A Masked Pure-Hardware Implementation of Kyber Cryptographic Algorithm

Tendayi Kamucheka, Alexander Nelson, David Andrews, and Miaoqing Huang
Department of Computer Science and Computer Engineering
University of Arkansas
Fayetteville, Arkansas 72701
{tfkamuch,ahnelson,dandrews,mqhuang}@uark.edu

***Abstract*—Security against side-channel assisted attacks remains a focus and concern in the ongoing standardization process of quantum-computer-resistant cryptography algorithms. Hiding and masking techniques are currently under investigation to protect the Post-Quantum Cryptography (PQC) algorithms in the NIST PQC standardization process against sophisticated side-channel attacks. Between hiding and masking, masking is emerging as a popular option due to its simplicity and minimized cost of implementation compared with hiding, which often requires duplication of hardware resources and advanced analysis and design techniques to implement correctly. This work presents a pure hardware implementation of masked CCA2-secure Kyber-512, a candidate chosen by NIST to be standardized. A novel hiding technique that leverages the advantages of FPGAs over micro-controllers and is demonstrably secure against Simple Power Analysis (SPA) and Differential Power Analysis (DPA) side-channel attacks is presented. Finally, a novel hybrid hiding-masking approach is presented that achieves a reduced hardware resource and clock-cycle penalty compared with previously reported figures for similar PQC candidates. The Test Vector Leakage Assessment (TVLA) is adopted to demonstrate the absence of side-channel leakage.**

***Keywords*—FPGA, Hiding, Masking, Post-Quantum Cryptography, Security**

I. Introduction

Quantum computing – specifically Shor's algorithm [1] – presents an existential threat to some standard cryptographic algorithms. To combat this, post-quantum cryptography (PQC) algorithms have been in development and are nearing mathematical and cryptanalytic maturity. Ongoing standardization efforts including the National Institute of Standards and Technology (NIST) PQC standardization process have chosen one PEK/KEMs algorithm (i.e., CRYSTALS-Kyber) and three digital signature algorithms (i.e., CRYSTALS-Dilithium, Falcon, and SPHINCS$^+$). CRYSTALS-Kyber [2] is a lattice-based key-encapsulation mechanism (KEM), an IND-CCA2-secure KEM based on the learning-with-errors problem over module lattices. Several FPGA hardware implementations of Kyber have been presented in the literature [3]–[6]. However, the robustness of these implementations to side-channel analysis (SCA) is unknown.

Side-channel leakage of private information – including private key material – is a consistent concern for robust cybersecurity. Otherwise cryptanalytically sound algorithms (e.g., AES, SHA) have been broken by side-channel attacks such as simple power analysis (SPA), differential power analysis [7], correlation power analysis [8], and machine learning [9]. Embedded devices that are provisioned with the selected PQC candidate may be deployed for dozens of years, and therefore should be hardened to these types of attacks. While no proof exists that can demonstrate full side-channel security, the Test Vector Leakage Assessment (TVLA) [10] can give an accurate representation of the likelihood that an implementation leaks private information.

Contributions: In this work, we present a masked pure hardware SCA-resistant implementation of CRYSTALS-Kyber on an FPGA. We compare our implementation's resources with a state-of-the-art hardware implementation to demonstrate differences in resource utilization to provide provable SCA resistance. Finally, we demonstrate that the implementation is SCA-resistant using the TVLA. In showing this, we also demonstrate the following:

- A novel approach to hiding using parallel processing and pipelining that proves effective and shows no side-channel leakage in testing. Our approach also results in significant savings in clock cycles compared with unprotected implementation.
- We apply hiding techniques to show overheads of 1.83x and 1.6x in clock cycles and hardware resources, respectively, compared with a baseline implementation of Kyber-512.
- We combine hiding and masking techniques to show an efficient design with only 1.08x and 1.06x overheads in clock cycles and hardware resources, respectively, compared with our hiding-only implementation to demonstrate a feasible low penalty masking technique.
- We verify the security of our approach with the TVLA on multiple power traces and show that it is within the bounds to have a 99.99999% probability of being SCA-resistant.
- All designs and data sets are open source and available at (link hidden for blind review).

II. Literature Review

In this section, we present a short review of the CRYSTALS-Kyber KEM, masking, the use of pipelining as a countermeasure against DPA side-channel attacks, and the effects of

TABLE I
PARAMETER SETS FOR THE THREE SECURITY LEVELS OF KYBER AND SIZES IN BYTES FOR PUBLIC KEY (PK), SECRET KEY (SK), AND CIPHERTEXT (CT).

Algorithm	Parameters (k,N,q)	PK, SK, CT (size in bytes)
Kyber-512	2, 256, 3329	800, 1632, 768
Kyber-768	3, 256, 3329	1184, 2400, 1088
Kyber-1024	4, 256, 3329	1568, 1568, 3168

measurement techniques on the success rates of DPA side-channel attacks. A good, more general review of the NIST-approved cryptography algorithms can be found in [11].

A. Notations

Let $q \in N$ be a prime and Z_q be the ring of integers modulo q. The ring of polynomials for some integer N is defined $\mathcal{R}_q = Z_q[x]/(x^N + 1)$. The polynomials have n coefficients, with each coefficient being modulo q. Regular font-weight lowercase letters (a) represent single polynomials, polynomial vectors boldface lowercase (**a**), and a matrix of polynomials boldface capitals (**A**). Multiplication in any ring is denoted using the $\cdot$ operator, and the $\oplus$ operator denotes bit-wise XOR.

B. CRYSTALS-Kyber

Kyber is a lattice-based IND-CCA2-secure KEM in the CRYSTALS suite, including the signature scheme Dilithium [12]. It guarantees security based on the hardness of the learning-with-errors (LWE) problem over module lattices [13]. The parameters for the different instantiations of Kyber, the polynomial length $n = 256$ and prime modulus $q = 3329$ are fixed. An additional parameter k determines the size of polynomial vectors and security level. e_i is a vector of noise polynomials sampled from the Centered Binomial Distribution. Table I lists all the parameters for the different security levels of Kyber. More rigorous mathematical descriptions of Kyber can be found here [2], [14].

C. Power Analysis

Power Analysis is a well-studied attack vector against cryptosystems. Measuring the current consumption of unprotected algorithms can reveal secret information about the algorithm's design and long secrets stored on the device.

Current consumption fluctuations resulting from charge and discharge cycles of transistors are observable within a 1% error margin with high sample rate oscilloscopes [15]. At the register level, the Hamming Weights of intermediate computational values are distinguishable with high accuracy. In the Hamming Weight model, observed power traces can be defined as $L = HW(op)+e$, where $HW(op)$ represents the Hamming Weight of the result of an operation, and e is noise present in the system caused by electromagnetic radiation emanating from other components present on the device. Power analysis techniques yield accurate results when the system noise is low.

Various techniques for power analysis have emerged over the years, namely, simple (SPA), Differential (DPA), and

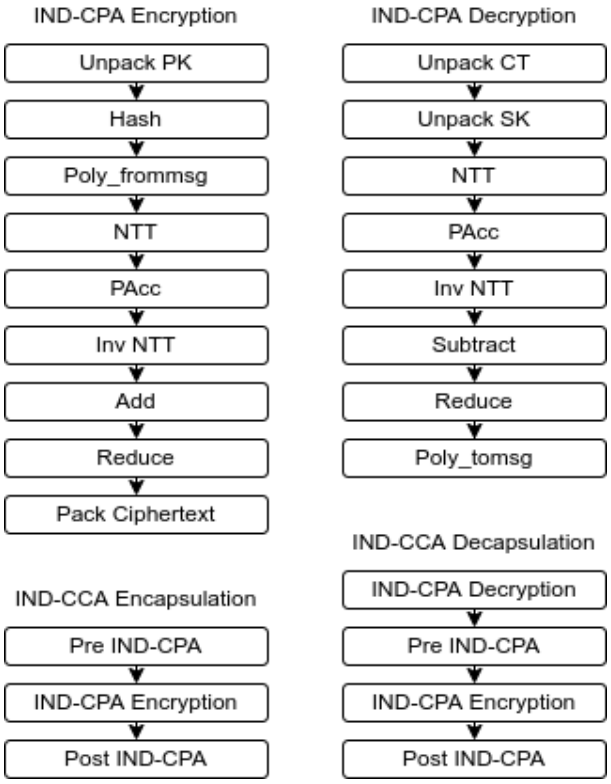


Fig. 1. Flowchart of Kyber KEM subroutines.

Correlation (CPA) Power Analysis. SPA is typically the simplest to implement. It involves analyzing the traces of current measurements collected by an oscilloscope. A trace here is the series of current values sampled over a period. The different stages of an unprotected program are identifiable without much analysis. Code branching related to inputs in unsafe implementations is often the easiest target as it reveals much about the program flow given some input by the user. The most effective countermeasures against SPA have been constant-time execution and avoiding conditional branch execution. Additionally, shortening the trace length of a routine is reportedly successful [15].

When the fluctuations in the traces are more subtle, more complex techniques like DPA are required. DPA operates on the differences between averages of two datasets separated by a random partitioning function. The datasets take known fixed and random inputs, respectively. When the random inputs are correct, a correlation function is needed to highlight correlations between the two datasets. When the inputs are correlated, the result of the correlation function is higher in magnitude. Uncorrelated inputs have a resulting magnitude closer to zero [15], [16]. Popular countermeasures against DPA often involve introducing noise in the form of randomness to the intermediate values such that correlation functions cannot reveal any correlations between the long-term secret key and malicious random inputs.

Heinz and Poppelman demonstrated that using a naive

redundant number representation is unsafe against DPA. They presented an updated efficient implementation that guaranteed the randomness of intermediate values, which was provably secure against DPA. They used the TVLA to demonstrate reduced side-channel leakage [16].

Pundir et al. presented a framework for power side-channel analysis. They highlight Masking, Threshold Implementation, and manipulating the signal-to-noise ratio (SNR) by introducing noise into the system to suppress signal information in power traces as successful countermeasures against DPA for AES and post-quantum candidate Saber [17].

D. Masking

First-order masking is a technique that can be used to avoid directly handling the sensitive secret polynomial sk. Instead, two statistically independent polynomials sk_1 and sk_2, both the same length as sk, are used in computation such that the decryption result is, $x_1 = \mathbf{A} \cdot sk_1 + e_1$, and $x_2 = \mathbf{A} \cdot sk_2 + e_2$. The relationship between the message m, the decryption result, and x_1 and x_2 is such that $m = x_1 + x_2$ in arithmetic masking or $m = x_1 \oplus x_2$ in Boolean masking. From the perspective of a side-channel observer, sk_1, sk_2, and subsequently x_1, and x_2 should not reveal any information about the long-term secret sk. For our purposes sk_1 and sk_2 are generated using the random mask R [18] such that: $sk_1 = sk \oplus R$, and $sk_2 = R$.

Oder et al. [19] demonstrated the general application of the dual-rail masking technique for Ring-LWE implementations. They point out that all Arithmetic-to-Boolean (A2B) conversion algorithms take in power-of-2 coefficients as inputs [19], [20]. Previously, this had been a major challenge in implementing masking for algorithms like Kyber because of the prime modulus q compared to Saber [21] and other implementations with a power-of-2 modulus. Algorithm 1 shows Oder et al.'s routine for converting coefficients to power-of-2 during the message decode routine *Poly_tomsg*. The routine MSB returns the most significant bit of the input coefficient and A2B is the arithmetic-to-binary (A2B) conversion of Debraize [22]. Q is the fixed prime modulus parameter of Kyber. The implementation shown is verified to be safe against side-channel power analysis attacks and is later applied to other PQC implementations [20].

Pessl and Prokop showed the application of masking techniques of Oder et al. to Kyber and NewHope. They demonstrated their implementation was resistant against fault injection through clock glitches and power analysis side-channel attacks on a micro-controller. Algorithm 2 shows a masked decode routine as adapted by Pessl and Prokop for Kyber [19], [20].

Our implementation extends the work of Oder et al. and Pessl and Prokop. Specifically, our work provides an inexpensive hardware implementation of the dual-rail masking techniques which were previously implemented in software and executed on the Cortex-M4 micro-controller for Kyber-512. In addition, we modify and optimize the initial software designs for hardware by leveraging the architectural benefits of the FPGA fabric. These modifications then lead to a new proposed efficient pipelined approach to masking, along with other hiding techniques. Our implementation is directly applicable to Kyber-{768 and 1024} without modification.

Algorithm 1 TransformPower2 routine converts coefficients to power-of-2 for A2B conversion [19].

procedure TRANSFORMPOWER2(x_1, x_2)
- $y_1 \leftarrow \{0,1\}^{16}$
- $y_2 \leftarrow x_1 - y_1$
- $y_2 \leftarrow y_2 + x_2$
- $z_1 \leftarrow y_1 - Q$
- $[z_1, z_2] \leftarrow$ A2B(z_1, y_2)
- $k_1 \leftarrow$ MSB(z_1) $\oplus 1$
- $k_2 \leftarrow$ MSB(z_2)
- $k_1' \leftarrow \{0,1\}^{16}$
- $k_1'' \leftarrow k_1 - k_1'$
- $k_2' \leftarrow \{0,1\}^{16}$
- $k_2'' \leftarrow k_2 - k_2'$
- $r \leftarrow \{0,1\}^{16}$
- $c_1 \leftarrow ((((((r + y_1) - k_1Q) - k_2Q) + 2k_1'k_2'Q) + 2k_1'k_2''Q) + 2k_1''k_2'Q) + 2k_1''k_2''Q)$
- $c_2 \leftarrow (y_2 - \{0,1\}^{16})$
- **return** c_1, c_2

end procedure

Algorithm 2 Masked Decode *(Poly_tomsg)* [20].

procedure MDECODE(c_1, c_2)
- $c_1 \leftarrow c_1 - \lfloor \frac{Q}{4} \rfloor$
- $[y_1, y_2] \leftarrow$ TransformPower2($c1, c2$)
- $y_1 \leftarrow y_1 - \lfloor \frac{Q}{2} \rfloor$
- $[y_1', y_2'] \leftarrow$ A2B(y_1, y_2)
- $m_1 \leftarrow$ MSB(y_1')
- $m_2 \leftarrow$ MSB(y_2')
- **return** m_1, m_2

end procedure

E. Pipelining as a Power Analysis Countermeasure

Standaert et al. [23] demonstrated a successful correlation power analysis attack on an FPGA. It was observed that when carefully applied, pipelining can be an effective countermeasure against DPA. While the outer stages of the pipeline remain partially predictable, the inner stages effectively function as noise generators. Depending on the exact implementation, the noise may not be sufficient to rule out the possibility of a successful DPA. Still, it is undeniable that it reduces the correlation between predictions and measurements.

Additional experimental results show that loop unrolling can further increase the robustness of pipelining as a DPA countermeasure. Observations also showed that resetting the internal state of registers results in leakage and therefore should be avoided for secure implementations [23].

We consider both observations made by Standaert et al. First, our hiding-only and hiding-plus-masking implementations do not reset internal registers between operations. Second, we use a combination of loop unrolling as a hiding

technique and efficient pipelining as a countermeasure. Our findings are reported in section IV.

F. Measurement Techniques

According to Mazur and Novotny, while experimenting on AES, the success rate of DPA depends on the measurement setup. Removing decoupling capacitors which tend to filter the output signal, plays a significant role in improving the success rate of the DPA. In addition, replacing switched-mode power supplies with linear power supplies improves the odds of success [24]. However, DPA is still possible without either modification while requiring more power traces.

Sun, Yan, and Zambreno highlight a significant amount of power analysis research on FPGAs has been theoretical, simulated [23], or experimental using customized hardware far removed from what would be found in real-world systems [25]–[28]. Our work is verified through practical experimentation, and we probe the target FPGA device without any physical change to the circuitry in our measurement setup.

G. Related Works

Several implementations exist for CRYSTAL-Kyber on the FPGA. Many of them have focused on improving area efficiency. However, provably secure designs remain few. Most published works on securing Kyber have been on the microcontroller [3], [29]–[32].

Jati et al. have presented an area-efficient design of Kyber secure against fault injection and power analysis on the FPGA. The authors argued that masking and other previously known techniques for protecting against side-channel-assisted attacks are expensive, so they proposed alternatives to guarantee security. Their design significantly reduced the area occupied by hashing functions using a common SHA3 core to perform all SHA3-(256,512) and SHAKE256 routines. They also used custom hashing functions based on a stripped-down version of AES-128 to detect fault injections. Jati et al. also used Random clock delays, address randomization, and instruction randomization to protect against side-channel assisted attacks [6].

The random clock delays would misalign power traces, making calculating any correlations in the time domain difficult. Address randomization shuffles coefficients with each round of execution, making using correlations functions difficult. Their instruction randomization, similar to out-of-order execution, would also further misalign coefficients on power traces. No experimental validation was given to demonstrate side-channel resistance. Nevertheless, the proposed countermeasures are sound.

III. Design & Implementation

A. Design

The implemented design combines both hiding and masking to improve side-channel resistance. Two designs were implemented for this work. The first used only hiding techniques and the second combined masking with hiding techniques. Results are presented showing the resistance against DPA for both implementations.

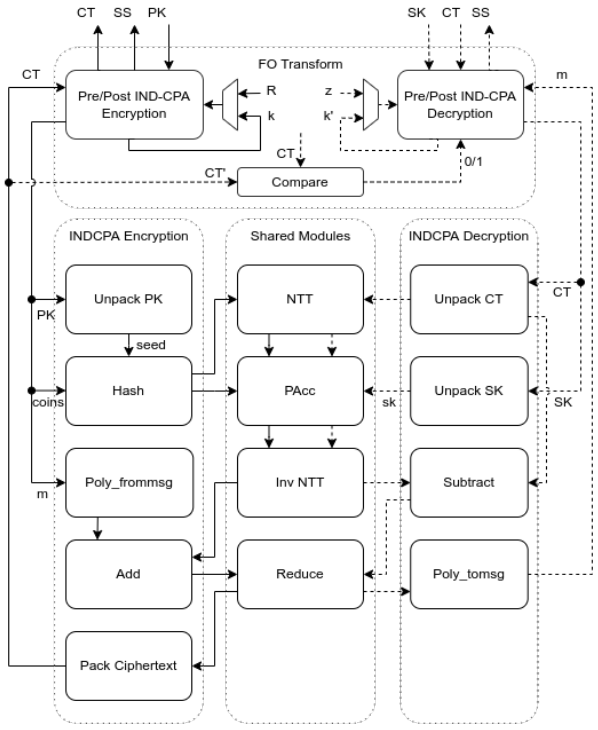


Fig. 2. Kyber KEM Encapsulation and Decapsulation Architecture. Solid and dashed lines show data flow in encapsulation and decapsulation modes respectively.

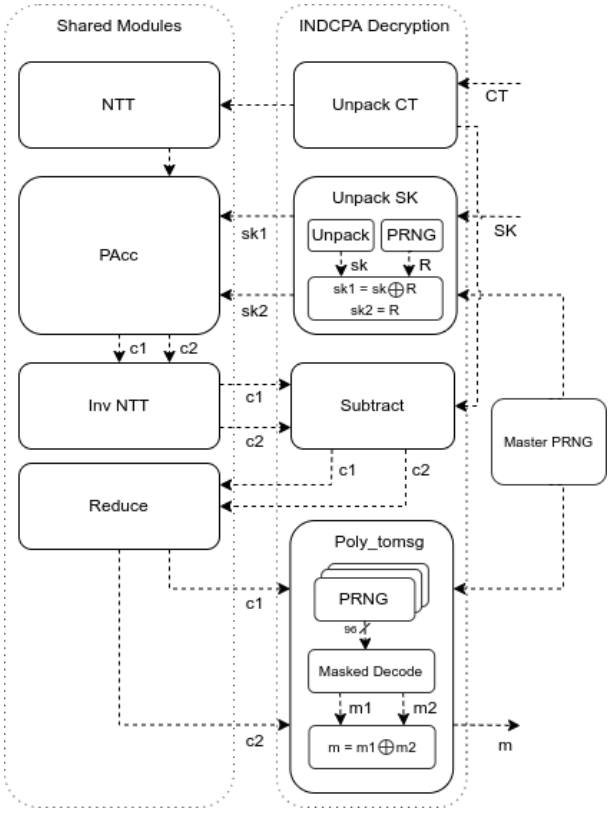


Fig. 3. Dual-rail first-order masking implementation for Kyber decapsulation.

Algorithm 3 Secure branch execution in Fujisaki-Okamoto transform.

if $c = c'$ **then**
    $SS \leftarrow \mathrm{H}(k'||H(c'))$
else
    $SS \leftarrow \mathrm{H}(z||H(c'))$
end if

1) Hiding Techniques: Power analysis attacks are dependent on direct correlations between underlying computations and observed device power draw. For SPA attacks, computations of individual coefficients of the secret vector sk are directly observed. In some cases, observations can be seen with the naked eye on power traces collected while the design is running. For this reason, implementations securing against such attacks avoid, for example, code branch executions dependent on the intermediate values and or directly handling the coefficients of the secret vector (masking).

The code snippet shown in Algorithm 3 demonstrates how the Fujisaki-Okamoto transformation maintains constant-time execution in the event of decryption failure by producing an incorrect shared key SS. To an attacker attempting a decryption oracle attack, [33], the observed power trace is not directly distinguishable between that of successful decryption and the production of correct SS. The attacker, in this case, would require further knowledge about the expected output to know that a decryption failure has occurred; this would require more than a single trace. The hiding techniques implemented rely on parallel processing and pipelining as security measures. Both these approaches prevent the processing of secret coefficients independently.

Parallel processing. The goal of this approach is to prevent coefficients from being handled independently. This technique is applied primarily to the 'Unpack SK' and polynomial addition and subtraction modules. Each of these modules processes up to eight arithmetic operations in parallel per clock cycle. For an attacker observing the power trace, any observed changes in the traces are the result of eight parallel operations [23].

For SPA attacks that rely on differences in the traces based on the hamming weights of the inputs and the difference between non-zero and partial zero inputs, the power trace offers no information about individual coefficients. Similarly, for DPA that relies on statistical techniques to identify leakages, no information is leaked about any single coefficient [23]. Therefore, we consider this approach to be secure for both attack methods. However, this approach is not without penalty since additional hardware resources are required. On the other hand, a coefficient vector of 256 elements can be processed efficiently in 32 clock cycles only.

Pipelining. Repeatedly used modules such as the Montgomery and Barrett reduction are protected using pipelining. Listing 1 shows the relevant Verilog code for a pipelined Montgomery reduction. Both Barrett and Montgomery reduction modules are implemented with a 4-stage pipeline to reduce the initialization interval from 4 to 1. A shift register is required to forward the input coefficient, which is used in the first and last stages of the pipeline. Reducing the initialization interval also has the benefit of significantly improving the throughput. When processed sequentially, a 256-element coefficient vector is processed in 1024 clock cycles while pipelining requires only 259. This implementation can further be optimized when synthesized to efficient DSP resources.

```
localparam K = 4;

reg [K-1:0] state = 0;

assign reduce_done = state[K-1];

always @(posedge clk)
  state <= {state[K-2:0], ce};

always @(posedge clk) begin
  a <= iCoeffs_a;
  u <= a * $signed(KYBER_QINV);
  t <= u * $signed(KYBER_Q);
  oCoeffs <= $signed(fifo_out - t) >>> 16;
end

Shift_Reg #(.MSB(K-1)) FIFO[31:0] (
  .clk(clk),
  .din(iCoeffs_a),
  .dout(fifo_out)
);
```

Listing 1. Code snippet showing pipelined Montgomery Reduction. K is the number of pipeline stages. KYBER_Q and KYBER_QINV are fixed constants.

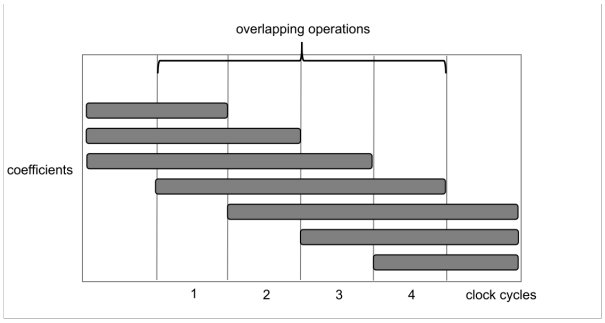


Fig. 4. The figure shows overlapping operations in Barrett and Montgomery modular reduction operations. The overlap in operation makes observing Hamming Weights of intermediate values difficult.

Pipelining is applied in 21 pipeline stages to a larger module *Poly_tomsg* in IND-CPA decryption. Without pipelining, the *Poly_tomsg* operation requires 6656 clock cycles to complete. With pipelining and masking applied, the operation can be reduced to 274 clock cycles only.

Considering the security of pipelining as a hiding technique, in any k-stage pipeline, only the first and last $k-1$ coefficients are exposed at the beginning and end of the operation. The number of potentially vulnerable clock cycles for a given operation with an input of N coefficients is $2k-2$. The remaining $N-k-1$ clock cycles effectively function as a noise generator. The noise generation is a result of processing multiple coefficients per clock cycle. The instantaneous current

consumption is uncorrelated to any one coefficient, or set of registers [23].

The basis of the security of pipelining is similar to that of parallel processing. Both techniques rely on hiding by processing multiple coefficients per clock cycle.

2) Masking Techniques: The dual rail masking technique is applied to the IND-CPA decryption algorithm as shown in Figure 3. The implementation follows a pipelined version of Algorithm 2. A Pseudo-Random Number Generator (PRNG) is added to generate the random values $R = \{0,1\}^{11}$. The masked secret vectors are produced as $sk_1 = sk \oplus R$ and $sk_2 = R$. We use only random numbers $R < 2^{12}$ because larger values resulted in decryption errors. The remainder of the decapsulation process, as shown in Figure 1, is unchanged.

A series of PRNGs are used to generate the randomness required to implement the masking algorithm. A suitable PRNG core[1] with good statistical properties was chosen [34]. One caveat to the hiding techniques described earlier, used in conjunction with masking, is that operations requiring randomness will require multiple random numbers per clock cycle. In the case of parallel processing, the PRNGs required are as many as half the coefficients processed in parallel. The relationship depends on the number of stages requiring random numbers in pipelining. These observations become limiting factors for the degree of parallelism achievable and an unavoidable overhead cost of pipelining.

Requiring multiple random numbers while working with PRNGs presents a set of challenges. First, PRNGs by nature repeat their output sequence unless re-seeded. Second, the PRNGs have identical outputs when seeded with the same seed. The PRNGs are organized in a master-slave arrangement to overcome these challenges and guarantee randomness. A single master PRNG IP, re-seeded with each round of execution and instantiated outside the Kyber-512 IP, supplies a stream of random seeds to the slave PRNG cores within the Kyber-512 IP. Seed buffers are placed close to the slave PRNGs. The size of the buffers is proportional to the number of PRNG IPs in the module, which is half the number of the random number required per clock cycle. Only half the number of generators is required because each slave 32-bit PRNG produces 2 16-bit random numbers. The slave PRNG cores internally have unique fixed random seeds as a guarantee for randomness.

B. Setup

We use the Xilinx Virtex-7 FPGA platform for our testing and implementation. The hiding-plus-masking Kyber-512 CCAKEM IP is instantiated together with a separate PRNG IP Core. The Kyber-512 core is clocked at 100 MHz. We reuse common modules as much as possible to reduce the utilization of hardware resources. In addition, for the hiding-plus-masking implementation we only duplicate smaller modules and serialize larger modules like the NTT core. We do this to achieve a good trade-off between hardware resource usage and additional clock cycles. A Tektronix MDO 3 series oscilloscope is used for capturing power traces from the FPGA development board. A probe is attached to the current output of the VCCAUX_IO and VCCBRAM power rails. We expected this probe placement closely mimics measuring current across a shunt as done in similar studies on the Cortex M4 microcontroller boards [18]. Xilinx Vivado 2019.1 and Xilinx SDK 2019.1 are used to generate the IP cores and writing the bitstream to the FPGA development board. The control program running on the Microblaze processor on the FPGA implements an interface with the Simple Serial version 1.1 of the ChipWhisperer platform. This choice offers a convenient communication and control interface API when capturing traces on the host PC. On the host PC, a Python script communicates with the oscilloscope to automate trace capturing over the USB VISA interface. Captured traces are transferred from the oscilloscope directly over the USB connection. The script also arms the oscilloscope and waits for a trigger signal from the FPGA board to trigger the trace capture on the oscilloscope. On the FPGA board, we use the XADC GPIO pins to output the necessary capture signals which are connected to the data probes on the oscilloscope.

TABLE II
FPGA-HOST PC COMMUNICATION UART API.

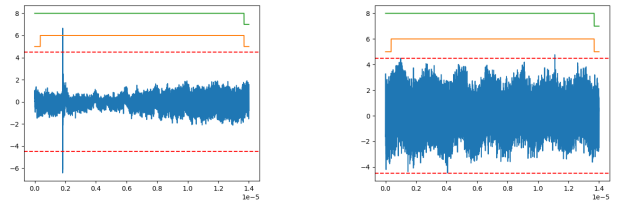
command	prefix	payload	bytes
device_reset	x	-	0
device_enable	e	-	0
set_mode	m	operation mode	1
set_rand_bytes_ciphertext	r	random bytes	32
set_public_key_secret_key	k	public key	800
set_rand_bytes_ciphertext	c	ciphertext	736
set_public_key_secret_key	s	secret key	1632
get_ciphertext	t	ciphertext	736
get_shared_secret	a	shared secret	32

C. Evaluation

The Test Vector Leakage Assessment (TVLA) has been put forward as a standard leakage assessment test. The test has known limitations, including risks of false positives and negatives. It is tempting to consider the TVLA a pass/fail security test. However, it is vital not to overestimate the meaning of the result. We use the TVLA only to measure side-channel leakage emanating from correlations of inputs.

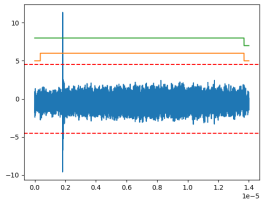
The test measures and evaluates side-channel leakage while maintaining a separation between the implementation and the device under test [10]. The test itself is a non-specific statistical test highlighting variations in power and EM traces attributed to some leakage of sensitive data over multiple traces. In general, independent leakage is assumed [35]. TVLA computes the univariate Welch's t-test for each point on a trace by the following equation: $TVLA = (\mu_r - \mu_f)/\sqrt{\sigma_r^2/n_r + \sigma_f^2/n_f}$, where μ_r, σ_r, and n_r are the mean, standard deviation, and number of traces collected for T_r; and likewise for trace set T_f. The two sets T_r and T_f represent the sets of traces collected with random inputs and fixed inputs, respectively.

[1]We acknowledge that incorporating a true random number generator with better statistical properties would likely improve the security of our masking implementation. We leave that as an exercise outside the scope of this work.

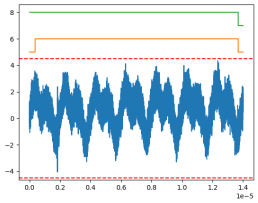


(a) VCCAUX_IO (VCC)

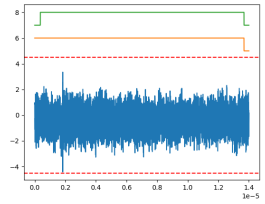
(b) VCCBRAM (VCC)



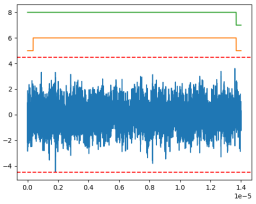
(c) VCCAUX_IO (I)



(d) VCCBRAM (I)

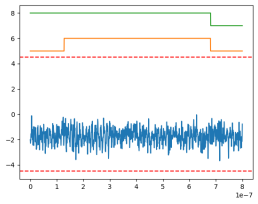


(e) VCCAUX_IO (GND)

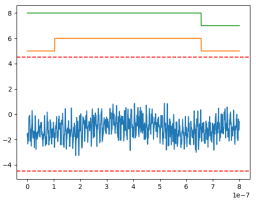


(f) VCCBRAM (GND)

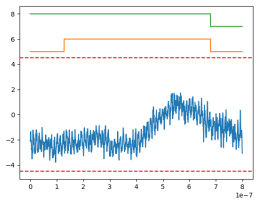
Fig. 5. Text vector leakage assessment of 40,000 fixed vs. random traces for **hiding-only** Kyber-512 on VC707. The x and y-axes show time and t-values respectively. (5a) Result of traces captured from the VCC terminal of the VCCAUX_IO power rail. (5b) Result of traces captured from the VCC terminal of the VCCBRAM power rail. (5c) Result of traces captured from the I (current) terminal of the VCCAUX_IO power rail. (5d) Result of traces captured from the I (current) terminal of the VCCBRAM power rail. (5e) Result of traces captured from the GND terminal of the VCCAUX_IO power rail. (5f) Result of traces captured from the GND terminal of the VCCBRAM power rail.



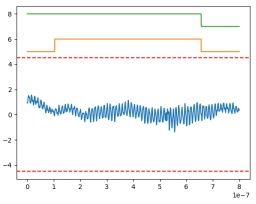
(a) VCCAUX_IO (VCC)



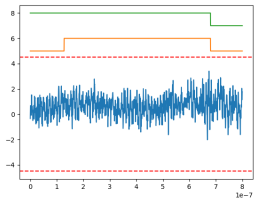
(b) VCCBRAM (VCC)



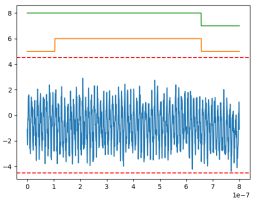
(c) VCCAUX_IO (I)



(d) VCCBRAM (I)



(e) VCCAUX_IO (GND)



(f) VCCBRAM (GND)

Fig. 6. Text vector leakage assessment of 40,000 fixed vs. random traces for **hiding-plus-masking** Kyber-512 on VC707. The x and y-axes show time and t-values respectively. (6a) Result of traces captured from the VCC terminal of the VCCAUX_IO power rail. (6b) Result of traces captured from the VCC terminal of the VCCBRAM power rail. (6c) Result of traces captured from the I (current) terminal of the VCCAUX_IO power rail. (6d) Result of traces captured from the I (current) terminal of the VCCBRAM power rail. (6e) Result of traces captured from the GND terminal of the VCCAUX_IO power rail. (6f) Result of traces captured from the GND terminal of the VCCBRAM power rail.

In our case, the inputs are a ciphertext and secret key pair run through Kyber's decryption algorithm. A pass or fail decision is given for each timestep on the trace by testing for a null hypothesis, such that of the two sets T_r and T_f are equivalent. The null hypothesis is rejected with a confidence level of 99.99999% if the absolute value of the t-test is greater than 4.5. The confidence level and t-test threshold are known values documented in literature for the TVLA test [36]. For each point on the trace, a rejected null hypothesis is a fail, suggesting the presence of some exploitable leakage from the underlying computation. We use the TVLA, first to identify leakage on traces measured on the oscilloscope, and second to confirm the elimination of said leakage after the application of masking.

IV. Results And Analysis

The focus of the results of TVLA presented is on the magnitude of t-values. Higher t-values point towards a correlation between the known fixed inputs and random inputs, while lower values indicate a lower correlation. The t-test fails when the t-values are larger than the 4.5 threshold. The result of the test vector leakage assessment on the $Poly\_tomsg$ routine is shown in Figure 5 and Figure 6. Traces were captured from the VCC, I, and GND terminals of the VCCAUX_IO and VCCBRAM power rails. Although we present the result of 40,000 traces, we also collected results from 10 and 20 thousand traces.

Figure 5 shows the result of the hiding-only Kyber-512. The hiding-only Kyber-512 failed TVLA at one point only on the TVLA test on all traces collected from VCCAUX_IO. The traces collected from VCCBRAM also showed leakage at different points in traces collected from VCC and GND. The traces collected from VCC and I rails are more revealing of the algorithm. Altogether the traces show fairly high t-values but are not conclusive to point out significant leakage. We conclude that the hiding-only countermeasure does offer protection against power analysis.

Figure 6 shows the result of traces captured on hiding-plus-masking Kyber-512. The $Poly\_tomsg$ routine in the

TABLE III
FPGA RESOURCE UTILIZATION AND CLOCK CYCLE COUNTS.

Implementation	Algorithm	Process	FPGA	Freq (MHz)	LUTs	Slices	DSPs	BRAMs	Div IPs	Cycle Counts
This work (Hiding-only)	Kyber-512	Encaps	Virtex7 VC707	100	153,939	107,804	53	264	0	88,176
		Decaps	Virtex7 VC707	100	143,112	81,746	60	294	0	126,619
This work (Hiding-plus-masking)	Kyber-512	Encaps	Virtex7 VC707	100	163,584	119,324	56	392	0	88,176
		Decaps	Virtex7 VC707	100	152,860	92,977	76	489.5	0	137,738
Huang et al. [3]	Kyber-512	Encaps	Artix7 AC701	155	80,322	141,825	54	200.5	2	49,015
		Decaps	Artix7 AC701	155	88,901	152,875	354	202	3	68,815
Huang et al. [3]	Kyber-768	Encaps	Artix7 AC701	155	97,085	153,867	36	200.5	2	77,481
		Decaps	Artix7 AC701	155	110,260	167,293	292	202	3	102,113
Huang et al. [3]	Kyber-1024	Encaps	Virtex7 VC707	192	119,189	162,636	36	200.5	2	107,054
		Decaps	Virtex7 VC707	192	132,918	172,489	548	202	3	135,553
Xing and Li [32]	Kyber-512	Encaps	Artix7 AC701	161	7,412	2126	2	3	-	5,100
		Decaps	Artix7 AC701	161	7,412	2126	2	3	-	6,700
Xing and Li [32]	Kyber-768	Encaps	Artix7 AC701	161	7,412	2126	2	3	-	7,900
		Decaps	Artix7 AC701	161	7,412	2126	2	3	-	10,000
Xing and Li [32]	Kyber-1024	Encaps	Artix7 AC707	161	7,412	2126	2	3	-	11,300
		Decaps	Artix7 AC707	161	7,412	2126	2	3	-	13,900
Jati et al. [6]	Kyber-512	Encaps	Artix7 XC7A35T-2	258	7,151	-	2	-	-	43,300
		Decaps	Artix7 XC7A35T-2	258	7,151	-	2	-	-	47,700
Jati et al. [6]	Kyber-768	Encaps	Artix7 XC7A35T-2	258	7,151	-	2	-	-	43,300
		Decaps	Artix7 XC7A35T-2	258	7,151	-	2	-	-	47,700
Jati et al. [6]	Kyber-1024	Encaps	Artix7 XC7A35T-2	258	7,151	-	2	-	-	43,300
		Decaps	Artix7 XC7A35T-2	258	7,151	-	2	-	-	47,700

hiding-plus-masking implementation is pipelined such that the measured traces show the result measured over 274 clock cycles compared to the 6656 for unmasked hiding-only results. The latency of the hiding-only routine is 0.014 ms, while the hiding-plus-masked is only 0.0008 ms. The masked traces are, therefore, significantly shorter. We observed smaller t-values in all the masked results, except for the VCCBRAM GND results. Additionally, the traces do not show any patterns relatable to the algorithm. The small t-values resulted from the pipelining and masking of the coefficients. We consider this evidence enough that our combined pipeline and masking approach is an effective countermeasure against power analysis.

We make two observations. First, the hiding-plus-masking implementation passes the TVLA at all points; No side-channel leakage was detected. Second, an overall reduced magnitude of the t-values in the leakage assessment results compared to the unmasked hiding-only results. The values sampled during trace collection came from the current consumption of 21 pipeline stages. On the other hand, four samples were captured per clock cycle for every coefficient in the unmasked traces. The smaller values indicate a lesser correlation between fixed and random inputs than the hiding-only traces. Moreover, smaller t-values overall suggest improved side-channel resistance against DPA.

We also show the utilization of FPGA resources and total clock cycles for hiding-only and hiding-plus-masking Kyber-512 in Table III. Our base design was inefficient and resource intensive. However, our goal is to highlight the cost of our masking approach, considering masking has been previously dismissed as too expensive to implement on the FPGA. A similar overhead factor can be expected when applied to more efficient designs. The cost of masking has previously been reported at 2x. We observe an overhead of 1.08x in terms of the clock cycles compared to the hiding-only design. Much of the savings come from pipelining. We also report an overhead of 1.06x for hardware resources compared to the hiding-only design—the savings result from reusing common hardware modules in the design. We duplicated the hardware resources for smaller modules to process the sk_1 and sk_2 in parallel. In the case of larger modules, we run the modules twice for sk_1 and then sk_2.

V. CONCLUSIONS AND FUTURE WORK

In this work, we demonstrated the first masked pure-hardware side-channel analysis resistant implementation of the CRYSTALS-Kyber post-quantum cryptography key-establishment mechanism. Further, we showed how the techniques did not greatly impact resource utilization and performance, while guaranteeing over 99.99999% probability of security against SCA attacks. We achieved these metrics using a novel combination of masking and hiding through pipelining operations in regions of the algorithm that utilize secret material.

Future Work: To extend the work presented in this paper, we intend to demonstrate these techniques applied to the other lattice-based KEM finalists, as well as the higher-bit security implementations of Kyber (Kyber-768 and Kyber-1024).

ACKNOWLEDGMENT

This work was supported in part by NIST Award 60NANB20D016. We also like to thank Daniel Apon (NIST and MITRE) for his help on project discussion and manuscript writing.

REFERENCES

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[2] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehle, "CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM," in *Proceedings - 3rd IEEE European Symposium on Security and Privacy, EURO S and P 2018*. Institute of Electrical and Electronics Engineers Inc., jul 2018, pp. 353–367.

[3] Y. Huang, M. Huang, Z. Lei, and J. Wu, "A pure hardware implementation of crystals-kyber pqc algorithm through resource reuse," *IEICE Electronics Express*, pp. 17–20 200 234, 2020.

[4] Y. Xing and S. Li, "A compact hardware implementation of cca-secure key exchange mechanism crystals-kyber on fpga," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 328–356, 2021.

[5] V. B. Dang, K. Mohajerani, and K. Gaj, "High-speed hardware architectures and fpga benchmarking of crystals-kyber, ntru, and saber," *Cryptology ePrint Archive*, 2021.

[6] A. Jati, N. Gupta, A. Chattopadhyay, and S. K. Sanadhya, "A configurable crystals-kyber hardware implementation with side-channel protection," Cryptology ePrint Archive, Report 2021/1189, 2021, https://ia.cr/2021/1189.

[7] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Annual international cryptology conference*. Springer, 1999, pp. 388–397.

[8] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *International workshop on cryptographic hardware and embedded systems*. Springer, 2004, pp. 16–29.

[9] H. Maghrebi, T. Portigliatti, and E. Prouff, "Breaking cryptographic implementations using deep learning techniques," in *International Conference on Security, Privacy, and Applied Cryptography Engineering*. Springer, 2016, pp. 3–26.

[10] T. Schneider and A. Moradi, "Leakage assessment methodology," *Journal of Cryptographic Engineering*, vol. 6, no. 2, pp. 85–99, 2016.

[11] H. Nejatollahi, N. Dutt, S. Ray, F. Regazzoni, I. Banerjee, and R. Cammarota, "Post-quantum lattice-based cryptography implementations: A survey," *ACM Comput. Surv.*, vol. 51, no. 6, jan 2019. [Online]. Available: https://doi.org/10.1145/3292548

[12] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS-Dilithium : A Lattice-Based Digital Signature Scheme," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2018, no. 1, pp. 238–268, feb 2018. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/839

[13] O. Regev, "The learning with errors problem," *Invited survey in CCC*, vol. 7, no. 30, p. 11, 2010.

[14] R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "Crystals-kyber algorithm specifications and supporting documentation," *NIST PQC Round*, vol. 2, p. 4, 2017.

[15] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology — CRYPTO' 99*, M. Wiener, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 388–397.

[16] D. Heinz and T. Pöppelmann, "Combined fault and dpa protection for lattice-based cryptography," Cryptology ePrint Archive, Paper 2021/101, 2021, https://eprint.iacr.org/2021/101. [Online]. Available: https://eprint.iacr.org/2021/101

[17] N. Pundir, J. Park, F. Farahmandi, and M. Tehranipoor, "Power side-channel leakage assessment framework at register-transfer level," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2022.

[18] M. V. Beirendonck, J.-P. D'anvers, A. Karmakar, J. Balasch, and I. Verbauwhede, "A side-channel-resistant implementation of saber," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 17, no. 2, pp. 1–26, 2021.

[19] T. Oder, T. Schneider, T. Pöppelmann, and T. Güneysu, "Practical cca2-secure and masked ring-lwe implementation," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 142–174, 2018.

[20] P. Pessl and L. Prokop, "Fault attacks on cca-secure lattice kems," Cryptology ePrint Archive, Report 2021/064, 2021, https://eprint.iacr.org/2021/064.

[21] J.-P. D'Anvers, A. Karmakar, S. S. Roy, and F. Vercauteren, "Saber. proposal to nist pqc standardization, round2, 2019."

[22] B. Debraize, "Efficient and provably secure methods for switching from arithmetic to boolean masking," in *Cryptographic Hardware and Embedded Systems – CHES 2012*, E. Prouff and P. Schaumont, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 107–121.

[23] F.-X. Standaert, S. B. Örs, and B. Preneel, "Power analysis of an fpga," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2004, pp. 30–44.

[24] L. Mazur and M. Novotnỳ, "Differential power analysis on fpga board: Boundaries of success," in *2017 6th Mediterranean Conference on Embedded Computing (MECO)*. IEEE, 2017, pp. 1–4.

[25] H. Guntur, J. Ishii, and A. Satoh, "Side-channel attack user reference architecture board sakura-g," in *2014 IEEE 3rd Global Conference on Consumer Electronics (GCCE)*. IEEE, 2014, pp. 271–274.

[26] M. Matsubayashi, A. Satoh, and J. Ishii, "Clock glitch generator on sakura-g for fault injection attack against a cryptographic circuit," in *2016 IEEE 5th Global Conference on Consumer Electronics*. IEEE, 2016, pp. 1–4.

[27] Y. Nomata, M. Matsubayashi, K. Sawada, and A. Satoh, "Comparison of side-channel attack on cryptographic cirucits between old and new technology fpgas," in *2016 IEEE 5th Global Conference on Consumer Electronics*. IEEE, 2016, pp. 1–4.

[28] S. Sun, Z. Yan, and J. Zambreno, "Experiments in attacking fpga-based embedded systems using differential power analysis," in *2008 IEEE International Conference on Electro/Information Technology*. IEEE, 2008, pp. 7–12.

[29] P. Ravi, R. Poussier, S. Bhasin, and A. Chattopadhyay, "On configurable sca countermeasures against single trace attacks for the ntt - a performance evaluation study over kyber and dilithium on the arm cortex-m4," Cryptology ePrint Archive, Paper 2020/1038, 2020, https://eprint.iacr.org/2020/1038. [Online]. Available: https://eprint.iacr.org/2020/1038

[30] A. Abdulrahman, V. Hwang, M. J. Kannwischer, and D. Sprenkels, "Faster kyber and dilithium on the cortex-m4," Cryptology ePrint Archive, Paper 2022/112, 2022, https://eprint.iacr.org/2022/112. [Online]. Available: https://eprint.iacr.org/2022/112

[31] L. Botros, M. J. Kannwischer, and P. Schwabe, "Memory-efficient high-speed implementation of kyber on cortex-m4," in *International Conference on Cryptology in Africa*. Springer, 2019, pp. 209–228.

[32] Y. Xing and S. Li, "A compact hardware implementation of cca-secure key exchange mechanism crystals-kyber on fpga," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2021, no. 2, p. 328–356, Feb. 2021. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8797

[33] Y. Qin, C. Cheng, and J. Ding, "An efficient key mismatch attack on the nist second round candidate kyber," *Cryptology ePrint Archive*, 2019.

[34] T. E. Tkacik, "A hardware random number generator," in *International Workshop on Cryptographic hardware and embedded systems*. Springer, 2002, pp. 450–453.

[35] M. V. Beirendonck, J.-P. D'Anvers, and I. Verbauwhede, "Analysis and comparison of table-based arithmetic to boolean masking," Cryptology ePrint Archive, Report 2021/067, 2021, urlhttps://eprint.iacr.org/2021/067.

[36] B. J. Gilbert Goodwill, J. Jaffe, P. Rohatgi *et al.*, "A testing methodology for side-channel resistance validation," in *NIST non-invasive attack testing workshop*, vol. 7, 2011, p. 115–136.