# Binary Ring-LWE Hardware with Power Side-Channel Countermeasures

Aydin Aysu, Michael Orshansky, and Mohit Tiwari
Department of Electrical and Computer Engineering
The University of Texas at Austin, Austin, TX, USA
{aydinay,tiwari,orshansky}@utexas.edu

***Abstract*—We describe the first hardware implementation of a quantum-secure encryption scheme along with its low-cost power side-channel countermeasures. The encryption uses an implementation-friendly Binary-Ring-Learning-with-Errors (B-RLWE) problem with binary errors that can be efficiently generated in hardware. We demonstrate that a direct implementation of B-RLWE exhibits vulnerability to power side-channel attacks, even to Simple Power Analysis, due to the nature of binary coefficients. We mitigate this vulnerability with a redundant addition and memory update. To further protect against Differential Power Analysis (DPA), we use a B-RLWE specific opportunity to construct a lightweight yet effective countermeasure based on randomization of intermediate states and masked threshold decoding. On a SAKURA-G FPGA board, we show that our method increases the required number of measurements for DPA attacks by 40× compared to unprotected design. Our results also quantify the trade-off between side-channel security and hardware area-cost of B-RLWE.**

## I. Introduction

Public-key encryption schemes using lattice problems enable quantum-secure alternatives of existing systems. Among lattice-based solutions, proposals relying on Ring-Learning-with-Errors (RLWE) problem [1] have been popular for their efficiency in hardware [2–5] and software [2, 6–8] implementations. A variant of RLWE, *binary*-RLWE (B-RLWE) [9], has recently been proposed providing even more efficient constructions due to its elimination of costly Gaussian samplings, reduction of operand sizes, and simplification of modular reduction. B-RLWE instead works with a smaller collection of bits sampled from a uniform distribution while achieving sufficient security levels against conventional [9, 10] and quantum [11] cryptanalysis.

Although lattice-based public-key schemes have theoretical guarantees against powerful quantum adversaries, their implementation can be vulnerable against physical attacks. Indeed, power-based side-channel attacks have been applied thoroughly for traditional public-key encryption algorithms like elliptic curve cryptosystems (ECC) [12]. These attacks extract secret keys by analyzing correlations between secret key values and power consumption of a target device. To that end, Simple Power Analysis (SPA) and Differential Power Analysis (DPA) are common and powerful threats that can extract keys from small correlations by evaluating several measurements with statistical methods.

Prior work on side-channel analysis of lattice-based pubic-key encryption focuses on RLWE or software specific attributes [13–18]. Saarinen *et al.* formulated possible blinding countermeasures for the Gaussian sampling and Number Theoretic Transform (NTT) of RLWE [13]. Reparaz *et al.* implemented two countermeasures, first based on arithmetic masking with a random key split [14], and second based on ciphertext-blinding by utilizing RLWE's additive homomorphism [15]. Oder *et al.* adapted a combination of these methods for an adaptive chosen-ciphertext-attack-secure (CCA-2) variant of RLWE [16]. Park *et al.* mounted an attack on RLWE implementation exploiting a modular addition vulnerability that occurs when RLWE is specifically implemented on 8-bit micro-controllers [17]. Most recently, Primas *et al.* showed a template attack on a software implementation of NTT that exploits a multitude of side-channels including a vulnerability of division instruction (`DIV`) on ARM Cortex-M4 [18].

Side-channel analysis of B-RLWE is different from RLWE as parts of the secret key (coefficients of the secret-key polynomial) are drawn from a binary distribution and thus have only two possible values: '0' or '1'. RLWE keys, however, are sampled from a larger set with a Gaussian distribution. Unfortunately, unlike RLWE, side-channel analysis of B-RLWE scheme has not been evaluated so far—this analysis is crucial for embedded deployment of B-RLWE since unprotected implementations will be vulnerable to such attacks. There is also no hardware realization of B-RLWE: the only existing work is a software implementation for Atmel-AVR and ARM-Cortex-M0 microcontrollers [19].

This paper presents the first B-RLWE hardware along with low-cost power side-channel countermeasures. We analyze the effect of B-RLWE operations for side-channel attacks and utilize a B-RLWE specific opportunity to design a low-cost DPA countermeasure. Specifically, we show that B-RLWE polynomial multiplication leaks information through partial product generation and intermediate sum update. While partial product generation causes SPA leaks, when it is mitigated, values of intermediate sum update become vulnerable to DPA attacks, which can be protected by its random initialization and masking. We evaluate the cost and efficiency of SPA and DPA attacks and countermeasures on a physical platform with real measurements. Compared to prior work, we claim the following 3 contributions.

**1. A Novel SPA Attack and Countermeasure for B-RLWE.** We describe a simple and effective SPA attack that becomes possible on B-RLWE. This attack can recover the secret key with a few power measurements due to binary coefficients of B-RLWE. We then propose an SPA countermeasure that uses *always-add-with-redundant-store* method to normalize power consumption.

**2. Novel DPA Attacks and Countermeasure for B-RLWE.** We develop novel DPA attacks that can break an SPA-secure B-RLWE implementation. Notably, these attacks can work for an adversary that has no knowledge on the secret key and can reduce the secret-key search-space from $2^n$ to only 2 options. We also provide a novel and efficient countermeasure via random initialization and masking that is secure against (first-order) DPA.

**3. First Hardware Implementations.** We design efficient hardware implementations of B-RLWE decryption. We provide two baseline designs for $(i)$ low-area and $(ii)$ high-performance applications, which are respectively the *smallest* and *fastest* lattice-based public-key decryptions to date. We extend the low-area design to apply proposed defenses and quantify their cost.

978-3-9819263-0-9/DATE18/©2018 EDAA

| | |
|---|---|
| KeyGen<br>$r_1, r_2 \leftarrow \mathcal{R}_q$<br>$p = r_1 - a \cdot r_2$ | Key generation samples the secret key $r_2$ and polynomial $r_1$ with binary coefficients, and produces public key $p$. $a$ is a global, public system parameter. $r_1$ can be discarded after this step. |
| Enc<br>$e_1, e_2, e_3 \leftarrow \mathcal{R}_q$<br>$c_1 = a \cdot e_1 + e_2$<br>$c_2 = p \cdot e_1 + e_3 + \tilde{m}$ | Encryption generates the ciphertext $c_1, c_2$ from message $m$ by using public key $p$ and by adding error nonces $e_1, e_2, e_3$ sampled with binary coefficients. The input message $m$, which is an $n$-bit binary string, is expanded to the polynomial $\tilde{m}$ by multiplying each coefficient with $q/2$ and adding $q + n/2 - 1 - i$, where $i$ is the degree associated with the coefficient. |
| Dec<br>$m = th(c_1 \cdot r_2 + c_2)$ | Decryption reconstructs the message $m$ by using the secret key $r_2$. The threshold decoder $th(.)$ processes each coefficient separately and it returns a binary value '1' if $c[i]$ lies in the range $(q/4,\ 3q/4)$, else it returns '0'. |

Fig. 1. B-RLWE public-key encryption scheme

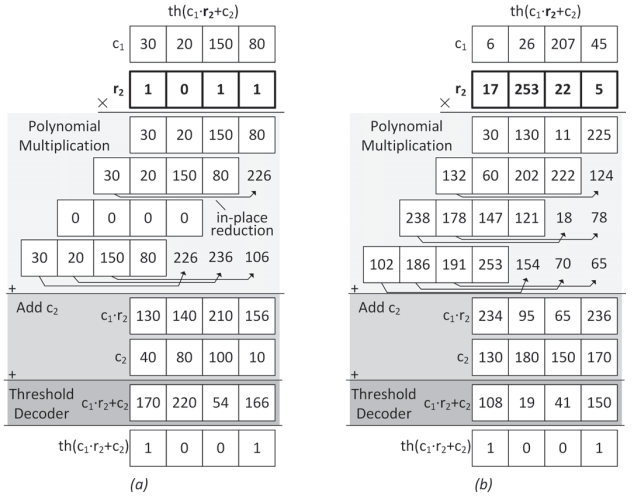


Fig. 2. (a) B-RLWE vs. (b) RLWE decryption example. Note the difference in secret key $r_2$; while B-RLWE works with binary coefficients, RLWE samples them from Gaussian distribution, which, for the example, is between 0 and 255.

## II. B-RLWE Public-key Encryption Scheme

B-RLWE[1] is a new, promising variant of RLWE that achieves smaller key sizes and more efficient computations [9]. The security analysis of B-RLWE is corroborated by several authors [10, 11, 20]. This section first provides a background on B-RLWE and introduces the public-key encryption scheme. We then highlight the difference of B-RLWE from RLWE and give a high-level motivation for its impact on power side-channels.

All Latin letters such as $r_i$, $c_i$, $c$ express polynomials, unless stated otherwise. The letter $m$ indicates a binary string, and $i$, $n$, and $q$ denotes an integer. The operations $+, -, \cdot$ respectively corresponds to polynomial addition, subtraction, and multiplication, if they operate with polynomials. The symbol $\leftarrow$ shows sampling a polynomial with binary coefficients and $\oplus$ is an XOR operation. We use $c[i]$ to represent a specific coefficient of polynomial $c$ with the degree $i$. To display multiple coefficients of a polynomial $c$ in a single notation, we use $c = [c[i]\, c[i-1]]$.

B-RLWE uses the ring $\mathcal{R}_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$, where an element inside this ring is a polynomial of degree $n-1$ with integer coefficients modulo $q$. The operations defined in this ring are polynomial addition, polynomial subtraction, and polynomial multiplication. The polynomial addition and subtraction is simply applying a coefficient-wise, modular addition and subtraction operations, respectively. Polynomial multiplication generates a $2n-2$ degree polynomial which is converted back to a polynomial of degree $n-1$ with the reduction function of $f(x) = x^n + 1$.

The B-RLWE public-key encryption scheme consists of three parts: Key Generation (`KeyGen`), Encryption (`Enc`), and Decryption (`Dec`). Note that, while encryption operates with public or ephemeral values, decryption uses the *long-term secret key* and thus is the target of side-channel attacks. Therefore, we implement and analyze the side-channel security of the decryption process. Figure 1 summarizes the B-RLWE scheme.

For the scope of this paper, we use the parameter set-II in [19], which achieves a relatively lower decoding failure probability of $2^{-32}$, a security level of 84-bits (88-bits according to Wunderer [10]) against conventional computers, and a security level of 73-bits [11] against quantum computers. Since we target *lightweight applications* (eg. constrained IoT nodes) where typical security level is 80-bits (eg. [21]), these security levels suffice for our case. The target configuration sets $n = 256$ and $q = 256$; it works with polynomials degree 255 and with coefficients modulo 256. *The same 8-bit arithmetic we analyze supports 190-bit security level so our side-channel analysis covers B-RLWE instantiations of higher security*. Note that these parameters are smaller than state-of-the-art RLWE settings [22], which is favorable for lightweight applications. But they do not, by default, allow a Number Theoretic Transform (NTT) based polynomial multiplication. The parameters, however, may be tweaked to enable NTT for trading off area for performance; such optimizations and their side-channel analysis are out of scope of this work.

Figure 2 illustrates a decryption example for B-RLWE and RLWE. For simplicity, the example uses the same toy setting $n = 4$ and $q = 256$ both for B-RLWE and RLWE. The figure shows the multiplication of $c_1 \cdot r_2$ with in-place reductions. Each row of multiplication shows the product of a single $r_2$ coefficient with all coefficients of $c_1$; we refer to this method as *row-wise* multiplication. The in-place reduction is a (modulo 256) sign change of the reduced coefficient. Result of polynomial multiplication is the addition of all row-wise computations. In practice, this occurs in a *row-first* manner by computing a row of partial products and by adding them to the intermediate sum that holds the result of previous row. The polynomial multiplication thus has an *accumulative nature*; intermediate sum values at each row depend on all previous rows, which will later in Section III-C play a key role in DPA attacks.

The main difference of B-RLWE vs. RLWE decryption is the secret-key polynomial $r_2$ of B-RLWE sampled with binary coefficients. Hence, polynomial multiplication of B-RLWE is a sequence of additions. If key bit $r_2[i]$ is equal to '0', no partial products are generated and intermediate sum keeps its value (third row in the example). Else, partial products (ie. coefficients of $c_1$) are added into the intermediate sum. Although being more efficient, this causes SPA leak as we discuss in Section III-B.

## III. Side-channel Analysis of B-RLWE Hardware

### A. Area-optimized Hardware Design

Parallelization choice of polynomial multiplication derives the hardware design of B-RLWE. Since we primarily focus on lightweight applications, we design and analyze *coefficient-serial*

[1]To emphasize its *binary* aspect, we refer the scheme as binary-ring-learning-with-errors, and abbreviate it as B-RLWE. We do not cover CCA-2 security of the scheme and refer interested readers to [16] for possible CCA-2 extensions.

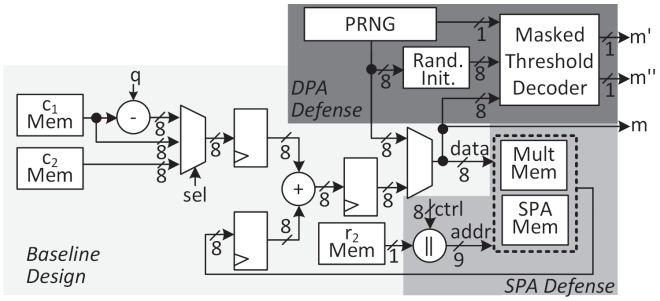


Fig. 3. Hardware architecture of B-RLWE including side-channel defenses

architectures that compute a single coefficient of a single row of multiplication in a clock cycle. This design uses one adder as its main processing unit. Therefore, it takes $n$ cycles to compute one row of polynomial multiplication and $n \times n$ cycles to compute the entire operation. Figure 3 demonstrates the details of the hardware architecture for lightweight B-RLWE decryption. This figure shows our baseline architecture and also highlights the side-channel security defenses in the same diagram. Section III-B and III-C elaborate SPA and DPA defense, respectively.

The main processing unit of the architecture is the 8-bit adder that first computes the polynomial multiplication and later the polynomial addition. Since the modular reduction $q$ is set to 256, it is simply cutting off the most significant bit of the sum (the 9-bit adder output is truncated into 8-bits). The sign change for in-place reduction is handled by subtracting $c_1[i]$ from $q$ and either, $c_1[i]$, in-placed $c_1[i]$, or $c_2[i]$ is selected as the input of the adder. The second input of the adder is the output of memory that stores intermediate sum. In the baseline design, threshold decoding is simple: the output $m$ is generated by XORing the most significant two bits of the coefficients after adding $c1$.

### B. SPA Attack and Countermeasure of B-RLWE

The switching activity is reduced if $r_2[i]$ is '0' because addition and memory update will not occur by default. An SPA attack can thus extract the secret key with a few measurements by interpreting differences in power traces caused by these operations.

To normalize the power activity, we introduce the "always-add-with-redundant-store" method that always computes the addition of coefficients and performs a memory update. Figure 3 demonstrates the architecture of this datapath. For SPA security, our architecture is designed with two key insights: (1) Adder will always, ie. independent of the key value, compute the partial product and (2) if the key is '0', the result will be stored in a redundant memory (SPA Mem in Figure 3). To minimize the power variance, this redundant memory is forced to be placed within the same memory block (BRAM) that holds the intermediate sums, ie. they are separated with address space. Therefore, regardless of the value of $r_2[i]$, there is always an addition and memory write that takes place in the SPA-resistant architecture, preventing variances exploited by SPA in the power-trace caused by inactivity. Section V-A evaluates the effectiveness of our SPA defense. Our countermeasure may be vulnerable to EM or photonic attacks but they are out of scope of this work.

### C. DPA Attack and Countermeasure of B-RLWE

DPA can exploit small power correlations to key by using a statistical analysis on a sufficiently large number of power traces, and break a design that has SPA countermeasure.

*1) Attacking SPA-secure Executions with DPA:* Due to the accumulative nature of B-RLWE polynomial multiplication, the computation carried out at a certain row depends on the current key bit and all the previous key bits that have been used in the multiplication. For example, the result of the second row, depends on the first key bit ($r_2[0]$) and the second key bit ($r_2[1]$). The crux of our DPA attack is to exploit this notion: a different key history will lead to a different computation. Using this attack, the adversary, starting from the first bit, can subsequently recover the entire secret key.

TABLE I
DPA ATTACK INTUITION FOR SPA-SECURE CIRCUIT. VALUE OF $r_2[i-1]$ CORRELATES WITH POWER CONSUMPTION AT ROW $i$

| | Attacking first row ($r_2[0]$) | | Attacking second row ($r_2[1]$,$r_2[0]$) | | | |
|---|---|---|---|---|---|---|
| $r_2[i]$ | 0 | 1 | 00 | 01 | 10 | 11 |
| Adder Operation | $0 \rightarrow 30$ | $0 \rightarrow 30$ | **$0 \rightarrow 20$** | **$30 \rightarrow 50$** | **$0 \rightarrow 20$** | **$30 \rightarrow 50$** |

Table I visualizes the intuition behind the proposed DPA attack, which details an attack to the toy example in Figure 2 with $c_1$=[30 20 150 80] and $r_2$=[1 0 1 1]. Let the DPA adversary target the computation of $mult[3]$, which denotes the most significant byte of the intermediate sum of polynomial multiplication. Recall that, due to the SPA countermeasure, the intermediate sum's accumulation by $c_1[i]$ will occur regardless of the key $r2[i]$ value.

Attacking the first row of computations fails since regardless of the value of $r_2[0]$ the hardware will compute 30=0+30. Therefore, SPA defense will also protect against the attack that targets computations in the first row. However, by attacking the second row of polynomial multiplication, the adversary can extract the value of the first bit of secret key (ie. $r_2[0]$). This is possible because the computation that takes place in the second row depends on previous bit due to the accumulative nature of the computation—even though SPA countermeasure makes it independent of the current key value. For the example, if $r_2[0]$ is '1', the datapath will compute 50=30+20, else, if $r_2[0]$ is '0', the datapath computes 20=0+20. Thus, the adversary can effectively execute a *differential attack*; it can apply a random input to the decryption, make a hypothesis on an intermediate sum from input value and key guesses, and check those hypothesis through the power trace. This attack will recover the value of each key bit, starting from $r_2[0]$, and it can successively recover consecutive bits of the key. For example, after recovering $r_2[0]$, the attacker can target the computations of the third row and extract $r_2[1]$. When this DPA attack completes; only 2 key candidates remain depending if $r_2[n-1]$=0 or $r_2[n-1]$=1. The brute force search space will thus reduce from $2^n$ to 2 options.

The proposed DPA attack extracts the secret key by making and testing hypothesis on a single intermediate computation, which is typically referred to as *first-order* DPA attacks.

*2) Novel DPA countermeasure with Random Initialization and Masking:* We propose a novel first-order DPA countermeasure for B-RLWE decryption based on random initialization and a masked threshold decoder. Our approach is to initialize the intermediate sum with a random value at the start of each decryption process and to recover from the effect of this random value at the final step using a masked threshold decoder. Figure 4 illustrates this method and shows its effect on the same example used in Figure 2. Since polynomial operations have additive homomorphism, the initialized random values will be preserved right before the threshold decoder (ie. each coefficient is shifted by a fixed, random amount). However, simply subtracting the random numbers before the threshold may allow a DPA-adversary

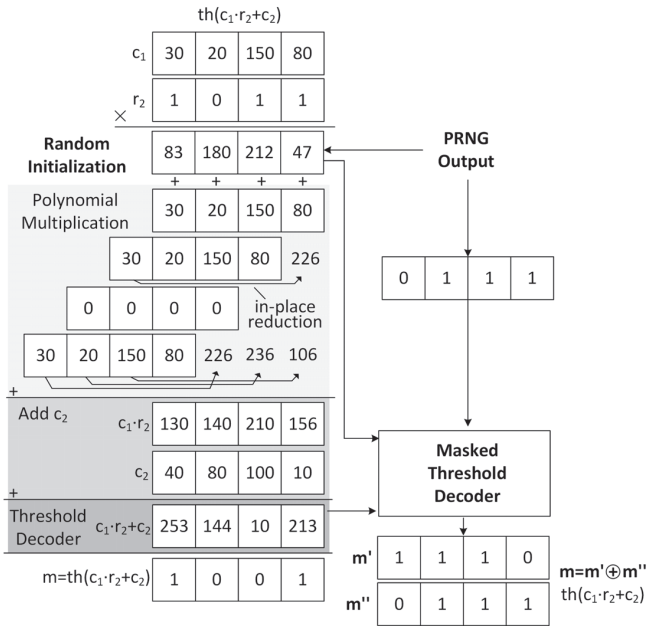


Fig. 4. Example of random initialization with masked threshold decoder.

to correlate against the output values; DPA defenses remove *all* correlations between intermediate computations and the key. The masked threshold decoder takes random initialization, shifted result, and a random bit as input for each coefficient to generate two bits $m'$ and $m''$ such that $m = m' \oplus m''$. Since it uses a random bit for each new execution, the masked threshold decoder successfully de-correlates the final operation from the secret key. The masked threshold decoder can be realized with a $128K \times 2$ ROM. If the final XOR is applied on the two output shares $m'$ and $m''$, the proposed method would be functionally equivalent to an unmasked threshold decoder. Hence, our method does not increase the error rate of thresholding. Attacks on intermediate coefficients of $r_2 * c_1$ before adding them to accumulator is also not possible for two reasons. First, due to SPA defense, $c_1$ coefficients are always loaded and added. Second, the intermediate values change for each execution (due to random initialization), hence adversary cannot build accurate hypothesis.

Figure 3 shows the implementation of our DPA defense. We used a PRNG based on TRIVIUM [23] to optimize area, which is also used in previous lightweight lattice-based implementations [4]. Any cryptographically secure PRNG can be used; this selection has *no impact* on our side-channel analysis. PRNG supplies random bits to initialize memory contents that store intermediate sum (both real and redundant memory) and to apply masked threshold decoder. The initialized values are also stored in a memory ($Rand.\ Init.$, Figure 3) to be later used in the masked threshold decoder.

*3) Comparison of Proposed Countermeasure with Prior Methods:* Our DPA countermeasure has a lower execution-time overhead compared to previous masked solutions applied for RLWE [14], [16]. Both of these schemes perform random splitting: the key is split into two random shares and each share is computed separately, thus doubling the cycle count. In contrast, the cycle count overhead of our method is much smaller ($< 4\%$), which is the time it takes to generate random numbers to properly initialize the memory. We also do not opt for the threshold decoder proposed in [14] for several reasons. First, it reduces the reliability of decryption. Second, it further increases the execution time. Third, it may be vulnerable to horizontal DPA attacks. Moreover, the operand size of B-RLWE is 8-bits, thus a straightforward threshold decoder with a table lookup occupies relatively smaller area. The DPA-resistance approach of [15] drastically reduces the reliability because the decryption has to correct the aggregated errors of two encryptions. Moreover, it also enforces a decryption system to include overheads of a full encryption module. Our method does not have these drawbacks.

TABLE II
COMPARISON WITH PREVIOUS LWE-BASED DECRYPTION HARDWARE

| Reference | SCR[a] | LUT/FF/SLICE | BRAM /DSP | Cycles | MHz | Op/s | Device |
|---|---|---|---|---|---|---|---|
| [25][b] | ✗ | 63/58/32 | 13/1 | 32768 | 144 | 4395 | S6LX45 |
| [2] | ✗ | 124158/65174/- | - | - | - | - | V6LX240T |
| [5] | ✗ | 4549/3624/1506 | 12/1 | 4404 | 262 | 59492 | V6LX75T |
| [4] | ✗ | 94/87/32 | 1/1 | 66338 | 189 | 2849 | S6LX9 |
| [3] | ✗ | 1349/860/- | 2/1 | 2800 | 313 | 109890 | V6LX75T |
| [14] | ✓ | 2014/959/- | 2/1 | 7500 | 100 | 13333 | XC2VP7 |
| **Our High-perf.** | ✗ | 6728/6813/1874 | 0/0 | **262** | 101 | 385496 | S6LX75 |
| **Our Low-area** | ✗ | **57/30/19** | 2/0 | 65797 | 135 | 2051 | S6LX75 |
| **Our Low-area +SPA** | ✓ | **58/30/22** | 2/0 | 65797 | 133 | 2021 | S6LX75 |
| **Our Low-area +SPA+DPA** | ✓ | **115/78/38** | 21/0 | 68105 | 119 | 1747 | S6LX75 |

[a] Side-channel resistance
[b] Howe *et al.* uses standard lattices instead of ideals, hence they implement LWE decryption instead of RLWE

*4) Higher-order DPA Attacks:* Higher-order DPA attacks can break systems that are protected with lower-order masking. We describe a second-order DPA attack to our first-order DPA defense (Sec. V-B3 shows results). The goal of this attack is to negate the effect of random initialization by picking two points in the power trace and by combining these two points to make direct correlations to the secret key value. The proposed attack uses absolute-difference based combination [24], which exploits the correlation of Hamming weight ($HW$) of absolute differences ($HW(a) \sim |HW(a+r \bmod 256) - HW(r)|$), to correlate directly to the target value. To achieve this attack, adversary correlates against the combination of random numbers generated by the PRNG ($HW(r)$) and the random intermediate addition $HW(a+r \bmod 256)$. Second-order DPA-adversary thus applies the first-order DPA attack method on the two combined points.

## IV. IMPLEMENTATION COST OF B-RLWE HARDWARE

The hardware architectures we propose are implemented in Verilog HDL and mapped on to the Xillinx Spartan-6 XC6SLX75 FPGA. The synthesis, placement, and routing of the proposed designs to the target FPGA is performed using Xilinx Integrated Synthesis Environment (ISE) version 14.6.

Due to different optimization goals, automation tools, and target FPGA device, a fair comparison among different hardware architectures is difficult make purely on the basis of technology. Moreover, different parameters used in RLWE implementations, which may also result in different pre/post-quantum security levels and decoding failure rates, complicates this task. Nevertheless, we list previous RLWE implementations in addition to reporting our results to provide a first-order comparison. To give an essence of the design space of B-RLWE, we also implement a baseline parallel architecture that uses 256 adders, which can compute a complete row of partial products in one clock cycle. This design also follows a row-first polynomial multiplication.

Table II reflects the potential of B-RLWE as a promising alternative for post-quantum public-key scheme. The baseline

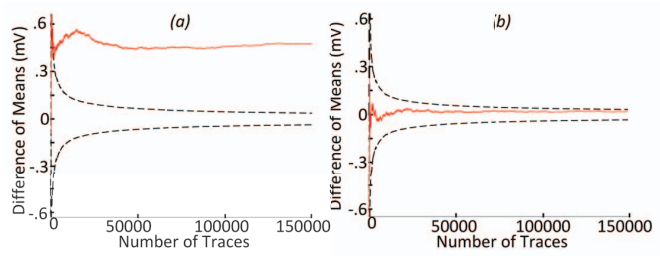


Fig. 5. Difference-of-means based SPA-resistance tests without (a) and with (b) the SPA countermeasure. Figures show the evolution of the difference-of-means at the maximum leak point for 150000 traces. Dashed lines mark the confidence interval for difference-of-means equal to 0 with 99.99%. In the baseline design (a), starting from 200 traces, the attack can successfully estimate the correct guess with a confidence of 99.99%. After applying the SPA countermeasure (b), the SPA leak is mitigated even after 150000 measurements. This test is applied on $r_2[2]$ when $r_2[1]$ and $r_2[0]$ is fixed to '1' and '0', respectively.

low-area architecture, which does not incorporate side-channel defenses, is the *smallest* lattice-based hardware implementation to date. This design reduces the slice cost and does not use a DSP macro. Likewise, the baseline high-performance architecture is the *fastest* lattice-based public-key decryption to date. The SPA defense increases the slice count by 15% and the DPA defense (including PRNG), increases the slice count by 200%. The major cost of the DPA defense is the increase in the number of BRAM. However, even with the included side-channel defenses, our architectures are still relatively small. Recall that we target lightweight applications hence the parameters are set to $n = 256$ and $q = 256$, providing 80-bits of security. By setting $n = 512$, B-RLWE instantiation can achieve higher security levels at the expense of quadrupling the execution time.

## V. Evaluation of Side-channel Attack Resistance

To evaluate the power attacks on a real environment, we mapped our implementations on the SAKURA-G board, which includes a Xilinx Spartan-6 (XC6SLX75-2CSG484C) FPGA. We measure the voltage drop on a 1-Ω resistance and make use of the on-board amplifiers on the SAKURA-G platform. The measurements are taken with a low-end digital USB oscilloscope (PicoScope 2204A) that can sample at 20 ns intervals (50 MS/s). The design is clocked at a constant 1.5 MHz operating frequency.

### A. Testing SPA Resistance

To evaluate SPA-resistance, we perform a difference-of-means test [26] on the baseline and SPA-resistant design. This test aims to differentiate two sets—measurements taken when a particular key bit is equal to '1' vs '0'—by comparing their means separately for each point in the mean power trace. Figure 5 reflects the outcome of this test for up to 150000 measurements. The results validates our SPA countermeasure: At the maximum leak point (ie, at the point where maximum difference occurs in time), while the baseline design shows a large difference in the mean power consumption for two different key values, this effect is mitigated in the SPA-resistant design.

### B. Testing DPA Resistance

To evaluate DPA-resistance, we follow the methodology presented in [14] that applies a three-pronged test and uses the same Pearson correlation in the DPA attack [27]. We target both the input and output registers of the adders (ie. intermediates before and after addition) and make hypothesis based on the Hamming distance of register values. Our security model, similar to previous work, assumes that DPA adversary knows details of hardware like data-flow, parallelization choice, and pipelines.

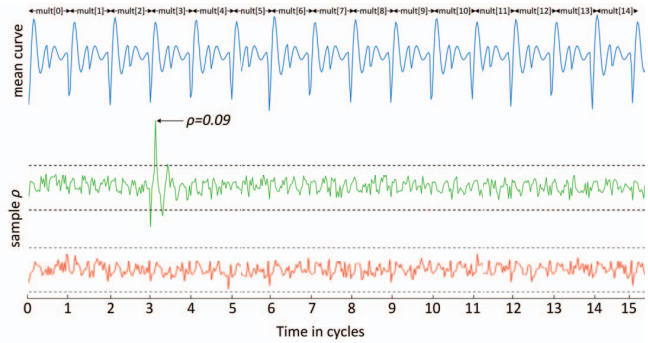


Fig. 6. First-order DPA attack while the masks turned off. This attack makes hypothesis on intermediate computation $mult[3]$ where $mult = c_1 \cdot r_2$. The top trace, depicted in blue, shows the mean power consumption of the FPGA at each evaluation time. In the middle, green trace presents the correlation of correct guess which leaks the value of the key bit $r_2[0]$ at the expected time interval. At the bottom, the red trace displays the correlation of wrong guess; no correlations are observed. Dashed lines mark the confidence interval of $\rho = 0$ with 99.99%.

Adversary can therefore estimate when the targeted computation will likely occur and apply the attack around those clock cycles. We test the capability of a real adversary that has no *a priori* knowledge of the key. In contrast, previous attack evaluations either assume that adversary knows all key segments except the targeted one [14], [15], or assesses a lower bound on the potential of risk [16] via the non-specific t-test [28].

*1) DPA with PRNG off:* We first test the effectiveness of first-order DPA attack by turning off the PRNG in the masked architecture. After this deactivation, PRNG keeps a constant output value of '0'. Therefore, the DPA security of the design will be equivalent to an unmasked, baseline implementation. Figure 6 illustrates the result of DPA attack using 15000 traces. Figure shows the first 15 clock cycles of execution during the second row of polynomial multiplication. DPA attack targets the intermediate value of $mult[3]$; it will make a hypothesis on the fourth coefficient of intermediate sum, based on the value of key bits ($r_2[i-1]$), starting from $r_2[0]$. Recall that, as we described in Section III-C, the two hypothesis $r_2[i-1]=0$ and $r_2[i-1]=1$ will generate different switching activity and DPA attack can check which of these hypothesis correlates with real power measurements. The selection of $mult[3]$ is not special, the same attack works using any other coefficient, of course, by building accurate intermediate guesses for that target coefficient. Results validate the DPA vulnerability. There is indeed a significant correlation for the correct guess occurring at the expected clock cycle, which is statistically sufficient to extract the correct guess with over 99.99% confidence. On the other hand, correlation with incorrect key guess is within the bounds of confidence interval, which demonstrates that, statistically, there is no correlation for the incorrect key guess.

Figure 7 (a) presents the change in the maximum correlation for both correct guess and wrong guess, which is useful to asses how many traces are needed for DPA to succeed with the target confidence. After 2000 traces, the correlation of correct key guess exceeds the confidence interval, while incorrect key guess is always within its bounds. Therefore, this value puts an upper limit on first-order DPA-resistance of the entire system. The figure reflects the attack to the first two bits and demonstrates its extraction within a window of 15 clock cycles. By focusing on other parts of the same power trace (eg. third-row computations) the adversary can successively recover all bits of the key.

*2) DPA with PRNG on:* After verifying the DPA-vulnerability, we activate the PRNG and repeat the same process. This attack

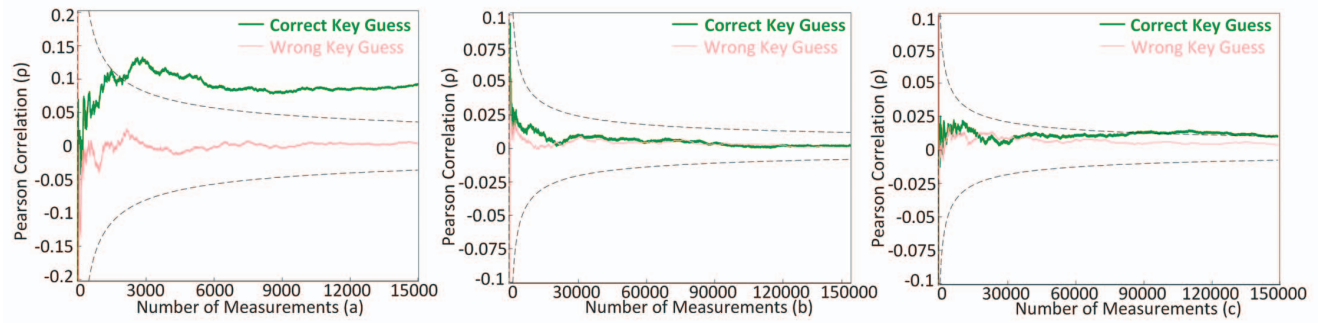


Fig. 7. DPA resilience evaluation using (a) first-order attack on the baseline design, (b) first-order attack on the masked design, and (c) second-order attack on the masked design. Results show the evolution of the correlation coefficient $\rho$ at the maximum leak point, note the increase in number of measurements for (b) and (c). Dashed lines mark the confidence interval of $\rho = 0$ with 99.99%; DPA attack succeeds if correlation crosses this threshold for the correct key guess. Starting from 2000 traces in (a), DPA estimates the correct key. In (b), even after 150000 traces, DPA fails. In (c), starting from 80000 traces, DPA estimates the correct key.

shall not succeed if the DPA defense is sound. Since we cannot prove the non-existence of first-order leaks, we empirically validate, by using 150000 traces, if there is a meaningful correlation for both guesses. Figure 7 (b) displays the result of this experiment. As expected, random initialization and the masked threshold decoder prevents first-order DPA attacks. There is indeed no first-order DPA leak as the correlation of both key guesses converge to 0.

*3) Second-order DPA with PRNG on:* Finally, we test the success of our second-order attack—see section III-C4 for the description of the attack. This process also verifies if the number of traces used in first-order attack is sufficient. Figure 7 (c) shows the result of second-order DPA attack. The correlation of correct key guess is lower than the first-order DPA thus it requires more traces to break the system with the same confidence. Therefore, using approximately 80000 traces, which is substantially more than first-order DPA, second-order attacks can still succeed with 99.99% confidence. Note that we make a strong assumption on the second-order DPA adversary: it can predict the two fixed points in time causing leaks and combine them. In practice, first-order defenses are typically implemented with methods that circumvent this prediction such as inserting random idle states, randomizing the order of executions, and adding dummy operations [29].

## VI. Conclusions

As new post-quantum cryptosystems for next-generation of applications are being formulated, it is essential to investigate the cost and effectiveness of side-channel analysis, especially for lightweight, mobile platforms like RFID tags, smart-cards, and constrained IoT nodes. This paper performs this crypto-engineering tasks, for the first time, for binary-ring-learning-with-errors, a new post-quantum public-key encryption scheme. We propose a hardware design and conduct an in-depth analysis of different side-channel attack vectors including SPA, DPA, and higher-order DPA attacks, provide novel countermeasures, and analyze their efficiency and hardware cost. The results show that, the accumulative nature of polynomial multiplication with binary coefficients, which is unique to B-RLWE, enables both new attack vectors and more efficient countermeasures.

## VII. Acknowledgments

We thank Ashay Rane and Andreas Gerstlauer for helpful discussions. This work is sponsored in part by Lockheed Martin, the National Science Foundation (Award #1453806, #1314709, #1527888, and #1441484), Semiconductor Research Corporation (SRC), and C-FAR: one of the six SRC STARnet Centers, sponsored by MARCO and DARPA. We thank anonymous reviewers for their feedback.

## References

[1] V. Lyubashevsky, C. Peikert, and O. Regev, *On ideal lattices and learning with errors over rings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 1–23.
[2] N. Göttert, T. Feller, M. Schneider, J. Buchmann, and S. Huss, *On the design of hardware building blocks for modern lattice-based encryption schemes*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 512–529.
[3] S. S. Roy, F. Vercauteren, N. Mentens, D. D. Chen, and I. Verbauwhede, *Compact Ring-LWE cryptoprocessor*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 371–391.
[4] T. Pöppelmann and T. Güneysu, "Area optimization of lightweight lattice-based encryption on reconfigurable hardware," in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, June 2014, pp. 2796–2799.
[5] T. Pöppelmann and T. Güneysu, *Towards practical lattice-based public-key encryption on reconfigurable hardware*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 68–85.
[6] T. Pöppelmann, T. Oder, and T. Güneysu, *High-performance ideal lattice-based cryptography on 8-bit ATxmega microcontrollers*. Cham: Springer International Publishing, 2015, pp. 346–365.
[7] Z. Liu, H. Seo, S. Sinha Roy, J. Großschädl, H. Kim, and I. Verbauwhede, *Efficient Ring-LWE encryption on 8-bit AVR processors*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 663–682.
[8] R. De Clercq, S. S. Roy, F. Vercauteren, and I. Verbauwhede, "Efficient software implementation of Ring-LWE encryption," in *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2015, pp. 339–344.
[9] J. Buchmann, F. Göpfert, R. Player, and T. Wunderer, "On the hardness of LWE with binary error: revisiting the hybrid lattice-reduction and meet-in-the-middle attack," in *International Conference on Cryptology in Africa*. Springer, 2016, pp. 24–43.
[10] T. Wunderer, "Revisiting the hybrid attack: Improved analysis and refined security estimates," Cryptology ePrint Archive, Report 2016/733, 2016.
[11] F. Göpfert, C. van Vredendaal, and T. Wunderer, *A Hybrid Lattice Basis Reduction and Quantum Search Attack on LWE*. Cham: Springer International Publishing, 2017, pp. 184–202.
[12] J. Fan and I. Verbauwhede, *An updated survey on secure ECC Implementations: Attacks, countermeasures and cost*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 265–282.
[13] M.-J. O. Saarinen, "Arithmetic coding and blinding countermeasures for lattice signatures," Cryptology ePrint Archive, Report 2016/276, 2016.
[14] O. Reparaz, S. Sinha Roy, F. Vercauteren, and I. Verbauwhede, *A masked Ring-LWE implementation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 683–702.
[15] O. Reparaz, R. De Clercq, S. S. Roy, F. Vercauteren, and I. Verbauwhede, *Additively homomorphic Ring-LWE masking*. Cham: Springer International Publishing, 2016, pp. 233–244.
[16] T. Oder, T. Schneider, T. Pöppelmann, and T. Güneysu, "Practical CCA2-secure and masked Ring-LWE implementation," Cryptology ePrint Archive, Report 2016/1109, 2016.
[17] A. Park and D. G. Han, "Chosen ciphertext simple power analysis on software 8-bit implementation of Ring-LWE encryption," in *2016 IEEE Asian Hardware-Oriented Security and Trust (AsianHOST)*, Dec 2016, pp. 1–6.
[18] R. Primas, P. Pessl, and S. Mangard, "Single-trace side-channel attacks on masked lattice-based encryption," Cryptology ePrint Archive, Report 2017/594, 2017.
[19] J. Buchmann, F. Göpfert, T. Güneysu, T. Oder, and T. Pöppelmann, "High-performance and lightweight lattice-based public-key encryption," in *Proceedings of the 2Nd ACM International Workshop on IoT Privacy, Trust, and Security*, ser. IoTPTS '16. New York, NY, USA: ACM, 2016, pp. 2–9.
[20] M. R. Albrecht, *On Dual Lattice Attacks Against Small-Secret LWE and Parameter Choices in HElib and SEAL*. Cham: Springer International Publishing, 2017, pp. 103–129.
[21] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe, *PRESENT: An Ultra-Lightweight Block Cipher*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 450–466.
[22] R. Lindner and C. Peikert, *Better key sizes (and attacks) for LWE-based encryption*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 319–339.
[23] C. De Cannière and B. Preneel, *Trivium*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 244–266.
[24] T. S. Messerges, *Using second-order power analysis to attack DPA resistant software*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 238–251.
[25] J. Howe, C. Moore, M. O'Neill, F. Regazzoni, T. Güneysu, and K. Beeden, "Lattice-based encryption over standard lattices in hardware," in *Proceedings of the 53rd Annual Design Automation Conference*, ser. DAC '16. New York, NY, USA: ACM, 2016, pp. 162:1–162:6.
[26] P. Kocher, J. Jaffe, and B. Jun, *Differential power analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 388–397.
[27] E. Brier, C. Clavier, and F. Olivier, *Correlation power analysis with a leakage model*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 16–29.
[28] T. Schneider and A. Moradi, *Leakage assessment methodology*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 495–513.
[29] P. C. Kocher, J. M. Jaffe, and B. C. Jun, "Using unpredictable information to minimize leakage from smartcards and other cryptosystems," Dec. 4 2001, uS Patent 6,327,661.