Probabilistic Algorithms with applications to countering Fault Attacks on Lattice based Post-Quantum Cryptography

Nimish Mishra[1] and Debdeep Mukhopadhyay[1]

Indian Institute of Technology Kharagpur
nimish.mishra@kgpian.iitkgp.ac.in, debdeep@cse.iitkgp.ac.in

Abstract. Fault attacks that exploit the propagation of effective/ineffective faults present a richer attack surface than Differential Fault Attacks, in the sense that the adversary depends on a single bit of information to eventually leak secret cryptographic material. In the recent past, a number of propagation-based fault attacks on Lattice-based Key Encapsulation Mechanisms have been proposed; many of which have no known countermeasures. In this work, we propose an orthogonal countermeasure principle that does not follow adhoc strategies (like shuffling operations on secret coefficients), but rather depends on cryptographically-backed guarantees to provide quantifiable defence against aforementioned fault attacks. Concretely, we propose a framework that uses rejection sampling (which has been traditionally used as alternatives to trapdoors) to convert otherwise deterministic algorithms to probabilistic ones. Our specific goals allow careful selection of distributions such that our framework functions with a constant number of retries (around $2-3$) for unfaulted executions. In other words, should a fault be injected, the probability of success is negligible; for correct execution however, the probability of success is overwhelmingly high. Using our framework, we hence enable probalistic decryptions in Kyber, NewHope, and Masked Kyber, and completely cut-off fault propagation in known attacks on these constructions, allowing a sound defence against known fault attacks in literature.

Furthermore, we show the extension of our framework to discrete distributions, and demonstrate it against recent fault attacks on Dilithium. In similar semantics as with KEMs, our framework, to the best of our knowledge, allows development of the first (sound) countermeasure strategy against Differential Fault Analysis attacks on Dilithium. Concretely, use of our framework converts the otherwise deterministic sampling of blinding vector coefficients in Dilithium to probabilistic sampling, and ties this probability to presence of faults. As with KEMs, in absence of faults, our rejection samplers allow operation of Dilithium in a constant number of retries. In presence of faults however, the number of retries grows unacceptably high for an adversary.

Finally, although demonstrated at applications for countering faults in Post-Quantum Cryptography, our framework is rather generic and may be of interest independently from fault analysis. In some sense, given a-priori knowledge of a target distribution f, our framework converts a

deterministic algorithm to its probabilistic variant that "verifies" whether it executes on distributions statistically indistinguishable from f. Should the algorithm attempt to operate upon an incorrect distribution, the probability of success drops sharply to negligible, essentially rendering the algorithm ineffective and unable to make progress. For correct execution, however, the design of the framework guarantees a constant number of retries, thereby preventing a non-constant blowup of execution time due to rejection sampling. Finally, the exact description of f helps in further implementation optimizations, reducing the computation overhead of evaluating the probabilities involved in rejection sampling.

Keywords: Fault Attacks, Kyber, Dilithium, Post-Quantum Cryptography, Rejection Sampling

1 Introduction

Physical Attacks in Post-Quantum Cryptography (PQC). PQC comprises cryptosystems that derive their security from cryptographic problems considered to be intractable by sizable quantum computers. As such, much effort has been expended by the National Institute of Standards and Technology (NIST) in the past few years into standardizing such post-quantum implementations, broadly under the following primitives [1]: Key-Encapulation Mechanisms (KEM), Digital Signatures (DS), and Public-Key Encryption (PKE). The computationally hard problems used to construct instantiations of these primitives are derived usually from either code-based crpytographic constructions (like general/syndrome decoding problem), or from multivariate cryptography (like multivariate quadratic polynomial problem and the min-rank problem), or from lattice-based cryptography (as in short-integer solutions, learning from errors, and so on).

As a consequence, over the years, the various submissions to NIST have undergone rigorous scrutiny wrt. both adherence to theoretical security guarantees of the associated primitives, as well as resilience against physical attacks (including passive side-channel attacks, and active fault injection attacks). Physical attacks, while not strictly voiding security of the primitives, have been considered important since PQC is expected to operate in the evolving landscape of embedded systems, which are expected to be functionally operational without explicit human supervision. As consequence, such embedded systems are usually assumed accessible by a *physical* adversary. In fact, physical attacks have been traditionally considered as welcome contributions to NIST's standardization process [2].

Working Principle of Fault Attacks. NIST finalists have therefore received considerable attention wrt. susceptibility to fault injection attacks [24]. For this work, we mainly consider the attacks on lattice-based primitives [1]. Fault attacks

[1] Lattice-based primitives total up to 9 out of 15 NIST's 3-rd round finalists.

on such primitives rely upon the capability of the adversary to force erroneous computation on secret-key dependent operations, which in turn causes a statistical bias distinguishable from that of non-faulty execution.

Fault attacks are then classified based on how the adversary exploits such statistical biases. In case of signature schemes like Dilithium [18], for instance, the adversary has access to the *faulty* output from the signing oracle, and can perform differential analysis of the faulty output against the non-faulty output to extract the secret cryptographic material. As such, fault models like instruction skips [27,8,29,14] and skews in randomness used by set-to-constant faults [25,26] have been typically used in such attack settings. These attacks can therefore be grouped under the umbrella term: ① "Differential Fault Attacks". On the other hand, specifically in the case of KEMs, the adversary injects faults in the decapsulation oracle, but the IND-CCA[2] security provided by the Fujisaki-Okamoto (FO) Transform [23] does the allow the adversary any access to the faulty output. Rather, failure in verification of the FO transform essentially leaks *one* bit of information to the adversary: whether the injected fault *propagated* the the FO verification. This fault propagation allows the adversary to classify injected faults as *effective* or *ineffective*, and essentially links the secret cryptographic material to such propagation. Such attacks can be referred to as ② "Propagation-based Fault Attacks". Several works like [15,10,23,11,4,9] have used effective/ineffective fault propagation to first leak the sign of *linear* decryption noise (of the Learning with Errors (LWE) samples), then create a system of inequations in the unknown error and secret, and finally use statistical solvers like Belief Propagation [23] or Lattice Reduction [10].

Existing Countermeasures. The existing countermeasures in literature closely follow the principles upon which ① Differential Fault Attacks (DFA), and ② Propagation-based Fault Attacks operate. For instance, to counter DFA, it is sufficient to prevent the adversary from obtaining the faulty output. This can be achieved by **detection-based** countermeasures which verify the extent of functional correctness of the victim primitive. For instance, in case of digital signatures, explicit verification of the random coins [24] as well as of the NTT[3] factors [12] prevents attacks like [25,26] which use faults to force skewness in randomness essential to *hiding* the secret key within the concerned operations. Likewise, explicit equality checks [24] are useful in defending against fault attacks which *skip* initialization of sensitive cryptographic material. We note that the countermeasure principle (i.e. explicit verification) against DFA therefore also neatly ties to the functioning of the FO transform in Lattice-based KEMs. FO transform essentially recomputes and verifies the correctness of the encapsulated ciphertext, thereby guaranteeing security against chosen ciphertext attacks. As an unintended consequence however, the FO transform *randomizes* the output

[2] Indistinguishability against Chosen Ciphertext Attacks
[3] Number Theoretic Transform

if a fault is *detected*[4], thereby offering a natural line of defence against DFA on Lattice-based KEMs.

Countering ② propagation-based attacks is, however, trickier. In fact, the very design of detection-based countermeasures *aids* attack strategies relying on propagation. For instance, the FO Transform in Lattice-based KEMs, while providing `IND-CCA` security and resilience to DFA, becomes the main aide of propagation based attacks [15,10,23,11,4,9]. Concretely, by reverifying correctness of ciphertext and randomizing the output in case of failure, the FO Transform leaks information on whether the injected fault was effective or ineffective, eventually leading to key recovery. To the best of our knowledge, the only known countermeasure principle in literature is to *shuffle* the order of ciphertext coefficients in the decapsulation routine. This prevents the adversary from understanding the actual secret key coefficient involved with the effective/ineffective fault propagation. While this countermeasure principle is successful for some attacks like [23], it fails in context of other attacks like [4] and [15]. Precisely, the attack in [4] relies on *fault correction* to infer the actual faulted coefficient, and hence bypasses shuffling. On the other hand, the attack in [15] suggests use of side-channels to infer the shuffled permutation of secret coefficients.

Our goal is henceforth to develop a countermeasure principle that provides sound defence against the generic class of propagation based attacks on Lattice-based cryptography; and we want to achieve this ① through cryptographically backed tools (instead of implementation afterthoughts such as shuffling), ② with quantifiable analysis of the extent of the defence in preventing the attack, while being ③ lightweight (i.e. little overhead), and ④ with potential extension to DFA.

1.1 Our Contributions

Generic Framework for Probabilistic variants of Deterministic Algorithms. The main result of this work is a generic framework that, with a-priori knowledge of the target distribution f, converts a deterministic algorithm $\mathcal{M}$ to its probabilistic variant $\mathcal{A}$. In some sense, this amounts to "verifying" whether $\mathcal{A}$ operates upon the "correct" distribution f.

We achieve this by adapting rejection sampling [16,17] to our use-case. Rejecting sampling allows to sample from an arbitrary target distribution f, given a source distribution g. If $\forall \mathbf{x}, f(\mathbf{x}) \leq M.g(\mathbf{x})$, then should the sample be output with probability $\frac{f(\mathbf{x})}{M.g(\mathbf{x})}$, the distribution of $\mathbf{x}$ is statistically indistinguishable from a random sample from f. Additionally, the expected number of retries for the rejection sampling to output $\mathbf{x}$ is M. As such, prior works [16,17] tailor g to be as close as possible to f. Their end goal is to set g to a distribution dependent on some secret cryptographic material, while f is independent of the secret. Thus, drawing from g is equivalent to using the secret cryptographic material to

[4] Note that some recent works like [20] enable DFA on KEM by skipping FO Transform itself. We do not consider these attacks here, as they are trivially bypassed by replicating the FO Transform check in the temporal domain.

generate some output $\mathbf{x}$. However, outputting $\mathbf{x}$ with distribution statistically indistinguishable from f ensures nothing about the secret is leaked.

Our goal, however, is different: we simply want to enable a probabilistic variant of $\mathcal{M}$ on distribution f (i.e. ensure $\mathcal{A}$ operates upon f). This shift in goal allows us to tailor g as a "slightly smoothed"[5] variant of f (unlike markedly different distribution, as in [16,17]). Now since f and g are "almost" close, we show that $\forall \mathbf{x}, f(\mathbf{x}) \leq M.g(\mathbf{x})$ for $M = \mathcal{O}(1)$. That is, our rejection sampling requires a *constant* number of retries for $\mathcal{A}$ to produce an output statistically indistinguishable from f. From hereon, we first *move* from deterministic $\mathcal{M}$ to probabilistic $\mathcal{F}$ (without rejection sampling), and then invoke the framework from [16] to move from $\mathcal{F}$ to $\mathcal{A}$ (with rejection sampling). Formally:

Theorem 1 (Framework for "moving" from deterministic $\mathcal{M}$ to probabilistic $\mathcal{A}$ with a-priori knowledge of f).

Let f and g be probability distributions with the property that:

$$Pr\Big[f(\mathbf{z}) \leq M.g(\mathbf{z});\ \mathbf{z} \xleftarrow{\$} f, M = \mathcal{O}(1)\Big] \geq 1 - \epsilon(.)$$

then the distribution of the following algorithm $\mathcal{M}$:

1. $\mathbf{z} \xleftarrow{\$} f$
2. *Operate upon* $\mathbf{z}$

is within statistical distance $\mid 1 - \frac{1}{M} \mid$ of the following algorithm $\mathcal{F}$:

1. $\mathbf{z} \xleftarrow{\$} f$
2. *With probability* $\frac{1}{M}$, *operate upon* $\mathbf{z}$

which is, in turn, within statistical distance $\frac{\epsilon(.)}{M}$ of the following algorithm $\mathcal{A}$:

1. $\mathbf{z} \xleftarrow{\$} g$
2. *With probability* $\texttt{min}\Big(1, \frac{f(\mathbf{z})}{M.g(\mathbf{z})}\Big)$, *operate upon* $\mathbf{z}$

The probability for $\mathcal{A}$ to operate upon $\mathbf{z}$ is $\geq \frac{1-\epsilon(.)}{M}$.

It is easy to see from Theorem 1, that for some $\mathbf{z}^* \xleftarrow{\$} f^*$, where f^* is at non-negligible statistical distance from f, algorithm $\mathcal{A}$ operates upon $\mathbf{z}^*$ with negligible probability. This provides us with a handle to construct the remaining contributions of this work.

Applications to preventing Propagation-based Fault Attacks. To the best of our knowledge, current propositions to defend against propagation based fault attacks on Lattice-based KEMs rely upon shuffling of coefficients. However, attacks like [4] and [15] bypass shuffling. In this work, however, we present a subsuming argument: fault attacks in [15,10,23,11,4,9] rely on injecting faults to

[5] Refer to Sec. 3, Sec. 4, and Sec. 5 for concrete instantiations.

force the decapsulation oracle of KEMs to operate upon some distribution f^* (statistically distinguishable from f), and exploit the effective/ineffective fault propagation to leak secret cryptographic material. It is straightforward to see then, that our framework essentially prevents the decapsulation oracle of Lattice-base KEMs to operate upon f^*. Concretely, for the *unfaulted* execution (i.e. $\mathbf{z} \xleftarrow{\$} f$), $\mathcal{A}$ operates upon $\mathbf{z}$ with overwhelming probability (i.e. a constant number of retries; empirically $2-3$); for faulted execution (i.e. $\mathbf{z}^* \xleftarrow{\$} f^*$), $\mathcal{A}$ *rejects* $\mathbf{z}$ with overwhelming probability. Essentially, our framework *stops* fault propagation, stripping the adversary with the foundation piece central to mounting the attack.

Extension to DFA. We also show that the framework in Theorem 1 is equally applicable over discrete distributions. As such, we extend it to countering fault attacks on Dilithium. Essentially, Dilithium relies upon sampling a blinding vector $\mathtt{y}$ from a "safe" set $\mathcal{G}$ to prevent leaking any information about $\mathtt{s}$ from $\mathtt{z}$ ($= \mathtt{y} + \mathtt{sc}$). Recent attacks [27,8,14] introduce faults (either instruction skips or stuck-at faults) to force sampling of $\mathtt{y} \notin \mathcal{G}$, eventually leaking $\mathtt{s}$. Without further assumptions on $\mathcal{G}$, we extend our framework in Theorem 1 by superposing a discrete distribution upon $\mathcal{G}$, and thereby "verify" that Dilithium operates upon $\mathtt{z}$ (with overwhelming probability) iff $\mathtt{y} \in \mathcal{G}$. To the best of our knowledge, this presents the first cryptographically-backed countermeasure to recent fault attacks on Dilithium [27,8,14].

Overhead. Essentially, we would like the empirical overhead of $\mathcal{A}$ to be as close as possible to that of $\mathcal{M}$. We hence discuss overhead in two dimensions: ① number of retries M before $\mathcal{A}$ produces an output, and ② the additional computational cost of evaluating the sampler probabilities.

As already mentioned, our specific goal of moving from deterministic $\mathcal{M}$ to probabilistic $\mathcal{A}$ allows choice of g to be very "close" to f, thus bounding the number of retries (i.e. M) by a constant. In other words, for unfaulted execution, a constant number of retries ($2-3$ in our empirical validation) is sufficient for $\mathcal{A}$ to produce output. Likewise, for faulted execution, because of the precise choices of f and g, $\mathcal{A}$ effectively does not generate output (with all but negligible probability).

Finally, as later sections detail, distributions f and g are Bimodal Gaussian in case of Lattice-based KEMs, and are discrete distributions over safe set $\mathcal{G}$ for Dilithium. For Bimodal Gaussian, we perform a constant number of additional exponentiation operations. For Dilithium, we use the contiguous structure of $\mathcal{G}$ over $\mathbb{Z}$ to reduce computing probabilities over $\mathcal{G}$ to simple range-checks (i.e. if $\mathtt{min}(\mathcal{G}) \leq \mathbf{x} \leq \mathtt{max}(\mathcal{G})$, it is assigned a non-zero constant probability. Otherwise, it is assigned probability 0. More details follow in Sec. 5). All in all, computing f and g incur overhead asymptotically constant overhead wrt. parameterization of the respective constructions (i.e. overhead is indepedent of parameterization), and is thus acceptable.

2 Preliminaries

We capture the essential background used as building blocks for this work.

2.1 LPR Public-Key Encryption

The LPR public-key encryption is a lattice based primitive based off the Ring-LWE (RLWE) problem [19]. The LPR encryption scheme syntax consists of a tuple of three algorithms: `LPR.PKE.KeyGen`, `LPR.PKE.Enc`, and `LPR.PKE.Dec`, which are summarized in Algo. 1, Algo. 2, and Algo. 3 respectively. For an integer q, the set $\mathbb{Z}_q$ represents a ring of integer modulo q[6]. By extension, R_q is then the polynomial ring $\mathbb{Z}_q[X]/(X^n+1)$. As evident from Algo. 1, `LPR.PKE.KeyGen` first samples secret polynomial `s` and noise polynomial `e` from a narrow distribution $\mathcal{X}$ over R_q. Likewise, the public polynomial `a` is sampled from a uniform distribution over R_q. Finally, `b` is computed as `as + e`. The decisional version of RLWE problem then (informally) states: it is computionally intractable for an adversary to distinguish `b` from $\mathcal{U}(R_q)$.

Algorithm 1 `LPR.PKE.KeyGen`

1: $\mathtt{a} \leftarrow \mathcal{U}(R_q)$
2: $\mathtt{s}, \mathtt{e} \leftarrow \mathcal{X}(R_q)$
3: $\mathtt{b} = (\mathtt{as} + \mathtt{e})$
4: **return** ($\mathtt{pk} = (\mathtt{a}, \mathtt{b})$, $\mathtt{sk} = (\mathtt{a}, \mathtt{s})$)

`LPR.PKE.Enc`, as stated in Algo. 2, then proceeds to *encode* the message m onto R_q by first centering the message bit around 0 or $\frac{q}{2}$, and then adding a *narrow* noise to it.

Algorithm 2 `LPR.PKE.Enc`

1: **Input**: Public Key `pk`, message $m \in \mathbb{Z}_2^n$
2: $\mathtt{r}, \mathtt{e}_1, \mathtt{e}_2 \leftarrow \mathcal{X}(R_q)$
3: $\mathtt{u} = \mathtt{ar} + \mathtt{e}_1$
4: $\mathtt{v} = \mathtt{br} + \mathtt{e}_2 + m \cdot \lfloor \frac{q}{2} \rfloor$
5: **return** `(u, v)`

Finally, `LPR.PKE.Dec`, as stated in Algo. 3, extracts $m^{'}$, and *decodes* it to the actual bit depending on whether $m^{'}$ is centered about 0 or $\frac{q}{2}$. Correctness follows by ensuring the parameter q to be much larger than the *narrow* noise around 0 and $\frac{q}{2}$, preventing a ciphertext corresponding to bit 1 from being incorrectly decoded as bit 0 (and vice-versa). We refer to [19] for a detailed security reduction to RLWE.

[6] q, or the ring-modulo, is one of the most important parameters in security analysis of RLWE, and by extension, the cryptosystems depending upon it.

Algorithm 3 `LPR.PKE.Dec`

1: **Input**: Secret key `sk`, ciphertext `(u, v)`
2: $m^{'}$ = `v - us`
3: **return** `DECODE`($m^{'}$)

2.2 KEMs from LPR

LPR is secure against Chosen Plaintext attacks. In order to construct a `IND-CCA` secure KEM from LPR, the post-quantum variant of the Fujisaki-Okamoto (FO) transformation [13] is used. A CCA-secure KEM is thus syntactically described as a tuple of three algorithms: `LPR.KEM.KeyGen`, `LPR.KEM.Encaps`, and `LPR.KEM.Decaps`, summarized in Algo. 4, Algo. 5, and Algo. 6. `LPR.KEM` therefore uses `LPR.PKE` with modifications for the end-goal of establishing a symmetric key `K`. To this end, `LPR.KEM.KeyGen`, in addition to invoking `LPR.PKE.KeyGen` as a black-box, also initializes `pkh` (using a public hash function $\mathcal{H}$) and a uniform n-bitstring `z`.

Algorithm 4 `LPR.KEM.KeyGen`

1: (`pk`, `sk`) = `LPR.PKE.KeyGen()`
2: `pkh` = $\mathcal{H}$(`pk`)
3: `z` $\leftarrow \mathcal{U}(\{0,1\}^n)$
4: **return** (`pk`, `sk`, `pkh`, `z`)

`LPR.KEM.Encaps` then samples a uniform message m and encrypts it using black-box `LPR.PKE.Enc`. The ciphertext, along with the hashed public key material `pkh`, is then used in a sequence of transformations to eventually extract the symmetric key `K` through a public hash oracle $\mathcal{G}$ and a key derivation function $\mathcal{KDF}$. Following the communication model established by `LPR.PKE`, the encapsulation oracle then communicates the ciphertext `c` to the decapsulation oracle.

Algorithm 5 `LPR.KEM.Encaps`

1: **Input**: `pk`, `pkh`
2: Message `m` = $\mathcal{H}(\mathcal{U}(\{0,1\}^n))$
3: ($\mathtt{K}_H$, `r`) = $\mathcal{G}$(`pkh`, `m`)
4: `c` = `LPR.PKE.Enc`(`pk`, `m`, `r`)
5: `K` = $\mathcal{KDF}$ ($\mathtt{K}_H$, $\mathcal{H}$(`c`))
6: **return** (`c`, `K`)

The decapsulation oracle, represented by `LPR.KEM.Decaps` as in Algo. 6, first decrypts the message through a black-box invocation of `LPR.PKE.Enc`. Then, the public hash oracle $\mathcal{G}$ and the key-derivation function $\mathcal{KDF}$ are invoked in manner similar to `LPR.KEM.Encaps` to create the symmetric key `K`.

To defend against chosen-ciphertext attacks, post-decryption, a re-encryption through `LPR.PKE.Enc` is performed and the correct key `K` is output iff the input ciphertext `c` was indeed correct. Otherwise, a random key (using a random vector `z`) is generated. It is straightforward to see that the security guarantees of a generic KEM can be derived from the semantic security of the underlying `LPR.PKE`, with the additional `IND-CCA` being derived from the Fujisaki-Okamoto (FO) transform. Finally, correctness of `LPR.KEM` is (informally) the same key `K` established for the *correct* ciphertext `c`; correctness is henceforth derived from the correctness of `LPR.PKE`. Likewise, it is straightforward to reduce `LPR.KEM` to RLWE through `LPR.PKE`. We note that all candidates for NIST's standardization for post-quantum Lattice-based KEMs are founded upon `LPR.KEM` as discussed here.

Algorithm 6 `LPR.KEM.Decaps`.

1: **Input**: (`sk`, `c`)
2: `m` = `LPR.PKE.Dec`(`sk`, `c`) // Point of Fault Injection
3: ($\mathtt{K}_H^{'}$, $\mathtt{r}^{'}$) = $\mathcal{G}$(`pkh`, `m`)
4: $\mathtt{c}_*$ = `LPR.PKE.Enc`(`pk`, `m`, $\mathtt{r}^{'}$)
5: **if** $\mathtt{c}_*$ = `c` **then**
6: `K` = $\mathcal{KDF}$ ($\mathtt{K}_H^{'}$, $\mathcal{H}$(`c`)) // Ineffective Fault
7: **else**
8: `K` = $\mathcal{KDF}$ (`z`, $\mathcal{H}$(`c`)) // Effective Fault
9: **return** (`c`, `K`)

2.3 Propagation based Fault Attacks

Most prior works on fault attacks [15,10,23,11,4,9] on KEMs follow the same underlying principle: ① inject a fault in `LPR.PKE.Dec` (refer Algo. 6), and ② query the victim which has established `K` using `LPR.KEM.Decaps`. Should decryption failures arise, the adversary infers that fault was effective (Line 8 in Algo. 6); otherwise, the fault was ineffective (Line 6 in Algo. 6). The core underlying difference between the attacks in [15,10,23,11,4,9] is the fault model.

Without loss of generality, we concretely establish the attack in [23] on Kyber[7]. Refer Fig. 1. The lower horizontal axis is associated with the ciphertext distribution as Kyber's decoding proceeds; likewise, the upper horizontal axis is associated with the corresponding plaintext distribution. The solid curve denotes the ciphertext distribution for plaintext message bit 0 (notice the narrow distribution around 0), while the dotted curve denotes the ciphertext distribution for plaintext message bit 1 (notice the narrow distribution around $\frac{q}{2}$). Conversion of the ciphertext distribution to plaintext domain then proceeds in three steps: ① $\times 2$, ② $+\frac{q}{2}$, and ③ observing the least-significant bit of the transformed ciphertext distribution to recover the corresponding plaintext message bit.

[7] A Lattice-based NIST Round-3 finalist for KEMs.

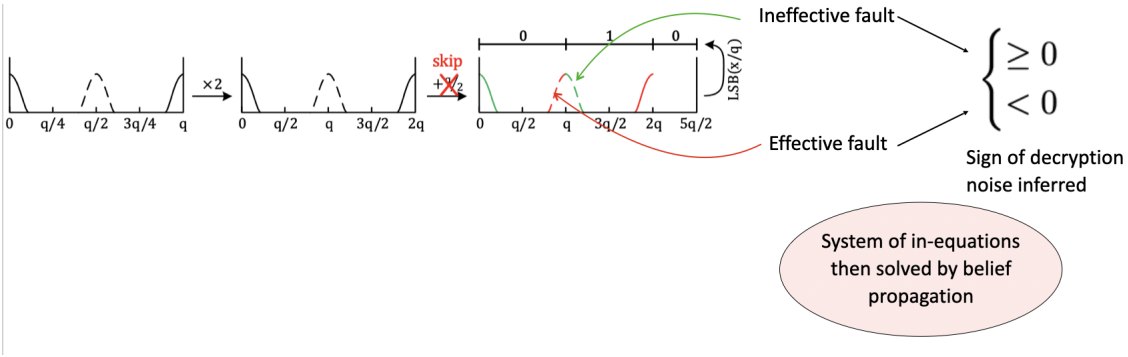

Fig. 1. Schematic of the propagation-based fault attack in [23] on Kyber.

The attack in [23] uses an instruction skip fault model to skip the second step, causing the ciphertext distribution to *left-shift* by $\frac{q}{2}$. An attempt to decode this faulty ciphertext distribution then causes either an ineffective fault or an effective fault. The adversary hence is able to infer the following inequation:

$$\Big\langle \mathbf{r}[i], \mathbf{e} \Big\rangle - \Big\langle \mathbf{e_1}[i], \mathbf{s} \Big\rangle + \mathbf{e_2}[i] \begin{cases} \geq 0, & \text{if fault is ineffective} \\ < 0, & \text{if fault is effective} \end{cases}$$

where bold-faced randomness $\mathbf{r}$ and error terms $\mathbf{e}/\mathbf{e}_1/\mathbf{e}_2$ are polynomials specific to Kyber's `LPR.KEM.KeyGen` routine, i represents the faulted coefficient, and $\langle . \rangle$ represents dot product. As is the case with `LPR.PKE.Enc` (Algo. 2), $\mathbf{r}$, $\mathbf{e}_1$ and $\mathbf{e}_2$ are known to the adversary, while $\mathbf{e}$ and $\mathbf{s}$ are secret (being a part of `LPR.PKE.KeyGen`). Over several fault injections, a system of such inequations is created by the adversary, post which statistical solvers are used to recover $\mathbf{s}$.

2.4 Rejection Sampling

Rejection sampling was first introduced in [28] to sample from an arbitrary target distribution f, given a sample from a distinct source distribution g. Rejection sampling was later used in a number of works [17,6] to construct efficient Lattice-based signatures, and was also a core tool used in CRYSTALS-Dilithium [18,7]. We state the core result from [17]:

Lemma 1 (Rejection Sampling). *Let V be an arbitrary set, and $h : V \to \mathbb{R}$ and $f : \mathbb{Z}^m \to \mathbb{R}$ be probability distributions. If $g_v : \mathbb{Z}^m \to \mathbb{R}$ is a family of probability distributions indexed by $v \in V$ with the property that there exists a $M \in \mathbb{R}$ such that:*

$$\forall v \in V, \forall \mathbf{z} \in \mathbb{Z}^m, M.g_v(\mathbf{z}) \geq f(\mathbf{z})$$

then the output distributions of the following two algorithms are statistically indistinguishable (within statistical distance $\frac{\epsilon}{M}$):

1. $v \xleftarrow{\$} h$, $\mathbf{z} \xleftarrow{\$} g_v$, *output* $(\mathbf{z}, v)$ *with probability* $\frac{f(\mathbf{z})}{M.g_v(\mathbf{z})}$
2. $v \xleftarrow{\$} h$, $\mathbf{z} \xleftarrow{\$} f$, *output* $(\mathbf{z}, v)$ *with probability* $\frac{1}{M}$

2.5 Discrete Gaussian Distribution

The Gaussian Distribution for some $x \in \mathbb{R}$, centered at $c \in \mathbb{R}$ with standard deviation $\sigma \in \mathbb{R}$ is defined by $\rho_{c,\sigma}(x) = e^{-\frac{(x-c)^2}{2\sigma^2}}$. A straightforward extension to $\mathbb{R}^n$ is then given by $\rho_{\mathbf{c},\sigma}(\mathbf{x}) = e^{-\frac{\|\mathbf{x}-\mathbf{c}\|^2}{2\sigma^2}}$ for $\mathbf{x}, \mathbf{c} \in \mathbb{R}^n$. The discrete Gaussian distribution centered at $\mathbf{c}$ is then given by $D^n_{\mathbf{c},\sigma}(\mathbf{x}) = \rho_{\mathbf{c},\sigma}(\mathbf{x})/\rho_\sigma(\mathbb{Z}^n)$.

3 Probabilistic Decryption with Rejection Sampling

We present our main result now: enabling probabilistic decryption in `LPR.KEM.Decaps` using rejection sampling. Careful choice of the rejection sampler is then tied to defending against fault attacks; concretely, the rejection sampler accepts (with overwhelming probability) non-faulted ciphertext distribution, while rejects (with overwhelming probability) the faulted distribution. In some sense, our construction allows cryptographic *verification* of the correctness of the ciphertext distribution, and prevents an adversary from introducing exploitable bias (cf. Sec. 2.3). With reference to prior attacks exploiting effective/ineffective fault propagation [15,10,23,11,4,9], probabilistic decryption in `LPR.KEM.Decaps` effectively *eliminates* fault propagation, and thus entirely prevents these attacks.

3.1 Intuition

Ciphertext Distribution. First, we establish the ciphertext distribution for various KEM instantiations based upon `LPR.KEM`, namely Kyber [3], NewHope [22], and Masked Kyber [21]. While the exact error distributions chosen does not play a part in the hardness of the underlying LWE problem [3], we remark that all implementations prefer a *narrow* discrete Gaussian about two peaks in the ciphertext space (one for each bit 0 and 1; refer Fig. 1). Formally, we can represent the ciphertext distribution of NIST KEMs as:

$$\mathcal{BG}(\mathbf{x}; \mathbf{c}_1, \mathbf{c}_2, \sigma, \lambda_1, \lambda_2) = \lambda_1.D^n_{\mathbf{c}_1,\sigma}(\mathbf{x}) + \lambda_2.D^n_{\mathbf{c}_2,\sigma}(\mathbf{x})$$

Where $\mathbf{c}_1$ and $\mathbf{c}_2$ are centers of the Bimodal Gaussian ($\mathcal{BG}$) corresponding to plaintext bit 0 and 1 without loss of generality, and $\lambda_i : i \in \{1, 2\}$ are the *mixing* parameters of the two Gaussians. Without forcing any assumption on the sample $\mathbf{x}$[8], we use $\lambda_1 = \lambda_2 = \frac{1}{2}$ hereon.

Rejection Sampler for ciphertext distribution "verification". It follows from Lemma 1, Sec. 2.4 that if $\exists M \in \mathbb{R}, \forall \mathbf{z} \xleftarrow{\$} f : f(\mathbf{z}) \leq M.g_v(\mathbf{z})$, then

[8] The ciphertext $\mathbf{x}$ has equal probability of encoding the plaintext bit 0 or bit 1.

the rejection sampling procedure produces an output distribution statistically indistinguishable from f. Intuitively, the *farther* apart the curves $M.g_v$ and f are, for any $\mathbf{z}$, *higher* is the expected number of retries (i.e. M) of the rejection sampling before an output is produced. As such, prior works [17,6] focus on carefully crafted f and g_v to reasonably bound M. The main goal of rejection sampling is to sample $\mathbf{z}$ from a secret dependent distribution g_v, but output $(\mathbf{z}, v)$ iff $\mathbf{z}$ is statistically indistinguishable from a random sample from secret independent distribution f.

In our case, we target an orthogonal goal- *use of rejection sampling to "verify" ciphertext distribution* such that `LPR.KEM.Decaps` operates (with overwhelming probability) only for the non-faulted distribution. To do so, for any given ciphertext $\mathbf{z}$, we allow sampling of $\mathbf{z}$ in `LPR.KEM.Encaps` from a *"slightly" smoother* $\mathcal{BG}(\mathbf{x}; \mathbf{c}_1, \mathbf{c}_2, \frac{\sigma}{\beta}, \frac{1}{2}, \frac{1}{2})$ for a small constant $\beta = \mathcal{O}(1)$ such that $\beta < 1$, while `LPR.KEM.Decaps` is allowed to operate on $\mathbf{z}$ with probability $\frac{\mathcal{BG}(\mathbf{z};\mathbf{c}_1,\mathbf{c}_2,\sigma,\frac{1}{2},\frac{1}{2})}{M.\mathcal{BG}(\mathbf{z};\mathbf{c}_1,\mathbf{c}_2,\frac{\sigma}{\beta},\frac{1}{2},\frac{1}{2})}$. In other words, $\mathbf{z}$ is sampled from $\mathcal{BG}(\mathbf{z}; \mathbf{c}_1, \mathbf{c}_2, \frac{\sigma}{\beta}, \frac{1}{2}, \frac{1}{2})$, but the rejection sampler ensures that the output distribution is statistically indistinguishable from $\mathcal{BG}(\mathbf{z}; \mathbf{c}_1, \mathbf{c}_2, \sigma, \frac{1}{2}, \frac{1}{2})$, which is exactly the ciphertext distribution in `LPR.KEM`.

We also note that since $\mathcal{BG}(\mathbf{z}; \mathbf{c}_1, \mathbf{c}_2, \frac{\sigma}{\beta}, \frac{1}{2}, \frac{1}{2})$ is forced to be "slightly smoother"[9] by virtue of larger standard deviation $\frac{\sigma}{\beta}$, it is straightforward to establish $\mathcal{BG}(\mathbf{z}; \mathbf{c}_1, \mathbf{c}_2, \sigma, \frac{1}{2}, \frac{1}{2}) \leq M.\mathcal{BG}(\mathbf{z}; \mathbf{c}_1, \mathbf{c}_2, \frac{\sigma}{\beta}, \frac{1}{2}, \frac{1}{2})$ for $M = \mathcal{O}(1)$. This implies that such a sampler requires only a *constant* expected number of retries to produce output. We formally prove this in Sec. 3.2. Moreover, $M = \mathcal{O}(1)$ also plays an important role in not putting much overhead in benign execution of `LPR.KEM.Decaps` due to repeated rejections. We provide experimental details in Sec. 4.

Finally, by parameterization of NIST KEMs, $\mathcal{BG}(\mathbf{z}; \mathbf{c}_1, \mathbf{c}_2, \sigma, \frac{1}{2}, \frac{1}{2})$ is usually very narrow[10] about $\mathbf{c_1}$ and $\mathbf{c_2}$. As detailed in Sec. 2.3, prior attacks utilize fault injection to force `LPR.KEM.Decaps` to operate upon a faulty distribution $\mathcal{BG}(\mathbf{z}; \mathbf{c}_1^*, \mathbf{c}_2^*, \sigma, \frac{1}{2}, \frac{1}{2})$; subsequent fault propagation leaks sign of linear decryption noise, and consequently the secret key. However, it is straightforward to see that in presence of rejection sampling of form $\frac{\mathcal{BG}(\mathbf{z};\mathbf{c}_1,\mathbf{c}_2,\sigma,\frac{1}{2},\frac{1}{2})}{M.\mathcal{BG}(\mathbf{z};\mathbf{c}_1,\mathbf{c}_2,\frac{\sigma}{\beta},\frac{1}{2},\frac{1}{2})}$, any sample $\mathbf{z}^*$ drawn from $\mathcal{BG}(\mathbf{z}; \mathbf{c}_1^*, \mathbf{c}_2^*, \sigma, \frac{1}{2}, \frac{1}{2})$ has a negligible probability of being accepted by the rejection sampler, thereby *preventing* any fault propagation. In some sense, such form of rejection sampling offers *verification* of the ciphertext distribution, and is particularly useful in discarding ciphertexts from *faulted* ciphertext distributions, thereby eliminating the possibility of fault propagation and eventual key recovery. We provide experimental details in Sec. 4.

[9] Since our goal is just to "verify" $\mathcal{BG}(\mathbf{z}; \mathbf{c}_1, \mathbf{c}_2, \sigma, \frac{1}{2}, \frac{1}{2})$, and not remove dependence of the output on secret cryptographic material (as in [17,6]), we can take the liberty to choose g_v as a "slightly" smoothed variant of f, thereby bounding $M = \mathcal{O}(1)$. In [17,6], g_v is radically different from f, hence resulting in non-constant M.

[10] In other words, all but negligible probability mass is centered about $\mathbf{c_1}$ and $\mathbf{c_2}$.

3.2 Enabling LPR.KEM.Decaps with Rejection Sampling

We now establish a *probabilistic* LPR.KEM.Decaps by virtue of rejection sampling in LPR.PKE.Dec. As mentioned in Sec. 3.1, we aim to integrate a rejection sampler of form $\frac{\mathcal{BG}(\mathbf{z};\mathbf{c}_1,\mathbf{c}_2,\sigma,\frac{1}{2},\frac{1}{2})}{M.\mathcal{BG}(\mathbf{z};\mathbf{c}_1,\mathbf{c}_2,\frac{\sigma}{\beta},\frac{1}{2},\frac{1}{2})}$ in LPR.KEM.Decaps. Details on the same follow.

Lemma 2. *In* LPR.KEM.Decaps *[3,22,21], for some* $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{Z}^n$ *and some* $\sigma, \beta \in \mathbb{R} : \beta < 1$*, the following holds for some negligible function* $\epsilon(.)$*:*

$$Pr\Big[\frac{\mathcal{BG}(\mathbf{z};\mathbf{c}_1,\mathbf{c}_2,\sigma,\frac{1}{2},\frac{1}{2})}{\mathcal{BG}(\mathbf{z};\mathbf{c}_1,\mathbf{c}_2,\frac{\sigma}{\beta},\frac{1}{2},\frac{1}{2})} \leq \mathcal{O}(1);\ \mathbf{z} \xleftarrow{\$} \mathcal{BG}(\mathbf{z};\mathbf{c}_1,\mathbf{c}_2,\sigma,\frac{1}{2},\frac{1}{2})\Big] \geq 1-\epsilon(.)$$

Proof. From Sec. 3.1, we expand $\mathcal{BG}(\mathbf{z};\mathbf{c}_1,\mathbf{c}_2,\sigma,\frac{1}{2},\frac{1}{2})$ as such:

$$\frac{1}{2}.D^n_{\mathbf{c}_1,\sigma}(\mathbf{z}) + \frac{1}{2}.D^n_{\mathbf{c}_2,\sigma}(\mathbf{z})$$

$$= \frac{1}{2}.\mathtt{exp}\Big(-\frac{\parallel \mathbf{z}-\mathbf{c}_1 \parallel^2}{2\sigma^2}\Big)\Big/\rho_\sigma(\mathbb{Z}^n) + \frac{1}{2}.\mathtt{exp}\Big(-\frac{\parallel \mathbf{z}-\mathbf{c}_2 \parallel^2}{2\sigma^2}\Big)\Big/\rho_\sigma(\mathbb{Z}^n)$$

While $\mathcal{BG}(\mathbf{z};\mathbf{c}_1,\mathbf{c}_2,\frac{\sigma}{\beta},\frac{1}{2},\frac{1}{2})$ can be likewise expanded as:

$$\frac{1}{2}.D^n_{\mathbf{c}_1,\frac{\sigma}{\beta}}(\mathbf{z}) + \frac{1}{2}.D^n_{\mathbf{c}_2,\frac{\sigma}{\beta}}(\mathbf{z})$$

$$= \frac{1}{2}.\mathtt{exp}\Big(-\frac{\parallel \mathbf{z}-\mathbf{c}_1 \parallel^2}{2(\frac{\sigma}{\beta})^2}\Big)\Big/\rho_{\frac{\sigma}{\beta}}(\mathbb{Z}^n) + \frac{1}{2}.\mathtt{exp}\Big(-\frac{\parallel \mathbf{z}-\mathbf{c}_2 \parallel^2}{2(\frac{\sigma}{\beta})^2}\Big)\Big/\rho_{\frac{\sigma}{\beta}}(\mathbb{Z}^n)$$

It is not straightforward to further reduce the given terms as they are. However, we derive another observation from the structure of ciphertext distributions in Kyber [3], NewHope [22], and Masked Kyber [21]: the centers of the bimodals can be interpreted as $(\frac{q}{4}-\frac{q}{4})$ and $(\frac{q}{4}+\frac{q}{4})$ (cf. Fig. 6, Fig. 8, and Fig. 9 for a schematic). To make analysis simpler, a *left-shift* of $\mathcal{BG}(\mathbf{z};\mathbf{c}_1,\mathbf{c}_2,\sigma,\frac{1}{2},\frac{1}{2})$ and $\mathcal{BG}(\mathbf{z};\mathbf{c}_1,\mathbf{c}_2,\frac{\sigma}{\beta},\frac{1}{2},\frac{1}{2})$ by $\frac{q}{4}$ allows centers of the bimodals to be around $-\frac{q}{4}$ and $\frac{q}{4}$. In other words, around the origin, the two modes reside at equal offset in opposite directions. We stress that the shifting is performed to ease up analysis; the bounds on the distributions are *not* affected by shifts in modal centers. The bounds are rather the property of scaling of the distributions (i.e. the standard deviations), which we do not disturb. The equation thus resolves to:

$$\frac{1}{2}.\mathtt{exp}\Big(-\frac{\parallel \mathbf{z}-\mathbf{c} \parallel^2}{2\sigma^2}\Big)\Big/\rho_\sigma(\mathbb{Z}^n) + \frac{1}{2}.\mathtt{exp}\Big(-\frac{\parallel \mathbf{z}+\mathbf{c} \parallel^2}{2\sigma^2}\Big)\Big/\rho_\sigma(\mathbb{Z}^n)$$

And

$$\frac{1}{2}.\texttt{exp}\Big(-\frac{\parallel \mathbf{z}-\mathbf{c} \parallel^2}{2(\frac{\sigma}{\beta})^2}\Big)\Big/\rho_{\frac{\sigma}{\beta}}(\mathbb{Z}^n) + \frac{1}{2}.\texttt{exp}\Big(-\frac{\parallel \mathbf{z}+\mathbf{c} \parallel^2}{2(\frac{\sigma}{\beta})^2}\Big)\Big/\rho_{\frac{\sigma}{\beta}}(\mathbb{Z}^n)$$

The ratio in the lemma is then obtained by:

$$\frac{\frac{1}{2}.\texttt{exp}\Big(-\frac{\|\mathbf{z}-\mathbf{c}\|^2}{2\sigma^2}\Big)\Big/\rho_{\sigma}(\mathbb{Z}^n) + \frac{1}{2}.\texttt{exp}\Big(-\frac{\|\mathbf{z}+\mathbf{c}\|^2}{2\sigma^2}\Big)\Big/\rho_{\sigma}(\mathbb{Z}^n)}{\frac{1}{2}.\texttt{exp}\Big(-\frac{\|\mathbf{z}-\mathbf{c}\|^2}{2(\frac{\sigma}{\beta})^2}\Big)\Big/\rho_{\frac{\sigma}{\beta}}(\mathbb{Z}^n) + \frac{1}{2}.\texttt{exp}\Big(-\frac{\|\mathbf{z}+\mathbf{c}\|^2}{2(\frac{\sigma}{\beta})^2}\Big)\Big/\rho_{\frac{\sigma}{\beta}}(\mathbb{Z}^n)}$$

For $\beta < 1$, note that the terms in the numerator are dominated by their respective counterparts in the denominator[11]. Thus, this equation holds with all but negligible probability $\geq 1-\epsilon(.)$, for correct parameterization of β. Informally, this iterates the fact that a *smoother* Gaussian (i.e. a higher standard deviation) will strictly dominate a *sharper* Gaussian (i.e. a lower standard deviation) for its entire input range except for a negligible fraction around the center (where the sharper curve is dominant). However, as stated in Corollary 1, scaling of the smoother Gaussian by some constant factor[12] adjusts this relational dominance at the center (as well as at every other point in the input domain).

Corollary 1. *In* `LPR.KEM.Decaps` *[3,22,21], for some* $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{Z}^n$*, some* $\sigma, \beta \in \mathbb{R} : \beta < 1$*, and some constant* $M \in \mathbb{Z}$*, the following holds for some negligible function* $\epsilon(.)$*:*

$$Pr\Big[\mathcal{BG}(\mathbf{z}; \mathbf{c}_1, \mathbf{c}_2, \sigma, \frac{1}{2}, \frac{1}{2}) \leq M \ . \ \mathcal{BG}(\mathbf{z}; \mathbf{c}_1, \mathbf{c}_2, \frac{\sigma}{\beta}, \frac{1}{2}, \frac{1}{2});$$

$$M = \mathcal{O}(1) \ , \mathbf{z} \xleftarrow{\$} \mathcal{BG}(\mathbf{z}; \mathbf{c}_1, \mathbf{c}_2, \sigma, \frac{1}{2}, \frac{1}{2})\Big] \geq 1 - \epsilon(.)$$

Proof. This follows directly from Lemma 2. The exact value of M, in our case, is empirically determined to allow the rejection sampler to work as close as possible with the target distribution $\mathcal{BG}(\mathbf{z}; \mathbf{c}_1, \mathbf{c}_2, \sigma, \frac{1}{2}, \frac{1}{2})$.

[11] The normalization factors cancel out without much effect from the change in standard deviation, since the total probability mass in both cases remains same. Only its distribution changes.

[12] Since, by design, the source and the target distributions differ not in modal centers but just in standard deviations, the scaling factor is constant. This allows bounding the number of retries of the rejection sampling to a constant, allowing an acceptable overhead on `LPR.KEM.Decaps`.

Lemma 3. *Let f and g be probability distributions with the property that:*

$$Pr\Big[f(\mathbf{z}) \leq M.g(\mathbf{z});\ \mathbf{z} \xleftarrow{\$} f\Big] \geq 1 - \epsilon(.)$$

then the distribution of the following algorithm $\mathcal{M}$:

1. $\mathbf{z} \xleftarrow{\$} f$
2. $\texttt{PROCESS}(\mathbf{z})$

is within statistical distance $\mid 1 - \frac{1}{M} \mid$ of the following algorithm $\mathcal{F}$:

1. $\mathbf{z} \xleftarrow{\$} f$
2. *With probability $\frac{1}{M}$,* $\texttt{PROCESS}(\mathbf{z})$

The expected number of tries for $\mathcal{F}$ to $\texttt{PROCESS}(\mathbf{z})$ is M.

Proof. The design of $\mathcal{M}$ and $\mathcal{F}$ follows from an abstraction of `LPR.KEM.Encaps` and `LPR.KEM.Decaps`. The distribution f abstracts the ciphertext distribution; transitively, $\mathbf{z}$ then abstracts the actual ciphertext output by `LPR.KEM.Encaps`. Finally, `PROCESS` abstracts the operations of `LPR.KEM.Decaps` upon $\mathbf{z}$. For this abstraction, note then that algorithm $\mathcal{M}$ is essentially the current operation of `LPR.KEM`, where the decapsulation operates upon the ciphertext output by `LPR.KEM.Encaps` with certainty. Intuitively, this lemma allows moving from a deterministic algorithm to a probabilistic one (but without the rejection sampling for now).

Let $\mathcal{M}(\mathbf{z})$ denote the distribution of $\mathcal{M}$ when operating upon $\mathbf{z}$; likewise for $\mathcal{F}$. Let $\mathcal{N}_{\mathcal{M}}$ denote the probability of $\mathcal{M}$ *not* operating upon $\mathbf{z}$; likewise for $\mathcal{F}$. We now bound the statistical distance between $\mathcal{M}$ and $\mathcal{F}$:

$$\Delta(\mathcal{M}, \mathcal{F}) = \frac{1}{2}\Big(\sum_{\mathbf{z} \in \mathbb{Z}^n} |\mathcal{M}(\mathbf{z}) - \mathcal{F}(\mathbf{z})| + |\mathcal{N}_{\mathcal{M}} - \mathcal{N}_{\mathcal{F}}|\Big)$$

Note that $\mathcal{M}$, by design, has no failure probability. Thus $\mathcal{N}_{\mathcal{M}} = 0$. We thus obtain:

$$\begin{aligned}
\Delta(\mathcal{M}, \mathcal{F}) &= \frac{1}{2}\Big(\sum_{\mathbf{z} \in \mathbb{Z}^n} \Big|f(\mathbf{z}) - f(\mathbf{z}).\frac{1}{M}\Big| + \Big|1 - \frac{1}{M}\Big|\Big) \\
&\leq \frac{1}{2}\Big(\sum_{\mathbf{z} \in \mathbb{Z}^n} f(\mathbf{z})\Big|1 - \frac{1}{M}\Big| + \Big|1 - \frac{1}{M}\Big|\Big) \\
&= \frac{1}{2}\Big|1 - \frac{1}{M}\Big|\Big(\sum_{\mathbf{z} \in \mathbb{Z}^n} f(\mathbf{z}) + 1\Big) \\
&= \frac{1}{2}\Big|1 - \frac{1}{M}\Big|.2 = \Big|1 - \frac{1}{M}\Big|
\end{aligned}$$

Intuitively, statistical distance bounds the difference between probabilities that algorithms $\mathcal{M}$ and $\mathcal{F}$ assign to the same event (i.e. $\texttt{PROCESS}(\mathbf{z})$). By design, $\mathcal{M}$ assigns probability 1, while $\mathcal{F}$ assigns probability $\frac{1}{M}$ for a single try; the difference between the two thus upper bounded by $\left|1 - \frac{1}{M}\right|$.

Furthermore, *multiple* tries of $\mathcal{F}$ are *independent* of each other, thereby allowing to represent $\texttt{PROCESS}(\mathbf{z})$ as a binomial random variable (say $\mathbb{X}$). As such, the expected number of tries of the binomial variable $\mathbb{X}$ equal to the inverse of probability of success for a single try (i.e. M). To summarize, $\mathcal{F}$ performs $\texttt{PROCESS}(\mathbf{z})$ in expected M[13] trials.

Lemma 4. *Let f and g be probability distributions with the property that:*

$$Pr\Big[f(\mathbf{z}) \leq M.g(\mathbf{z});\ \mathbf{z} \xleftarrow{\$} f\Big] \geq 1 - \epsilon(.)$$

then the distribution of the following algorithm $\mathcal{F}$:

1. $\mathbf{z} \xleftarrow{\$} f$
2. *With probability* $\frac{1}{M}$, $\texttt{PROCESS}(\mathbf{z})$

is within statistical distance $\frac{\epsilon(.)}{M}$ *of the following algorithm $\mathcal{A}$:*

1. $\mathbf{z} \xleftarrow{\$} g$
2. *With probability* $\texttt{min}\Big(1, \frac{f(\mathbf{z})}{M.g(\mathbf{z})}\Big)$, $\texttt{PROCESS}(\mathbf{z})$

The probability for $\mathcal{A}$ to $\texttt{PROCESS}(\mathbf{z})$ *is* $\geq \frac{1-\epsilon(.)}{M}$.

Proof. The design of $\mathcal{F}$ follows the same goal as Lemma 3 did. In addition, the design of $\mathcal{A}$ abstracts the operation of the rejection sampler. Note how $\texttt{PROCESS}(\mathbf{z})$, which abstracts the operations of `LPR.KEM.Decaps` as in Lemma 3, is encapsulated within the operation of the rejection sampler. This abstracts the main result of this work: enabling a probabilistic operation `LPR.KEM.Decaps` through carefully crafted probability distributions engaging in rejection sampling. Intuitively, this lemma allows moving from a "blind" probabilistic `LPR.KEM.Decaps` (i.e. $\mathcal{F}$) to a rejection sampling enabled `LPR.KEM.Decaps`, which can then be used to cryptographically "verify" distribution of $\mathbf{z}$.

We first bound the probability that $\mathcal{A}$ performs $\texttt{PROCESS}(\mathbf{z})$. We partition the set $\mathbb{Z}^n$ such that subset S contains all $\mathbf{z} \in \mathbb{Z}^n$ for which $Pr\Big[f(\mathbf{z}) \leq M.g(\mathbf{z});\ \mathbf{z} \xleftarrow{\$} f\Big] \geq 1 - \epsilon(.)$ holds. We thus derive the following, after splitting this summation over $\mathbf{z}$ belonging over S and over $\mathbb{Z}^n \backslash S$ [16]:

[13] This reinforces the need to bound $M = \mathcal{O}(1)$.

$$Pr\Big[\mathcal{A}\ \texttt{performs PROCESS}(\mathbf{z})\Big] = \sum_{\mathbf{z}\in S} g(\mathbf{z})\,.\,\texttt{min}\Big(1, \frac{f(\mathbf{z})}{M.g(\mathbf{z})}\Big) + \sum_{\mathbf{z}\notin S} g(\mathbf{z})$$

$$= \sum_{\mathbf{z}\in S} \frac{f(\mathbf{z})}{M} + \sum_{\mathbf{z}\notin S} g(\mathbf{z})$$

Where the first summation follows from the generic rejection sampling theorem (cf. Lemma 1). For $\mathbf{z} \in \mathbb{Z}^n \backslash S$ where the relation $f(\mathbf{z}) \leq M.g(\mathbf{z})$ does not hold, $\mathcal{A}$ simply relies upon the probability distribution $g(\mathbf{z})$. For our use-case, however, it follows from Lemma 2 that $\forall \mathbf{z}$, the relation $f(\mathbf{z}) > M.g(\mathbf{z})$ holds with negligible probability. Thereby, we obtain:

$$Pr\Big[\mathcal{A}\ \texttt{performs PROCESS}(\mathbf{z})\Big] \geq \sum_{\mathbf{z}\in S} \frac{f(\mathbf{z})}{M}$$

Note, finally, that the premise of $Pr\Big[f(\mathbf{z}) \leq M.g(\mathbf{z});\ \mathbf{z} \xleftarrow{\$} f\Big] \geq 1 - \epsilon(.)$ is that $\mathbf{z}$ is sampled from f, thereby lower bounding $f(\mathbf{z})$ with all but negligible probability:

$$Pr\Big[\mathcal{A}\ \texttt{performs PROCESS}(\mathbf{z})\Big] = \sum_{\mathbf{z}\in S} \frac{f(\mathbf{z})}{M} \geq \frac{1-\epsilon(.)}{M}$$

We now bound the statistical distance between $\mathcal{A}$ and $\mathcal{F}$. As with Lemma 3, let $\mathcal{A}(\mathbf{z})$ denote the distribution of $\mathcal{A}$ when operating upon $\mathbf{z}$; likewise for $\mathcal{F}(\mathbf{z})$. Likewise, let $\mathcal{N}_{\mathcal{A}}$ denote the probability of $\mathcal{A}$ not operating upon $\mathbf{z}$; likewise for $\mathcal{F}$. The statistical distance hence is given as:

$$\Delta(\mathcal{A}, \mathcal{F}) = \frac{1}{2}\Big(\sum_{\mathbf{z}\in\mathbb{Z}^n} |\mathcal{A}(\mathbf{z}) - \mathcal{F}(\mathbf{z})| + |\mathcal{N}_{\mathcal{A}} - \mathcal{N}_{\mathcal{F}}|\Big)$$

$$= \frac{1}{2}\Big(\sum_{\mathbf{z}\in S}\Big|g(\mathbf{z})\,.\,\texttt{min}\Big(1, \frac{f(\mathbf{z})}{M.g(\mathbf{z})}\Big) - \frac{f(\mathbf{z})}{M}\Big| + \sum_{\mathbf{z}\notin S}\Big|g(\mathbf{z}) - \frac{f(\mathbf{z})}{M}\Big| + |\mathcal{N}_{\mathcal{A}} - \mathcal{N}_{\mathcal{F}}|\Big)$$

Using Lemma 1 upon all but negligible $\mathbf{z} \in S$:

$$\leq \frac{1}{2}\Big(\sum_{\mathbf{z}\in S}\Big|\frac{f(\mathbf{z})}{M} - \frac{f(\mathbf{z})}{M}\Big| + \sum_{\mathbf{z}\notin S}\Big|g(\mathbf{z}) - \frac{f(\mathbf{z})}{M}\Big| + |\mathcal{N}_{\mathcal{A}} - \mathcal{N}_{\mathcal{F}}|\Big)$$

$$= \frac{1}{2}\Big(\sum_{\mathbf{z}\notin S}\Big|g(\mathbf{z}) - \frac{f(\mathbf{z})}{M}\Big| + |\mathcal{N}_{\mathcal{A}} - \mathcal{N}_{\mathcal{F}}|\Big)$$

It follows from Lemma 2 that $\forall \mathbf{z}$, the relation $f(\mathbf{z}) > M.g(\mathbf{z})$ holds with negligible probability.

$$\leq \frac{1}{2}\Big(\sum_{\mathbf{z}\notin S}\Big|\frac{f(\mathbf{z})}{M}\Big| + |\mathcal{N}_{\mathcal{A}} - \mathcal{N}_{\mathcal{F}}|\Big)$$

Substituting $\mathcal{N}_{\mathcal{A}}$ and $\mathcal{N}_{\mathcal{F}}$, the equation now becomes:

$$\leq \frac{1}{2}\Big(\sum_{\mathbf{z}\notin S}\Big|\frac{f(\mathbf{z})}{M}\Big| + \Big|\Big(1 - \frac{1-\epsilon(.)}{M}\Big) - \Big(1 - \frac{1}{M}\Big)\Big|\Big)$$

From Lemma 2, it is straightforward to infer that $\mathbb{Z}^n \backslash S$ is an empty set in all but negligible probability. The final bound hence becomes:

$$\leq \frac{1}{2}\Big(\frac{\epsilon(.)}{M} + \Big|\Big(1 - \frac{1-\epsilon(.)}{M}\Big) - \Big(1 - \frac{1}{M}\Big)\Big|\Big) = \frac{\epsilon(.)}{M}$$

This completes the proof. We now state the main result formally.

Theorem 2 (Probabilistic Decryption in `LPR.KEM.Decaps` using Rejection Sampling).
In `LPR.KEM.Decaps` *[3,22,21], for some* $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{Z}^n$ *and some* $\sigma, \beta \in \mathbb{R} : \beta < 1$, *let the following property hold, given some negligible function* $\epsilon(.)$*:*

$$Pr\Big[\frac{\mathcal{BG}(\mathbf{z}; \mathbf{c}_1, \mathbf{c}_2, \sigma, \frac{1}{2}, \frac{1}{2})}{\mathcal{BG}(\mathbf{z}; \mathbf{c}_1, \mathbf{c}_2, \frac{\sigma}{\beta}, \frac{1}{2}, \frac{1}{2})} \leq \mathcal{O}(1);\ \mathbf{z} \xleftarrow{\$} \mathcal{BG}(\mathbf{z}; \mathbf{c}_1, \mathbf{c}_2, \sigma, \frac{1}{2}, \frac{1}{2})\Big] \geq 1 - \epsilon(.)$$

then the distribution of the following algorithm $\mathcal{M}$*:*

1. $\mathbf{z} \xleftarrow{\$} \mathcal{BG}(\mathbf{z}; \mathbf{c}_1, \mathbf{c}_2, \sigma, \frac{1}{2}, \frac{1}{2})$
2. *Execute* `LPR.KEM.Decaps`$(\mathbf{z})$

is within statistical distance $|\ 1 - \frac{1}{M}\ |$ *of the following algorithm* $\mathcal{F}$*:*

1. $\mathbf{z} \xleftarrow{\$} \mathcal{BG}(\mathbf{z}; \mathbf{c}_1, \mathbf{c}_2, \sigma, \frac{1}{2}, \frac{1}{2})$
2. *With probability* $\frac{1}{M}$*, execute* `LPR.KEM.Decaps`$(\mathbf{z})$

Which is, in turn, within statistical distance $\frac{\epsilon(.)}{M}$ *of the following algorithm* $\mathcal{A}$*:*

1. $\mathbf{z} \xleftarrow{\$} \mathcal{BG}(\mathbf{z}; \mathbf{c}_1, \mathbf{c}_2, \frac{\sigma}{\beta}, \frac{1}{2}, \frac{1}{2})$
2. *With probability* $\mathtt{min}\Big(1, \frac{\mathcal{BG}(\mathbf{z}; \mathbf{c}_1, \mathbf{c}_2, \sigma, \frac{1}{2}, \frac{1}{2})}{M.\mathcal{BG}(\mathbf{z}; \mathbf{c}_1, \mathbf{c}_2, \frac{\sigma}{\beta}, \frac{1}{2}, \frac{1}{2})}\Big)$*, execute* `LPR.KEM.Decaps`$(\mathbf{z})$

Here, $M = \mathcal{O}(1)$ *is the expected number of tries for* $\mathcal{A}$ *to successfully execute* `LPR.KEM.Decaps`.

Proof. The proof follows from Lemma 2, Lemma 3, Lemma 4, and Corollary 1.

3.3 Illustrative Example

We now present an illustrative example of our rejection sampler $\frac{\mathcal{BG}(\mathbf{z};\mathbf{c}_1,\mathbf{c}_2,\sigma,\frac{1}{2},\frac{1}{2})}{M.\mathcal{BG}(\mathbf{z};\mathbf{c}_1,\mathbf{c}_2,\frac{\sigma}{\beta},\frac{1}{2},\frac{1}{2})}$ to better establish a hypothetical distribution and the operation of the rejection sampler. For the toy example, we consider $\mathcal{BG}(\mathbf{z};-4,4,1.5,\frac{1}{2},\frac{1}{2})$ as the target distribution, visualized in Fig. 2. The normalization factor to convert this into a probability distribution is omitted: it bears no effect over the overall operation of the sampler.

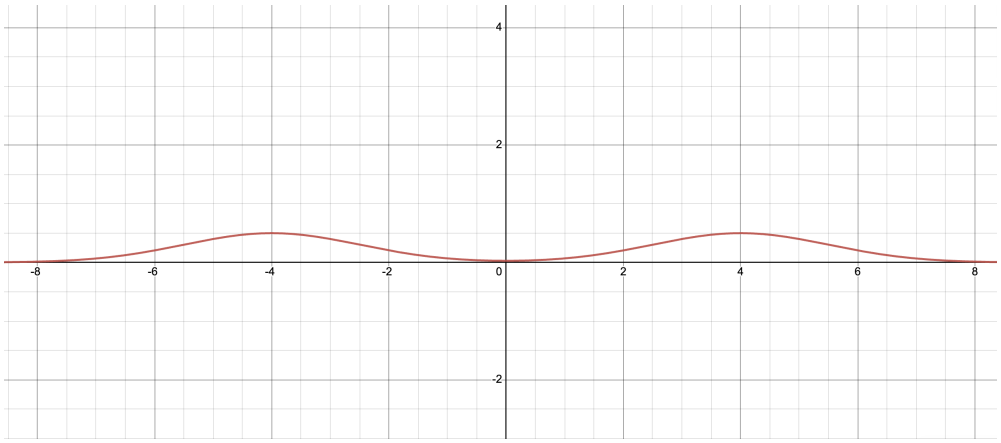

Fig. 2. The target distribution [5].

We set $\beta = 0.5$; our source distribution hence becomes $\mathcal{BG}(\mathbf{z};-4,4,\frac{1.5}{0.5},\frac{1}{2},\frac{1}{2})$. This distribution is visualized in Fig. 3. The normalization factor to convert this into a probability distribution is also omitted. Finally, with an empirically determined value $M = 2$, the distribution enforced by the rejection sampler $\frac{\mathcal{BG}(\mathbf{z};-4,4,1.5,\frac{1}{2},\frac{1}{2})}{M.\mathcal{BG}(\mathbf{z};-4,4,\frac{1.5}{0.5},\frac{1}{2},\frac{1}{2})}$ as visualized in Fig. 4 is very close to the target distribution. Fig. 5 captures the required number of trials (maximum possible trials being 100000[14]) for a sample to be output by $\frac{\mathcal{BG}(\mathbf{z};-4,4,1.5,\frac{1}{2},\frac{1}{2})}{M.\mathcal{BG}(\mathbf{z};-4,4,\frac{1.5}{0.5},\frac{1}{2},\frac{1}{2})}$. As evident, not only is the sampler capable of generating the correct distribution in *constant* number of retries, but also in preventing generation of samples from *incorrect* distributions. In other words, our rejection sampler, by design, provides *verification* of the sample distribution before it can be operated upon.

[14] In a realistic setting, no such upper bound will be placed. Hence, the sampler effectively ceases all operations until $\mathbf{z}$ belongs to the correct distribution.

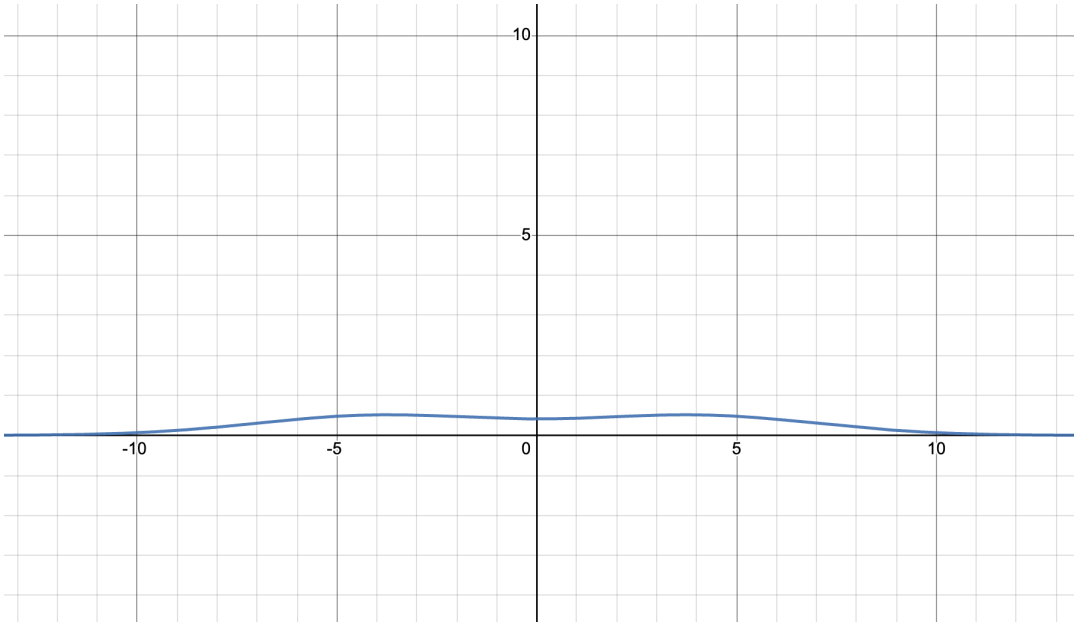

Fig. 3. The source distribution [5].

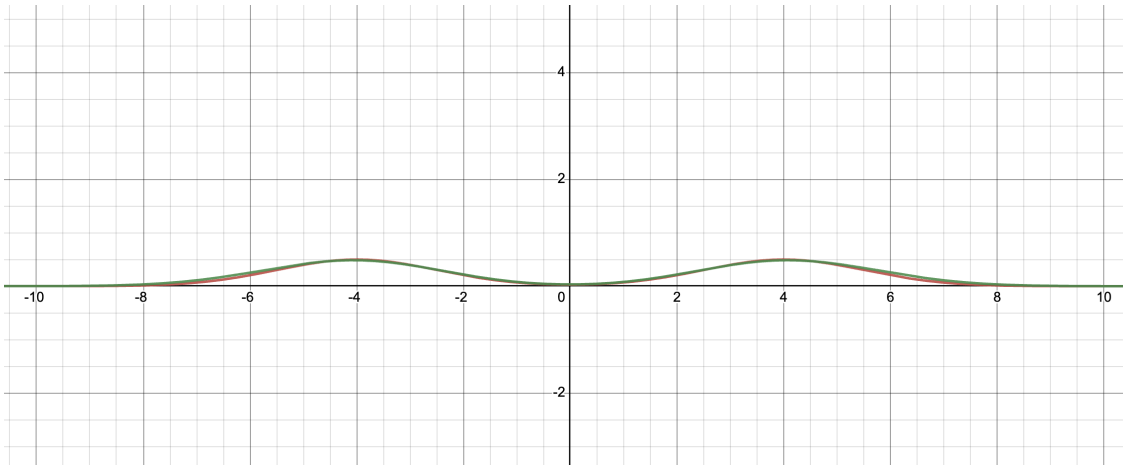

Fig. 4. The distribution enforced by the rejection sampler (in green), superimposed over the target distribution (in red) [5].

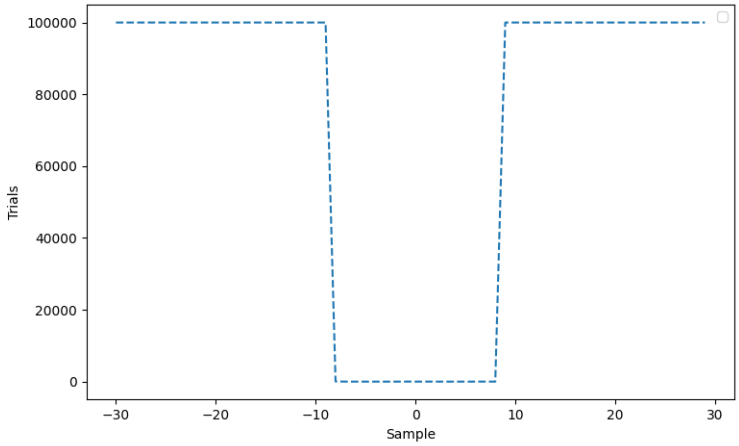

Fig. 5. The number of trials needed by the rejection sampler $\frac{\mathcal{BG}(\mathbf{z};-4,4,1.5,\frac{1}{2},\frac{1}{2})}{M.\mathcal{BG}(\mathbf{z};-4,4,\frac{1.5}{0.5},\frac{1}{2},\frac{1}{2})}$ to output a certain sample (the experiment aborts upon hitting a maximum of 100000 trials).

4 Case Studies

We now capture the specifics of applying probabilistic decryptions to reference implementations of Lattice-based KEMs, and by extension, preventing the known fault attacks on such instantiations.

4.1 Kyber

Kyber [3] is a Lattice-based KEM that closely follows the construction of LPR-styled public key encryption scheme, with an additional FO transform to achieve CCA security.

Structure and Differences with LPR . There are two main differences in Kyber's structure wrt. classic LPR construction. First, Kyber relies on Module Learning-with-Errors (MLWE). Informally, in Algo. 1, `a` is no longer a polynomial, rather a *square matrix* of polynomials in $R_q^{k\times k}$ for some module rank k. Likewise, `s`, `r`, `e`, $\mathtt{e}_1$, and $\mathtt{e}_2$ are column vectors of polynomials from R_q^k. Secondly, Kyber employs *lossy* ciphertext compression. Informally, for both `u` and `v` in Algo. 2, the most-significant bits d_u and d_v are (respectively) considered in Kyber's encapsulation and decapsulation routines. Hereafter, we abstract such differences wrt. LPR with the algorithm triple $\left\{$ `Kyber.KeyGen`, `Kyber.Encaps`, `Kyber.Decaps` $\right\}$. We note these differences bear no impact on the description and working of the rejection sampling, and refer to [3] for a detailed exposition.

Attacks on Kyber . Attacks on Kyber exploiting fault propagation have been explored in [23,4,11,10]. The attack in [23] (also summarized in Fig. 1) adds an additional error $\frac{q}{2}$ to the ciphertext, and exploits consequent effective/ineffective fault propagation to derive a system of inequations and leak the secret key. A similar attack principle is followed in [11] except for the fact that the error is introduced in the encapsulation oracle. This allows the adversary to have greater control over the faulted coefficient of the ciphertext, thereby allowing the strategy in [11] to resist countermeasures like *shuffling* that [23] cannot. Finally, the attacks in [11,10] follow the same attack principle, albeit with improvements in fault model used, exploitable fault locations, tractability of key recovery in presence of experimental setup hysterisis, and so on.

Kyber with Rejection Sampling . Fig. 6 captures the distributions of ciphertexts generated by `Kyber.Encaps` and the subsequent transformations performed in `Kyber.Decaps`. The lower horizontal line depicts the ciphertext space, while the upper horizontal line depicts the plaintext space. Within the semantics of our rejection sampling, it is clear that the expected distributions for Kyber can be approximated as a Bimodal about centers 0 and $\frac{q}{2}$. This leads to a natural extension of Theorem 2 as Corollary 2.

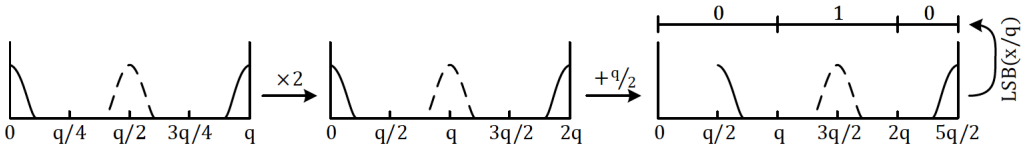

Fig. 6. Schematic of transformations performed during `Kyber.Decaps` [23].

Corollary 2 (Probabilistic Decryption in `Kyber.Decaps` using Rejection Sampling).

In `Kyber.Decaps` *[3], for centers* $\mathbf{c}_1 = 0$ *and* $\mathbf{c}_2 = \frac{q}{2}$ *and some* $\sigma, \beta \in \mathbb{R} : \beta < 1$*, let the following property hold, given some negligible function* $\epsilon(.)$*:*

$$Pr\Big[\frac{\mathcal{BG}(\mathbf{z}; 0, \frac{q}{2}, \sigma, \frac{1}{2}, \frac{1}{2})}{\mathcal{BG}(\mathbf{z}; 0, \frac{q}{2}, \frac{\sigma}{\beta}, \frac{1}{2}, \frac{1}{2})} \leq \mathcal{O}(1);\ \mathbf{z} \xleftarrow{\$} \mathcal{BG}(\mathbf{z}; 0, \frac{q}{2}, \sigma, \frac{1}{2}, \frac{1}{2})\Big] \geq 1 - \epsilon(.)$$

then the distribution of the following algorithm $\mathcal{M}$*:*

1. $\mathbf{z} \xleftarrow{\$} \mathcal{BG}(\mathbf{z}; 0, \frac{q}{2}, \sigma, \frac{1}{2}, \frac{1}{2})$
2. *Execute* `Kyber.Decaps`$(\mathbf{z})$

is within statistical distance $|\ 1 - \frac{1}{M}\ |$ *of the following algorithm* $\mathcal{F}$*:*

1. $\mathbf{z} \xleftarrow{\$} \mathcal{BG}(\mathbf{z}; 0, \frac{q}{2}, \sigma, \frac{1}{2}, \frac{1}{2})$
2. *With probability* $\frac{1}{M}$*, execute* `Kyber.Decaps`$(\mathbf{z})$

Which is, in turn, within statistical distance $\frac{\epsilon(.)}{M}$ *of the following algorithm* $\mathcal{A}$*:*

1. $\mathbf{z} \xleftarrow{\$} \mathcal{BG}(\mathbf{z}; 0, \frac{q}{2}, \frac{\sigma}{\beta}, \frac{1}{2}, \frac{1}{2})$
2. *With probability* $\texttt{min}\Big(1, \frac{\mathcal{BG}(\mathbf{z}; 0, \frac{q}{2}, \sigma, \frac{1}{2}, \frac{1}{2})}{M.\mathcal{BG}(\mathbf{z}; 0, \frac{q}{2}, \frac{\sigma}{\beta}, \frac{1}{2}, \frac{1}{2})}\Big)$*, execute* `Kyber.Decaps`$(\mathbf{z})$

Here, $M = \mathcal{O}(1)$ *is the expected number of tries for* $\mathcal{A}$ *to successfully execute* `Kyber.Decaps`*.*

We implement probabilistic decryption for Kyber in line with Corollary 2, and set parameters $q = 3329$[15], $\sigma = 1.5$[16], $\beta = 0.5$, and $M = 2$. When we repeat the attacks from [23,4,11,10], the rejection sampler reliably prevents fault propagation, thereby completely cutting off these attacks. Fig. 7 plots the number of retries needed for probabilistic `Kyber.Decaps` to execute. As evident, around 0 and $\frac{q}{2}$, the average number of retries is about 2. Elsewhere (specifically about $\frac{q}{4}$ where prior attacks operate), note that the number of retries has maxed out (at 100000) in our experimental setup. In a real-world scenario, `Kyber.Decaps` will essentially keep retrying with all but negligible probability. From all practical perspectives, the adversary does not observe fault propagation at all.

4.2 NewHope and Masked Kyber

The attacks in [15,10,23,11,4,9] also translate to NewHope and Masked Kyber. We omit the specifics of porting our rejection sampling to NewHope and Masked Kyber, by observing that the ciphertext distributions in NewHope (cf. Fig. 8) and Masked Kyber (cf. Fig. 9) can also be approximated by samplers similar to Kyber. In our experimentation, attacks on NewHope and Masked Kyber are also reliably prevented.

5 Extension to DFA: A Case Study of Dilithium

So far, we have focused on enabling probabilistic decryptions for Lattice-based KEMs, and established how cryptographically-backed guarantees could be given at countering prior attacks that exploit effective/ineffective propagation. In this section, we show the genericness of our approach by extending it to countering DFA based attacks on Dilithium [27,8,14]. To the best of our knowledge, this is the first work to offer defences against these attacks on Dilithium.

[15] In line with parameterization in [3]
[16] In line with parameterization in [3]

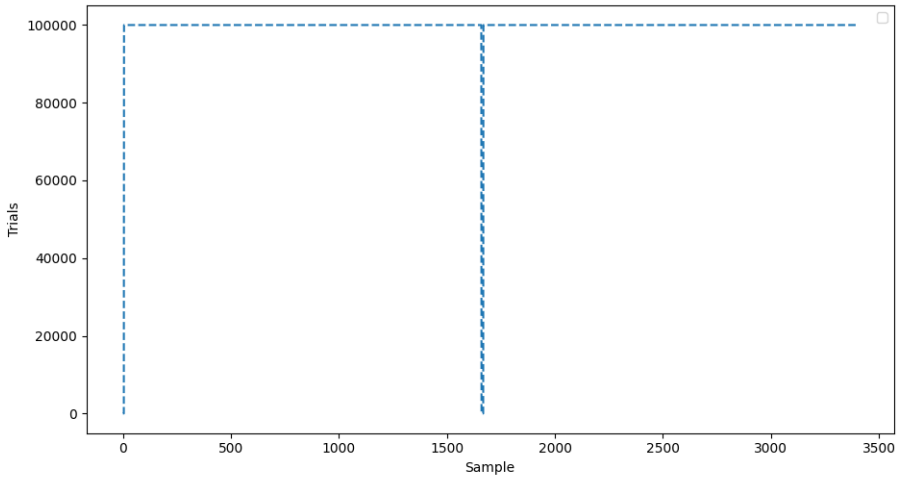

Fig. 7. Number of trials needed by `Kyber.Decaps` before generating output. For empirical evaluation, the number of retries is maxed at 100000.

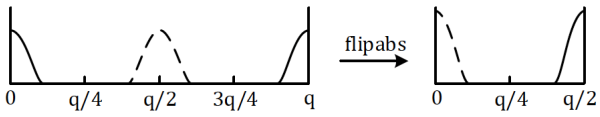

Fig. 8. Schematic of transformations performed during `NewHope.Decaps` [23].

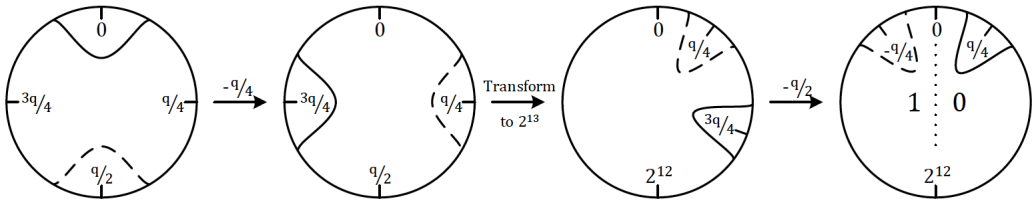

Fig. 9. Schematic of transformations performed during `MaskedKyber.Decaps` [23].

5.1 Attacks on Dilithium

Dilithium [18] is a Lattice-based instantiation in NIST standardization for digital signatures. Dilithium culminates a rich line of work [16,17,6] which rely on Fiat-Shamir transformation of interactive protocols (along with aborts) to construct primitives such as identification schemes, digital signatures, and so on.

The constructions in [16,17,6], as well as Dilithium [18], follow a similar foundation which attacks like [27,8,14] void and eventually exploit. Concretely, for some secret scalar $\mathtt{s}$ and some scalar *challenge* $\mathtt{c} \in \{-1, 0, 1\}$, the constructions sample some *blinding* scalar $\mathtt{y}$ and output $\mathtt{z} = \mathtt{y} + \mathtt{sc}$ iff $\mathtt{z} \in \mathcal{G}$ for some *safe* set $\mathcal{G}$. The set $\mathcal{G}$ varies from instantiation to instantiation, but ensures nothing useful about $\mathtt{sc}$ is leaked from $\mathtt{z}$. Extension of $\mathtt{z} = \mathtt{y} + \mathtt{sc}$ to vectors is straightforward and natural, and we refer to [16,17,6,18] for details. The fault attacks in [27,8,14] force the distribution of $\mathtt{y}$ to be skewed such that the output $\mathtt{z}$ from the construction belongs no longer to $\mathcal{G}$ (i.e. $\mathtt{z} \notin \mathcal{G}$). Eventually, this allows recovery of $\mathtt{s}$, for both settings where $\mathtt{c}$ is deterministic as well as randomized.

5.2 "Verifying" distribution of $\mathtt{y}$ through Rejection Sampling

Approximating $\mathtt{y}$. The first step towards extending our rejection sampler to Dilithium is to approximate the distribution of $\mathtt{y}$. From [18], it follows that a series of operations sample $v_j : j \in \{0, \cdots, 255\}$ from the set $\{0, \cdots, 2\gamma_1 - 2\}$. The coefficients of $\mathtt{y}$ are then sampled as $q + \gamma_1 - 1 - v_j : j \in \{0 \cdots 255\}$. Hereon, we refer the "safe" set $\mathcal{G} = q + \gamma_1 - 1 - v_j : j \in \{0 \cdots 2\gamma_1 - 2\}$[17] as the set of all coefficients of $\mathtt{y}$ that *hides* $\mathtt{sc}$ in $\mathtt{z} = \mathtt{y} + \mathtt{sc}$.

Definition 1. *For any $\mathtt{x} \in \mathbb{Z}$, the discrete probability distribution $D_{\mathcal{G}}$ approximates the sampling of coefficients of $\mathtt{y}$ in Dilithium's reference implementation:*

$$D_{\mathcal{G}} = \mathtt{Pr}_{\mathcal{G}}(\mathtt{x}) \begin{cases} \frac{1}{|\mathcal{G}|}, \mathtt{if\ x} \in \mathcal{G} \\ 0, \mathtt{otherwise} \end{cases}$$

In other words, without assuming further skewness or constraints over the sampling of coefficients of $\mathtt{y}$, we assume a uniform distribution over $\mathcal{G}$.

Approximating a "smooth" distribution $\widehat{D_{\mathcal{G}}}$ over $\mathtt{y}$. As with KEMs (cf. Sec. 3.2), the next step is to approximate a slightly "smoothed" distribution $\widehat{D_{\mathcal{G}}}$ of $\mathtt{y}$ which eventually becomes the source distribution for the sampler.

Definition 2. *For any $\mathtt{x} \in \mathbb{Z}$ and for the set $\mathcal{P} = \{-\alpha, \cdots, \alpha\}$, the "smoothed" discrete probability distribution $\widehat{D_{\mathcal{G}}}$ approximates the sampling of coefficients of $\mathtt{y}$ from the coset $\mathtt{p}.\mathcal{G}$ defined over the addition operator $+$:*

$$\widehat{D_{\mathcal{G}}} = \mathtt{Pr}_{\widehat{\mathcal{G}}}(\mathtt{x}) \begin{cases} \frac{1}{(2\alpha+1)|\mathcal{G}|}, \mathtt{if\ x} \in \mathtt{p}.\mathcal{G} \\ 0, \mathtt{otherwise} \end{cases}$$

Intuitively, for each integer $k \in \mathcal{G}$, $\widehat{D_{\mathcal{G}}}$ redistributes its probability mass to additional 2α integers *centered* about k. We now establish the rejection sampler for Dilithium formally.

[17] Both q and γ_1 are constants in Dilithium's parameterization.

Lemma 5. *For any* $\mathtt{y} \in \mathbb{Z}$*, the following holds for some negligible function* $\epsilon(.)$*:*

$$Pr\Big[\frac{D_{\mathcal{G}}(\mathtt{y})}{\widehat{D_{\mathcal{G}}}(\mathtt{y})} \leq \mathcal{O}(1);\ \mathtt{y} \xleftarrow{\$} D_{\mathcal{G}}\Big] \geq 1 - \epsilon(.)$$

Proof. We begin by remarking that the set $\mathcal{G}$ is a proper subset of $\mathtt{p}.\mathcal{G}$ for all $\alpha > 0$. As such, for all $\mathtt{y} \notin \mathcal{G}$, $D_{\mathcal{G}}(\mathtt{y}) = 0$ by definition and the upper bound is established trivially. For $\mathtt{y} \in \mathcal{G}$:

$$\frac{D_{\mathcal{G}}(\mathtt{y})}{\widehat{D_{\mathcal{G}}}(\mathtt{y})} = \frac{\frac{1}{|\mathcal{G}|}}{\frac{1}{(2\alpha+1)|\mathcal{G}|}} = (2\alpha + 1) \leq \mathcal{O}(1);\ \forall \alpha = \mathcal{O}(1)$$

It is straightforward to bound M thus, which abstracts the number of retries before output is produced.

Corollary 3. *For any* $\mathtt{y} \in \mathbb{Z}$ *and for some* $M = \mathcal{O}(1)$*, the following holds for some negligible function* $\epsilon(.)$*:*

$$Pr\Big[D_{\mathcal{G}}(\mathtt{y}) \leq M.\widehat{D_{\mathcal{G}}}(\mathtt{y});\ \mathtt{y} \xleftarrow{\$} D_{\mathcal{G}}\Big] \geq 1 - \epsilon(.)$$

By extension, $M \geq (2\alpha + 1)$*.*

Lemma 6. *Let* $D_{\mathcal{G}}$ *and* $\widehat{D_{\mathcal{G}}}$ *be probability distributions with the property that:*

$$Pr\Big[D_{\mathcal{G}}(\mathtt{y}) \leq M.\widehat{D_{\mathcal{G}}}(\mathtt{y});\ \mathtt{y} \xleftarrow{\$} D_{\mathcal{G}}\Big] \geq 1 - \epsilon(.)$$

then the distribution of the following algorithm $\mathcal{M}$

1. $\mathbf{y} \xleftarrow{\$} D_{\mathcal{G}}$
2. $\mathtt{PROCESS}(\mathbf{y})$

is within statistical distance $|\ 1 - \frac{1}{M}\ |$ *of the following algorithm* $\mathcal{F}$*:*

1. $\mathbf{y} \xleftarrow{\$} D_{\mathcal{G}}$
2. *With probability* $\frac{1}{M}$, $\mathtt{PROCESS}(\mathbf{y})$

Finally, the expected number of tries for $\mathcal{F}$ *to* $\mathtt{PROCESS}(\mathbf{y})$ *is* M*.*

Proof. The designs of $\mathcal{M}$ and $\mathcal{F}$ abstract operations of Dilithium's signing oracle. Concretely, after sampling $\mathbf{y}$ from the *safe* set $\mathcal{G}$, $\mathtt{PROCESS}(\mathbf{y})$ abstracts the sequence of operations used to generate the signature $\mathtt{z}$, perform additional checks as necessary, and output the signature. Intuitively, this lemma allows moving from a deterministic algorithm to a probabilistic one (but without the rejection sampling for now).

Let $\mathcal{M}(\mathbf{y})$ denote the distribution of $\mathcal{M}$ when operating upon $\mathbf{y}$; likewise for $\mathcal{F}$. Let $\mathcal{N}_{\mathcal{M}}$ denote the probability of $\mathcal{M}$ *not* operating upon $\mathbf{y}$; likewise for $\mathcal{F}$. We now bound the statistical distance between $\mathcal{M}$ and $\mathcal{F}$:

$$\Delta(\mathcal{M},\mathcal{F}) = \frac{1}{2}\Big(\sum_{\mathbf{y}\in\mathcal{G}} |\mathcal{M}(\mathbf{y}) - \mathcal{F}(\mathbf{y})| + |\mathcal{N}_{\mathcal{M}} - \mathcal{N}_{\mathcal{F}}|\Big)$$

Note that $\mathcal{M}$, by design, has no failure probability. Thus $\mathcal{N}_{\mathcal{M}} = 0$. We thus obtain:

$$\begin{aligned}
\Delta(\mathcal{M},\mathcal{F}) &= \frac{1}{2}\Big(\sum_{\mathbf{y}\in\mathcal{G}} \Big|D_{\mathcal{G}}(\mathbf{y}) - D_{\mathcal{G}}(\mathbf{y}).\frac{1}{M}\Big| + \Big|1-\frac{1}{M}\Big|\Big) \\
&\leq \frac{1}{2}\Big(\sum_{\mathbf{y}\in\mathcal{G}} D_{\mathcal{G}}(\mathbf{z})\Big|1-\frac{1}{M}\Big| + \Big|1-\frac{1}{M}\Big|\Big) \\
&= \frac{1}{2}\Big|1-\frac{1}{M}\Big|\Big(\sum_{\mathbf{y}\in\mathcal{G}} D_{\mathcal{G}}(\mathbf{z}) + 1\Big) \\
&= \frac{1}{2}\Big|1-\frac{1}{M}\Big|.2 = \Big|1-\frac{1}{M}\Big|
\end{aligned}$$

Intuitively, statistical distance bounds the difference between probabilities that algorithms $\mathcal{M}$ and $\mathcal{F}$ assign to the same event (i.e. $\texttt{PROCESS}(\mathbf{y})$). By design, $\mathcal{M}$ assigns probability 1, while $\mathcal{F}$ assigns probability $\frac{1}{M}$ for a single try; the difference between the two thus upper bounded by $\big|1-\frac{1}{M}\big|$.

Furthermore, *multiple* tries of $\mathcal{F}$ are *independent* of each other, thereby allowing to represent $\texttt{PROCESS}(\mathbf{z})$ as a binomial random variable (say $\mathbb{X}$). As such, the expected number of tries of the binomial variable $\mathbb{X}$ equal to the inverse of probability of success for a single try (i.e. M). To summarize, $\mathcal{F}$ performs $\texttt{PROCESS}(\mathbf{z})$ in expected M[18] trials.

Lemma 7. *Let $D_{\mathcal{G}}$ and $\widehat{D_{\mathcal{G}}}$ be probability distributions with the property that:*

$$Pr\Big[D_{\mathcal{G}}(\mathrm{y}) \leq M.\widehat{D_{\mathcal{G}}}(\mathrm{y});\ \mathrm{y} \xleftarrow{\$} D_{\mathcal{G}}\Big] \geq 1-\epsilon(.)$$

then the distribution of the following algorithm $\mathcal{F}$:

1. $\mathbf{y} \xleftarrow{\$} D_{\mathcal{G}}$
2. *With probability* $\frac{1}{M}$, $\texttt{PROCESS}(\mathbf{y})$

is statistically indistinguishable from the algorithm $\mathcal{A}$:

[18] This reinforces the need to bound $M = \mathcal{O}(1)$.

1. $\mathbf{y} \xleftarrow{\$} \widehat{D_{\mathcal{G}}}$
2. *With probability* $\texttt{min}(1, \frac{D_{\mathcal{G}}}{M.\widehat{D_{\mathcal{G}}}})$ $\texttt{PROCESS}(\mathbf{y})$

The probability for $\mathcal{A}$ *to* $\texttt{PROCESS}(\mathbf{y})$ *is* $\frac{1}{M}$

Proof. Note that the design of $\mathcal{A}$ abstracts the working of the rejection sampler for Dilithium, and encapsulates the main result of this section. Intuitively, this lemma allows moving from a "blind" probabilistic signing (i.e. $\mathcal{F}$) to a rejection sampling enabled signing, which can then be used to cryptographically "verify" distribution of $\mathbf{y}$. To prove statistically indistinguishability, we begin by bounding the probability of $\mathcal{A}$ to execute $\texttt{PROCESS}(\mathbf{y})$. We partition the set $\mathbb{Z}^n$ such that subset S contains all $\mathbf{y} \in \mathbb{Z}^n$ for which $Pr\Big[D_{\mathcal{G}}(\mathbf{y}) \leq M.\widehat{D_{\mathcal{G}}}(\mathbf{y});\ \mathbf{y} \xleftarrow{\$} D_{\mathcal{G}}\Big] \geq 1 - \epsilon(.)$ holds. We thus derive the following, after splitting this summation over $\mathbf{y}$ belonging to S and to $\mathbb{Z}^n \backslash S$ [16]:

$$Pr\Big[\mathcal{A}\ \texttt{performs}\ \texttt{PROCESS}(\mathbf{y})\Big] = \sum_{\mathbf{y} \in S} \widehat{D_{\mathcal{G}}}(\mathbf{y}) \,.\, \texttt{min}\Big(1, \frac{D_{\mathcal{G}}(\mathbf{y})}{M.\widehat{D_{\mathcal{G}}}(\mathbf{y})}\Big) + \sum_{\mathbf{y} \notin S} \widehat{D_{\mathcal{G}}}(\mathbf{y})$$

$$= \sum_{\mathbf{y} \in S} \frac{D_{\mathcal{G}}(\mathbf{y})}{M} + \sum_{\mathbf{y} \notin S} \widehat{D_{\mathcal{G}}}(\mathbf{y})$$

Where the first summation follows from the generic rejection sampling theorem (cf. Lemma 1). From Lemma 5, it is clear that the set $\mathbb{Z}^n \backslash S$ is null for the considered discrete distributions $D_{\mathcal{G}}$ and $\widehat{D_{\mathcal{G}}}$. We hence obtain:

$$Pr\Big[\mathcal{A}\ \texttt{performs}\ \texttt{PROCESS}(\mathbf{y})\Big] = \sum_{\mathbf{y} \in S} \frac{D_{\mathcal{G}}(\mathbf{y})}{M} = \frac{1}{M}$$

We now bound the statistical distance between $\mathcal{A}$ and $\mathcal{F}$. As with Lemma 6, let $\mathcal{A}(\mathbf{y})$ denote the distribution of $\mathcal{A}$ when operating upon $\mathbf{y}$; likewise for $\mathcal{F}(\mathbf{y})$. Likewise, let $\mathcal{N}_{\mathcal{A}}$ denote the probability of $\mathcal{A}$ not operating upon $\mathbf{y}$; likewise for $\mathcal{F}$. The statistical distance hence is given as:

$$\Delta(\mathcal{A}, \mathcal{F}) = \frac{1}{2}\Big(\sum_{\mathbf{y} \in \mathbb{Z}^n} |\mathcal{A}(\mathbf{y}) - \mathcal{F}(\mathbf{y})| + |\mathcal{N}_{\mathcal{A}} - \mathcal{N}_{\mathcal{F}}| \Big)$$

$$= \frac{1}{2}\Big(\sum_{\mathbf{y} \in S} \left| \widehat{D_{\mathcal{G}}}(\mathbf{y}) \,.\, \texttt{min}\Big(1, \frac{D_{\mathcal{G}}(\mathbf{y})}{M.\widehat{D_{\mathcal{G}}}(\mathbf{y})}\Big) - \frac{D_{\mathcal{G}}(\mathbf{y})}{M} \right| + \sum_{\mathbf{y} \notin S} \left| \widehat{D_{\mathcal{G}}}(\mathbf{y}) - \frac{D_{\mathcal{G}}(\mathbf{y})}{M} \right| + |\mathcal{N}_{\mathcal{A}} - \mathcal{N}_{\mathcal{F}}| \Big)$$

It follows from Lemma 5 that $D_{\mathcal{G}}$ is a proper subset of $\widehat{D_{\mathcal{G}}}$, and that the set $\mathbb{Z}^n \backslash S$ is null for the considered discrete distributions $D_{\mathcal{G}}$ and $\widehat{D_{\mathcal{G}}}$. Moreover, by Definition 1, $D_{\mathcal{G}}$ evaluates to 0 for any $\mathbf{y} \in \mathbb{Z}^n \backslash \mathcal{G}$. We hence reduce the above equation to:

$$\Delta(\mathcal{A},\mathcal{F}) = \frac{1}{2}\Big(\sum_{\mathbf{y}\in\mathcal{G}}\left|\widehat{D_{\mathcal{G}}}(\mathbf{y})\,.\,\mathtt{min}\Big(1,\frac{D_{\mathcal{G}}(\mathbf{y})}{M.\widehat{D_{\mathcal{G}}}(\mathbf{y})}\Big) - \frac{D_{\mathcal{G}}(\mathbf{y})}{M}\right| + |\mathcal{N}_{\mathcal{A}} - \mathcal{N}_{\mathcal{F}}|\Big)$$

Using Lemma 1, we reduce further to

$$\Delta(\mathcal{A},\mathcal{F}) = \frac{1}{2}\Big(\sum_{\mathbf{y}\in\mathcal{G}}\left|\frac{D_{\mathcal{G}}(\mathbf{y})}{M} - \frac{D_{\mathcal{G}}(\mathbf{y})}{M}\right| + |\mathcal{N}_{\mathcal{A}} - \mathcal{N}_{\mathcal{F}}|\Big) = \frac{1}{2}\,|\mathcal{N}_{\mathcal{A}} - \mathcal{N}_{\mathcal{F}}|$$

By definition, probability of $\mathcal{F}$ giving an output is $\frac{1}{M}$. Moreover, as established earlier, $Pr\Big[\mathcal{A}\ \mathtt{performs\ PROCESS}(\mathbf{y})\Big] = \frac{1}{M}$. We hence obtain:

$$\Delta(\mathcal{A},\mathcal{F}) = \frac{1}{2}\Big(\left|1 - \frac{1}{M}\right| - \left|1 - \frac{1}{M}\right|\Big)$$

Finally establishing

$$\Delta(\mathcal{A},\mathcal{F}) = 0$$

This concludes the proof. We now state the main result formally.

Theorem 3 ("Verification" in Dilithium using Rejection Sampling). *Let $D_{\mathcal{G}}$ and $\widehat{D_{\mathcal{G}}}$ be probability distributions with the property that:*

$$Pr\Big[D_{\mathcal{G}}(\mathtt{y}) \leq M.\widehat{D_{\mathcal{G}}}(\mathtt{y});\ \mathtt{y} \xleftarrow{\$} D_{\mathcal{G}}\Big] \geq 1 - \epsilon(.)$$

then the distribution of the following algorithm $\mathcal{M}$

1. $\mathbf{y} \xleftarrow{\$} D_{\mathcal{G}}$
2. $\mathtt{PROCESS}(\mathbf{y})$

is within statistical distance $|\ 1 - \frac{1}{M}\ |$ of the following algorithm $\mathcal{F}$:

1. $\mathbf{y} \xleftarrow{\$} D_{\mathcal{G}}$
2. *With probability* $\frac{1}{M}$, $\mathtt{PROCESS}(\mathbf{y})$

which is in turn statistically indistinguishable from the following algorithm $\mathcal{A}$

1. $\mathbf{y} \xleftarrow{\$} \widehat{D_{\mathcal{G}}}$
2. *With probability* $\mathtt{min}(1, \frac{D_{\mathcal{G}}}{M.\widehat{D_{\mathcal{G}}}})$ $\mathtt{PROCESS}(\mathbf{y})$

Here, $M = \mathcal{O}(1)$ is the expected number of tries for $\mathcal{A}$ to successfully execute $\mathtt{PROCESS}(\mathbf{y})$.

Proof. The proof follows from Lemma 5, Lemma 6, Lemma 7, and Corollary 3.

We implement verification in Dilithium using rejection sampling in line with Theorem 3 and set parameters $q = 8380417$[19], $\gamma_1 = \left(\frac{q-1}{16}\right)$[20], $\alpha = 1$, and $M \geq (2\alpha+1) = 3$. When we mount attacks from [27,8,14], our rejection sampler for Dilithium is able to completely stop skewed sampling of coefficients in $\mathtt{y}$, thereby stopping these attacks. The number of retries for an unfaulted execution is on average 3, while retries for faulted execution are effectively infinite (because of negligible probability of success).

6 Discussion

The main contribution of this work is a framework to convert a deterministic algorithm $\mathcal{M}$ (with a-priori knowledge of some distribution f) to its probabilistic variant $\mathcal{A}$, such that the latter can "verify" its execution on f. In other words, the probability of $\mathcal{A}$ executing on some distribution f^* (that is statistically distinguishable from f) is negligible. Moreover, the specific goals for this framework allow us to remodel rejection sampling in this context, where the number of retries for the samplers to generate output is constant (by carefully crafting the distributions of the sampler). We demonstrate the use of our framework to prevent fault attacks on Lattice-based Key-Encapsulation Mechanisms (including Kyber, NewHope, and Masked Kyber), as well as on Lattice-based digital signature construction Dilithium. By design of our framework and implementation optimizations of the distributions specific to KEMs and Dilithium, we achieve negligible overhead in our framework. Concretely, for unfaulted execution, the probabilistic variants of these PQC algorithms generate output in a constant number of retries $(2-3)$. However, when faults are injected, the number of retries grows unacceptably large for an adversary. We remark that, to the best of our knowledge, our framework provides the first cryptographically-backed and quantifiable defence strategy against recent fault attacks on KEMs and Dilithium.

We conclude by remarking that our framework is of independent interest for applications that require "verification" of complex distributions in the execution of their algorithms. Moreover, the design of the framework achieves this "verification" in an additional constant overhead (that is independent of the parameterization of the underlying construction). We believe that our framework's perspective on rejection sampling could lead to a richer class of constructions from a security viewpoint.

References

1. Alagic, G., Apon, D., Cooper, D., Dang, Q., Dang, T., Kelsey, J., Lichtinger, J., Miller, C., Moody, D., Peralta, R., et al.: Status report on the third round of

[19] In line with parameterization in reference implementation [7].
[20] In line with parameterization in reference implementation [7].

the nist post-quantum cryptography standardization process. US Department of Commerce, NIST (2022)
2. Apon, D., Howe, J..: Attacks on NIST PQC 3rd Round Candidates. https://iacr.org/submit/files/slides/2021/rwc/rwc2021/22/slides.pdf
3. Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé, D.: Crystals-kyber: a cca-secure module-lattice-based kem. In: 2018 IEEE European Symposium on Security and Privacy (EuroS&P). pp. 353–367. IEEE (2018)
4. Delvaux, J.: Roulette: A diverse family of feasible fault attacks on masked kyber. Cryptology ePrint Archive (2021)
5. Desmos.: Graphical Calculator. https://www.desmos.com/calculator
6. Ducas, L., Durmus, A., Lepoint, T., Lyubashevsky, V.: Lattice signatures and bimodal gaussians. In: Annual Cryptology Conference. pp. 40–56. Springer (2013)
7. Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: Crystals-dilithium: A lattice-based digital signature scheme. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 238–268 (2018)
8. ElGhamrawy, M., Azouaoui, M., Bronchain, O., Renes, J., Schneider, T., Schönauer, M., Seker, O., van Vredendaal, C.: From mlwe to rlwe: A differential fault attack on randomized & deterministic dilithium. IACR Transactions on Cryptographic Hardware and Embedded Systems **2023**(4), 262–286 (2023)
9. Fahr Jr, M., Kippen, H., Kwong, A., Dang, T., Lichtinger, J., Dachman-Soled, D., Genkin, D., Nelson, A., Perlner, R., Yerukhimovich, A., et al.: When frodo flips: End-to-end key recovery on frodokem via rowhammer. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. pp. 979–993 (2022)
10. Hermelink, J., Mårtensson, E., Samardjiska, S., Pessl, P., Rodosek, G.D.: Belief propagation meets lattice reduction: Security estimates for error-tolerant key recovery from decryption errors. Cryptology ePrint Archive (2023)
11. Hermelink, J., Pessl, P., Pöppelmann, T.: Fault-enabled chosen-ciphertext attacks on kyber. In: Progress in Cryptology–INDOCRYPT 2021: 22nd International Conference on Cryptology in India, Jaipur, India, December 12–15, 2021, Proceedings 22. pp. 311–334. Springer (2021)
12. Howe, J., Khalid, A., Martinoli, M., Regazzoni, F., Oswald, E.: Fault attack countermeasures for error samplers in lattice-based cryptography. In: 2019 IEEE International Symposium on Circuits and Systems (ISCAS). pp. 1–5. IEEE (2019)
13. Jiang, H., Zhang, Z., Chen, L., Wang, H., Ma, Z.: Post-quantum ind-cca-secure kem without additional hash. IACR Cryptol. ePrint Arch. **2017**, 1096 (2017)
14. Krahmer, E., Pessl, P., Land, G., Güneysu, T.: Correction fault attacks on randomized crystals-dilithium. Cryptology ePrint Archive (2024)
15. Kundu, S., Chowdhury, S., Saha, S., Karmakar, A., Mukhopadhyay, D., Verbauwhede, I.: Carry your fault: A fault propagation attack on side-channel protected lwe-based kem. arXiv preprint arXiv:2401.14098 (2024)
16. Lyubashevsky, V.: Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 598–616. Springer (2009)
17. Lyubashevsky, V.: Lattice signatures without trapdoors. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 738–755. Springer (2012)
18. Lyubashevsky, V., Ducas, L., Kiltz, E., Lepoint, T., Schwabe, P., Seiler, G., Stehlé, D., Bai, S.: Crystals-dilithium. Algorithm Specifications and Supporting Documentation (2020)

19. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29. pp. 1–23. Springer (2010)
20. Mondal, P., Kundu, S., Bhattacharya, S., Karmakar, A., Verbauwhede, I.: A practical key-recovery attack on lwe-based key-encapsulation mechanism schemes using rowhammer. arXiv preprint arXiv:2311.08027 (2023)
21. Oder, T., Schneider, T., Pöppelmann, T., Güneysu, T.: Practical cca2-secure and masked ring-lwe implementation. Cryptology ePrint Archive (2016)
22. Poppelmann, T., Alkim, E., Avanzi, R., Bos, J., Ducas, L., de la Piedra, A., Schwabe, P., Stebila, D., Albrecht, M., Orsini, E., et al.: Newhope-submission to the nist post-quantum cryptography standardization project. NIST National Institute of Standards and Technology (2019)
23. Prokop, L., Peßl, P.: Fault attacks on cca-secure lattice kems. IACR Transactions on Cryptographic Hardware and Embedded Systems **2021**(2), 37–60 (2021)
24. Ravi, P., Chattopadhyay, A., D'Anvers, J.P., Baksi, A.: Side-channel and fault-injection attacks over lattice-based post-quantum schemes (kyber, dilithium): Survey and new results. ACM Transactions on Embedded Computing Systems (2022)
25. Ravi, P., Roy, D.B., Bhasin, S., Chattopadhyay, A., Mukhopadhyay, D.: Number "not used" once-practical fault attack on pqm4 implementations of nist candidates. In: Constructive Side-Channel Analysis and Secure Design: 10th International Workshop, COSADE 2019, Darmstadt, Germany, April 3–5, 2019, Proceedings 10. pp. 232–250. Springer (2019)
26. Ravi, P., Yang, B., Bhasin, S., Zhang, F., Chattopadhyay, A.: Fiddling the twiddle constants-fault injection analysis of the number theoretic transform. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 447–481 (2023)
27. Ulitzsch, V.Q., Marzougui, S., Bagia, A., Tibouchi, M., Seifert, J.P.: Loop aborts strike back: Defeating fault countermeasures in lattice signatures with ilp. IACR Transactions on Cryptographic Hardware and Embedded Systems **2023**(4), 367–392 (2023)
28. Von Neumann, J.: Various techniques used in connection with random digits. John von Neumann, Collected Works **5**, 768–770 (1963)
29. Xagawa, K., Ito, A., Ueno, R., Takahashi, J., Homma, N.: Fault-injection attacks against nist's post-quantum cryptography round 3 kem candidates. In: Advances in Cryptology–ASIACRYPT 2021: 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6–10, 2021, Proceedings, Part II 27. pp. 33–61. Springer (2021)