# Bitstream Fault Injection Attacks on CRYSTALS Kyber Implementations on FPGAs

Ziying Ni*, Ayesha Khalid*, Weiqiang Liu† and Máire O'Neill*
Email: {zni03,a.khalid}@qub.ac.uk, liuweiqiang@nuaa.edu.cn, m.oneill@ecit.qub.ac.uk.
* Centre for Secure Information Technologies (CSIT), Queen's University Belfast, Belfast, UK
†College of Electronics and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, China

***Abstract*—CRYSTALS-Kyber is the only Public-key Encryption (PKE)/ Key-encapsulation Mechanism (KEM) scheme that was chosen for standardization by the National Institute of Standards and Technology initiated Post-quantum Cryptography competition (so called NIST PQC). In this paper, we show the first successfully malicious modifications of the bitstream of a Kyber FPGA implementation. We successfully demonstrate 4 different attacks on Kyber hardware implementations on Artix-7 FPGAs that either reduce the complexity of polynomial multiplication operations or enable direct secret key/ message recovery by: disabling BRAMs, disabling DSPs, zeroing NTT ROM and tampering with CBD2 results. Two of our attacks are generic in nature and the other two require reverse-engineering or a detailed knowledge of the design. We evaluate the feasibility of the four attacks, among which the zeroing NTT ROM and tampering with the CBD2 result attacks produce higher public key and ciphertext complexity and thus are difficult to be detected. Two countermeasures are proposed to prevent the attacks proposed in this paper.**

***Index Terms*—Post-quantum Cryptography (PQC), Lattice-based Cryptography (LBC), CRYSTALS-Kyber, FPGA security, bitstream fault injection, bitstream fault injection.**

## I. Introduction

Post-quantum cryptography (PQC) is the next generation of public key cryptographic solution, as it is believed to be capable of resisting attacks from quantum computers. The threat to the currently deployed public key cryptography arises from the significance computing capabilities of quantum computers; decomposition algorithms designed for quantum computers such as Shor's algorithm [1] and Grover's algorithm [2]. NIST commenced the standardization process for PQC in 2016 [3] and, by 2022, announced the first four algorithms that will be standardized in the near future. CRYSTALS-Kyber (hereafter called Kyber) [4] is the only key encapsulation mechanism (KEM) protocol among these four chosen algorithms.

Kyber is a lattice-based cipher (LBC) that derives its security from the module learning with errors (MLWE) problem, offering a more robust algebraic structure compared to traditional lattice ciphers based on the Ring-LWE (RLWE) problem. Despite its inherent algorithmic security, potential vulnerabilities may arise during physical implementation due to side-channel and fault attacks, which can compromise critical information or even the secret key [5], [6]. In 2018, Ravi $et\ al.$ successfully reported launching the first fault attack on Kyber on the ARM Cortex processor [7]. By skipping the instruction controlling the central binomial distribution (CBD) sampling counter, they made the secret key and noise polynomials become exactly equal, resulting in weakening Kyber's security to access the secret key and recover message $m$. In 2023, Ravi $et\ al.$ attacked the Read-Only Memory (ROM) in the Number theoratic transform or the NTT module on an ARM processor by setting all twiddle factors to zero [8]. This attack too was able to substantially undermine the security of Kyber and successfully recover message $m$.

Hardware implementations for Field Programmable Gate Arrays (FPGAs) store their complete designed circuit configuration in files known as *bitstreams*. One approach to fault injection attacks on FPGAs is to maliciously modify the bitstream [9]–[11]. In 2015, Swierczynski $et\ al.$ attacked AES [12] by reverse engineering the data from the ROMs used in AES bitstream [9]. In 2018, Ziener $et\ al.$ attacked AES by disabling block RAM (BRAMs) [10]. In 2020, Moraitis $et\ al.$ attacked SNOW 3G [13] by modifying the bitstream of the Lookup Table (LUT)-based implementation of the XOR gate [11]. These attacks targeted Virtex-5 or earlier FPGAs that carried out AES encryption of the bitstream without any CRC checksum, making bitstream tampering more easy to go unnoticed. In 7 series FPGAs, AES encryption as well as CRC checks and a HMAC are included when checking the integrity of the bitstream, making tampering easier to detect. In 2020, Ender $et\ al.$ found that a MultiBoot address register named WBSTAR in the FPGAs is not cleared during a reset, so the bitstream can be tampered with to write a 32-bit portion of the bitstream into the WBSTAR register. Since the decrypted bitstream is written at this time, the whole decrypted bitstream can be restored by multiple operations in sequence [14]. Consequently, it is possible to completely recover the non-encrypted bitstream in 7 series FPGAs in a matter of hours; in Artix-7 FPGA devices, which are a popular choice for several reported Kyber implementations, it takes roughly an hour to fully recover the non-encrypted bitstream. Therefore, our attack assume a fully decrypted bitstream is available. In addition, we thank Xing $et\ al.$ for generously open-sourcing their code [15].

To the best of the authors' knowledge, this is the first bitstream attack on the hardware design of a PQC such as Kyber. We investigate four schemes of bitstream attacks targeting the vulnerabilities of the Kyber accelerator: disabling BRAM, disabling DSP, zeroing NTT ROM, and manipulating

This work is grants from the Engineering and Physical Sciences Research Council (EPSRC) Quantum Communications Hub (EP/T001011/1), and supported by the National Natural Science Foundation of China (62022041) and the Fundamental Research Funds for the Central Universities (NP2022103).

979-8-3503-4859-0/DATE24/© 2024 EDAA

centered-2 binomial distribution sampling (CBD2) sampling. Through these attacks, we demonstrate that an adversary can fully recover the message $m$ by exploiting weaknesses during the key generation and key encapsulation phases, respectively. Moreover, we conduct a comparative analysis of these four attacks. Among them, the public key $pk$/ ciphertext $ct$ obtained by CBD2 sampling attack is difficult to detect whether it is under attack. In addition, CBD2 sampling successfully recovers the message $m$ without interfering with the recovery process observed by both communicating parties.

## II. Kyber KEM Protocol

Kyber builds its chosen-plaintext security (CPA)-secure PKE using a MLWE problem and converts it to the chosen-ciphertext attack (CCA)-secure KEM protocol using the Fujisaki-Okamoto transform [4]. Assume that the two parties in the Kyber protocol are Alice and Bob. During the key generation ($KeyGen.$) phase, Alice utilizes a random number $d$ to derive the matrix $A$, the secret key polynomial $s$, and the noise polynomial $e$. Subsequently, she computes the public key $pk = Encode_{12}(\hat{t}||\rho) = Encode_{12}((\hat{A} \circ \hat{s} + \hat{e})||\rho)$, where $\rho$ represents the random number calculated from SHA3 function $G(d)$. Alice then sends the $pk$ to Bob to finish the $KeyGen.$ phase. After receiving $pk$, Bob start the key encapsulation ($Encaps.$) phase. He regenerates the matrix $A^T$ and the secret key polynomial $r$, then calculates $u = A^T \circ r + e_1$, as well as $v = t^T \circ r + e_2 + m$. Bob packs the ciphertext $ct_{Bob} = (u||v)$, along with the shared key $K_{Bob}$, and transmits them back to Alice. After receiving $ct_{Bob}$ and $K_{Bob}$ from Bob, Alice uses the secret key vector $s$ to compute $m = v - s^T \circ u$ to recover the original message $m$. Following this, under the CCA security protocol, Alice repeats the computation of the new ciphertext $ct_{Alice}$ and $K_{Alice}$ using the $pk$ and the recovered $m$, then compares them with the $ct_{Bob}$ and $K_{Bob}$ from Bob to complete the verification process. For more details of Kyber, the reader is kindly referred to [4].

## III. Key Units Bitstream Tampering Attack

In this section, we demonstrate two attack scenarios.

### A. Disabling BRAMs and DSPs in the Bitstream

The first attack disables customizable IP core components used in the Kyber design, i.e., **(1) disabling BRAM** and **(2) disabling DSPs** to allow an attacker to obtain a weak ciphertext to break the message. BRAMs are used to store and read data. In the Xilinx FPGAs, a single 36K BRAM block can act as two independent 18K BRAM blocks; if the 36K BRAM is disabled, then both 18K BRAMs are disabled. Since the input signals to BRAMs include clock, reset, enable, address, and input data; modifying either reset or enable specific bits in the bitstream can completely disable a BRAM. DSPs are used to perform multiplication and addition calculations. Kyber requires a 12-bit multiplication during polynomial multiplication (Number Theoretic Transform (NTT)/ Inverse-NTT (INTT), point-wise multiplication (PWM)) calculations, which are likely to be configured on DSP units of FPGAs for all security levels. Since the multiplication of 12 bits does not exceed the maximum bit-width a single DSP can process (18 $\times$ 25-bit), no cascading of multiple DSP blocks is needed in the Kyber accelerator. The DSP module inputs are, clock, enable, reset and two multiplication operands. As with BRAM blocks, either the enable or reset input of the DSP can be modified in the bitstream to set the null-operation.

To initiate the malicious modification of the bitstream attack, we experimentally find out the offsets of the enable signals ($en$) of each BRAM and the reset signal ($rst$) of the DSPs in the bitstream for the xc7a100t FPGA (Nexys 4) device. To find the BRAM offset, we instantiate all BRAM modules available on that FPGA and switch $en$ on the BRAMs to '1' and the '0' to get two different bitstreams. A bit by bit comparison of the two bitstreams is then carried out. The header of the bitstreams contain the compile time information of the current program and the CRC checksum, which is ignored during the comparison. For a single 36K BRAM there is a difference of 4 bits between specifying the $en$ signal being set or not. Using a similar approach, we find out how the two bitstreams differ with all DSP $rst$ signals enabled in one and disabled in the other. A DSP requires the modification of the only 1 bits in the bitstream to disable it.

### B. Tampering with LUT-based units in Bitstream

FPGA logic consists of configurable logic blocks or the CLBs, each of which contains two slices. Each SliceM (memory slice) or SliceL (logic slice) comprises 4 LUT cells, mapping the main logic and part of the storage function of the design. Kyber requires between 7,400-12,000 LUT cells on FPGAs. In 2015, Swierczynski $et\ al.$ constructed the truth table of the S-BOX in AES implementations and deduced the possible bitstream values of the LUTs used by the S-BOX by reverse engineering the bitstreams, then performed a matching search on the bitstreams, and finally zeroed out the bitstream values of the searched LUTs in order to successfully attack AES [9]. In this section, we will show how to modify the NTT ROM and CBD2 sampling units in a Kyber design to launch our attack.

*1) **Attack the distributed ROM**:* To launch this attack, the attacker needs to know the address of each LUT in the current FPGA in the bitstream. To do so, the attacker can use only a single LUT design, e.g. a full adder, then change the address of the LUTs using a constraint file, and finally find the associated bitstream block in the whole bitstream to obtain the address information of all LUTs in the FPGA. This does not mean that a single LUT design needs to be dragged over all LUTs to find the address. In any section of the FPGA, such as X0Y0 area, the order of the LUT in the bitstream is always from bottom to top, left to right, and each BRAM and DSP block between LUTs always occupy the same bitstream length. Therefore, the addresses of all LUTs in the entire FPGA can be quickly obtained through derivation.

Secondly, the attacker must know the shuffled ordering of the LUT mapping. In Xilinx FPGAs, a single LUT has 6 data input ports ($A1$, $A2$, $A3$, $A4$, $A5$, $A6$) and two outputs ($O5$, $O6$). In the bitstream, the output of a single LUT is determined

by a sequence of 64 bits, representing the truth table generated by the 6 inputs; when the output of the truth table is '1', a '1' is recorded in the 64-bit sequence, and vice versa. It should be noted that the contents of the truth table represented by the 64-bit sequence are not sequential, and the order can be referred to the Table I in [16]. The 128 twiddle factors in the Kyber NTT will have 7 bits of address ($tf\_Address$) data, so the highest bit of $tf\_Address$, instead of being connected to any LUT, will be used as a selection signal in the 2in1 MUX to pick the values of two LUTs. The lower 6 bits of the address are connected to two different LUTs, each of the two LUTs will determine one bit of the ROM output. Hence storing a 128 twiddle factor array, 12-bit wide, requires 24 LUTs, where each LUT requires a 6-input truth table. The first step is to extract the 64-bit truth table $TruthTable$ results (OutputBit). At this time the order of the truth table is natural binary encoded. Then, the natural binary encoding order of the truth table is converted to the order in the bitstream (OrderFromBitstream). It is necessary to find the reordered address using index $BitIndx$, as specified in Table I of [16]. Finally, the resulting truth table is generated as a 64-bit binary sequence (HexToBit).

As the design is synthesized and implemented on a device, the 6 ports of each LUT and their $tf\_Address$[0:5] do not linearly match. For example, after once synthesis, the six ports in the LUT may be mapped to the $tf\_Address$ as (2, 3, 4, 0, 1, 5), where 2 represents A0 connected to $tf\_Address$[2]; this order may completely change after the next synthesis, causing a completely different bitstream to be generated. Therefore, before calculating the OrderFromBitstream function, it is necessary to perform a BitRestrct function, in which we include the pairing information ($A1$, $A2$, $A3$, $A4$, $A5$, $A6$), and then construct the mapping $r$ of the default truth table ($TruthTable$). After that, $TruthTable$ is converted into the order in the bitstream according to the mapping $r$ and the bitstream shuffling order $BitIndx$. After traversing all possible 64-bit combinations, these bitstreams are added to the bitstream library. It is important to note that the bitstream library can be constructed before hand when the attacker determines which module to attack. The attack is performed directly using the bitstream library. For each input initial truth table, there will be at most 720 64-bit bitstream blocks. In the generated bitstream library, only one identical bitstream will be retained to shorten the number of matches.

The Kyber hardware accelerator design of [15] requires more than 7,000 LUTs, and although we already know the exact locations of the LUTs in the bitstream, comparing them one by one takes a significant amount of time. However, no matter what the input of the ROM truth table corresponding to a single 64-bit bitstream block is, the Hamming weight of the bitstream block corresponding to the same output is certain. Therefore, before the bitstreams of the extracted LUTs are compared with the generated bitstream library, the extracted bitstreams are first verified that the Hamming weights match to minimize comparison time.

*2) **Attack the CBD2 sampling**:* The attack scheme proposed in [7] keep the nonce counter in CBD sampling unchanged, so that the sampling results of key $s$ and noise vector $e$ are completely equal, that is, $t = A \circ s + s$. Therefore, it can be deduced that the specific value of the secret key $s$. However, in hardware design, there are often other counters, such as state machines, etc., which pose challenges to finding the nonce counter. But we noticed that the uniqueness of the truth table of the CBD sampler itself made it easier to detect in bitstream. In [15], the three different security levels are merged into one architecture, so that both CBD2 and centered-3 binomial distribution sampling ($CBD_3$) exist in that architecture. While in most of the other work [17], [18], the three different security levels are separated into three different architectures to be designed, so for the Kyber768 and Kyber1024 architectures, only CBD2 sampling is included. For the purposes of this paper, the CBD sampling in [15] is replaced with a CBD2 sampling module to provide attacks under Kyber768 and Kyber1024. The formula for calculating CBD2 is as follows:

$$CBD_{Res} = a0 + a1 - b0 - b1 \tag{1}$$

The final output of the CBD is therefore 12 bits ($CBD_{Res}$), but in the synthesized implementation, a portion of the output bits will be optimized out since some of the bits in the $CBD_{Res}$ have exactly the same value. In CBD2 mode, the result only be {0xd00, 0xcff, 0x0, 0x1, 0x2}. As we combine the identical bits, the output comprises only {0, 1, 2-7, 8, 10-11}, where the ninth bit is always '0' and will not be require a Flip-Flop for storage. Furthermore, a single LUT6 comprises two LUT5s, and the input port A6 is utilized to select the outputs of these two LUT5s, i.e., O6 and O5. While forming CBD2 module from LUT6, only four input ports are used. Thus, there is a possibility that the two LUT5s within a LUT6 may produce different output bits of CBD2 results. Note that due to the deterministic nature of input connections in LUT6, the inputs utilized to generate the result in CBD2 cannot produce any other results. The outputs of the two LUT5s only arise from different bits of the same result. Since a single CBD2 output contains only [0, 1, 2, 8, 11] 5 valid output bits out of the 12 output bits, when two LUT5s in the same LUT6 are used for each of the two bits, the remaining bit may be paired with either bit [3-7] or bit [10], even if the output of bit [3-7] is identical to [2], and the output of bit [10] is identical to the output of [11].

Due to bitstream shuffling, employing comparisons with shuffled bitstreams will increase the complexity of the process, particularly when dealing with LUT5 outputs involving two outputs i.e., shared LUTs. To address this, we have modified the resulting generated bitstream library. This adjustment ensures that the bitstream library no longer produces sequences exactly in the bitstream format. Therefore, the values in the bitstream library are sourced exclusively from BitRestrct rather than OrderFromBitstream. The sequence of BitRestrct outputs is arranged in truth table order and is split by the 0's and 1's of input A6 into two 5-input truth tables. For the original bitstream, however, the bitstream needs to be rearranged in truth table order (Re_OrderFromBitstream). After that, the 64-bit bitstreams are compared with the generated bitstream

library one by one according to the order of the LUTs in the address. The comparison has two steps, the first step checks the Hamming weight of the original bitstream with 6-input truth table. In CBD2 sampling of Kyber, the corresponding Hamming weights of the [0, 1, 2, 8, 11] bits of $CBD\_Res$ are [20, 8, 4, 16, 20], respectively. Only if the Hamming weights are the same, the current 64-bit original bitstream is compared with the bitstream library. The second step involves merging two LUT5s, where the first 32 bits and the last 32 bits of the 64-bit results from Re_OrderFromBitstream respectively indicate the truth tables of distinct bits within the CBD2 result. The Hamming weights of the 5-input truth tables are half the values of the 6-input truth tables, specifically [10, 4, 2, 8, 10] ($order\_list$). When searching for the LUT address corresponding to a particular bit, it is necessary to exclude the Hamming weight of the currently compared bit from the $order\_list$. For instance, when comparing $CBD\_Res[1]$, the $order\_list$ would be modified to [10, 2, 8, 10]. Afterwards, if the following three conditions are simultaneously met, the address of the current LUT is recorded:

1. The Hamming weight of the first 32 bits or the last 32 bits of the current original bitstream is equal to half the Hamming weight of the current bitstream library values.
2. The Hamming weight of the remaining 32 bits of the current original bitstream is equal to one of values from $order\_list$.
3. The first/last 32 bits of the original bitstream are the same as the first/last 32 bits of values in the bitstream library.

The address in the bitstream of the current matching LUT will be recorded along with whether the current matching LUT uses the first or last 32 bits of the current address.

It's important to highlight that once we identify the LUT utilized by CBD2, we can manipulate with the results of CBD2 at will. For instance, post-attack, the mapping between the three LUTs employed by one of the CBD2 samplers and the respective bits in the result is as follows: [0, 2] (LUT1), [7, 8] (LUT2), [1, 11] (LUT3). Among them, 2/7 of $CBD\_Res$ produce identical outputs, we don't care whether they are in LUT1 or LUT2. For LUT1, we can readily manipulate 32 bits to control $CBD\_Res[0]$ as either 0 or 1. Similarly, precise data manipulation can be executed for all bits, so that inject the any possible secret key values.

## IV. Attack Kyber on PKEs and KEMs

In this section, we will discuss the effects of performing four different attacks on Kyber PKE and KEM schemes.

### A. Attacks on the Key Generation

During key generation, Alice needs to calculate the following polynomial operations:

$$\begin{cases} \hat{s} = NTT(s) \\ \hat{e} = NTT(e) \\ \hat{t} = \hat{s} \circ \hat{A} + \hat{e} \end{cases} \tag{2}$$

We use the following four attack methods in Section III to attack:

- *Attack on **BRAMs**:* If all the BRAMs are disabled, the results of these calculations will be '0', and then the public key $pk$ transmitted by Alice to Bob is $(NTT(A \circ s + e)||\rho) = (0||\rho)$.
- *Attack on **DSPs**:* Disabling the DSP will result in simpler NTT calculations. Assume that the input of NTT is $\{a_0, a_1, ..., a_n\}$, then the calculation result of NTT is $\{a_0, a_1, ..., a_0, a_1\}$, which can be written as $NTT^*$. In addition, since PWM and NTT share the DSP unit in the design of [15] (as well as most of Kyber designs), therefore, the PWM calculation result will be zero, and the output of $pk$ will be $(NTT(A \circ s + e)||\rho) = (NTT^*(e)||\rho)$.
- *Attack on **ROM in NTT**:* In most Kyber designs, including [15], NTT, INTT, and PWM use independent ROMs, and the internal data of the ROMs are different, so attacking the ROM of the NTT will only affect the result of NTT calculation. In the case of known $pk$, it is easy to calculate $pk = \hat{A} \circ NTT^*(s) + NTT^*(e)$ to solve the secret key $s$.
- *Attack on **CBD2 sampling**:* The attack on CBD2 can only works on Kyber768 and Kyber1024. After finding the LUT that constitutes CBD2, attacker could set the result of CBD2 to any data that fits the key interval [-2, 2] (the attacker could know the secret key $s$ when changing the bitstream of the LUTs), and especially setting it to '0' will weaken $pk$.

Bob will perform the following operation during encryption:

$$\begin{cases} u = INTT(\hat{A^T} \circ \hat{r}) + e_1 \\ v = INTT(\hat{t^T} \circ \hat{r}) + e_2 \\ \quad + Decomperss_q(Decode_1(m), 1) \end{cases} \tag{3}$$

After attacking BRAM, $t^T$ received from Alice becomes all '0', resulting in $v = e_2 + Decompress_q(Decode_1(m), 1)$. With disabling the DSP, although $pk$ is not entirely composed of 0s, the range of each data element in $pk$ is [-2, 2], which aligns with the range of $r$ in Eq. (3). Thus, when the attacker obtains the ciphertext $ct^*$ after attack in theBRAMs and DSPs, they can decode and decompress $ct^*$ to directly retrieve the original message $m$. For NTT ROM zeroing and CBD sampling attacks, the attacker can directly obtain the secret key $sk^*$. Thus, the ciphertext $ct^*$ can be directly used to recover the message $m$.

On Alice's side, the message $m$ needs to be recovered first, and Alice needs to calculate:

$$m = Compress_q(v - INTT(\hat{s^T} \circ NTT(u)), 1) \tag{4}$$

With BRAM and DSP disabling, polynomial multiplication operation is set to zero. This results in the outcome of $NTT(\hat{s^T} \circ NTT(u))$ being zero, enabling Alice to directly recover the message $m$ from $v$. Regarding CBD2 sampling, Alice can recover the message $m$ directly as the attack on CBD2 does not affect polynomial operations. However, it's important to note that the $KeyGen.$ and $Decaps.$ phases are often integrated into the same architecture. Thus, if a module in the $KeyGen.$ phase is attacked, the $Decaps.$ phase is also affected. For instance, setting the twiddle factor in NTT to

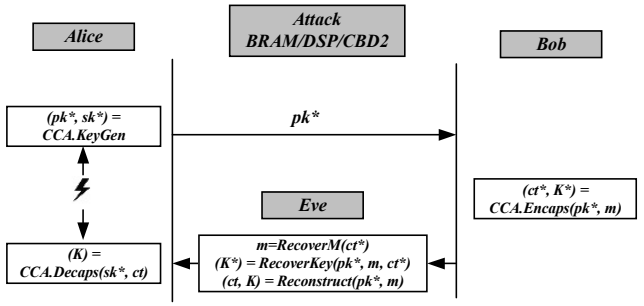


Fig. 1. RecoverM attack on $KeyGen.$ of CCA secure Kyber KEM.

zero impacts the computation of $NTT(u)$ in Eq. (4), posing difficulties for Alice in recovering the message $m$. Furthermore, the attack on $KeyGen.$ also affects the verification stage within the KEM protocol. As a result, even if Alice could recover the message $m$, the recomputed shared key $K_{Alice}$ will not match the shared key $K_{Bob}$ transmitted by Bob. Thus, the Man-In-The-Middle (MITM) attacker, Eve, must serve as an intermediary between Alice and Bob to manipulate the communication. As shown in Fig. 1, Eve already possesses knowledge of Alice's public key $pk^*$ and message $m$. Using this information, Eve can compute a new ciphertext following the attacked Reconstruct scheme and then relay it to Alice, ensuring that Alice's $K_{Alice}$ matches that of Eve.

### B. Attacks on Key Encapsulation

Disabling the BRAM and DSP units during the $Encaps.$ phase will lead to an incorrect polynomial computation in Eq. (3). When the BRAM is disabled, all intermediate data in the INTT computation becomes inaccessible, resulting in an $INTT(\hat{t^T} \circ \hat{r})$ value of 0. Moreover, the architecture in [15], regardless of whether it needs to perform an NTT/INTT computation or not, the result of the sampling will be stored in BRAM, leading to $INTT(\hat{t^T} \circ \hat{r}) + e_1$ becoming 0 as well. Thus, Alice will receive the ciphertext $ct^* = (u||v) = (0||m)$ from Bob, enabling attackers to directly obtain the message $m$ from $ct$. Similarly, if the DSP unit is disabled, the result of $\hat{t^T} \circ \hat{r}$ will be set to zero, causing $v$ to be solely from the sum of $e_2$ and $m$. The result of $u$ will be $e_1$, and Alice will receive the ciphertext $ct^* = (u||v) = (e_1||e_2+m)$ from Bob. Similarly, Eve can easily deduce the message $m$ from $ct^*$.

When the NTT ROM in the $Encaps.$ phase is set to zero, $\hat{r}$ will contain only two valid data, ranging from -2 to 2. Meanwhile, in the $Encaps.$ phase, $e_1$, $e_2$ will not perform NTT computation, and the zeroing of the NTT ROM is equivalent to weakening the key $r$. An attacker can employ an exhaustive attack technique to deduce Bob's key $r$ from $u = INTT(\hat{A^T} \circ \hat{r})$ + $e_1$, then use the key to parse $v$ to get the message $m$. At this point, the ciphertext transmitted by Bob will follow the change of matrix $A$, which conforms to the features of the generated ciphertext and it is therefore difficult to detect [8]. In the CBD2 attack, polynomial operations are not affected. The attacker only inserts a known key into Bob's architecture, allowing Eve to effortlessly obtain the $m$ from the ciphertext.

For Alice, under each of the four attack methods, when Alice receives the ciphertext $ct^*$ from Bob, she could analyze $ct^*$ and try to recover the message $m$ in Eq. (4) as follows:

- *Attack on **BRAMs**:* When the BRAM is disabled, then $ct^* = (0||m)$. As a result, from Eq. (4), Alice can successfully recover $m$.
- *Attack on **DSPs**:* When the DSP is disabled, $ct^* = (e_1||e_2 + m)$, and Eq. 4, simplifies to $m = Compress_q(e_2 + m - (e_1 \circ s^T))$. Since $s^T$, $e_1$, $e_2$ take values in the range of -2 to 2, Alice can recover $m$ normally.
- *Attack on **NTT ROM** or **CBD2**:* When both the NTT ROM and CBD2 modules are subject to an attack, it essentially means the attacker has acquired Bob's secret key $r$. However, this does not compromise any of the polynomial operations on Bob's end. Consequently, Alice can still recover the message $m$ securely.

Similar to the attack on the $KeyGen.$ phase, if the architecture of $Encaps.$ phase is attacked, Alice can no longer obtain the same ciphertext $ct$ and shared key $K$ as Bob, even if she has parsed the message $m$. As depicted in Fig. 2, at this time, Eve acts as an MITM in the communication. After receiving Bob's $ct^*$, Eve recovers the original message $m$ (RecoverM) and then utilizes the $pk$ transmitted by Alice and message $m$ to recalculate $ct$ and $K$, which she then transmits back to Alice. Subsequently, Alice recalculates $ct$, $K$ using $sk$ and message $m$ and treats it as her own. Thus, the ciphertext $ct$ and key $K$ calculated by Alice during the verification stage match those transmitted by Eve.

## V. Physical Experiment and Discussion

For the physical experiment, we employed a Nexys 4 FPGA (xc7a100t) with 63,400 LUTs, 135 BRAMs, and 240 DSPs. To monitor the data obtained from the modified bitstream, we established a connection between the FPGA and PC using the UART module. Table I presents the complexity of the generated $pk$/$ct$ for the four attack methods and whether Eve and Alice are able to successfully recover the message $m$. In the table, we specify the number of malicious bit modifications required. For the ROM and CBD2 attacks, as they only reveal the secret

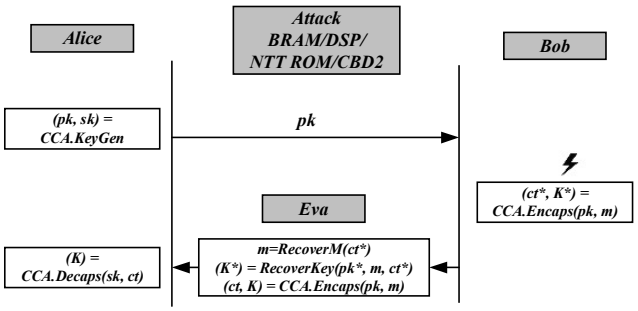


Fig. 2. RecoverM attack on $Encaps.$ of CCA secure Kyber KEM.

TABLE I
COMPLEXITY OF $pk$/$ct$ UNDER FOUR ATTACK METHODS AND WHETHER EVE AND ALICE COULD RECOVER MESSAGE $m$

| Attack Methods | KeyGen | | | Encaps. | | | Bits |
|---|---|---|---|---|---|---|---|
| | $pk$ | m(Eve) | m(Alice) | $ct$ | m(Eve) | m(Alice) | |
| BRAM | Weak | ✓ | ✓ | Weak | ✓ | ✓ | 540 |
| DSP | Weak | ✓ | ✓ | Weak | ✓ | ✓ | 240 |
| NTT ROM | Strong | ✓ | × | Strong | ✓ | ✓ | 807 |
| CBD2 | Strong | ✓ | ✓ | Strong | ✓ | ✓ | 768 |

key to Eve without affecting the effective key generation, the resulting $pk$/$ct$ varies with the matrix $A$/$A^T$, enhancing the complexity of the attacked $pk$/$ct$. Conversely, the BRAM attack leads to the lowest $pk$/$ct$ complexity due to the architecture in [15], where the polynomial multiplication and sampling results are fed into the BRAM before computation, leading to the majority of data in the $pk$/$ct$ result being set to zero. Moreover, zeroing the NTT ROM during the key generation phase poses challenges for Alice in recovering the message $m$, making the attack non-continuous. In Xilinx FPGAs, each cluster contains an equal number and identical internal addressing of BRAMs, DSPs and LUTs. This enables attackers to pre-test various FPGA types, allowing them to ascertain specific BRAM and DSP addresses and pre-store this information. Thus, when a particular device is identified from the bitstream, the database can be invoked to facilitate the attack. For the Artix-7 FPGAs, the available models include only the following 8 devices, i.e., xc7a12t, 15t, 25t, 35t, 50t, 75t, 100t, and 200t; making it feasible to construct the database in advance.

Table I also present the number of bits requiring modifications under the four attack methods. For the NTT ROM, the bits needing alteration correspond to the total number of bits in the 24 LUTs occupied by the ROM, minus the Hamming weight of each bit. In the case of CBD2, the maximum number of bits to be modified is demonstrated for a total of 12 LUTs utilized across 4 parallel sampling units. Unlike the other attacks, CBD2 attack does not allow simple changes to the LUT bitstreams to 0. As seen in Table I, the CBD2 attack alters relatively fewer bitstreams, and has high complexity in generating the attacked $pk$/$ct$. In addition, both Eve and Alice can recover $m$, although the attack solely works on Kyber768 and Kyber1024. However, it it important to note that some designs independently implement CBD2 and CBD3 in Kyber512, making it possible to execute this part of the sampling attack.

**Countermeasures:** For hardware core in FPGA (BRAM, DSP), it is necessary to add zero test to the output of hard cores. However, for LUT-based modules, since the attacker knows the truth table of a single LUT in the design, the attacker can easily find the specific LUT. Thus, this process could be spread out over two or more LUTs, or the counter could be completed using two or more clock cycles.

## VI. CONCLUSION

We performed the first malicious bitstream modification based fault attack on the FPGA bitstream of Kyber, the only KEM standardized protocol for post-quantum cryptography (PQC) by NIST. We analyze the effect of Kyber under four attacks: disabling BRAMs, disabling DSPs, zeroing the NTT ROM and tampering with CBD2 results. In addition, we evaluate these four attacks of the $pk/ct$ generated after the attack and whether Eve and Alice can recover the message $m$ correctly. The results show that all four attacks can be successfully completed, and that $pk/ct$ after the zeroed NTT ROM and tampered CBD2 attacks are difficult to recognize due to their high complexity. Finally, we propose two countermeasures.

## 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] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *28th ACM Symposium on Theory of Computing*, pp. 212–219, 1996.

[3] D. Moody, "Post-quantum cryptography: NIST's plan for the future," in *Talk given at PQCrypto'16 Conference*, 2016.

[4] 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 3*, 2020.

[5] A. Khalid, T. Oder, F. Valencia, M. O'Neill, T. Güneysu, and F. Regazzoni, "Physical protection of lattice-based cryptography: Challenges and solutions," in *Proceedings of the 2018 on Great Lakes Symposium on VLSI*, pp. 365–370, 2018.

[6] P. Ravi, A. Chattopadhyay, J. P. D'Anvers, and A. Baksi, "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.

[7] P. Ravi, D. B. Roy, S. Bhasin, A. Chattopadhyay, and D. Mukhopadhyay, "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.

[8] P. Ravi, B. Yang, S. Bhasin, F. Zhang, and A. Chattopadhyay, "Fiddling the twiddle constants-fault injection analysis of the number theoretic transform," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 447–481, 2023.

[9] P. Swierczynski, M. Fyrbiak, P. Koppe, and C. Paar, "FPGA trojans through detecting and weakening of cryptographic primitives," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 8, pp. 1236–1249, 2015.

[10] D. Ziener, J. Pirkl, and J. Teich, "Configuration tampering of bram-based AES implementations on fpgas," in *2018 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, pp. 1–7, IEEE, 2018.

[11] M. Moraitis and E. Dubrova, "Bitstream modification attack on SNOW 3G," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1275–1278, 2020.

[12] V. Rijmen and J. Daemen, "Advanced encryption standard," *Proceedings of Federal Information Processing Standards Publications, National Institute of Standards and Technology*, pp. 19–22, 2001.

[13] ETSI/SAGE, "Specification of the 3gpp confidentiality and integrity algorithms uea2 & uia2," 2006.

[14] M. Ender, A. Moradi, and C. Paar, "The unpatchable silicon: a full break of the bitstream encryption of xilinx 7-series FPGAs," in *29th USENIX Security Symposium (USENIX Security 20)*, pp. 1803–1819, 2020.

[15] 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.

[16] M. Jeong, J. Lee, E. Jung, Y. H. Kim, and K. Cho, "Extract LUT logics from a downloaded bitstream data in FPGA," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, 2018.

[17] V. B. Dang, K. Mohajerani, and K. Gaj, "High-speed hardware architectures and FPGA benchmarking of CRYSTALS-Kyber, NTRU, and Saber," *IEEE Transactions on Computers*, vol. 72, no. 2, pp. 306–320, 2023.

[18] Z. Ni, A. Khalid, D.-e.-S. Kundi, M. O'Neill, and W. Liu, "HPKA: A high-performance CRYSTALS-Kyber accelerator exploring efficient pipelining," *IEEE Transactions on Computers*, pp. 1–14, 2023.