# Fault Attack Countermeasures for Error Samplers in Lattice-Based Cryptography

James Howe*, Ayesha Khalid†, Marco Martinoli*, Francesco Regazzoni‡, and Elisabeth Oswald*
*Department of Computer Science, University of Bristol, UK,
†Centre for Secure Information Technologies (CSIT), Queen's University Belfast, Northern Ireland,
‡Advanced Learning and Research Institute, Università della Svizzera Italiana, Switzerland.

***Abstract*—Lattice-based cryptography is one of the leading candidates for NIST's post-quantum standardisation effort, providing efficient key encapsulation and signature schemes. Most of these schemes base their hardness on variants of LWE, and thus rely heavily on error samplers to provide necessary uncertainty by obfuscating computations on secret information. Because of this it is a clear and obvious target for side-channel analysis, with numerous types of attacks targeting this component to gain secret-key information. In order to bring potential lattice-based cryptographic standards to practical realisation, it is important to protect these modules from past and future fault and side-channel attacks. This paper proposes countermeasures that exploit the distributions expected from these error samples, that is either Gaussian or binomial, by using statistical tests to verify the samplers are operating properly. The novel countermeasures are designed to protect against all previous fault attacks on error samplers. We optimize hardware implementation of the proposed tests to avoid division and square root calculations, however, the countermeasure we propose is sufficiently generic to be suitable also for software. We measure the impact of these countermeasures on performance and area consumption on a Xilinx Artix-7 FPGA. Our countermeasure achieve promising performance while resulting in a minimal overhead.**

***Index Terms*—Lattice-based cryptography, fault attacks, countermeasures, FPGA, hardware security.**

## I. Introduction

Post-quantum (or quantum-safe) cryptography has seen a substantial expansion recently, in part due to the NIST call for quantum-safe algorithms [NIS16a]. This call is essential to secure the future of secure communications that use public-key cryptography since the schemes currently used, based on the hardness of factoring prime numbers (RSA) or the discrete logarithm problem (ECC/ECDSA), will be solved in polynomial time with a scalable quantum computer.

Amongst the submissions to the NIST call, schemes based on the hardness of lattice problems seem to be the leading candidates and make up the largest proportion of submissions. Lattice-based cryptography is a very appealing candidate due to its security; offering average-case to worst-case hardness, its efficiency; outperforming many other candidates in software or hardware, and versatility; having schemes for advanced cryptographic services such as identity-based encryption, as well as standard primitives such as encryption, signatures, and key encapsulation [HPO+15].

This research was supported in part by EPSRC via grant EP/N011635/1 and by the European Union Horizon 2020 SAFEcrypto project (grant no. 644729).

Error sampling is one of the main modules within lattice-based cryptographic schemes, and are critical components as these are used to hide computations on secret information and make the scheme computationally hard. This can be seen from Table I, which shows those lattice-based cryptographic schemes, which rely on error sampling, that were submitted to NIST for post-quantum standardisation. It can be seen that Gaussian or binomial is typically used, i.e. a distribution with a statistically normal shape, however this can lead to a number of side-channel vulnerabilities. Many of the side-channel attacks of lattice-based cryptographic schemes have exploited the vulnerabilities of the error sampler, mainly targeting modules such as Gaussian sampling [Pes16], [EFGT17], [EFGT16], [BHLY16], [BBK16], [PBY17]. This research proposes countermeasures to all previous physical attacks on error samplers.

The main contribution of this paper is to propose a number of countermeasures, via statistical tests, for the normally distributed error samplers used in lattice-based cryptography. The tests are grouped by their computational complexity, categorised as *low cost*, *standard*, and *expensive*, which should be used depending on the application device and/or security level. The tests are applicable to Gaussian and binomial error samplers which cover most, if not all, lattice-based cryptographic schemes, especially those submitted to NIST for post-quantum standardisation. The proposed tests are then demonstrated in conjunction with the most efficient error sampler in hardware [HKR+18], a look-up based technique, to evaluate the latency and area costs in hardware. The results show that error samplers are still able to remain practical, with a minimal overhead in speed and area, whilst operating in a sound and genuine fashion.

In Section II we provide further motivation and preliminaries, which is followed by a summary of related work in Section III. In Section IV we discuss the countermeasures proposed in this research, which is then followed by details on their hardware designs, specifically those optimisations used to ensure the smallest impact in performance. The results of these hardware designs are given in Section VI, followed by details on how these countermeasures can be integrated in those lattice-based cryptographic schemes considered state-of-the-art.

TABLE I
LATTICE-BASED KEY ENCAPSULATION (KEM) AND SIGNATURE SCHEMES SUBMITTED TO NIST FOR POST-QUANTUM STANDARDISATION, SEPARATED BY THEIR NIST SECURITY LEVELS, ERROR TYPE (GAUSSIAN OR BINOMIAL), AND PARAMETER VALUE (FOR ALL SCHEMES $\mu = 0$).

| Crypto. Type | Cryptographic Scheme | Security Level | Error Type | Error Parameter ($\sigma$) |
|---|---|---|---|---|
| KEM | Ding Key Ex. [DTGW17] | 1,3,5 | G | $2.6, 4.19$ |
| | (R-)EMBLEM [SPL+17] | 1 | G | $25, 3$ |
| | FrodoKEM [NAB+17] | 1,5 | G | $2.75, 2.3$ |
| | Kyber [SAB+17] | 1,3,5 | B | $\sqrt{2}$ |
| | LAC [LLJ+17] | 1,3,5 | B | $1/\sqrt{2}, 1/2$ |
| | LIMA [SAL+17] | 3 | B | $3.16$ |
| | Lizard [CPL+17] | 1,5 | G | $\approx 2$ |
| | LOTUS [PHAM17] | 1,2,3,4,5 | G | $3$ |
| | KCL [ZjGS17] | 3,5 | B | $\sqrt{8}, \sqrt{6}$ |
| | NewHope [PAA+17] | 1,3 | B | $2$ |
| | NTRU-RSS-KEM [SHRS17] | 1 | B | $1$ |
| | NTRUEncrypt [ZCHW17a] | 1,3,5 | G | $724$ |
| | Mersenne-756839 [AJPS17] | 1 | G | $28.64$ |
| | Titanium [SSZ17] | 1,3,5 | B | $\sqrt{2}$ |
| Sign-ature | Dilithium-G [LDK+17] | 1,2,3 | G | 19200, 17900,12400 |
| | Falcon [PFH+17] | 1,2,3,4,5 | G | $171.8, 213.1$ |
| | pqNTRUSign [ZCHW17b] | 1 | G | $107$ |
| | qTESLA [BAA+17] | 1,3,5 | B | $8.5, 10$ |

## II. PRELIMINARIES

We denote the sample size, $n$, and standard deviation, $\sigma$, which is typical notation in lattice-based cryptography. Table I provides all potential NIST post-quantum candidates that utilise either a Gaussian or binomial sampler, for which this research is applicable. This table also provides parameters for the standard deviation, but to simplify the calculations we propose in Section IV we instead use the statistical variance, $s$, which translates simply as $s = \sigma^2$. Calculations on variance have no impact on the test results and mean we can avoid explicitly calculating square roots.

We use typical statistics notations to define the target mean, $\mu$, the target variance, $s$, the sample mean, $\bar{x}$, and the sample variance, $\bar{s}$. We also require pre-stored values for specific t-tests, $t_{\alpha/2}$, and chi-squared tests, $\chi^2_\alpha$, for $n-1$ degrees of freedom and significance level $\alpha$ (here we use $\alpha = 0.99$), which are used for comparisons against our calculated statistics. These pre-stored test statistics can be changed to the whims of the implementer, with no impact on performance. Definitions for the equations we use to calculate the test statistics can be found in Table II.

Most lattice-based schemes have their security foundations on the learning with errors (LWE) problem [Reg05]: Given some uniformly distributed vectors $\mathbf{a}_i \in \mathbb{Z}_q^n$, integers $n$ and $q$, and $b_i \equiv \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i \bmod q$, where the secret-key $\mathbf{s}$ is chosen uniformly at random from $\mathbb{Z}_q^n$ and each $e_i$ follow some small error distribution, find $\mathbf{s}$ given access to pairs $(\mathbf{a}_i, b_i)$. The problem asks an adversary to either find $\mathbf{s} \in \mathbb{Z}_q^n$ given $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{b} \equiv \mathbf{A}^T\mathbf{s}+\mathbf{e} \bmod q$ *or* to distinguish between $(\mathbf{a}_i, b_i)$ and $(\mathbf{a}_i, u_i)$ where $u_i$ is chosen uniformly at random.

In other words, solving a system of linear equations is usually easy, but as soon as an error ($e_i$) is added to the equations, it becomes a hard mathematical problem and there exists no quantum or classical algorithm that could solve this problem in polynomial time. Thus, schemes based on LWE, with large enough parameters, are considered quantum-secure.

The error generated to hide these secret-key operations has a statistically 'normal' shape, which are typically either Gaussian or binomial. From Table I, it can be seen that half of the lattice-based key encapsulation (KEM) schemes submitted to the NIST post-quantum standardisation use binomial sampling, the other half use Gaussian sampling. Lattice-based signature schemes typically require Gaussian sampling. A number of side-channel attacks have targeted these modules [BHLY16], [EFGT16], [PBY17], [EFGT17], [EFGT18] in order to gain secret-key information or to break the cryptographic scheme. Moreover, there has been little work on countermeasures for these attacks beyond shuffling [RVV13], [Saa17], which may not be sufficient [Pes16].

NIST have also stressed for the need for efficient side-channel protected schemes [NIS16b]: "Schemes that can be made resistant to side-channel attack at minimal cost are more desirable than those whose performance is severely hampered by any attempt to resist side-channel attacks."

Binomial samplers are generally realized by uniformly sampling two $k$-bit vectors and computing their respective Hamming weights. The binomial distributed variable is obtained by subtracting the Hamming weights of both $k$-bit vectors. Gaussian samplers on the other hand are much more complex, with the number of different techniques ranging from arithmetic-based to table lookup-based. In hardware, it was shown by Howe et al. [HKR+18] that the cumulative distribution table (CDT) lookup-based sampler (shown in Algorithm 1) is the most efficient method, and later by Khalid et al. [KHR+18] the same method was shown to be scalable for various parameters. This method for error sampling has been adopted by at least four of the NIST post-quantum candidates, including FrodoKEM [NAB+17].

In any case, irrespective of the method used to derive the Gaussian or binomial variables, it is important they are correct and secure from any type of side-channel analysis, such as fault attacks, which is the goal of this paper. Moreover, it is also important to merely ensure these error samplers are operating correctly, following requirements from strict theoretical foundations. Providing assurances for this is an important attribute in itself and is essential if lattice-based cryptographic schemes are implemented into real-world applications.

## III. RELATED WORK

Implementations of lattice-based cryptographic schemes have been investigated in the past using side-channel analysis. Many of these have targeted error sampling modules, such as Gaussian samplers, in order to gain secret-key information.

Fault analysis attacks are particularly prominent when attacking error samplers, as this could mean outputting all zeroes, outputting error samples with smaller variance which are easier to solve, or outputting values that are significantly more predicable to an adversary. This can even be seen in Algorithm 1; where if one were to zero the `max` variable or instantiate a fault in the `while`-loop, then the error sampler

would not sufficient hide the secret-key computations and deteriorate its zero-knowledge.

**Algorithm 1** CDT Sampling via Binary Search

**Require:**
Three integers $min$, $mid$, and $max$
CDT: $0 = S[0] < S[1] < \cdots < S[N] = 1$
Uniform $r \in \{0, \ldots, (2^\lambda - 1)\}$ and a bit $b \leftarrow \{0, 1\}$
**Ensure:** $min \leftarrow 0$; $max \leftarrow N$; $mid \leftarrow (min + max)/2$;
**while** $(max > min)$ **do**
  **if** $(r \geq S[mid])$ **then**
    $min \leftarrow (mid + 1)$;
  **else**
    $max \leftarrow mid$;
  **end if**
**end while**
**return** $x = (-1)^b(mid - 1)$

Fault analysis applied to lattice-based signature schemes is described in depth by Bindel et al. [BBK16]. From their analysis they conclude that zeroing of the error samplers is applicable to *all* the signature schemes they consider with a small number of faults (either 1 or 2).

This attack is shown in practice by Espitau et al. [EFGT16], reporting fault attacks on the error sampling components of a number of lattice-based signature schemes.

Some countermeasures have been suggested for error samplers, the most used technique employs shuffling [RVV13], [Saa17]. However, these countermeasures would do little against a fault attack and as Pessl [Pes16] shows, this countermeasure is still not sufficient, recovering secret-key information in a practical real-world setting.

Many other side-channel attacks on lattice-based cryptographic schemes have been shown that are specific to software implementations [BHLY16], [Pes16], [EFGT17], exploiting information leakage via cache memory, power leakage, template attacks, or using branch tracing. The error sampler is not always the attack target here and other components can be used by an adversary, this is also illustrated in the analysis by Bindel et al. [BBK16].

## IV. Proposed Countermeasures

In this section we discuss the proposed countermeasures for error samplers. The countermeasures are categorised, where each level deploys increasingly powerful statistical analysis, hence aims at thwarting more powerful adversaries, but comes at a higher cost in performance and hardware area consumption. The discussions about these tests are generic enough to be applied to hardware or software, but optimizations are specific to hardware. A summary of the equations we use for these tests is shown in Table II.

### A. Low Cost

This test counts the number of repetitions in the observation and raises an alarm flag if the repetitions exceed an improbable value (user defined, for example; 10). Fault attacks including *zeroing attacks* and *early loop abort* result in a constant stream of the same value(s), leaving secret-key information visible or predictable to an adversary.

### B. Standard

This countermeasure will calculate the sample mean ($\bar{x}$) and sample variance ($\bar{s}$), whilst also checking for repetitions. Checking the sample variance is particularly important, as this parameter is linked to the hardness of the cryptographic scheme's LWE problem. Minimising the variance of the error sampler has the potential to make the LWE instances easier to solve. These countermeasures would also spot errors and bugs in the sampler, and any malicious implementations. In hardware, this test will be computed after a power-of-two sample size so we do not have to explicitly compute the division. This is convenient for schemes like Kyber [SAB$^+$17] and Dilithium(-G) [LDK$^+$17] which require $n = 256$ samples for each respective key encapsulation or signature.

For the sample mean calculation, we require one accumulation of all of the $n$ outputs, followed by a power-of-two division: $(\sum_{i=1}^{n} x_i)/n$. For the sample variance calculation, we use the same register for the mean calculation, as well as the power-of-two division, the only extra element we need is an extra accumulator to accumulate the sum of the squared outputs to calculate: $(\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2/n)/n$.

Once we have approximations for the mean and variance, we can perform statistical tests to see whether these values are acceptable. For the mean, we perform a t-test to see whether the sample mean is within (pre-calculated) acceptable bounds (i.e. confidence intervals). For the variance, we perform a chi-squared test to see if our sample variance is within an acceptable predefined value. The null hypothesis ($H_0$) for these tests are for our target mean (similarly, target variance) to equal our sample mean (similarly, sample variance).

These tests are much more effective than the low cost variant at spotting bugs and errors in the sampler, as well as more sophisticated physical attacks. For example, an attack which could minimise the range of values output from an error sampler would make the LWE instances much more easier to solve. The calculations here would spot an attack of this type and report an error. These tests will also ensure that the Gaussian distribution we have in practice is the same distribution we require theoretically.

There are *online* alternatives for these equations, proposed by Welford [Wel62]. However, these equations require explicit division (thus, floating point numbers) which would be inefficient in hardware. For software testing, this would be a better option, which would also provide consistently updated testing, rather than testing after a fixed number of trials.

### C. Expensive

This test will include those in the previous categories, as well as a chi-squared test for comparing observed and expected values. Essentially, ensuring the output distribution is exactly what we require from a theoretical standpoint. It will also spot poor pseudo-randomness, as well as any programming

TABLE II
DETAILS OF THE PROPOSED HARDWARE AND SOFTWARE TESTS.

| Test Level | Test Description | Test Formula |
|---|---|---|
| Low Cost | Check for repetitions | A counter for if $x_i = c$ |
| Standard | Sample Mean ($\bar{x}$) | $(\sum_{i=1}^{n} x_i)/n$ |
| | Sample Variance ($\bar{s}$) | $(\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2)/n$ |
| | Standard Error of $\bar{x}$ | $\text{SE}_{\bar{x}} = \bar{s}/\sqrt{n}$ |
| | Test Statistic for $\bar{s}$ | $T = (n/s)\bar{s}$ |
| | Null Hypothesis | Check if $\lvert\mu\rvert < \bar{x} + t_{\alpha/2}\text{SE}_{\bar{x}}$ |
| | Null Hypothesis | Check if $T < \hat{\chi}^2_{n,\alpha/2}$ |
| Expensive | Chi-Squared Test | $\hat{\chi}^2 = \sum_{k=1}^{n} \frac{(\text{obs}(k) - \exp(k))^2}{\exp(k)}$ |
| | Test Statistic for $\hat{\chi}^2$ | $\chi^2(df = n-1, p\text{-value})$ |
| | Null Hypothesis | Check if $\hat{\chi}^2 < \chi^2(n-1, p\text{-value})$ |

errors, malicious error sampler designs, or erroneous activity caused by damage to the device. For this we require a look-up table to store a histogram of counts for each output of the error sampler, i.e. the observed values. This is then compared to the CDT table, i.e. the expected values, to verify that the frequencies of these outputs are valid. These observed and expected values are then used in a chi-squared goodness-of-fit test via the calculation: $\sum_{k=1}^{n} \frac{(\text{obs}(k) - \exp(k))^2}{\exp(k)}$. If this test statistic is within certain bounds we fail to reject our null hypothesis for normality. Like the previous tests, these bounds are also known in advanced and are hence pre-stored on the device. Observed values that have been generated via a source of poor pseudo-randomness would be spotted here due to the fact that our expected values have been calculated based on theoretically sufficient randomness.

## V. HARDWARE DESIGN

The hardware design of fault attack countermeasures has been undertaken as separate synthesizable HDL modules for the three categories previously discussed. The user may opt to choose any of these in the design as the security level or resource budget of the system allows. The hardware results are shown in Table III and a brief overview now follows.

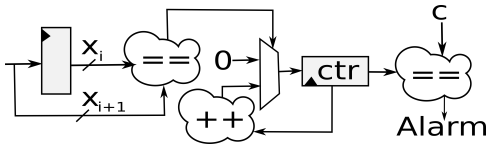


Fig. 1. Hardware architecture of the proposed low cost countermeasure.

The low cost countermeasure, shown in Figure 1, is the simplest of the three and requires a pipeline register to hold the previous valid value of sampler ($x_i$). A comparator compares the $i^{\text{th}}$ and $i+1^{\text{th}}$ valid sampler outputs and in case of a match a counter ($ctr$) is incremented. An $alarm$ flag is raised as the counter exceeds a legal user defined bound $c$.

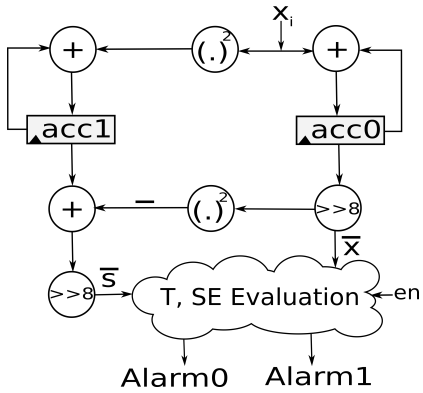


Fig. 2. Hardware architecture of the proposed standard countermeasure.

For the standard level countermeasure, shown in Figure 2, uses the calculation of $\bar{x}$ and $\bar{s}$ and requires accumulation of sampler outputs and its squared values, in two separate accumulation registers, namely $acc0$ and $acc1$, respectively. For $\bar{x}$, every $n = 256$ samples, a power-of-two division (by 256) is carried out by simply *right shifting* the accumulator ($acc0$) value by 8 (to evaluate $(\sum_{i=1}^{n} x_i)/n$). Similarly for the $\bar{s}$ calculation, we use the squared output of $\bar{x}$ (in $acc0$) and subtract that from the sum of the squared samples held in $acc1$, to calculate: $(\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2/n)/n$. By keeping $n$ as a power of two, the actual division is avoided, which is preferable since division is slow and expensive in hardware. Hence, the division by $n$ (to get approximations for $\bar{x}$, $\bar{s}$) and by $\sqrt{n}$ (in the $\text{SE}_{\bar{x}}$ approximation) is simply a right shift by 8 and 4, respectively. The values of $t_{\alpha/2}$ and $n/s$ are pre-calculated values. To keep the hardware generic, DSP multipliers are avoided. The standard test can raise either of the two alarms in case the hypothesis bound is exceeded.

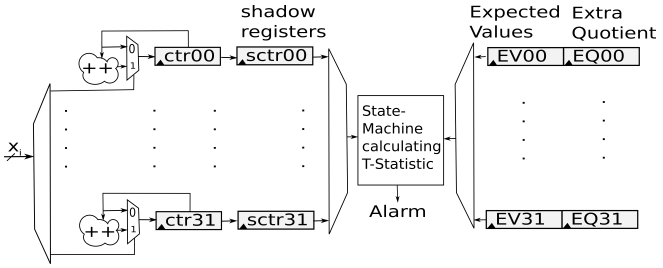


Fig. 3. Hardware architecture of the proposed expensive countermeasure.

The expensive countermeasure is shown in Figure 3, and is rightly named so for the cost it incurs in terms of the hardware overhead. This countermeasure requires an array of registers, array depth being as many as number of possible error sampler outputs (32 for $\sigma = 3.33$). This register array ($ctr00 - ctr31$) holds the sample count and is initialized with zeros as the system resets. For each sampler output generated ($x_i$), a register in the $ctr$ array, respective to the sample output value observed is incremented, to keep count of the observed values. As soon as the $n^{\text{th}}$ sampler output is used to update the $ctr$ array, the observed values are stored in a shadow array ($sctr00 - sctr31$) for further calculations of the chi-squared

test, while the original observed values array ($ctr00 - ctr31$) is cleared (to reset values of all zeros) to start counter update for the future sampler outputs, concurrently. The expected values ($EV00 - EV31$) are an array of pre-calculated values stored as a ROM. In order to avoid the division by the expected values, we store $\frac{256}{\exp(k)}$, instead of $\frac{1}{\exp(k)}$ as extra quotients ($EQ00 - EQ31$). The final computation steps through all the array values requires 32 clock cycles for the 32 bins. These calculations are carried out by a state machine. For each of the 32 observed values held in $sctrn$, subtraction from the pre-stored expected values in $EVn$ is carried out, the difference is squared and a division is simply carried out by multiplication by pre-calculated quotients in $EQn$. After 32 cycles, the accumulated value is used to calculate the test statistic, failure of result to lie within the legal bound raises an $alarm$.

## VI. Results

The hardware results for the proposed countermeasures are shown in Table III, demonstrated on a Xilinx Artix-7 FPGA. In order to fully realise the performance analysis of the proposed countermeasures, we integrate each test with a constant-time CDT sampler. The CDT sampler design has been shown in the past to be the most efficient method in hardware [HKR$^+$18] utilising minimal hardware resources whilst maintaining a high throughput performance, requiring 6 clock cycles per sample.

The low cost countermeasure can be implemented in 3 slices on the FPGA, which is an increase of $8\%$. Moreover, this countermeasure has no impact on the throughput performance of the error sampler. Essentially making this countermeasure extremely relevant to hardware implementations of any lattice-based cryptographic scheme, having almost zero degradation in area or performance, as well as protecting many potential attacks, faults, or errors.

The standard countermeasure is realised using slightly more hardware resources, however still remains relatively inexpensive. On its own, the countermeasure requires 24 slices, adding an $44\%$ to the CDT sampler with the integrated countermeasures. However, this countermeasure, as well as the low cost one, has a fixed cost and will not require additional hardware resources if the error sampler were slower and/or larger. The standard countermeasure also has minimal impact on performance, as most of the calculations are computed in parallel to the running of the error sampler. Overall it requires only one additional clock cycle to complete the calculations after the error sampler has completed generating its $n$ samples.

The expensive countermeasures require significantly larger hardware resources than the ones previously proposed. This is essentially due to the requirements of keeping a histogram of all the observed values output from the error sampler. On its own, the countermeasure utilises 126 slices, which is nearly 4x larger than just the CDT sampler. Once integrated into the CDT sampler, the hardware resources required are either 129 or 149 slices, depending on whether block-RAM is used, where the countermeasure takes around $85\%$ of the overall area consumption. The expensive countermeasure has a small impact on performance; the calculations needed at the end of $n$ error samples is just 32 clock cycles.

TABLE III
Post-place and route (PAR) results for the proposed countermeasures, separately, and the countermeasures integrated with a CDT sampler with $\sigma = 3.33$.

| Countermeasure Category | LUT/FF | Slices | DSP/ BRAM | Freq. (MHz) | Clock Cycles | Ops/sec ($\times 10^6$) |
|---|---|---|---|---|---|---|
| Plain CDT Sampler | 115/81 | 33 | 0/0 | 297 | 6 | 49.5 |
| Low Cost | 6/10 | 3 | 0/0 | - | $+0^\dagger$ | - |
| CDT with Low Cost | 123/91 | 36 | 0/0 | 297 | 6 | 49.5 |
| Standard | 74/58 | 24 | 0/0 | - | $+1^\dagger$ | - |
| CDT with Standard | 182/139 | 55 | 0/0 | 297 | 6 | 49.5 |
| Expensive | 226/436 | 126 | 1/0 | - | $+32^\dagger$ | - |
| CDT with Expensive | 315/517 | 149 | 1/0 | 297 | 6 | 49.5 |
| CDT with Expensive | 251/453 | 129 | 1/1 | 193 | 6 | 38.6 |

## VII. Applications

Typically, a side-channel countermeasure would flag malicious or erroneous activity by invalidating the output by outputting logical false ($\perp$) instead. This symbol is also typically used in cryptography to signify incorrect decryption or verification. In hardware, countermeasures could also be linked to the reset of the design in order to ensure the device's countermeasures correctly withstand the attack.

Alternatively, many lattice-based cryptographic schemes instantiate conditionals in the final stages of their protocol, such as an if-statement, typically for security purposes. This is especially true for signature schemes which require rejection conditions so that no information about the secret-key is leaked with the signature. With respect to NIST post-quantum candidates, this can be seen in Kyber [SAB$^+$17, Alg. 9, Line 7], FrodoKEM [NAB$^+$17, Alg. 14, Line 15], and Dilithium [LDK$^+$17, Fig. 4, Line 22]. Adding to this conditional would have little or no impact on the performance of the cryptographic scheme and would be a suitable place to include the results of our proposed countermeasures. Specifically, the ciphertext or signature should only be output if the countermeasures output an 'accept', i.e. that the error sampler is operating correctly, in combination with the conditionals that are already in place.

Another application of these countermeasures is on cryptographic outputs that follow a normal distribution. For example, the lattice-based signature scheme, BLISS [DDLL13], has outputs that follow a Gaussian distribution. The proposed tests here could also be applied to validate the signature outputs of BLISS, and any other lattice-based cryptographic outputs that have a normal structure, which which might be more efficient than other countermeasures, such as verify-after-sign.

## VIII. Conclusions

The aim of this paper is to propose cheap and efficient fault attack countermeasures for use in error samplers in lattice-based cryptography. These types of tests are not only important to protect against fault attacks and side-channel analysis, but

$^\dagger$This is a one-off clock cycle count at the end of $n$ samples, whereas others are clock cycles per sample.

also to ensure confidence that the error distributions produced in real-world applications are correct. The results for the proposed designs show that it is possible to protect against attacks on error samplers as well as having little or no effect on the efficiency or area consumption of the module. The proposed novel countermeasures not only thwart fault attacks but also mean other errors (such as bugs, environmental damage, and programming errors) are observed which is particularly important for IoT applications.

## REFERENCES


[AJPS17] Divesh Aggarwal, Antoine Joux, Anupam Prakash, and Mikos Santha. Mersenne-756839. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

[BAA+17] Nina Bindel, Sedat Akleylek, Erdem Alkim, Paulo S. L. M. Barreto, Johannes Buchmann, Edward Eaton, Gus Gutoski, Juliane Kramer, Patrick Longa, Harun Polat, Jefferson E. Ricardini, and Gustavo Zanon. qtesla. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

[BBK16] Nina Bindel, Johannes Buchmann, and Juliane Krämer. Lattice-based signature schemes and their sensitivity to fault attacks. In *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2016 Workshop on*, pages 63–77. IEEE, 2016.

[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 *International Conference on Cryptographic Hardware and Embedded Systems*, pages 323–345. Springer, 2016.

[CPL+17] Jung Hee Cheon, Sangjoon Park, Joohee Lee, Duhyeong Kim, Yongsoo Song, Seungwan Hong, Dongwoo Kim, Jinsu Kim, Seong-Min Hong, Aaram Yun, Jeongsu Kim, Haeryong Park, Eunyoung Choi, Kimoon kim, Jun-Sub Kim, and Jieun Lee. Lizard. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

[DDLL13] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal Gaussians. In *Advances in Cryptology–CRYPTO 2013*, pages 40–56. Springer, 2013.

[DTGW17] Jintai Ding, Tsuyoshi Takagi, Xinwei Gao, and Yuntao Wang. Ding key exchange. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

[EFGT16] Thomas Espitau, Pierre-Alain Fouque, Benoît Gérard, and Mehdi Tibouchi. Loop-abort faults on lattice-based fiat-shamir and hash-and-sign signatures. In *International Conference on Selected Areas in Cryptography*, pages 140–158. Springer, 2016.

[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 *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1857–1874. ACM, 2017.

[EFGT18] Thomas Espitau, Pierre-Alain Fouque, Benoit Gerard, and Mehdi Tibouchi. Loop-abort faults on lattice-based signature schemes and key exchange protocols. *IEEE Transactions on Computers*, (11):1535–1549, 2018.

[HKR+18] James Howe, Ayesha Khalid, Ciara Rafferty, Francesco Regazzoni, and Máire O'Neill. On practical discrete Gaussian samplers for lattice-based cryptography. *IEEE Transactions on Computers*, 67(3):322–334, 2018.

[HPO+15] James Howe, Thomas Pöppelmann, Máire O'Neill, Elizabeth O'Sullivan, and Tim Güneysu. Practical lattice-based digital signature schemes. *ACM Trans. Embedded Comput. Syst.*, 14(3):41:1–41:24, 2015.

[KHR+18] Ayesha Khalid, James Howe, Ciara Rafferty, Francesco Regazzoni, and Máire O'Neill. Compact, scalable, and efficient discrete gaussian samplers for lattice-based cryptography. In *Circuits and Systems (ISCAS), 2018 IEEE International Symposium on*, pages 1–5. IEEE, 2018.

[LDK+17] Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, and Damien Stehle. CRYSTALS-Dilithium. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

[LLJ+17] Xianhui Lu, Yamin Liu, Dingding Jia, Haiyang Xue, Jingnan He, and Zhenfei Zhang. Lac. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

[NAB+17] Michael Naehrig, Erdem Alkim, Joppe Bos, Léo Ducas, Karen Easterbrook, Brian LaMacchia, Patrick Longa, Ilya Mironov, Valeria Nikolaenko, Christopher Peikert, Ananth Raghunathan, and Douglas Stebila. FrodoKEM. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

[NIS16a] NIST. Post-quantum crypto project. http://csrc.nist.gov/groups/ST/post-quantum-crypto/, 2016.

[NIS16b] NIST. Submission requirements and evaluation criteria for the post-quantum cryptography standardization process. https://csrc.nist.gov/csrc/media/projects/post-quantum-cryptography/documents/call-for-proposals-final-dec-2016.pdf, 2016.

[PAA+17] Thomas Pöppelmann, Erdem Alkim, Roberto Avanzi, Joppe Bos, Léo Ducas, Antonio de la Piedra, Peter Schwabe, and Douglas Stebila. Newhope. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

[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 *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1843–1855. ACM, 2017.

[Pes16] Peter Pessl. Analyzing the shuffling side-channel countermeasure for lattice-based signatures. In *INDOCRYPT 2016*, pages 153–170. Springer, 2016.

[PFH+17] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

[PHAM17] Le Trieu Phong, Takuya Hayashi, Yoshinori Aono, and Shiho Moriai. Lotus. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93. ACM, 2005.

[RVV13] Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. High precision discrete Gaussian sampling on FPGAs. In *International Conference on Selected Areas in Cryptography*, pages 383–401. Springer, 2013.

[Saa17] Markku-Juhani O Saarinen. Arithmetic coding and blinding countermeasures for lattice signatures. *Journal of Cryptographic Engineering*, pages 1–14, 2017.

[SAB+17] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, and Damien Stehle. CRYSTALS-Kyber. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

[SAL+17] Nigel P. Smart, Martin R. Albrecht, Yehuda Lindell, Emmanuela Orsini, Valery Osheter, Kenny Paterson, and Guy Peer. Lima. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

[SHRS17] John M. Schanck, Andreas Hulsing, Joost Rijneveld, and Peter Schwabe. NTRU-HRSS-KEM. Technical report, National Institute of Standards and Technology, 2017. available

at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

[SPL+17] Minhye Seo, Jong Hwan Park, Dong Hoon Lee, Suhri Kim, and Seung-Joon Lee. EMBLEM and R-EMBLEM. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

[SSZ17] Ron Steinfeld, Amin Sakzad, and Raymond K. Zhao. Titanium. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

[Wel62] BP Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420, 1962.

[ZCHW17a] Zhenfei Zhang, Cong Chen, Jeffrey Hoffstein, and William Whyte. NTRUEncrypt. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

[ZCHW17b] Zhenfei Zhang, Cong Chen, Jeffrey Hoffstein, and William Whyte. pqNTRUSign. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

[ZjGS17] Yunlei Zhao, Zhengzhong jin, Boru Gong, and Guangye Sui. KCL (pka OKCN/AKCN/CNKE). Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.