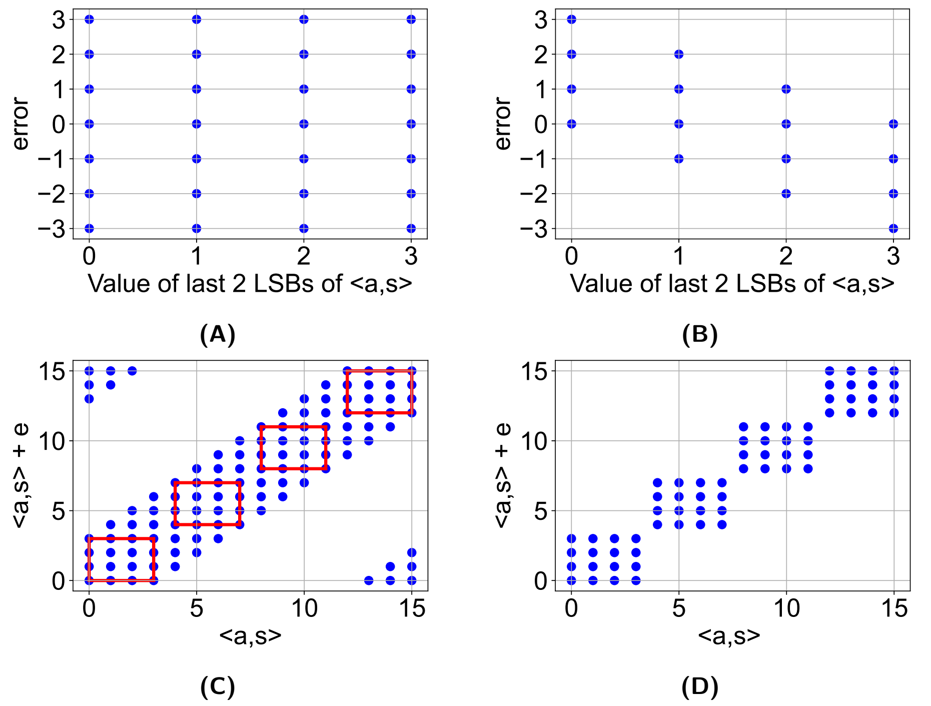

Figure 4: Errors of (a) LWE and (b) $\mathrm{LWPE_A}$ versus the correct value of the last two LSBs of $\langle a, s\rangle$, and incorrect output of (c) LWE and (d) $\mathrm{LWPE_A}$ versus the correct output for an example where $q = 16$ and $\eta = 3$. The LWPE output values obtained in (d) are superimposed on the ones of the LWE in (c) as red rectangles.

We note that the errors of the ϵ-PIP generating LPPN samples did not feature such mathematical output dependencies in [BHK+21] (only physical output dependencies were observed). The reason is twofold. First, these errors were generated by sampling a single bit (i.e., the output). Second, these errors (almost) followed a Bernoulli distribution that does not take into consideration whether the actual value of the error was $+1$ or -1.

We conclude that generalizing LPPN prototypes into LWPE ones to approximate practically-relevant noise distributions (e.g., for PQ applications) using this approach is not ideal. First, the error distribution of an $\mathrm{LWPE_A}$ instance does not approximate well the target (e.g., CBD_3) distribution. More critically, the approximated distributions exhibit mathematical dependencies that may affect the security of the assumption. We next discuss an alternative approach to LWPE prototypes addressing these limitations.

3.2 Design B

To mitigate the previous mathematical error dependencies, we next propose to generate the errors by sampling the LWPE processor incorrectly at an internal stage rather than at its output. We first discuss the implementation details of the resulting $\mathrm{LWPE_B}$ processor followed by simulation results. Then, we explain why this approach allows canceling the mathematical output data dependencies of the error distributions.

3.2.1 $\mathrm{LWPE_B}$ implementation

Figure 5 illustrates the proposed architecture of the $\mathrm{LWPE_B}$ processor. It is quite similar to the $\mathrm{LWPE_A}$ design where the inner product block takes 128-byte vectors of the secret $\boldsymbol{s}$

and the challenge $\boldsymbol{a}$, multiplies them (using 128 parallel 8-bit multipliers), and then adds them two by two in a seven stage adder tree to output a single byte y. However, there are two main differences between the two approaches. The first one is that the flip-flops responsible for the incorrect sampling are placed after an internal adder stage (Add3 in this case) instead of at the output of the $\mathrm{LWPE_A}$ processor. The second difference is that the errors are generated from sampling the LSB of η bytes ($\eta = 2$ in the current example of Figure 5) rather than multiple LSBs of a single byte (as in the $\mathrm{LWPE_A}$ processor). Indeed it is easier to ensure that the generated errors of the $\mathrm{LWPE_B}$ processor follow a CBD_η distribution by sampling only the LSB of multiple bytes (b_0 and not higher-order bits), in which case the error probability needs to be equal to 0.5 for all the erroneous bits.

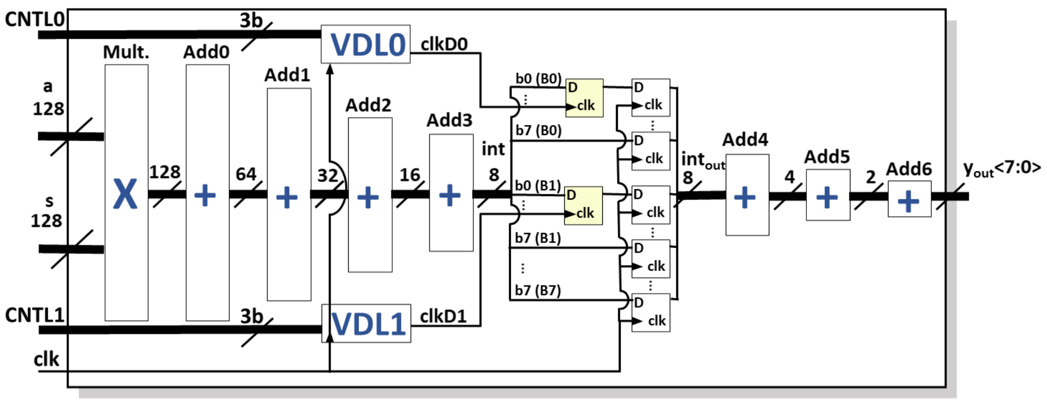

Figure 5: Architecture of the $\mathrm{LWPE_B}$ processor.

Next, the output of the incorrect sampling flip-flops along with the remaining bits of the concerned stage is sampled with the system clock clk to synchronize them before being passed to the subsequent adder stages to produce the final $\mathrm{LWPE_B}$ output y_{out}. The incorrect sampling flip-flops are clocked by delayed versions of the system clock. Similar to the $\mathrm{LWPE_A}$ processor, the current approach uses variable delay lines with control signals to adjust the delay of the sampling clocks. Figure 6 shows the error distribution of the $\mathrm{LWPE_B}$ processor compared to the required CBD_2 and CBD_3 distributions. Clearly, the current design of the $\mathrm{LWPE_B}$ processor leads to a much tighter fit of the output error distribution compared to that of the previously described $\mathrm{LWPE_A}$ processor.[4]

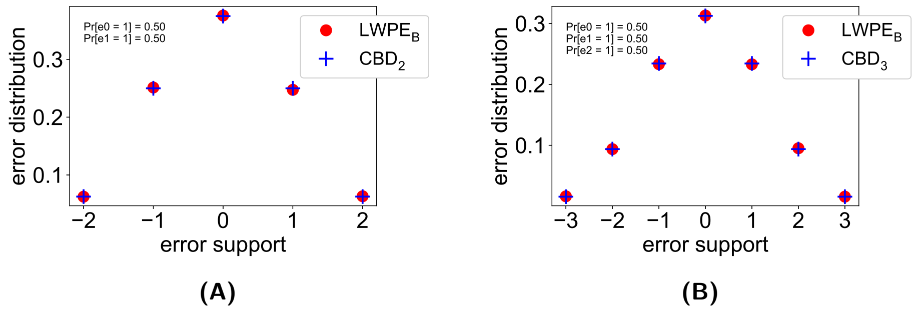

Figure 6: $\mathrm{LWPE_B}$: Error distribution approximating (a) CBD_2 and (b) CBD_3.

[4] The statistical distances with CBD_2 and CBD_3 are 3.5×10^{-3} and 5.1×10^{-3}, respectively.

We note that sampling incorrectly at the output of other internal stages is possible. Yet, opting for a later stage in the adder chain is in general a good approach since it limits the risk of exploitable input data dependencies mentioned in Footnote 1. Besides, it reduces the need to add registers (and therefore decreases the area and power consumption).

3.2.2 Mathematical output data-dependencies of the errors

Sampling incorrectly at an internal stage of the LWPE_B processor causes the errors in the sampled output of this specific stage to encounter the same mathematical data dependencies as previously explained in section 3.1.1. However, since this is not the final output and the remaining bytes of the same and subsequent stages are yet to be added, the mathematical data dependencies are expected to be reduced thanks to diffusion and eventually canceled out, which we illustrate with the next example. Let $n = 128$, $q = 256$ and $\eta = 2$. Let also the error be generated by sampling incorrectly the output of the fourth adder (Add3) at b_0 of bytes B_0 and B_1. We show in Figure 7 three different scenarios (marked by distinct colored rectangles) in case the correct output is equal to 125 as highlighted in yellow.

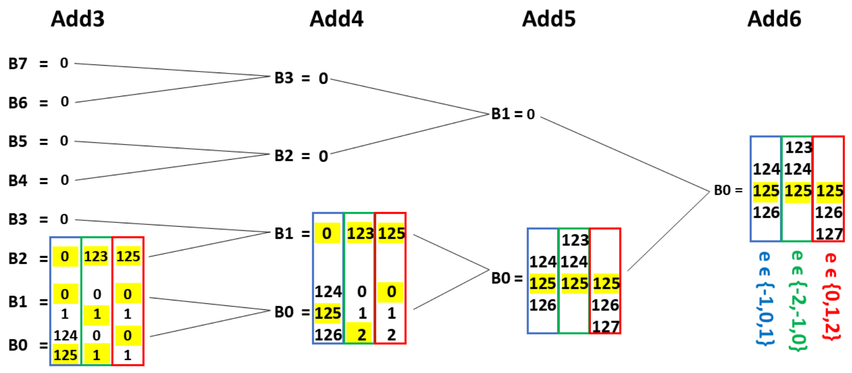

Figure 7: Illustration of the reduced output data dependencies in LWPE_B, for an exemplary case where $\eta = 2$ and the errors are introduced at the Add3 stage of the processor.

The error at bytes B_0 and B_1 of Add3 $\in$ either to $\{-1, 0\}$ or to $\{0, 1\}$, depending on the value of their LSB being either 1 or 0, respectively. Therefore, if $B_2 = 0$, the correct values of $B_1 = 0$ and the one of $B_0 = 125$, then the error of the final output $\in \{-1, 0, 1\}$ (marked in blue). Certainly, multiple combinations of byte values at Add3 will lead to the same output (125 in this example). We will consider two cases such that the error at the final output indeed $\in \{-2, \ldots, 2\}$. If $B_2 = 123$, the correct values of $B_1 = 1$ and the one of $B_0 = 1$, then the error of the final output $\in \{-2, 0, 1\}$ (marked in green). Finally, if $B_2 = 125$, the correct values of $B_1 = 0$ and the one of $B_0 = 0$, then the error of the final output $\in \{0, 1, 2\}$ (marked in red). Hence, the error at the final output can take values from the full error support of a CBD_2 distribution (despite it does not for the intermediate stages) and these scenarios may occur with other combinations of byte values. Simply stated, the adder tree is such that in the final stage, an erroneous byte X is added with an independent and uniformly distributed byte Y to produce the output. So even if the error distribution is dependent on X, it is not dependent on the output $X + Y$. As a result, without side-channel leakage on Y (or other physical defaults that we discuss in Section 5.3), there are no exploitable data-dependencies in the LWPE_B samples.

We complement this example with the results of Figure 8, where the outputs of the LWPE_B processor targeting a CBD_2 are plotted versus the correct value of $(\langle \boldsymbol{a}, \boldsymbol{s} \rangle)$. As can be seen from the inset plots, the outputs extend across the full error support range of the CBD_2 for all correct values, indicating no mathematical output dependencies.

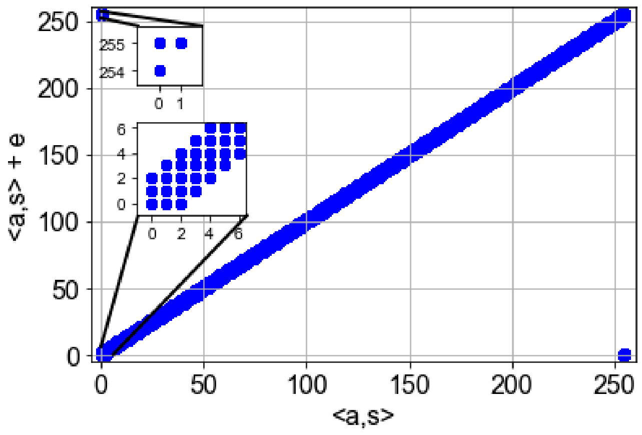

Figure 8: $\mathrm{LWPE_B}$: incorrect outputs versus correct outputs in case $\eta = 2$.

3.3 Security requirements

Despite a physical inner product may not produce an error that exactly follows a CBD (or any other target distribution), the assumption that "the performance of the best-known attacks against LWE-based encryption does not depend on the exact distribution of noise, but rather on its standard deviation" is commonly admitted [ABD+21b]. More precisely, the property that is required for LWE to be hard is that the error distribution is B-bounded [BGV12]. It is assumed that a B-bounded error distribution provides security in both the QROM [Reg09] and the ROM [Pei09] for rounded Gaussians. The use of other B-bounded distributions, with sufficient standard deviation, does not seem to lead to better attacks [BGV12, DT14]. In the $\mathrm{LWPE_B}$ case, the CBD_η distribution is approximated by flipping η LSBs, which guarantees the resulting distribution is B-bounded [BGV12]. The empirical standard deviation of the error distribution will be given with the prototype hardware implementation of LWPE in Section 5 in order to assess its security level.

4 LWPE design Space

The previous section put forward the feasibility to design a χ-PIP where χ is a CBD_η distribution, in order to generate useful LWPE samples. In this section, we further expand our design space and explore the possibility of applying inexact computing to other scenarios. Figure 9 summarizes the dimensions of this design space (marked in different color codes). Elements highlighted in green imply high-level compatibility with the proposed concept of inexact computing. We mark in yellow the parts of the design space that may be less efficiently implemented thanks to incorrect sampling. Finally, elements emphasized in orange indicate stronger implementation challenges still to be solved. We next discuss these different dimensions of the design space in more detail.

Lattice structures. LWE-based cryptosystems can rely on different variants: the standard LWE, the ring-LWE or the module-LWE problem. The standard (also called plain) LWE is a hard learning problem relying on algebraically unstructured lattices. Consequently, the main operations are simple matrix-vector products in ring $\mathbb{Z}_q$. On the other hand, algebraically structured lattices such as RLWE and MLWE use a polynomial ring $R_q =$

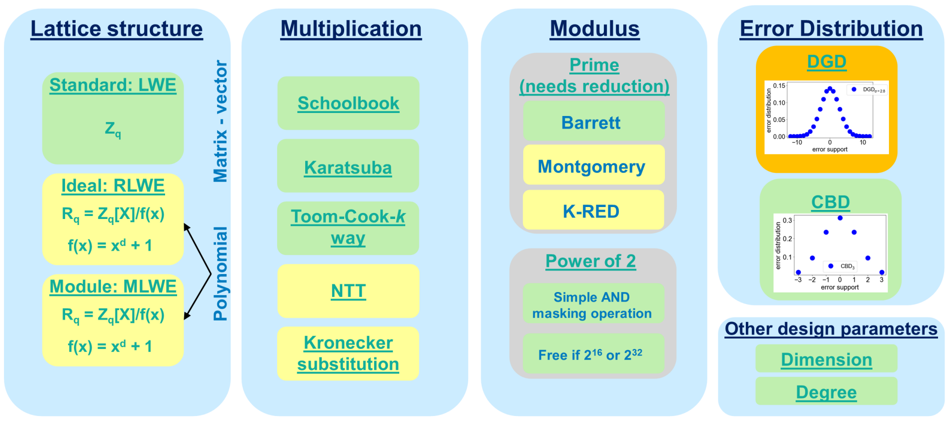

Figure 9: LWPE design space.

$\mathbb{Z}_q[x]/(x^d + 1)$, where d is a power of 2, instead of $\mathbb{Z}_q$. In theory, inexact computing can be used in both cases to generate erroneous samples. Yet, the efficient implementation of RLWE or MLWE schemes may require to exploit more advanced algorithms for the polynomial operations, especially the multiplication, as we discuss next.

Multiplication algorithms. Various algorithms can be used to perform the multiplication operations of LWE-based schemes. In case the lattice-based cryptosystem relies on the standard LWE problem, the main arithmetic operations are simple matrix-vector products. This enables the use of simple Schoolbook multiplication which directly supports the previously described use of inexact computing to generate errors.

On the other hand, for algebraically structured lattices, multiplications take place in R_q. This generally necessitates the use of fast polynomial multiplication algorithms such as the Number Theoretic Transform (NTT) if the cryptosystem parameters are NTT-friendly. In this case, the polynomial coefficients are first converted to the NTT domain before pointwise multiplications take place. The result is then converted out of the NTT domain (i.e., we compute $\mathrm{NTT}^{-1}(\mathrm{NTT}(\boldsymbol{a}) \times \mathrm{NTT}(\boldsymbol{s}))$). In cases where a polynomial matrix is multiplied by a polynomial vector (e.g., in Kyber), further addition operations are done before transforming the final result to polynomial domain (i.e., $\mathrm{NTT}^{-1}(\Sigma_{i=0}^{n-1} \mathrm{NTT}(\boldsymbol{a_i}) \times \mathrm{NTT}(\boldsymbol{s}))$).[5] Consequently, error generation via incorrect sampling is possible at one of the (multiple) stages of the multiplication. As previously, one can sample the final output after the inverse NTT operation. However, this approach leads to errors featuring mathematical output dependencies, because the incorrect sampling takes place at the LSB of the output and will not propagate to higher-order bits, as previously discussed in Section 3.1. It is also feasible to sample incorrectly at an intermediate stage, between two addition operations (i.e., $\mathrm{NTT}(noisy(\mathrm{NTT}^{-1}(\mathrm{NTT}(\boldsymbol{a_0}) \times \mathrm{NTT}(\boldsymbol{s})))) + \Sigma_{i=1}^{n-1} \mathrm{NTT}(\boldsymbol{a_i}) \times \mathrm{NTT}(\boldsymbol{s}))$. In that case, the mathematical output dependencies are expected to be diffused thanks to the subsequent addition operation(s), as previously explained in section 3.2.

Other popular multiplication algorithms include Karatsuba [KO63], Toom-Cook-k-way [Too63]and the Kronecker substitution. Unlike the NTT, these multiplication algorithms do not require any restrictions on the cryptosystem modulus: they apply to any prime or power of two modulus. For the rest, the Kronecker substitution leads to similar challenges as the NTT. They both use pointwise multiplication after evaluating

[5] In Kyber, the result of the polynomial multiplication is kept in NTT domain to further compute the public key and ciphertext. As a result, an extra NTT^{-1} operation is necessary to sample incorrectly in the polynomial domain as well as an extra NTT to go back to NTT domain if necessary.

the polynomial using predefined points. As a result, it does not directly support the use of inexact computing to generate errors and requires the multiplication result to be converted back to polynomial. Both the Karatsuba and Toom-Cook algorithms are based on a divide-and-conquer strategy. The Karatsuba multiplication splits each of the multiplicand polynomials into two degree-$d/2$ polynomials, whereas the Toom-Cook-k-way one splits them into k equal parts. Further splitting is necessary to reduce the number of multiplication steps. Both algorithms are compatible with our incorrect sampling techniques to generate errors since the coefficients are multiplied without any transformation.

We note that it is possible to combine algorithms as in [AHH$^+$19], where the authors used Kronecker substitution (with improved techniques [Har09]) together with either Karatsuba or Schoolbook multiplications for implementing Kyber. Another example in [MKV20], where the authors carry out the polynomial multiplications of Saber [DKRV18] (a public key encryption scheme, NIST Round 3 finalist, that relies on the hardness of the module learning with rounding problem), by combining Toom-Cook and Karatsuba with an evaluation step (similar to the Kronecker substitution) to perform pointwise multiplication. In yet another work, Karatsuba is used with NTT in order to reduce the number of multiplications needed [DMG21, XL21]. Whether generating errors with inexact computing is supported or not by these approaches largely depends on the individual algorithms of such combinations, which we leave as a scope for further investigations.

Modulus. Modular computations used in lattice-based cryptosystems either use a prime or a power of two modulus. Prime moduli are usually chosen to satisfy either $q \equiv 1 \; mod \; d$ (as in Kyber) or $q \equiv 1 \; mod \; 2d$ (as in NewHope), where d is the ring dimension in both cases. This is required to enable the fast NTT-based polynomial multiplication employed in several post-quantum cryptosystems. Accordingly, modular operations necessitate the use of reduction algorithms such as Barrett [Bar86], Montgomery [Mon85] or K-RED [LN16]. The Barrett algorithm optimizes the modular reduction by replacing the expensive divisions (used in classical modular reduction) with multiplications and bit shifts. Concretely, to compute $(a \times b) mod q$ we need to evaluate $(a \times b) - ((a \times b \times p) \gg l) \times q$, where $p = \lfloor \frac{2^l}{q} \rfloor$, $\gg$ is the right bit shift operation and l needs to be chosen to reduce the difference between $\frac{1}{q}$ and $\frac{p}{2^l}$. Such a reduction technique is compatible with the inexact computing concept without any extra cost. The Montgomery reduction rather avoids expensive division operations by transforming the input multiplicands to a "Montgomery form" (i.e., $\bar{a} = (a \times R) \; mod \; q$ and $\bar{b} = (b \times R) \; mod \; q$), which is basically the residue class of the multiplicands, using a constant $R > q$ that is coprime to q, and is always a power of two. Thus the only division necessary is the division by R which is a simple bit shift. The Montgomery reduction is then applied to evaluate the result of the modular multiplication which is in Montgomery form (i.e., $\bar{a} \times \bar{b} \times R^{-1} \; mod \; q$). In cryptosystems such as Kyber, several multiplications are performed in a row, so the intermediate results can be left in Montgomery form while only the final output is transformed out of it. Consequently, sampling incorrectly to generate errors is not directly possible in Montgomery form. To solve this issue, we need to convert the product (or a subsequent result of modular operations in the processor implementing a χ-PIP) out of the Montgomery form (using the Montgomery reduction) before the incorrect sampling step. This comes at a slight increase in the overall implementation cost. Eventually, the K-RED algorithm is somewhat similar to Montgomery as it operates on the residue class of the input. However, K-RED outputs a value that is congruent to a multiple of the input. So again, the K-RED algorithm does not support error generation via incorrect sampling unless the output is converted out of the K-RED representation.

We note that a power-of-two modulus facilitates the operations since reduction modulo q can then be computed by low-cost bit masking, or even at no cost if the size of the target platform matches the number of bits of the modulus (e.g., 16-bit or 32-bit). Such moduli facilitate the use of inexact computing to generate errors as well.

Error distribution. Natively, many LWE-based cryptosystems require the generation of errors from the (discrete) Gaussian distribution [BLP+13]. However, various works have shown that it is challenging to implement a discrete Gaussian sampler efficiently [BCNS15], especially if the implementation has to be protected against timing attacks (see for example [PBY17]). Therefore, PQ cryptosystems such as Frodo use a distribution that is very close to it and at the same time easier to implement via pre-computed lookup tables. We posit that a similar conclusion holds for implementations based on inexact computing. Namely, we expect the generation of discrete Gaussian errors to be significantly more challenging, especially due to the tail of the distribution. As a result, the centered binomial distribution (CBD) is gaining popularity, thanks to its easy, efficient, and secure implementation features [ADPS16]. It is used in both the Kyber and NewHope cryptosystems. From a security viewpoint, this choice is backed up by the argument that the performances of the best attacks against LWE-based encryption schemes do not depend on the exact distribution of the error, but rather on its standard deviation [ABD+21b].

Other design parameters. The ring dimension and the polynomial degree are not influencing the possibility to leverage inexact computing to generate erroneous samples.

5 FPGA prototype

The previous sections described how one could instantiate instances of LWPE based on simulations, and discussed the wide applicability of this concept for the efficient implementation of cryptosystems. In this section, we finally confirm the concrete feasibility of this idea by describing an FPGA prototype of an LWPE instance. Our target device is a Xilinx Artix-7 (28nm technology) mounted on a NewAE CW-305 board. We first describe the general architecture of our design and then discuss cost and security considerations.

Note that the adversary we consider in this paper is a passive one. But as discussed in [BHK+21], our implementation can be complemented with a fault detection scheme to prevent manipulations of the supply voltage targeting the calibration of the processor. More generally, previous work investigated the impact of other environmental (e.g., voltage) changes on LPPN samples [KBS+18]. Since our LWPE prototype leverages similar techniques to insert (bit-level) faults in inner product designs, we expect it to behave similarly, which we leave as an interesting scope for further investigations.

5.1 Architecture of the LWPE prototype

Our FPGA prototype instantiates a 1024-bit inner product modulo 2^8 with the same CBD-2 error distribution as discussed in Section 3.2. The choice of a power-of-two modulus allows us to simplify the architecture and, as will be clear later, the analysis of the output data dependencies. The prototype is composed of two main modules (see Figure 10): the Physical Inner Product (PIP) module and the Variable Delay Line (VDL).

Physical Inner Product. The inner product block receives the 128-byte input secret $\boldsymbol{s}$ and the 128-byte public input $\boldsymbol{a}$. Each byte of the input secret s_i, with $i \in [0, 127]$, is multiplied by the corresponding public input byte a_i, and the result is truncated to the least significant 8 bits. The output of the multiplication layer is then fed to 7 successive layers of 8-bit adders. The result at each intermediate adder layer is truncated to the least significant 8 bits as well. Clearly, the architecture of the χ-PIP in this configuration is byte-slice as any intermediate value is truncated at the least significant 8 bits, and adder/multiplier units are all 8-bit wide. It has to be noted that each intermediate wire has been declared with the *DONT_TOUCH* HDL constraint, in order to guarantee that

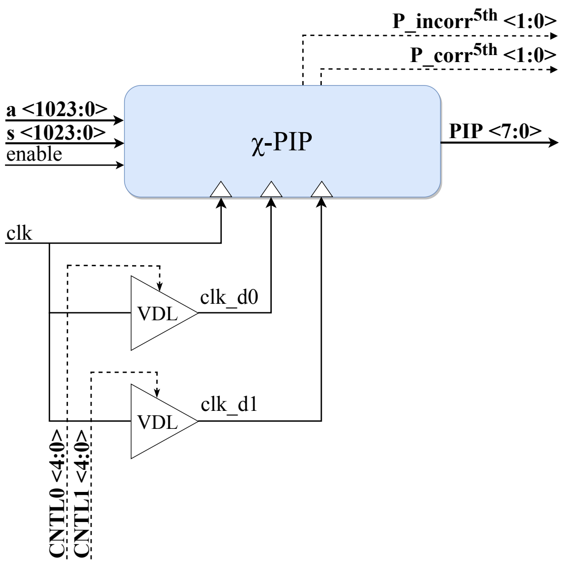

Figure 10: LWPE FPGA prototype (dashed lines are only for configuration and testing).

all the signals of the inner product are kept during the synthesis and the place&route steps of the design flow. This constraint is usually needed as we leverage the generation of combinatorial glitches to achieve inexact sampling and generate errors. As shown in Figure 11, the χ-PIP is organized as a two-stage pipeline. The first stage is composed of the multiplication layer and the first 4 adder layers. The second one is composed of the remaining adder layers. As also discussed in Section 3.2, the incorrect sampling is performed on the LSB of byte B_0 and B_1 of the 5^{th} adders layer. The final 8-bit output corresponds to the output of the last adder in the logical cone of the second stage.

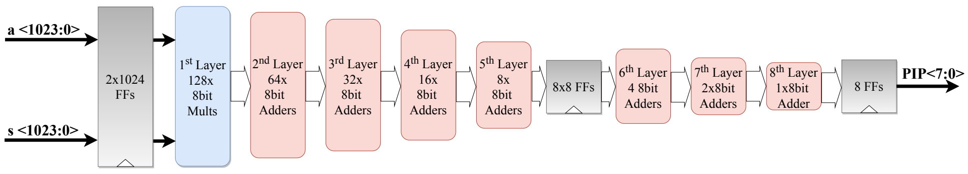

Figure 11: Architecture of the prototype LWPE's χ-PIP.

As the generation of physical errors in this design takes place in an intermediate stage of the pipeline, we must ensure that no metastable signals are propagated to the second part of the χ-PIP computation and Clock Domain Crossing (CDC) is safely done. For a classical digital design, the clock replica generated by the VDL can be considered as an additional clock domain, which should be correctly handled to avoid issues after the inexact sampling. Therefore, flip-flops that are devoted to inexact sampling are doubled, as shown in Figure 12. A first D-type flip-flop is fed by the delayed clock *clk_del*, which has the role to capture a non-stable intermediate and to generate a physical error with a certain probability (e.g., 0.5 in this specific case). A second D-type flip-flop, fed by the system's clock tree (*clk_sk*), resynchronizes the datum sampled by the first stage, in order to make it compliant with the rest of the architecture. The two flip-flops have to be instantiated with the Xilinx's *ASYNC_REG* attribute, so that this double stage sequential element is preserved from optimizations that would remove it from the implementation.

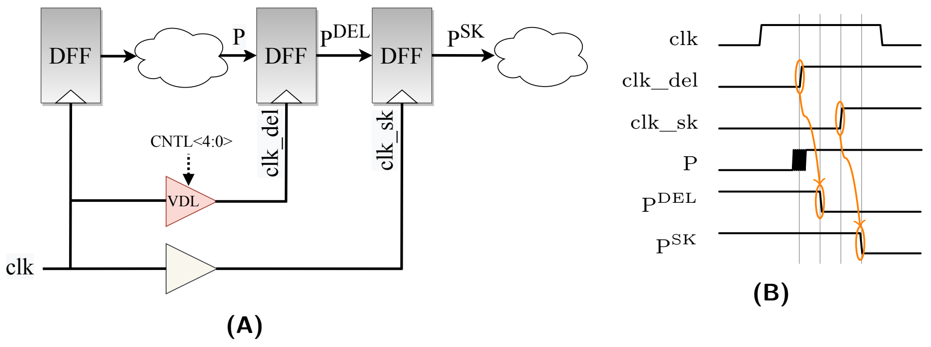

Figure 12: Inexact sampling module (A) and its relative timing diagram (B).

Variable Delay Line. The Variable Delay Line (VDL) makes use of a fixed delay component, provided by two LUT5 configured as buffers, and a programmable one, that is provided by a 16 *CARRY4* chain and a tree of F7/F8 and LUT-based multiplexers. A *CARRY4* unit is a specific logic unit that provides a very fast path for carrying propagation in arithmetic circuits. These specific units offer a variety of outputs that are characterized by a very fast propagation delay, making them suitable to implement fine-grain delay lines. In particular, 16 *CARRY4* chains provide 64 taps of ∼20ps each. The taps are selected by a 5-bit control word ($CNTL$). This compact structure has been recently proposed by Bellizia et al. in [BHK+21] for the design of an LPPN prototype. We currently used two independent VDLs in the LWPE prototype, but further optimization may lead to different choices. For example, the two incorrectly sampled bits may share the same VDL if the two-timing paths provide a very similar propagation delay in all PVT corners.

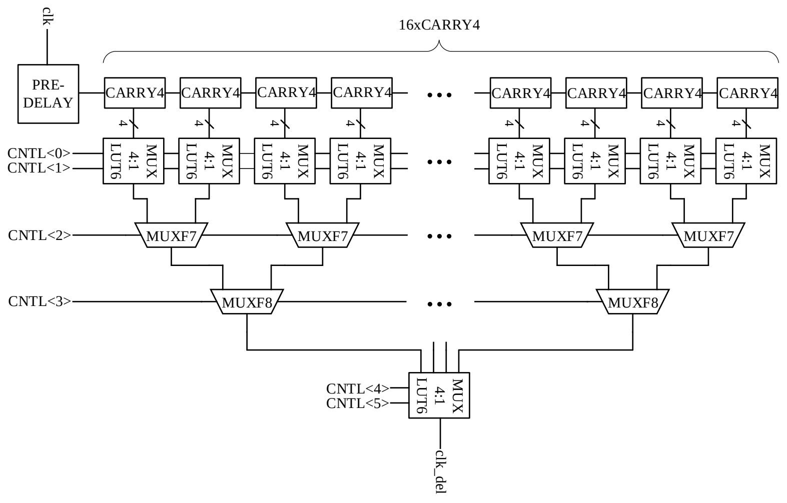

Figure 13: FPGA implementation of the Variable Delay Line (VDL) from [BHK+21].

5.2 FPGA Resources

We report a detailed summary of the resources needed to implement our LWPE prototype on the Xilinx Artix-7 FPGA in Table 1. As for the LPPN prototype in [BHK+21], we observe that the cost of the (unavoidable) inner product module is dominant and the cost of the VDDs is negligible. As in this previous work as well, we note that comparing an LWPE prototype with a classical LWE implementation is not straightforward since it would require comparing the cost of a dedicated RNG for which many solutions exist. Yet, we believe the general conclusions of LPPN prototypes extent to LWPE. Namely, hard physical problems can be implemented efficiently and reliably, and they gain interest when side-channel attacks are a concern since their design can be directly masked thanks to the homomorphic nature of the inner product computations, without the need to protect the generation of errors against leakage, since these errors are never computed explicitly.

Table 1: FPGA resources of our LWPE prototype.

Module	LUT	Regs
LWPE	5557	2127
χ-PIP	5515	2127
VDL1	21	0
VDL2	21	0

A bit more precisely, the standard alternatives to implement LWE samples are to use a TRNG or a PRNG. The conceptual advantages of LWPE over the TRNG solution is that it avoids the simple side-channel attack where the adversary would probe the error. Its quantitative advantages are more difficult to evaluate. For illustration, we report in Table 2 the results of the survey in [PMB+16] which compares the throughput, the area footprint, and the feasibility/reliability of different FPGA-suitable TRNGs. Overall, we observe that the low overheads needed to produce LWPE samples are in line with the ones of cheap / low throughput TRNGs. The quantitative comparison with PRNGs is simpler, especially if they have to be prevented against side-channel attacks, since it would typically require the implementation of either a masked primitive or a leakage-resilient stream cipher, both of them being significantly more expensive to implement [GMK17, YSPY10].

5.3 Empirical evaluation of the results

We first analyzed the standard deviation of the error samples collected from our FPGA prototype. For this purpose, we used $409,600$ samples (4 sets of $102,400$ samples under 4 different keys). For each of those sets we estimated a standard deviation of 1.00 (i.e., very close to the distribution of the CBD_2 which is exactly equal to 1). This is sufficient to guarantee the security against LWE attacks according to estimators like [APS15].

Next, and as a last contribution, we used our FPGA prototype to assess the output data dependencies of the errors based on a (much) larger number of samples than allowed by simulations. We ran the Kruskal-Wallis statistical test mentioned in Section 2.6 on the sets of samples in order to assess their potential output dependencies.

Concretely, and for each of the 4 keys, we considered the 256 empirical distributions

Table 2: Summary of RNGs implementations on Xilinx Spartan-6 from [PMB+16].

RNG Type	Area (LUTs/Regs)	Special Macros	Bit Rate [Mbps]	Feasibiliy & Repeatability
ERO	46/19	No	0.0042	Very Good
COSO	18/3	No	0.54	Very Poor
MURO	521/131	No	2.57	Good
PLL	34/12	Yes	0.44	Medium
TERO	39/12	No	0.625	Very Poor
STR	346/256	No	154	Poor

obtained by sorting the samples according to the correct value of their inner product. The Kruskal-Wallis test was then used with the null hypothesis that all those error distributions were identical. The test did not reject this null hypothesis despite a large number of samples available. As usual with such statistical tests, this does not prevent small dependencies could be observed with even more samples. Yet, the important observation here is that the solutions described in Section 3.2 to prevent mathematical output data dependencies extents to physical output data dependencies. So we can anyway conclude that our LWPE prototype generates samples with significantly lower output data dependencies than the LPPN prototype in [BHK$^+$21]. And as a result, as long as an adversary does not gain any leakage on the intermediate computations of the LWPE prototype, we can assert with confidence that the the LWPE samples are closer to LWE ones than LPPN samples are compared to LPN samples.[6] Combined with the observation that both LPPN and LWPE prototypes have excellent features for side-channel resistance thanks to masking (meaning that such leakage would be hard to exploit if these prototypes are masked), which will further reduce data dependencies (see the discussion of Section 5 in [BHK$^+$21]), we can therefore conclude that the output error distributions of our LWPE prototype has good features for integration in the implementation of actual cryptosystems.

We note that the limited physical data dependencies of the $\mathrm{LWPE_B}$ samples has a simple physical explanation. Looking back at Section 3.2.2, we know that mathematical data dependencies vanish due to the addition of uniformly distributed and random value Y with the erroneous intermediate computation X. So as long as X and Y are also physically independent (i.e., as long as there are no couplings / crosstalk in the circuit such that the errors of X would be influenced by the value of Y), the same observation holds for physical data dependencies. More precisely, these physical dependencies will occur over intermediate values, but the resulting error will be independent of the correct value of the inner product. Since we know from the leakage literature that such coupling effects are in general hard to measure and exploit [CBG$^+$17], it is natural that the coupling-induced data-dependent errors are significantly more difficult to detect than the glitch-induced data-dependent errors observed for the LPPN prototype of [BHK$^+$21] — they may even be impossible to detect without artificially amplifying them by design.

6 Conclusions and further research

Hard physical learning problems have been proposed as a provocative idea to circumvent certain implementation challenges in recent cryptosystems based on hard learning problems. First progresses in this direction were focused on the simpler LPN/LPPN case, and their generalization to LWE/LWPE was therefore an important open question (since the LWE assumption is much more versatile than the LPN one). In this paper, we showed that such a generalization is possible, although instances of LWPE require to tweak some of the design ideas used in LPPN prototypes. Namely, LWPE designs highly benefit from the introduction of the errors in an intermediate computation stage rather than at its output, in order to reduce the output data-dependencies of the errors' distributions.

These results lead to interesting scopes for further investigations. First, one could study whether the design tweaks we used for LWPE could be used in LPPN prototypes as well. This may not be direct since the simplicity of the LPPN computations leaves less room (i.e., critical path) to introduce errors in intermediate stages while keeping a sufficient control of the error probability. But combining this idea with longer serial parts than in [BHK$^+$21] could lead to positive outcomes. Second, we showed that the output data-dependencies of our LWPE prototype can be made small. But even if the

[6] Strong leakage on intermediate computations could make it possible to exploit their data dependencies before diffusion reduces them. Whether such attacks can be efficient is an interesting open problem.

exploitation of such small dependencies is not a direct security threat, a formal reduction from the standard LWE problem to a LWE-OD problem with output dependencies remains of theoretical interest (and may be motivated by other LWPE designs). Eventually, the understanding of the performances, reliability and side-channel security of hard physical learning problems remains at an early stage. So on the one hand, further challenging these ideas in concrete case studies, and possibly designing schemes that can better take advantage of approximate computing, would be beneficial to better specify the contexts in which they are useful. And on the other hand, investigating the robustness of LWPE against environmental changes (e.g., power supply, clock frequency, temperature) and formalizing security against input data dependencies are important long-term goals.

Acknowledgments. François-Xavier Standaert is a senior research associate of the Belgian Fund for Scientific Research (F.R.S.-FNRS). Pierrick Méaux was supported by the ERC Advanced grant CLOUDMAP (num. 787390). This work has been funded in parts by the European Union through the ERC consolidator grant SWORD (num. 724725) and by the Walloon Region through the Win2Wal project PIRATE (num. 1910082).

References

[AAB+20] Erdem Alkim, Roberto Avanzi, Joppe Bos, Léo Ducas, Antonio de la Piedra, Thomas Pöppelmann, Peter Schwavbe, Douglas Stebila, Martin R. Albrecht, Emmanuela Orsini, Valery Osheter, Kenneth G. Paterson, Guy Peer, and Nigel P. Smart. NewHope algorithm specifications and supporting documentation. https://newhopecrypto.org/data/NewHope_2020_04_10.pdf, 2020. 2020-04-10.

[ABD+21a] Erdem Alkim, Joppe W. Bos, Léo Ducas, Patrick Longa, Ilya Mirono, Micheal Naehri, Valeria Nikolaenko, Chris Peikert, Ananth Raghunathan, and Douglas Stebila. FrodoKEM learning with errors key encapsulation. https://frodokem.org/files/FrodoKEM-specification-20210604.pdf, 2021. 2021-06-04.

[ABD+21b] 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. https://pq-crystals.org/kyber/data/kyber-specification-round3-20210804.pdf, 2021. 2021-08-04.

[ADPS16] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - A new hope. In Thorsten Holz and Stefan Savage, editors, *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*, pages 327–343. USENIX Association, 2016.

[AHH+19] Martin R. Albrecht, Christian Hanser, Andrea Höller, Thomas Pöppelmann, Fernando Virdia, and Andreas Wallner. Implementing rlwe-based schemes using an RSA co-processor. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(1):169–208, 2019.

[AMS+11] Frederik Armknecht, Roel Maes, Ahmad-Reza Sadeghi, François-Xavier Standaert, and Christian Wachsmann. A formalization of the security features of physical functions. In *32nd IEEE Symposium on Security and Privacy, S&P 2011, 22-25 May 2011, Berkeley, California, USA*, pages 397–412. IEEE Computer Society, 2011.

[APS15] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *J. Math. Cryptol.*, 9(3):169–203, 2015.

[Bar86] Paul Barrett. Implementing the rivest shamir and adleman public key encryption algorithm on a standard digital signal processor. In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 311–323. Springer, 1986.

[BCNS15] Joppe W. Bos, Craig Costello, Michael Naehrig, and Douglas Stebila. Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 553–570. IEEE Computer Society, 2015.

[BGR+21] 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.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, pages 309–325. ACM, 2012.

[BHK+21] Davide Bellizia, Clément Hoffmann, Dina Kamel, Hanlin Liu, Pierrick Méaux, François-Xavier Standaert, and Yu Yu. Learning parity with physical noise: Imperfections, reductions and FPGA prototype. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(3):390–417, 2021.

[BHLY16] Leon Groot Bruinderink, Andreas Hülsing, Tanja Lange, and Yuval Yarom. Flush, gauss, and reload - A cache attack on the BLISS lattice-based signature scheme. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, volume 9813 of *Lecture Notes in Computer Science*, pages 323–345. Springer, 2016.

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 575–584. ACM, 2013.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106. IEEE Computer Society, 2011.

[CBG+17] Thomas De Cnudde, Begül Bilgin, Benedikt Gierlichs, Ventzislav Nikov, Svetla Nikova, and Vincent Rijmen. Does coupling affect the security of masked implementations? In *COSADE*, volume 10348 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2017.

[DKL+18] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):238–268, 2018.

[DKRV18] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. Saber: Module-lwr based key exchange, cpa-secure encryption and cca-secure KEM. In *AFRICACRYPT*, volume 10831 of *Lecture Notes in Computer Science*, pages 282–305. Springer, 2018.

[DLL$^+$21] Léo Ducas, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium algorithm specifications and supporting documentation. 2021. 2021-02-08.

[DMG21] BaKhacTrieu Dang, Kamyar Mohajerani, and Kris Gaj. High-speed hardware architectures and fair fpga benchmarking. In *Third PQC Standardization Conference, June 7 - 9, 2021*, 2021.

[DT14] Constantin Catalin Dragan and Ferucio Laurentiu Tiplea. Efficient key-policy attribute-based encryption for general boolean circuits from multilinear maps. *IACR Cryptol. ePrint Arch.*, page 462, 2014.

[EFGT17] Thomas Espitau, Pierre-Alain Fouque, Benoît Gérard, and Mehdi Tibouchi. Side-channel attacks on BLISS lattice-based signatures: Exploiting branch tracing against strongswan and electromagnetic emanations in microcontrollers. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 1857–1874. ACM, 2017.

[FO99] 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.

[GMK17] Hannes Groß, Stefan Mangard, and Thomas Korak. An efficient side-channel protected AES implementation with arbitrary protection order. In *CT-RSA*, volume 10159 of *Lecture Notes in Computer Science*, pages 95–112. Springer, 2017.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO (1)*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92. Springer, 2013.

[Har09] David Harvey. Faster polynomial multiplication via multipoint kronecker substitution. *J. Symb. Comput.*, 44(10):1502–1510, 2009.

[KBBS20] Dina Kamel, Davide Bellizia, Olivier Bronchain, and François-Xavier Standaert. Side-channel analysis of a learning parity with physical noise processor. *J. Cryptogr. Eng.*, pages 1–9, 2020.

[KBS$^+$18] Dina Kamel, Davide Bellizia, François-Xavier Standaert, Denis Flandre, and David Bol. Demonstrating an LPPN processor. In *ASHESCCS*, pages 18–23. ACM, 2018.

[KKB$^+$22] Dur-e-Shahwar Kundi, Ayesha Khalid, Song Bian, Chenghua Wang, Máire O'Neill, and Weiqiang Liu. Axrlwe: A multilevel approximate ring-lwe co-processor for lightweight iot applications. *IEEE Internet Things J.*, 9(13):10492–10501, 2022.

[KO63] Anatoly A. Karatsuba and Y. Ofman. Multiplication of multidigit numbers on automata. *Soviet Physics Doklady*, 7:595–596, 1963. URL: `http://cr.yp.to/bib/entries.html#1963/karatsuba`.

[KSD+20] Dina Kamel, François-Xavier Standaert, Alexandre Duc, Denis Flandre, and Francesco Berti. Learning with physical noise or errors. *IEEE Trans. Dependable Secur. Comput.*, 17(5):957–971, 2020.

[KW52] William H. Kruskal and W. Allen Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47(260):583–621, 1952.

[LN16] Patrick Longa and Michael Naehrig. Speeding up the number theoretic transform for faster ideal lattice-based cryptography. In Sara Foresti and Giuseppe Persiano, editors, *Cryptology and Network Security - 15th International Conference, CANS 2016, Milan, Italy, November 14-16, 2016, Proceedings*, volume 10052 of *Lecture Notes in Computer Science*, pages 124–139, 2016.

[LPR12] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *IACR Cryptol. ePrint Arch.*, page 230, 2012.

[LS12] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *IACR Cryptol. ePrint Arch.*, page 90, 2012.

[Lyu09] Vadim Lyubashevsky. Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, volume 5912 of *Lecture Notes in Computer Science*, pages 598–616. Springer, 2009.

[MKV20] Jose Maria Bermudo Mera, Angshuman Karmakar, and Ingrid Verbauwhede. Time-memory trade-off in toom-cook multiplication: an application to module-lattice based cryptography. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(2):222–244, 2020.

[MNS05] M. Maymandi-Nejad and M. Sachdev. A monotonic digitally controlled delay element. *IEEE Journal of Solid-State Circuits*, 40(11):2212–2219, 2005.

[Mon85] Peter L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44:519–521, 1985.

[PBY17] Peter Pessl, Leon Groot Bruinderink, and Yuval Yarom. To BLISS-B or not to be: Attacking strongswan's implementation of post-quantum signatures. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 1843–1855. ACM, 2017.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 333–342. ACM, 2009.

[PMB+16] Oto Petura, Ugo Mureddu, Nathalie Bochard, Viktor Fischer, and Lilian Bossuet. A survey of AIS-20/31 compliant TRNG cores suitable for FPGA devices. In Paolo Ienne, Walid A. Najjar, Jason Helge Anderson, Philip Brisk, and Walter Stechele, editors, *26th International Conference on Field Programmable Logic and Applications, FPL 2016, Lausanne, Switzerland, August 29 - September 2, 2016*, pages 1–10. IEEE, 2016.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):34:1–34:40, 2009.

[SPOG19] Tobias Schneider, Clara Paglialonga, Tobias Oder, and Tim Güneysu. Efficiently masking binomial sampling at arbitrary orders for lattice-based crypto. In *Public Key Cryptography (2)*, volume 11443 of *Lecture Notes in Computer Science*, pages 534–564. Springer, 2019.

[Too63] Andrei L. Toom. The complexity of a scheme of functional elements realizing the multiplication of integers. *Soviet Mathematics Doklady*, 3:714–716, 1963. http://www.de.ufpe.br/ toom/my-articles/engmat/MULT-E.PDF.

[UXT$^+$22] 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.

[XL21] Yufei Xing and Shuguo Li. A compact hardware implementation of cca-secure key exchange mechanism CRYSTALS-KYBER on FPGA. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(2):328–356, 2021.

[YSPY10] Yu Yu, François-Xavier Standaert, Olivier Pereira, and Moti Yung. Practical leakage-resilient pseudorandom generators. In *CCS*, pages 141–151. ACM, 2010.