Mask Conversions for $d+1$ shares in Hardware
with Application to Lattice-based PQC

Quinten Norga, Jan-Pieter D'Anvers, Suparna Kundu, and Ingrid Verbauwhede

COSIC, KU Leuven, Leuven, Belgium
firstname.lastname@esat.kuleuven.be

Keywords: Post-Quantum Cryptography · Hardware · (Higher-Order) Masking · Side-Channel Analysis

Abstract. The conversion between arithmetic and Boolean mask representations (A2B & B2A) is a crucial component for side-channel resistant implementations of lattice-based cryptography. In this paper, we present a first- and high-order masked, unified hardware implementation which can perform both A2B & B2A conversions. We optimize the operation on several layers of abstraction, applicable to any protection order. First, we propose novel higher-order algorithms for the secure addition and B2A operation. This is achieved through, among others, an improved method for repeated masked modular reduction and through the X2B operation, which can be viewed as a conversion from any type of additive masking to its Boolean representation. This allows for the removal of a full secure addition during B2A post-processing. Compared to prior work, our $B2A_q$ requires 51/46/45 % less fresh randomness at first through third protection order when implemented in software or hardware.
Secondly, on the circuit level, we successfully introduce half-cycle data paths and demonstrate how careful, manual masking is a superior approach for masking highly non-linear operations and providing first- and high-order security. Our techniques significantly reduce the high latency and fresh randomness overhead, typically introduced by glitch-resistant masking schemes and universally composable gadgets, including HPC3 by Knichel et al. presented at CCS 2022. Compared to state-of-the-art algorithms and masking techniques, our unified and high-throughput hardware implementation requires up to 89/84/86 % fewer clock cycles and 78/71/55 % fewer fresh random bits.
We show detailed performance results for first-, second- and third-order protected implementations on FPGA. Our proposed algorithms are proven secure in the glitch extended probing model and their implementations are validated via practical lab analysis using the TVLA methodology. We experimentally show that both our first- and second-order masked implementation is hardened against univariate and multivariate attacks using 100 million traces, for each mode of operation.

1 Introduction

The security of currently deployed public key cryptographic algorithms is typically based on the Integer Factorization or Elliptic Curve Discrete Logarithm problem. The

threat of large-scale quantum computers is ever-increasing, potentially leaving current algorithms and their implementations vulnerable to potential quantum attacks [63] in the (near) future. The term 'Post-Quantum Cryptography' (PQC) encompasses all alternative cryptographic algorithms, that can resist these attacks and are soon to replace vulnerable algorithms and their implementations.

The National Institute of Standards and Technology (NIST) has recognized the need for replacing the existing public-key standards. Launching an initial PQC standardization effort in 2016 [28] and continuing with an additional Digital Signature (DS) competition in 2023 [32]. Noticeably, lattice-based schemes and their promising security and performance features, are popular candidates for both competitions. Kyber [30], Dilithium [31] and Falcon [57] will be standardized, while seven out of 40 accepted submissions for the PQC DS competition (Round 1) are lattice-based. One of the challenges for the deployment of new post-quantum schemes is protection against (physical) side-channel attacks.

Side-Channel Analysis (SCA) attacks aim at extracting sensitive information from electronic devices performing security-critical applications, by observing the physical characteristics of the calculations. First discovered and published by Kocher [47] in 1996, many types of physical behavior exist and can be abused by adversaries: execution time, instantaneous power consumption [48], Electromagnetic (EM) radiation [36] or temperature and heat dissipation [44]. The security and confidentiality of a cryptographic implementation can be completely broken if its physical characteristics correlate to a secret key, typically called (side-channel) *leakage*. Many insecure implementations, including of lattice-based schemes, have been successfully attacked using side channels [25,42,58,59,65].

As a result, there is an urgent need for developing efficient countermeasures and protection mechanisms. The importance of these protection mechanisms is emphasized by NIST including them as a major evaluation criterion in the PQC standardization process [1]. Protection against SCA attacks is a critical factor for the security of a physical device and remains an open challenge in academia and industry.

Masking is an algorithmic and well-studied approach for protecting cryptographic hardware or software implementations against (passive) EM or power side-channel attacks. Following the concept of secret sharing by Shamir et al. [62], a sensitive variable x is split into $(d+1)$ uniform random shares $(x^{\{i\}})$ for achieving security order d. Each of the shares separately is uncorrelated to the secret and only if an adversary combines information of all $d+1$ shares, it can learn something about the original secret x. Operations are performed on individual shares, resulting in physical characteristics being uncorrelated to the original secret. The masking countermeasure [45,56,60,39,40] is popular because it can provide physical security through formal security and adversary models. These aim at capturing real-world attack scenarios in a precise yet abstract manner. Hence, such theoretical models allow for elegant and high-level reasoning of SCA resilience of designs and (hardware) implementations.

Masking the operations of lattice-based crypto schemes requires a mix of both Boolean and arithmetic mask representations. More precisely, polynomial multiplication and addition are preferably performed on arithmetic shares, whereas hashing inherently is a bitwise operation and thus prefers Boolean masking. Hence, there is

a need for converting between both sharing types: from arithmetic to Boolean (A2B) and Boolean to arithmetic (B2A). These conversions are costly, even more so at higher protection orders, and are one of the major bottlenecks in masked implementations.

Related Work. Masking techniques have been applied to lattice-based cryptography in other work, mostly targeting software implementations. This includes PQC candidates Dilithium [53], Saber [7,22,50,35], Kyber [10,43,35,12] and NTRU [24,49].

A (secure) first-order A2B conversion was originally proposed by Goubin in [38], with Coron et al. proposing higher-order conversions [20,21] for power-of-two moduli ($q=2^k$). They propose to construct the A2B conversion from the Secure Addition (SecADD) operation, which can be seen as an arithmetic addition of two Boolean shared variables.

However, most lattice-based schemes (incl. Kyber and Dilithium) operate on polynomials with coefficients modulo a prime integer q. A secure addition modulo a prime integer q is indicated as SecADD_q and can be constructed from a regular SecADD and additional explicit modular reduction. This expensive procedure typically involves a combination of additional SecADD's and Secure Multiplexers (SecMUX). Techniques for the $\text{A2B}_q/\text{B2A}_q$ operation have been proposed by Barthe et al. [6] and in [61]. Alternatively, methods have been proposed for first performing a modulus conversion from a prime integer to a power-of-two one [12,53]. This allows all masked operations to be performed modulo 2^k, which is typically cheaper than the prime modulo variant where explicit modular reduction is required. More recently, table-based approaches have received more attention as they are becoming viable for high-order conversions [64,23,33], yet not as efficient compared to computational approaches for now. These techniques are out-of-scope for this work.

Contribution. We propose improvements from the algorithmic level down to the circuit level, applicable to arbitrary protection orders. Firstly, we present an improved secure addition for prime moduli, leading to a more efficient A2B_q and B2A_q. Secondly, our novel B2A method does not require performing a secure addition in the post-processing stage. We also leverage the inclusion of memory elements in the datapath as a masking countermeasure to maximize performance, essentially at no additional overhead cost. We evaluate the security of all proposed techniques both formally and experimentally.

Our unified, streaming hardware architecture can be *dynamically* configured to perform any type of mask conversion: $\text{A2B}_{2^k}/\text{A2B}_q/\text{B2A}_{2^k}/\text{B2A}_q$. Our work is directly applicable to any lattice-based PQC scheme, we specifically target Kyber parameters in our unified implementation. To the best of our knowledge, our design strategy results in the lowest overhead cost (latency, fresh randomness and area) compared to the current state-of-the-art. Our findings and contributions are listed as follows:

- SecADD_q requires explicit modular reduction: 2 × SecADD or a SecADD and SecMUX. We propose a novel gadget, $\textbf{SecADDImp}_q$, which utilizes *implicit modular reduction* and is well-suited for As a result, the modular reduction requires strictly 1 × SecADD operation at all protection orders, resulting in up to 25% less fresh random bits and 20% less clock cycles for the A2B_q.
- **B2A** conversions require an A2B operation with expensive *pre-and post-processing stages*. We simplify the post-processing stage by removing the secure addition,

improving (among others) latency and randomness cost through our novel *X2B* gadget. At first protection order, the latency and randomness requirements are halved, for second order the randomness is reduced by 40% and latency by 33%. We also propose a low-latency variant that has order-independent latency, which leverages pre-computation of (random) data shares.
- By introducing **circuit-level** techniques, such as *half-cycle data paths*, we demonstrate how the latency in glitch-resistant masking schemes can be significantly reduced, at nearly no additional cost. We propose targeting highly non-linear operations, like secure additions, resulting in 89% less clock cycles. We also demonstrate that universal composability comes with an (unacceptable) high cost and is unsuitable for masking complex operations like A2B/B2A. Instead, by carefully masking all operations, we significantly reduce the masking overhead (area, latency, randomness): up to 78/71/55 % less fresh randomness for first through third protection order.
- The side-channel resistance of our implementation is formally proven and experimentally verified in our Security Evaluations Lab using the Test Vector Leakage Assessment (TVLA) methodology. Our RTL source code will be made available at the time of publication.

Outline. Section 2 will briefly introduce the notations used throughout this work, give necessary background information and highlight other related works. In Section 3, we present our novel secure gadgets and how they are used to construct secure mask conversions. We discuss and argue about the security and efficiency of our proposed methods and compare them to prior art. Next, in Section 4, we discuss and demonstrate how they are efficiently implemented in hardware. This detailed performance evaluation is followed by the security evaluation of our novel design in Section 5. We conclude our work in Section 6.

2 Background & Preliminaries

2.1 Notation

The bit position (index) is indicated by the subscript, with the LSB at bit 0 (x_0) and MSB at position $k-1$ (x_{k-1}) (k bit data words). All operations and units/costs are expressed in terms of k-bit data words/shares, unless explicitly specified. Rounding up to the next integer is denoted by $\lceil \cdot \rceil$.

2.2 Arithmetic, Boolean and Composite Sharing

At protection order d, a secret value $x \in \mathbb{F}_k^n$ is arithmetically masked by converting it into $d+1$ shares $x^{\{0:d\}}$, such that $x = \sum_{i=0}^{d} x^{\{i\}}$ modulo a predefined integer q. For Boolean masking, the sharing of a secret value x can be reconstructed as $x = \bigoplus_{i=0}^{d} x^{\{i\}}$. Throughout this work, all sharing is considered uniformly random.

We introduce the term composite sharing for secret values that consist of a combination of arithmetic and Boolean shares. $x^{\{a,b\}}$ corresponds to a secret value x consisting of a arithmetic shares, each shared as b Boolean shares. Or alternatively:

$x = \sum_{i=1}^{a} (\bigoplus_{j=1}^{b} x^{\{i,j\}})$ with a (total) masking order $d = (a*b) - 1$. Note that Boolean masking can be seen as a special form of composite masking where all $d+1$ Boolean shares belong to the same arithmetic share ($a = 1, b = d+1$). An arithmetically shared variable consists of $d+1$ arithmetic shares ($a = d+1, b = 1$).

2.3 (Extended) *d*-Probing Model

The most prominent and well-studied adversary and security model, the Ishai, Sahai, and Wagner (ISW) d-probing model [45], aims at capturing the capabilities of real-world adversaries. In such a context, the adversary can probe and observe up to d wires (intermediate values) of an ideal (glitch-less) circuit performing sensitive operations. In this model, a (masked) circuit is d^{th}- order probing secure if and only if the information gained from d (noise-free and instantaneous) probes does not reveal any information of any secret variable.

However, the discrepancy between theoretical and practical security has been shown to be problematic in the case of the original ISW d-probing model. This has resulted in the compromised security of theoretically secure designs and implementations [51,55]. Several extensions to this original model have been proposed, aiming towards (more accurately) capturing different physical effects (naturally) present in digital logic circuits (CMOS) and hardware. These *unintentional and undesired* defaults include:

- **Glitches**: signal transitions due to different delay paths and switching delays in combinational logic.
- **Transitions**: memory contents recombining over time (in sequential clock cycles).
- **Coupling**: signals in separate, but neighboring wires recombining.

An extended (and more robust) security model that introduces more powerful adversary probes was proposed by Faust et al. in [34]. It introduces glitch-extended [51,52], transition-extended [19,3] and coupling-extended probes [26], and incorporate such (natural) physical defects as part of the adversarial model.

2.4 Masking: a Side-Channel Leakage Countermeasure

By introducing masking countermeasures, an attacker can only obtain information about any sensitive value if they have access to all shares at once, while an incomplete set of shares results in statistically random information. Chari et al. proved that increasing the protection order of a circuit d results in an exponential increase in the effort and number of traces required for an attacker to derive sensitive intermediate values [17].

Algorithmic Masking: Threshold Implementations (TI) & Domain-Oriented Masking (DOM) A Threshold Implementation is a masked circuit that is inherently immune against glitches in hardware. It performs a certain function securely on shared data and was introduced by Nikova et al. [56]. A major challenge has been extending this masking scheme for higher protection orders and against multivariate attacks [8,18,9], especially without requiring expensive and tedious redesign/analysis of the entire circuit.

In contrast, following the DOM scheme [40], achieving d-order secure masked circuits requires splitting sensitive variables into $d+1$ (independent) shares. Each share is assigned to an independent share domain, while secure operations operate in a domain-independent fashion. Non-linear operations require that shares cross domain borders, which can be done securely by blinding these shares with fresh randomness and synchronizing them using registers. Interestingly, this strategy can be trivially extended and applied for any protection order which is why we utilize it in this work.

Creating a complex circuit, consisting of multiple such secure (DOM) gates, requires careful analysis and introduction of countermeasures to achieve d-probing security. In essence, the non-completeness property should always be respected and additional mask refresh stages should be introduced in order to withstand multivariate attacks. For hardware circuits, several security notions for composability have been proposed: Non-Interference (NI) [4] and Strong Non-Interference (SNI) [5] in the presence of glitches.

Definition 1 (t-(Strong)-Non-Interference [5]). *A gadget with one output sharing and m_i input sharings is t-Non-Interferent (t-NI) (resp. t-Strong Non-Interferent (t-SNI)) if any set of at most t_1 probes on its internal wires and t_2 probes on wires from its output sharings such that $t_1+t_2 \leq t$ can be simulated with t_1+t_2 (resp. t_1) shares of each of its m_i input sharings.*

Gate-Level Masking & Hardware Private Circuits (HPC) A different approach is based on 'trivial composability' and the security notion of Probe-Isolating Non-Interference (PINI) [13] and HPC gadgets, which are derived from the DOM scheme. Introduced by Cassiers et al. in [16], the proposed gadgets can be instantiated at arbitrary protection orders and trivially combined into a larger circuit. In general, trivial composability and its low verification cost and guaranteed d-probing security comes at a high (overhead) cost, due to being overly conservative in applying certain countermeasures. We target 'optimized composition' in the glitch and transition-robust probing model in this work and as a result the overhead, introduced when masking A2B/B2A operations, is significantly reduced compared to strictly using (PINI) HPC gadgets.

2.5 Masking Lattice-based PQC: ML-KEM

Mask conversions are required when masking any lattice-based PQC scheme. Figure 1 illustrates the impact of different mask domains and the need for switching between both during several sub-operations for Kyber (or ML-KEM). The decryption procedure requires performing Boolean operations, like binomial sampling and hashing. The re-encryption stage requires performing polynomial multiplication, after which a masked comparison is performed. Note again, these conversions are extremely costly and result in being (one of) the main contributor(s) of run time latency. For the pseudocode of the full algorithms of all (future) PQC standards, we refer to their Round 3 Submissions or the official (drafts of the) NIST FIPS standards [29] respectively.

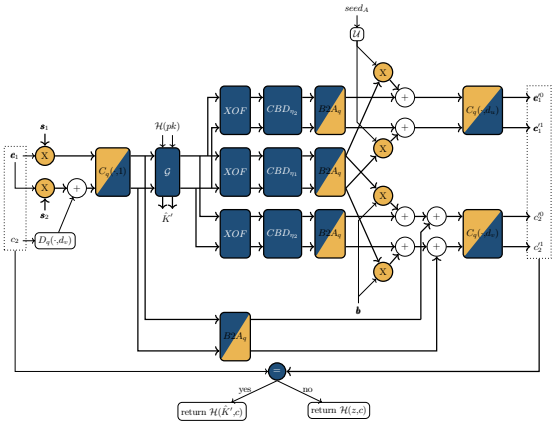


Fig. 1: The masked Decapsulation procedure for Kyber, or ML-KEM (FIPS 203). Operations that require Boolean masking are highlighted in blue, operations that prefer arithmetic masking are highlighted in yellow. Mask conversions are required to convert between these representations. [35]

3 Optimized Secure Gadgets for Mask Conversions

The following section will follow a bottom-up approach. First, we optimize the Secure Addition (SecADD$_q$) by focusing on the masked modular reduction. Our strategy can be directly applied for arbitrary moduli q and arbitrary protection orders d, including Kyber ($q=3329$) and Dilithium ($q=8380417$). By performing the modular reduction implicitly, our approach requires strictly 2 SecADDs in total, instead of an additional (third) SecADD or SecMUX.

Secondly, we propose two novel B2A gadgets: one low-latency (B2APreCompute) and one area-efficient (B2AComposite) variant. The composability of all proposed gadgets is proven secure in the glitch extended probing model and tested in TVLA setting. We introduce a new primitive $X2B$, which eliminates a full SecADD/SecADD$_q$ from the post-processing stage, reducing the latency and randomness requirements at all protection orders.

Interestingly, all proposed gadgets lend themselves to highly flexible and efficient hardware implementations, which we demonstrate in the subsequent section.

3.1 Secure Addition (SecADD)

The secure addition is equivalent to performing an arithmetic addition ($s=x+y \mod q$) in the Boolean domain (Equation 1). As we will demonstrate, it serves as the primary building block for higher-order mask conversions.

$$s^{\{0:d\}}=x^{\{0:d\}}+y^{\{0:d\}} \bmod q=\bigoplus_{i=0}^{d} x^{\{i\}}+\bigoplus_{i=0}^{d} y^{\{i\}} \bmod q \tag{1}$$

For power-of-two moduli ($q=2^k$), the modular reduction is taken care of implicitly during computation and prime moduli require explicit (masked) modular reduction. We first introduce strategies proposed in literature, including SecMUX-based masked modular reduction. We demonstrate its high overhead cost and highlight a state-of-the-art SecMUX-less strategy, applicable to first-order implementations. We extend this method for arbitrary protection orders and demonstrate why it is not a suitable strategy when performing multiple secure additions in succession. Finally, we introduce the optimized $SecADDImp_q$ gadget. The 'implicit' modular reduction of prime moduli allows them to be directly and efficiently chained together, which is required for any mask conversion operation.

SecADD$_q$ = 2 × SecADD + SecMUX, from [6] Barthe et al. introduced a simple, yet costly method for performing the SecADD$_q$ at arbitrary protection orders. It requires calculating both $s=x+y$ and $s'=x+y-q$ securely, one of which will be in range $[0,q)$. A costly SecMUX (Equation 2) securely selects the desired shared data (s or s') that lies in the $[0:q-1]$ interval, based on the carry bits c.

$$\text{SecMUX}(s^{\{0:d\}},s'^{\{0:d\}},c_k^{\{0:d\}}) = \text{SecXOR}(\text{SecAND}(s,c),\ \text{SecAND}(s',\ \text{SecNOT}(c))) \tag{2}$$

This secure gadget requires two k-bit SecAND, one SecXOR and one SecNOT operations, whereas the secure gates required to construct the secure adder itself typically operate on smaller (1 or $k/2$ bit) chunks of data.

SecADD$_q$ = 3 × SecADD, from [35] Subsequently, Fritzmann et al. introduced a method for performing a first-order SecADD$_q$, which does not involve a SecMUX gadget. By pre-processing the input data, which requires access to the initial masking of either input y (or x), the SecMUX operation can be removed.

In practice, we need one of the inputs to be in range $[-q,0)$. This can be achieved by subtracting q from one of the inputs before it is shared: $y'=y-q$. The first SecADD operates on y' and x (or y and x') and computes $z=x+y'$ (Alg. 1, Line 1). Next, a correction term c' is constructed and is added to this intermediate result z, ensuring that the result of the second SecADD $s=z+c'$ lies in $[0,q)$ (Line 4). c' is computed based on the carry bits of the intermediate result z: $c'=z_{k-1}\cdot q$. If z is still negative, indicated by the carry bit of the two's complement representation being equal to one, c' will be q. As such q is securely added back to z during the second SecADD. If the intermediate result is greater than zero, c' will remain zero and the result will be in $[0:q)$.

The carry bits of the intermediate result z essentially acts as a select signal for a standard MUX, determining if zero or q is securely added back to z (Line 3). In conclusion, the result, which lies in $[0,q)$, is directly computed during the second SecADD, based on the output of the first secure addition:

$$s=z+c'=x+y-q(+q)=x+y \bmod q \tag{3}$$

Note that the modular reduction and the construction of c' now is a linear (e.g. mask-friendly) operation. There is no longer any need to explicitly select the correct

Algorithm 1 SecADD$_q$ (without SecMUX) (extended from [35])

Input Parameter : q ▷ q is prime
Input Data : $x^{\{0:d\}}$ and $y'^{\{0:d\}} = y^{\{0:d\}} + (2^w - q)$ ▷ Initial masking.
Output Data : $s^{\{0:d\}}$ such that $s = x + y \bmod q$

1: $z^{\{0:d\}} \leftarrow \text{SecADD}(x^{\{0:d\}}, y'^{\{0:d\}})$
2: $c_0^{\{0:d\}} \leftarrow z^{\{0:d\}} \gg (k-1)$ ▷ Carry bit (share-wise).
3: $c'^{\{0:d\}} \leftarrow c^{\{0:d\}} \cdot q$ ▷ Share-wise.
4: $s^{\{0:d\}} \leftarrow \text{SecADD}(z^{\{0:d\}}, c'^{\{0:d\}})$

result, using a SecMUX. It is share-wise: no domain borders are crossed, removing the need for fresh random shares or delay elements, except for the secure additions themselves. As such, we can extend this method to arbitrary protection orders, as demonstrated in Algorithm 1 by generalizing and duplicating the operation in each share domain.

The main issue with this method arises when one of the Boolean masked inputs is not in range $[-q,0)$. This is the case if the output of a SecADD$_q$ operation, in range $[0,q)$, is directly used as the input for another SecADD$_q$, as is the case during an A2B (and B2A) conversion. To subtract q from a Boolean masked variable, an additional secure addition with $(2^k - q)$ is required, as proposed in [14] (Algorithm 11). As a result, a full SecADD$_q$ now requires three SecADDs, which is costly in the context of mask conversions.

SecADDImp$_q$ = 2 × SecADD We now propose our SecADDImp$_q$ gadget (Algorithm 2), which requires only two SecADD operations even when performing multiple secure additions successively, by exploiting 'implicit' modular reduction. We also extend the notion of a secure addition so that two outputs are possible, one of which is calculated in practice. One is calculated if the SecADD is one of many subsequent secure additions that need to be calculated, or the other if it is the final one.

For the algorithm in the previous section, if the secure addition is the final operation, the goal is to calculate $s = x + y$ which lies in $[0,q)$ with x and y consisting of $d+1$ Boolean shares. As described above, this can be achieved using strictly two SecADDs (Eq. 3 & Alg. 1) if one of the inputs is pre-processed: $y' = y - q$.

If another SecADD$_q$ needs to be performed subsequently, the result of the operation needs to be pre-processed (by subtracting q) as it is one of the inputs of the next SecADD$_q$. Instead of doing this explicitly, our novel gadget SecADDImp$_q$ allows for this to be computed implicitly. The result will now be $s' = s + (2^k - q)$, which lies in $[-q,0)$, allowing for the output to be used directly as an input for the next SecADD$_q$.

More specifically, first $z = x + y' (= x + y - q)$ is calculated. Using this intermediate result z, a different correction term c' is constructed in Line 4: $c' = (\sim z_{k-1}) \cdot (-q)$. This term is eventually added together with the intermediate result, in order to obtain the final result: $s' = z + c'$. Intuitively, if the intermediate result lies in $[-q,0)$, the unshared correction term should be zero. If positive, $-q$ should be added back to the intermediate result in order to ensure the final result (s') lies in $[-q,0)$. This is

achieved by using the Boolean inverse[1] of the carry bits (z_{k-1}) as a share-wise select signal for a MUX. If an uneven amount of carry bits are one, the unshared value is negative and an even amount of shares in c' is set to $-q$. As a result, the unshared c' is equal to zero, which is desired.

Algorithm 2 SecADDImp$_q$

Input Parameter :q ▷ q is prime
Input Data :$x^{\{0:d\}}$ and $y'^{\{0:d\}} = y^{\{0:d\}} + (2^w - q)$ ▷ Initial masking.
Output Data 1 :$s^{\{0:d\}}$ such that $s = x + y \bmod q$
Output Data 2 :$s'^{\{0:d\}}$ such that $s' = s + (2^k - q)$

1: $z^{\{0:d\}} \leftarrow \text{SecAdd}(x^{\{0:d\}}, y'^{\{0:d\}})$
2: $cc_0^{\{0:d\}} \leftarrow z^{\{0:d\}} \gg (k-1)$ ▷ Carry bit.
3: $c^{\{0:d\}} \leftarrow cc^{\{0:d\}} \cdot q$ ▷ Share-wise.
4: $c'^{\{0:d\}} \leftarrow (\sim cc^{\{0:d\}}) \cdot (-q)$ ▷ SecNOT; Share-wise.
5: $s^{\{0:d\}} \leftarrow \text{SecAdd}(z^{\{0:d\}}, c^{\{0:d\}})$ ▷ Final Output/Operation.
6: $s'^{\{0:d\}} \leftarrow \text{SecAdd}(z^{\{0:d\}}, c'^{\{0:d\}})$

This extension allows for multiple secure additions to be directly chained in succession, without the need for repeated and explicit pre-processing of one of the inputs and thus strictly requiring two SecADDs. Such a thing is useful for A2B$_q$/B2A$_q$ conversions. The only time when access to the initial masking is required is one of the inputs, y, of the very first secure addition of which many are performed in succession. The input is corrected with $-q$ before the initial sharing, so that $\bigoplus_{i=0}^{d} y'^{\{i\}} = y - q$. If not possible, a one-time pre-processing using the SecADD is required.

Robust Probing Security: We now show that the SecADDImp$_q$ gadget is correct and glitch-extended probing secure considering the leakage effects from Section 2.3.
Correctness. For prime q, explicit modular reduction is performed on $z = x + y' = x + y - q \in [-q : q-2]$, because y' lies in $[-q : -1]$.

- $z \in [-q : -1]$: $c = q$, so $s = z + q$ lies in $[0 : q-1]$. $c' = 0$, because an uneven amount of carry bits cc will be '1' as both x and y are mod q. This ensures $s' = z + 0$ lies in $[-q : -1]$.
- $z \in [0 : q-2]$: $c = 0$, so $s = z + q$ lies in $[0 : q-2]$. $c' = -q$, because an even amount of carry bits cc will be '1' as both x and y are mod q. This ensures $s' = z - q$ lies in $[-q : -2]$.

The algorithm returns either a value modulo q, or (mod q) - q.
Security. To argue about the higher-order security of Algorithm 2, we prove it to be $t-$NI with $t+1$ shares. This provides resistance against a probing adversary with t probes and allows the use of the gadget in larger compositions.

[1] SecNOT ($\sim$) on Boolean shared data is equivalent to performing binary invert on a single share.

Theorem 1 (). *The gadget SecADDImp$_q$ (Algorithm 2) is t-NI secure.*

Proof. We model Algorithm 2 as a sequence of t-(S)NI gadgets. For simplicity, we model the linear operations in Lines 2, 3 and 4 as t-NI gadgets in hardware, which can be trivially shown as the operations process the inputs share-wise, actually isolated in the gadget per domain. In the glitch-extended model, the secure addition is t-NI. An extended probe at the output of the secure adder, which is the most powerful one, can be simulated only with the input shares [20], making the SecADDImp$_q$ gadget t-NI.

3.2 B2A

A method for converting $d+1$ Boolean shares to $d+1$ arithmetic shares (mod 2^k) was introduced in [21] and extended for arbitrary moduli q in [6]. Generally speaking, the B2A conversion is equivalent to an A2B operation with additional (costly) pre- and post-processing stages. We make several modifications to this procedure and propose two new variants: A low-latency approach which requires pre-computation during the randomness generation (Section 3.2) and a more efficient, standard B2A conversion routine (Section 3.2). We can reduce the latency and randomness cost, mainly by modifying the pre-and post-processing routines so that essentially only the A2B (e.g. $X2B$) operation remains. Correctness and security proofs are also provided. We conclude by comparing the overhead of published work and our methods in Table 1.

Pre-Compute B2A (Low Latency) The goal of the B2A operation is to convert $d+1$ Boolean shares $B^{\{0:d\}}$ to $d+1$ arithmetic shares $A^{\{0:d\}}$. The first d output shares are newly sampled, random shares $R_A^{\{0:d-1\}}$: $A^{\{0:d-1\}} = R_A^{\{0:d-1\}}$. The final output share $A^{\{d\}}$ is computed during the remainder (and majority) of operations, which mainly involve the d previously sampled random, arithmetic shares $R_A^{\{0:d-1\}}$. In the following sections, we will denote with superscript-free variables (e.g. R_A) the unshared value: $R_A = \sum_{i=0}^{d-1} R_A^{\{i\}} \bmod q$.

More specifically, the final share $A^{\{d\}}$ is securely computed as $B - R_A$, so the output A equals $R_A + (B - R_A) = B$. In practice, $R_A^{\{0:d-1\}}$ is first converted to the Boolean domain (using an A2B), resulting in $R_B^{\{0:d\}}$: $\bigoplus_{i=0}^{d} R_B^{\{i\}} = \sum_{i=0}^{d-1} - R_A^{\{i\}} \bmod q$. Next, $B + R_B$ is computed using a secure addition, as both are Boolean shared operands. The only remaining step is to securely convert this result, using the FullXOR gadget [21], into one share: $A^{\{d\}} = \bigoplus_{i=0}^{d} (B+R_B)^{\{0:d\}}$.

We now remark that the A2B only involves random data and can be computed when the randomness is sampled and temporarily stored in memory (Algorithm 3). The main low-latency B2A algorithm now only involves operations on the actual, secret input data: a single SecADD/SecADD$_q$ and FullXOR need to be computed at run-time, independent of protection order (Algorithm 4) in order to obtain the final share $A^{\{d\}}$.

Next, propose an optimization by noting that computing the negation of R_A, input of the A2B, is required. This negation is typically performed in the arithmetic domain [21,6,35], which requires a share-wise effort ($\mathcal{O}(d)$). We propose performing this negation in the Boolean domain (SecNOT), requiring the Boolean inversion of only a single share ($\mathcal{O}(1)$). The relation between both operations is described in

Algorithm 3 B2APreCompute

Input Data : q ▷ $q=2^n$ $(n=1..k)$ or prime
Output Data : $R_A^{\{0:d-1\}}$ and $R_B^{\{0:d\}}$ such that $\bigoplus_{i=0}^{d} R_B^{\{i\}} = \sum_{i=0}^{d-1} -R_A^{\{i\}} \bmod q$

1: $R_A^{\{0:d-1\}} \leftarrow \text{Rand}([0:q-1])$
2: **if** q is prime **then** ▷ Modify initial masking for SecModALL.
3: **for** $i=0,2...d-2$ **do**
4: $z^{\{i\}} \leftarrow A^{\{i\}}$
5: $z^{\{i+1\}} \leftarrow A^{\{i+1\}} - q$ ▷ $-q$ correction.
6: **end for**
7: **else**
8: $z^{\{0:d-1\}} \leftarrow A^{\{0:d-1\}}$
9: **end if**
10: $z^{\{d\}} \leftarrow (2^w - 1)$ ▷ -1 in twos complement
11: $y^{\{0:d\}} \leftarrow A2B(z^{\{0:d\}})$
12: $R_B^{\{0:d\}} \leftarrow \sim y^{\{0:d\}}$ ▷ SecNOT, $R_B = A2B(-R_A)$

Equation 4. As R_A only consists of d shares, the $d+1$-th share can be set to -1 (Line 10, Algorithm 3). The negation is now achieved through a SecNOT of the Boolean representation of $R_A - 1$, which is computed during the A2B (Line 12). This ensures the equivalent result ($R_B = -R_A$) is obtained.

$$\sim x = -(x+1) \tag{4}$$

Algorithm 4 B2A [Low Latency]

Input Parameter : q ▷ $q=2^n$ $(n=1..k)$ or prime
Input Data : $B^{\{0:d\}}$
Input Data : $R_A^{\{0:d-1\}}$ and $R_B^{\{0:d\}}$ such that $\bigoplus_{i=0}^{d} R_B^{\{i\}} = \sum_{i=0}^{d-1} -R_A^{\{i\}} \bmod q$
Output Data : $A^{\{0:d\}}$ such that $\bigoplus_{i=0}^{d} B^{\{i\}} = \sum_{i=0}^{d} A^{\{i\}} \bmod q$

1: $z^{\{0:d\}} \leftarrow \text{SecADD/SecADD}_q(B^{\{0:d\}}, R_B^{\{0:d\}})$ ▷ Algorithm 2
2: $A^{\{0:d-1\}} \leftarrow R_A^{\{0:d-1\}}$
3: $A^{\{d\}} \leftarrow \text{FullXOR}(z^{\{0:d\}})$ ▷ [21]

Robust Probing Security: We now show that the B2A Low Latency gadget is correct and glitch-extended probing secure considering the leakage effects from Section 2.3.
Correctness. Algorithm 3 provides a Boolean (R_B) and arithmetic (R_A) representation of randomly sampled data. y is equivalent to $(R_A - 1)$ in the Boolean domain. The SecNOT operation ensures R_B is equal to $-(R_A - 1 + 1) = -R_A$, which is correct. In Algorithm 4, z is equal to $(B - R_A)$ through secure addition. The FullXOR operation combines all shares into a single one, ensuring the output A is equal to $R_A + B - R_A = B$ in a shared format, which is the correct result.

Security. To argue about the higher-order security of Algorithm (3 and) 4, we prove it to be $t-$SNI with $t+1$ shares. This provides resistance against a probing adversary with t probes. All pre-computations are only involving randomly sampled data, and can hence be perfectly simulated without any extended probes. Hence, we focus on the actual calculations in Algorithm 4.

Theorem 2 (). *The gadget B2A Low Latency (Algorithm 4) is t-SNI secure.*

Proof. The first d shares of the output A can be perfectly simulated without any extended probes, as they are randomly sampled. The final share (Line 3) is calculated with a t-SNI FullXOR gadget [21]: the SecADD/SecADD$_q$ gadget is t-NI, and is refreshed with a t-SNI refresh with a pre-computed all-zero input sharing, with a one-cycle latency in hardware [16]. These refreshed (registered) shares are combined (XOR'd) into one share, which can be perfectly simulated without any extended probes. As a result, the low-latency B2A gadget is t-SNI. □

Our Improved B2A Method Now, we propose a more efficient B2A method in Algorithm 5, which does not require any secure addition in the post-processing stage and does not rely on pre-computation. Compared to the state-of-the-art, the operation count is reduced and performance (latency, area and fresh randomness) is significantly improved, in both software and hardware implementations.

As described in the previous section, the goal of the B2A operation is to convert $d+1$ Boolean shares $B^{\{0:d\}}$ to $d+1$ arithmetic shares $A^{\{0:d\}}$. Again, $A^{\{0:d-1\}} = R_A^{\{0:d-1\}}$ is randomly sampled and the final share $A^{\{d\}}$ is computed as $B-R_A$. In practice, we introduce a new primitive $X2B$, which is a variant of the $A2B$ but operates on $d+1$ *composite* shares (a mix of arithmetic and Boolean shares): $z^{\{0:d\}}$. The composite shares $z^{\{0:d\}}$ are arithmetically shared, but each individual share $z^{\{i\}}$ consists in turn of a number of Boolean shares, that is $\sum_{i=0}^{d}(\bigoplus_{j=0}^{d} z^{\{i,j\}})$. During the $X2B$ operation these composite shares are added together, similar to the addition of strictly arithmetic shares during the $A2B$.

Our improved B2A is constructed by setting the $X2B$ input $z^{\{0:d-1\}}$ equal to $R_A^{\{0:d-1\}}$ and $z^{\{d\}}$ to $\sim B$. Note that B consist of $d+1$ Boolean shares which means that z is compositely shared, consisting of d arithmetic shares and one Boolean sharing. The $X2B$ is required to convert the compositely shared input, equal to $R_A+(\sim B)=R_A-B-1$, to a Boolean sharing. A negation in the Boolean domain is performed on the $X2B$ output to obtain the desired result $B-R_A$. As during regular post-processing, $d+1$ Boolean shares are combined into a single share using a FullXOR to obtain the final output share $A^{\{d\}}$.

This approach is an improvement over the state-of-the-art, as $B-R_A$ is directly computed during the $X2B$ and thus one does not need to perform the explicit secure addition during post-processing. In the original method one needs to compute one A2B and one secure addition, while our improved method requires only the X2B operation. The X2B operation has the same computational cost as A2B for first and second security order, and only slightly higher than A2B for higher orders. For first-order implementations, the amount of secure additions is halved, for second order one-third of secure additions is removed, etc. For prime moduli,

we give a comparison in Table 1. In all cases we obtain a more efficient end result.

Algorithm 5 B2AComposite

Input Parameter/Data : q ▷ $q{=}2^n$ $(n{=}1..k)$ or prime
Input Data : $B^{\{0:d\}}$
Output Data : $A^{\{0:d\}}$ such that $\bigoplus_{i=0}^{d} B^{\{i\}} {=} \sum_{i=0}^{d} A^{\{i\}} \bmod q$

1: $A^{\{0:d-1\}}, R_A^{\{0:d-1\}} \leftarrow \text{Rand}([0:q{-}1])$
2: **if** q is prime **then** ▷ Modify initial masking for SecADD$_q$.
3: **for** $i{=}0,2...d{-}2$ **do**
4: $z^{\{i\}} \leftarrow R_A^{\{i\}}$
5: $z^{\{i+1\}} \leftarrow R_A^{\{i+1\}} - q$ ▷ $-q$ correction.
6: **end for**
7: **else**
8: $z^{\{0:d-1\}} \leftarrow R_A^{\{0:d-1\}}$
9: **end if**
10: $z^{\{d,0:d\}} \leftarrow {\sim} B^{\{0:d\}}$ ▷ $z{=}R_A{-}B{-}1$
11: $y'^{\{0:d\}} \leftarrow X2B(z^{\{0:d\}})$
12: $y^{\{0:d\}} \leftarrow {\sim} y'^{\{0:d\}}$ ▷ $y{=}{-}z{-}1{=}B{-}R_A$
13: $A^{\{d\}} \leftarrow \text{FullXOR}(y^{\{0:d\}})$ ▷ $A^{\{d\}}{=}y$

Robust Probing Security: We now show that the B2AComposite gadget is correct and glitch-extended probing secure considering the leakage effects from Section 2.3.
Correctness. For the correctness of Algorithm 5 we largely refer to the proof in the previous section. The operations in both B2A methods are identical but merged into a single $X2B$ operation. y' is equivalent to $A{+}({\sim}B){=}A{-}B{-}1$. As a result y is equal to $({-}A{+}B{+}1){-}1$ or $B{-}A$. All shares are securely combined into a single one in Line 13, ensuring the output A is equal to $A{+}B{-}A{=}B$, which is the same data but shared differently.
Security. To argue about the higher-order security of Algorithm 5, we prove it to be $t{-}$SNI with $t{+}1$ shares. This provides resistance against a probing adversary with t probes.

Theorem 3 (). *The gadget B2AComposite (Algorithm 5) is t-SNI secure.*

Proof. All linear operations, including the SecNOT gadget in Lines 10 and 12, can be modeled as t-NI gadgets. This can be trivially shown as the operations process the inputs share-wise, actually isolated in the gadget per domain. The $X2B$ gadget is $t{-}$NI, as is the $A2B$ operation [61]. The FullXOR consists of a $t{-}$SNI refresh with all-zero input, ensuring the output can be perfectly simulated without any extended internal or input probes. □

[2] Pre-Compute

Table 1: B2A$_q$ Cost/Overhead Comparison (d+1 shares, k-bit words).

	Order	# SecADD	# SecMUX
[6]	d	$2(d+1)$	$d+1$
[35]	1	4	**0**
[14]	d	$2+3d$	**0**
B2A Precompute (Alg. 3 & 4)	d	$\mathbf{2} + 2(d+1)^2$	**0**
B2AmodALL (Alg. 5)	d	**2d**	**0**

4 High-Throughput & Low-Randomness Mask Conversions in Hardware

In this section, we first introduce our strategy and novel techniques for implementing the proposed secure gadgets and then demonstrate how (any d, any q) A2B/B2A operations can be combined in a unified accelerator in hardware. When implementing secure gadgets in hardware, it is preferred to maximally exploit its implicit parallelism by operating on all shares at once ('SIMD'). We introduce a high-order, layer-based implementation, in which all shares are operated on at once. It is highly flexible: by appropriately setting control signals, the data path is modified and the desired mask conversion is performed. The implementation is also highly compact, all four types of mask conversions can be performed on the same, single hardware unit with additional pre- and/or post-processing dynamically activated depending on the exact operation.

Next, we propose and discuss the introduction of circuit-level techniques to reduce the latency of masked implementations, at minimal additional cost. We exploit the mandatory inclusion of registers, preventing leakage in the presence of glitches, to increase performance. We provide the SystemVerilog source code for all designs, which we experimentally verify to be first- and high-order secure in the next section.

4.1 Secure Addition: SecADD$_x$

Several previous works have proposed algorithms and secure implementations of the secure addition. A masked ripple-carry adder was proposed by Coron et al. [21], and more hardware-focused parallel prefix-type adders by Bache et al. [2]. Essentially any type of adder can be selected and transformed into a masked 'SecADD' variant by carefully replacing its components with secure gadgets/gates. To the best of our knowledge, there exists no work (& implementation) targeting higher-order protection or maximally exploiting the parallel nature of hardware. We propose a Brent-Kung adder architecture [11] because it is more area-efficient than a Kogge-Stone or Schlansky architecture, at the cost of an increased latency yet high throughput.

We propose a unified and generic (hardware) design and implementation strategy, illustrated in Figure 2: 'SecADD$_x$'. The fully unrolled and pipelined implementation can compute the secure addition for power-of-two and prime moduli (using our SecADD$_q$, Algorithm 2), on the same hardware by setting the appropriate control signals and using dynamic reconfiguration. Again, this is relevant because both are necessary operations during a masked decapsulation of Kyber and can now be

performed using the same physical circuit. No physical hardware instances are reused over time for the same (shared) coefficient and only operate on data assigned to that specific share domain, avoiding transitional leakage in memory elements. Our novel streaming approach, in which data flows through the entire pipelined circuit, ensures all logic is maximally active.

Two SecADDs are instantiated, which are chained when the modulus is prime. As described in the previous section, either s or $s'=s-q$ can be calculated, depending on whether that secure addition is the final one or not. Alternatively, for the secure addition modulo a power-of-two integer, we propose using both SecADDs in parallel instead of one being idle in this mode. Throughput is doubled in this mode, as two shared data words (x_1,y_1 and x_2,y_2) can be accepted each clock cycle.

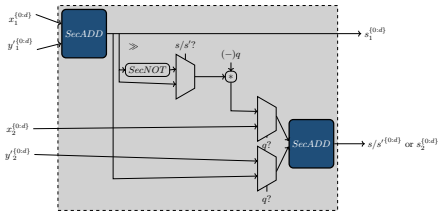


Fig. 2: Block diagram of SecADD$_x$: reconfigurability during operation (at runtime). If q is power-of-two, two inputs can be accepted (and outputs are produced) each clock cycle. If q is prime, the latency doubles and a single input is accepted each clock cycle, because both secure adders are chained. The output can be mod q or (mod q) - q.

4.2 Mask Conversions: $A2B_x/B2A_x$

We propose a fully pipelined, high throughput design which is related to the fact that many (different) mask conversions need to be performed in lattice-based PQC schemes. Specifically for Kyber, 256 coefficients per polynomial, and many polynomials during the entire masked decapsulation process require A2B's and B2A's. This section will first focus on the efficient and secure implementation of the A2B operation in hardware, as it is also the major component of the B2A operation. The B2A-specific pre- and post-processing will be discussed after.

A2B Layer We introduce the A2B Layer, the primary building block for constructing A2B (and B2A) conversions efficiently in hardware. It aims at maximizing the parallelism available in hardware by operating on all shares simultaneously. As described in Equation 5, each A2B layer doubles the level of Boolean shares and halves the level of arithmetic sharing of a masked variable. The total share count remains unchanged ($d+1$) from input to output, but the type of sharing does. By shares moving in parallel through $L=\lceil \log(d+1) \rceil$ 'A2B layers', all shares are converted to Boolean shares in parallel. This is in stark contrast with proposed strategies, which are fundamentally sequential and only operate on two shares at once.

$$x^{\{2a,b\}} = \text{A2BLayer}(x^{\{a,2b\}}) \tag{5}$$

An A2B Layer consists of two major operations: an expansion of Boolean shares ('Expand') and a reduction of arithmetic shares ('SecADD$_x$'). The expand operation doubles the number of Boolean shares of each arithmetic share using fresh randomness [21], after which the composite shares are securely added together in a pairwise fashion. Multiple layers can be instantiated and placed in succession for higher-order conversions, each operating on all shares in parallel while they move through all layers in a streaming fashion. The proposed A2B strategy (in hardware) is illustrated in Figure 3a and 3b for a two- and four-share conversion. All logic is maximally occupied and active, as the data streams through the instantiated logic. No logic is reused for a single shared input, avoiding any transitional leakage.

In the case of a prime modulus q, modular reduction is required. Before entering the first layer, during the pre-processing stage, the initial masking of one of the inputs for each SecADD$_x$ instance is explicitly corrected with $-q$. In between layers, the modulus reduction will be performed implicitly (SecADD$_q$, Algorithm 2).

(a) 2-share A2B. (b) 4-share A2B.

Fig. 3: Layer-based approach for first- and third-order A2B conversions: : $\bigoplus_{i=0}^{d} B^{\{i\}} = \sum_{i=0}^{d} A^{\{i\}} \bmod q$. Each layer (in gray) doubles the level of Boolean sharing and halves the level of arithmetic sharing. For prime moduli q, modular reduction is implicitly taken care of during computation in layers (s and s' are interleaved).

B2A An architecture diagram of the $X2B$ gadget, the main component of our B2AComposite gadget is provided in Figure 4 which mostly uses the instantiated A2B datapath. Again, some pre-processing is required before the initial masking. And a FullXOR, including MaskRefresh, is required as post-processing. Our approach results in the A2B and B2A operations now having identical latency when implemented in hardware. Interestingly, a significant portion of the computation (on randomly sampled data) can be a target for pre-computation, highlighted in yellow. This optimization is left as future work.

It is important to note that the minimal share count is no longer achieved during the $X2B$ operation at higher protection orders. Internally, the secure addition that

operates on the input B, is of order d. This means an additional $d+1$ share secure adder needs to be instantiated for $d \geq 3$ in all but the final layer, which always operates on $d+1$ shares. These adders are not naturally present there as they are not required for an A2B and are only active during the B2A operation. Yet, the randomness, area cost and latency of the full B2A is still significantly reduced.

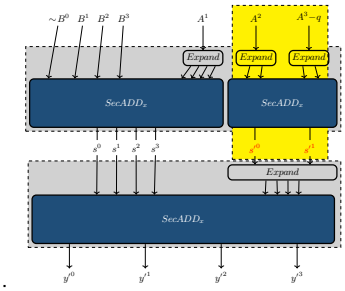


Fig. 4: High-order $X2B$: layers operate on both randomly sampled data (right) and shared inputs (left). The right side (yellow) can be pre-computed. $y=x-A$ is directly computed during $X2B$, removing the need for a secure addition in post-processing.

4.3 Half-Cycle Path

The Domain-Oriented Masking (DOM) scheme and Threshold Implementations (TI) both guarantee glitch-immunity, which means they proveably stay probing secure for every possible occurrence of a glitch. This is achieved by introducing register stages, which results in highly pipelined data paths. Instead of simply regarding these countermeasures as introducing undesired overhead and significantly increasing latency, they can also be leveraged positively to improve performance. In hardware, the mandatory registers, which are present as a leakage countermeasure, essentially result in a 'free' pipelining of the datapath.

The density of non-linear operations dictates the frequency of registers and hence the performance and cycle count. However, these are often not uniformly distributed throughout masked implementations and hence result in a non-optimal usage of the critical path 'budget'. Introducing additional registers to balance the datapath results in a non-minimal overhead and latency, and only further increases implementation cost.

We propose interleaving registers clocked at the positive and negative edge in select secure gadgets, resulting in half-cycle paths [41,27]. This circuit-level technique can ideally halve the latency (e.g. doubles the throughput) of a tightly pipelined secure gadget, as data is captured every half-clock cycle. As demonstrated in Figure 5, DOM and HPC3 require one cycle per masked non-linear gate and HPC1 requires two. As a result, our Brent-Kung SecADD design would require nine or 18 clock cycles if implemented with SOTA masking techniques. Our circuit-level optimizations result in a latency of only five cycles.

We implement half-cycle paths with minimal design effort, as it mainly corresponds to identifying highly pipelined stages of the masked implementation (successive non-linear operations) and alternating the clock edge at which the registers capture data. The maximal operating frequency (f_{max}) can remain largely unaffected if pipelining stages from masking are extremely small compared to other sections of the circuit, which is the case for mask conversions and secure additions. They consist of only a few boolean gates and can easily be completed within half a clock cycle of the original clock speed, which is dictated by I/O memory transfers. The modified sections, if identified correctly, are better utilized and operate on a high clock frequency, while other sections run at their natural, lower clock speed.

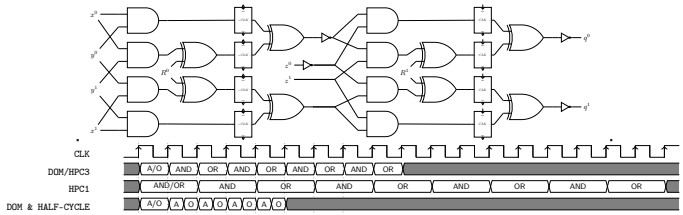


Fig. 5: Half-cycle data paths & Domain-Oriented Masking: the latency of highly non-linear, tightly pipelined data paths can be halved, for free (illustrated with SecAND & SecOR). The bottom figure illustrates the operation/gate latency for our (first-order) Brent-Kung SecADD: requiring only 5 clock cycles instead of 9/18 for na"ive DOM/HPC3 or HPC1.

4.4 Performance Evaluation

All of the types of mask conversions required in the Kyber decapsulation (or Dilithium/Falcon Sign) procedure can be computed using our efficient and secure streaming hardware design (Table **??**). Our first-order implementation has a utilization of 1638 LUT and 2874 FF, and can operate on a maximum clock frequency of 150 MHz. The second-order implementation has a utilization of 7946/18032 LUT/FF and a maximum clock frequency of 125 MHz. Through algorithmic, gadget- and circuit-level optimizations, we reduce latency, maximize throughput and minimize area cost. We compare other A2B/B2A algorithms with state-of-the-art masking techniques and our algorithms with our masking techniques, including HPC1 and HPC3. Compared to the DOM AND gate, HPC1 requires a significant amount of additional random bits and the latency is doubled (two cycles), due to an additional *Refresh* stage at the input. HPC3 has a latency of one cycle but requires double the randomness compared to the DOM AND gate [16,46].

[3] Not required for Kyber Decapsulation, but naturally supported in design.
[4] Section 3
[5] Section 4

Firstly, we can observe in the first and second row the overhead for several state-of-the-art A2B/B2A algorithms implemented using HPC1/3, which are high. These algorithms require both expensive modular reduction (SecMUX/SecADD) and post-processing for the B2A operation.

Secondly, these algorithms can be compared with our secure gadgets from Section 3, if naively implemented using HPC1/HPC3. The third row shows that our novel nethods result in a lower overhead compared to the state-of-the-art algorithms at all orders of protection. Both the latency and fresh randomness cost are reduced, due to implicit modular reduction and a reduced SecADD count. Compared to [6], which requires a SecMUX, the A2B_q requires around 9% less RND at first through third protection order. Our B2A approach, through the $X2B$ gadget, has half the latency (and randomness) at the first protection order, as it requires only one SecADD instead of two. Similar improvements can be expected for masked software implementations, as these are simply the result of the removal of operations.

Thirdly, we also introduce careful masking and several circuit-level optimizations (Section 4), resulting in the most efficient implementations for any type of A2B/B2A conversion, at arbitrary protection order, in hardware. By not relying on universal composability, we can reduce latency by $\pm$ 85% for all modes and at any protection order, as demonstrated in the final row. The randomness is reduced significantly: 45/54/47 % for the A2B mod 2^k, 47/48/62 % for the A2B mod q. Our novel B2A approach, combined with our careful masking approach results in a reduction of 70/68/51 % (mod 2^k) and 78/75/65 % (mod q) in fresh randomness requirements.

Interestingly, both 3- and 4-share implementations require the same latency for A2B/B2A conversions. This is because both require two A2B layers. The 4-share implementation does have a higher area utilization, as more & bigger adders are instantiated within these layers, also explaining the increased fresh randomness cost. Our B2A gadgets, based on simplified post-processing, have the same latency as the A2B operation, for any protection order d. Compared to prior work, for which the B2A requires around $\times 2$ more clock cycles for first order, $\times 1.33$ for second and third order implementations, more than the A2B operation.

5 Security Evaluation

5.1 Measurement Setup

In this section, we describe the practical evaluation of our masked designs. The synthesis results were obtained with the Xilinx Vivado v2021.1 compiler. We utilize the *keep_hierarchy* pragma to prevent the compiler from optimizing masking countermeasures away. This may result in a less-than-optimal overhead but ensures the desired security. We collect power traces from the measurement point on the Sakura-X evaluation board, containing a Xilinx Kintex7 FPGA. The traces are captured by a Tektronix DPO7254 oscilloscope at a sample rate of 1GS/s while the FPGA is externally clocked at 6MHz. We synchronized the oscilloscope and the external clock for all our measurements.

The mask conversion accelerator instance is duplicated 15 or 5 times[6] on FPGA for lab evaluations, to guarantee satisfactory SNR for statistical analysis, illustrated in the mean measurement traces. All instances operate on a single, identical input data and fresh randomness in parallel. No other operations or parts of the pipeline are activated during the entire operation. The randomness required by our design is supplied by a PRNG that runs on the crypto FPGA. The PRNG consists of a Trivium cipher implementation, which is re-seeded with fresh randomness for each mask conversion. We interleave the execution of the PRNG with the execution of the full mask conversion to decrease the impact of noise induced by the PRNG.

5.2 Test Vector Leakage Assessment Results

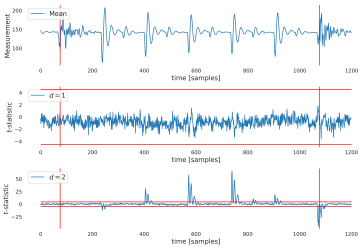
We verify that our implementations do not show first-order (or second-order) univariate and bivariate leakage. The *non-specific, fixed vs. random* t-test statistic [37] is calculated for the implementation of all different mask conversion operations. The threshold value of the t-test commonly used by the side-channel research community is 4.5 which provides a confidence of roughly 0.99999. If the t-test value of the measured power trace grows over 4.5, the implementation under test is considered as insecure. The regions of interest are indicated on all figures between vertical red lines, which indicate the start and end of mask conversion.

Figure 6 illustrates the TVLA results of the first-order masked A2B (Figure 6a & 6b) and B2A operations (Figure 6c & 6d) (mod 2^{13} and 3329, respectively), displaying the mean trace and first and second order statistical moments with the PRNG activated. Each of the subplots confirm our theoretical expectation, as no significant evidence of first-order leakage was detected for 100 million measurements. The second-order leakages show as anticipated. In contrast, we also include t-test results for the implementation with the randomness turned off (set to zero), guaranteeing that our test set-up is sound and can detect leakage (Figure 8a) with only 500K traces.

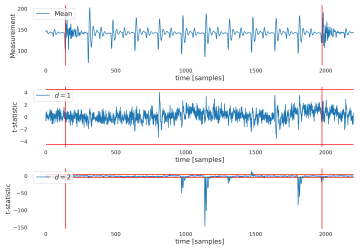
Figure 7 illustrates the TVLA result of the second-order masked A2B and B2A operations (mod 2^{13} and 3329), operating on three shares. The mean trace and first, second (and third) order statistical moments with the RNG activated are displayed. First- and second-order (univariate) leakages are not present. We want to bring the reader's attention to the complexities of observing higher-order leakages. For our second-order implementation, third-order leakages show for certain modes ($\mathrm{B2A}_q$), as anticipated, and not for others. We can attribute this phenomenon to effects described in [54]. More specifically, to observe higher-order leakage one needs to collect much more traces. One could expect that if we continued acquiring traces up to 500M or even 1 billion traces, our second-order implementation would exhibit third-order leakages more clearly in other modes of operation too. We do not include such figures due to the practical and computational infeasibility. Again, we verified our measurement setup by turning off the randomness source (Fig. 8b), with all present leakages not appearing when the randomness is turned on again.

We also performed second-order bivariate leakage detection tests [15], illustrated in Figure 9. To alleviate the computational complexity of this analysis, we set the

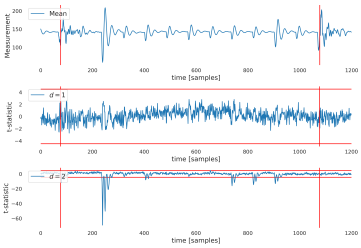
[6] for 2- & 3-share implementation



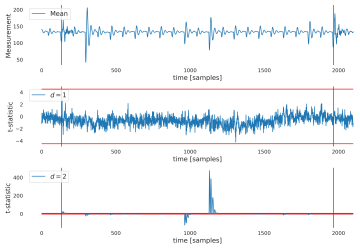
(a) A2B mod 2^{13}.



(b) A2B mod 3329.



(c) B2A mod 2^{13}.

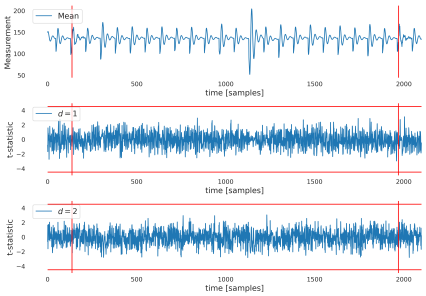


(d) B2A mod 3329.

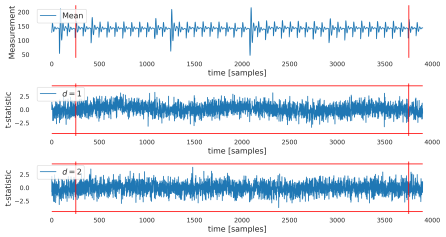
Fig. 6: 1^{st} & 2^{nd}-order univariate fixed-vs.-random TVLA results for all types of first-order mask conversions (2 shares) using 100M traces with PRNG ON. For each subfigure, the upper plot shows the mean trace. The $\pm$ 4.5 threshold is marked by red lines.

point of interests at every 2 (or 5) sample points, lowering the sampling rate to 500 (or 200) MS/s for the A2B and B2A operation. First, we verified that both our first-order implementation and second-order implementation with the PRNG turned off show leakages, with 500K traces. We confirm the measurement setup is sound and can detect bivariate leakages. With the PRNG switched on, no excursions of the t-values beyond $\pm$ 4.5 occur and thus the test is passed with 100M traces.

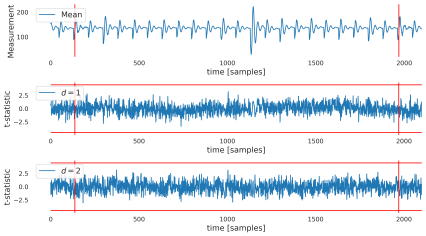
Conclusion. From these first-and high-order univariate and bivariate tests using TVLA methodology, we can conclude our proposed techniques and their implementations are secure. We demonstrate how our approach leads to efficient and secure first and high-order implementations.



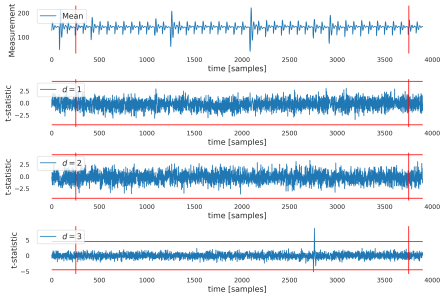
(a) A2B mod 2^{13}.



(b) A2B mod 3329.

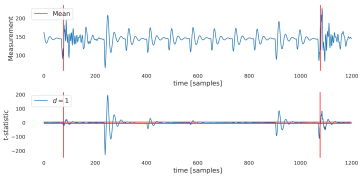


(c) B2A mod 2^{13}.

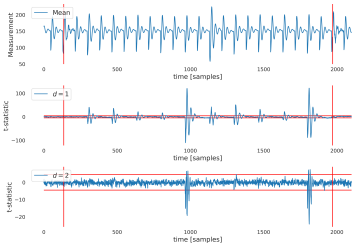


(d) B2A mod 3329.

Fig. 7: 1^{st}, 2^{nd} (&3^{rd})-order univariate fixed-vs.-random TVLA results for all types of second-order mask conversions (3 shares) using 100M traces with PRNG ON. For each subfigure, the upper plot shows the mean trace. The $\pm$ 4.5 threshold is marked by red lines.

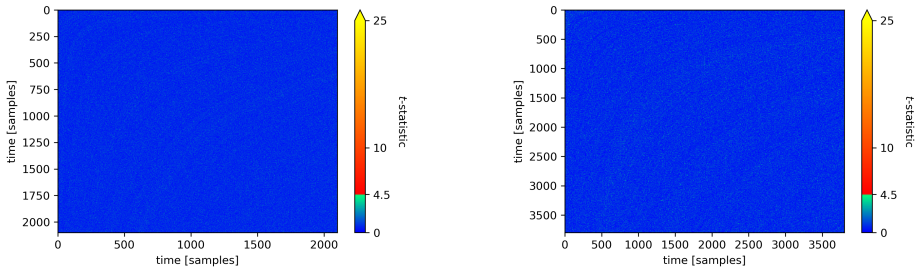


(a) 1^{st}-order (2 share) implementation



(b) 2^{nd}-order (3 share) implementation

Fig. 8: 1^{st} (& 2^{nd})-order univariate fixed-vs.-random TVLA results for A2B mod 2^{13} (2 & 3 shares) using 500K traces with PRNG OFF. For each subfigure, the upper plot shows the mean trace.



(a) A2B mod 2^{13}. (b) B2A mod 3329.

Fig. 9: Bivariate analysis of second-order mask conversion implementation (3 shares), 100M traces, PRNG ON. (Best viewed on screen.)

6 Conclusion

In this work, a first- and high-order hardware implementation of the mask conversion operation, secure against differential power analysis attacks were described. These leverage novel $d-$order secure gadgets and circuit-level optimizations to improve performance at all protection orders. Including a novel SecADD_q gadget, which relies on repeated, implicit modular reduction and a novel B2A algorithm, which relies on the novel $X2B$. The univariate and multivariate security is formally proven and experimentally validated in various modes.

Instead of utilizing state-of-the-art masking techniques, which rely on universal composability, this work leverages careful, manual masking to achieve first- and high-order protection. It is demonstrated that such an approach leads to more reasonable overheads and protects against side-channel leakage at the same degree. Also, no tedious re-design is required when extending our approach to higher protection orders, as we utilize the DOM-scheme. Half-cycle paths further exploit the masking countermeasures to increase the performance of highly non-linear operations, without requiring the explicit inclusion of additional pipelining registers.

In summary, the presented techniques result in hardware implementations with the lowest area utilization, fresh randomness cost and latency published to this date. Our first-order implementation requires only 1638/2874 [LUT/FF] when implemented on FPGA. The amount of clock cycles required for a mask conversion is reduced by up to 89%, the required amount of fresh randomness by up to 78%. The presented second-order implementation requires 7946/18032 [LUT/FF] on FPGA, which requires up to 84% less clock cycles and 71% fewer random bits.

Acknowledgements We thank Lennert Wouters, Zhenda Zhang, John Gaspoz and Siemen Dhooghe for the interesting discussions. This work was supported by CyberSecurity Research Flanders with reference number VR20192203. In addition, this work was supported by the European Commission through the Horizon 2020 research and innovation program under grant agreement Belfort ERC Advanced

Grant 101020005 695305, through the Horizon Europe research and innovation program under grant agreement HORIZON-CL3-2021-CS-01-02 101070008 ORSHIN. In addition, this work is supported in part by the European Commission through the DIGITAL-SIMPLE project Be-QCI with contract number 101091625.

References

1. Alagic, G., et al.: Status report on the second round of the NIST post-quantum cryptography standardization process. https://csrc.nist.gov/pubs/ir/8309/final (2020), [Online; accessed 17-August-2023]
2. Bache, F., Güneysu, T.: Boolean masking for arithmetic additions at arbitrary order in hardware. Applied Sciences **12**(5) (2022). https://doi.org/10.3390/app12052274, https://www.mdpi.com/2076-3417/12/5/2274
3. Balasch, J., Gierlichs, B., Grosso, V., Reparaz, O., Standaert, F.: On the cost of lazy engineering for masked software implementations. In: Joye, M., Moradi, A. (eds.) Smart Card Research and Advanced Applications - 13th International Conference, CARDIS 2014, Paris, France, November 5-7, 2014. Revised Selected Papers. Lecture Notes in Computer Science, vol. 8968, pp. 64–81. Springer (2014). https://doi.org/10.1007/978-3-319-16763-3_5, https://doi.org/10.1007/978-3-319-16763-3_5
4. Barthe, G., Belaïd, S., Dupressoir, F., Fouque, P.A., Grégoire, B., Strub, P.Y.: Verified proofs of higher-order masking. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology – EUROCRYPT 2015. pp. 457–485. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)
5. Barthe, G., Belaïd, S., Dupressoir, F., Fouque, P.A., Grégoire, B., Strub, P.Y., Zucchini, R.: Strong non-interference and type-directed higher-order masking. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016. pp. 116–129. ACM Press (Oct 2016). https://doi.org/10.1145/2976749.2978427
6. Barthe, G., Belaïd, S., Espitau, T., Fouque, P.A., Grégoire, B., Rossi, M., Tibouchi, M.: Masking the GLP lattice-based signature scheme at any order. In: Nielsen, J.B., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2018. pp. 354–384. Springer International Publishing, Cham (2018)
7. Beirendonck, M.V., D'anvers, J.P., Karmakar, A., Balasch, J., Verbauwhede, I.: A side-channel-resistant implementation of saber. J. Emerg. Technol. Comput. Syst. **17**(2) (apr 2021). https://doi.org/10.1145/3429983, https://doi.org/10.1145/3429983
8. Bilgin, B.: Threshold implementations: as countermeasure against higher-order differential power analysis. Ph.D. thesis, University of Twente, Enschede, Netherlands (2015)
9. Bilgin, B., Gierlichs, B., Nikova, S., Nikov, V., Rijmen, V.: Higher-order threshold implementations. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part II. LNCS, vol. 8874, pp. 326–343. Springer, Heidelberg (Dec 2014). https://doi.org/10.1007/978-3-662-45608-8_18

10. Bos, J.W., Gourjon, M., Renes, J., Schneider, T., van Vredendaal, C.: Masking kyber: First- and higher-order implementations. IACR Transactions on Cryptographic Hardware and Embedded Systems **2021**(4), 173–214 (Aug 2021). https://doi.org/10.46586/tches.v2021.i4.173-214, https://tches.iacr.org/index.php/TCHES/article/view/9064
11. Brent, Kung: A regular layout for parallel adders. IEEE Transactions on Computers **C-31**(3), 260–264 (1982). https://doi.org/10.1109/TC.1982.1675982
12. Bronchain, O., Cassiers, G.: Bitslicing arithmetic/boolean masking conversions for fun and profit with application to lattice-based kems. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2022**(4), 553–588 (2022). https://doi.org/10.46586/tches.v2022.i4.553-588, https://doi.org/10.46586/tches.v2022.i4.553-588
13. Cassiers, G., Standaert, F.X.: Trivially and efficiently composing masked gadgets with probe isolating non-interference. IEEE Transactions on Information Forensics and Security **PP**, 1–1 (02 2020). https://doi.org/10.1109/TIFS.2020.2971153
14. Cassiers, G.: Composable and efficient masking schemes for side-channel secure implementations. Ph.D. thesis, École polytechnique de Louvain and Université catholique de Louvain (2022)
15. Cassiers, G., Bronchain, O.: Scalib: A side-channel analysis library. Journal of Open Source Software **8**(86), 5196 (2023). https://doi.org/10.21105/joss.05196, https://doi.org/10.21105/joss.05196
16. Cassiers, G., Grégoire, B., Levi, I., Standaert, F.X.: Hardware private circuits: From trivial composition to full verification. IEEE Transactions on Computers **70**(10), 1677–1690 (2021). https://doi.org/10.1109/TC.2020.3022979
17. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards sound approaches to counteract power-analysis attacks. In: Wiener, M.J. (ed.) CRYPTO'99. LNCS, vol. 1666, pp. 398–412. Springer, Heidelberg (Aug 1999). https://doi.org/10.1007/3-540-48405-1_26
18. Cnudde, T., Bilgin, B., Reparaz, O., Nikov, V., Nikova, S.: Higher-order threshold implementation of the aes s-box. In: Revised Selected Papers of the 14th International Conference on Smart Card Research and Advanced Applications - Volume 9514. p. 259–272. CARDIS 2015, Springer-Verlag, Berlin, Heidelberg (2015). https://doi.org/10.1007/978-3-319-31271-2_16, https://doi.org/10.1007/978-3-319-31271-2_16
19. Coron, J.S., Giraud, C., Prouff, E., Renner, S., Rivain, M., Vadnala, P.K.: Conversion of security proofs from one leakage model to another: A new issue. In: Schindler, W., Huss, S.A. (eds.) COSADE 2012. LNCS, vol. 7275, pp. 69–81. Springer, Heidelberg (May 2012). https://doi.org/10.1007/978-3-642-29912-4_6
20. Coron, J.S., Großschädl, J., Tibouchi, M., Vadnala, P.K.: Conversion from arithmetic to boolean masking with logarithmic complexity. In: Leander, G. (ed.) Fast Software Encryption. pp. 130–149. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)
21. Coron, J.S., Großschädl, J., Vadnala, P.K.: Secure conversion between Boolean and arithmetic masking of any order. In: Batina, L., Robshaw, M. (eds.) CHES 2014. LNCS, vol. 8731, pp. 188–205. Springer, Heidelberg (Sep 2014). https://doi.org/10.1007/978-3-662-44709-3_11
22. Coron, J.S., Gérard, F., Montoya, S., Zeitoun, R.: High-order table-based conversion algorithms and masking lattice-based encryption. IACR Transactions on Cryptographic Hardware and Embedded Systems **2022**(2), 1–40 (Feb 2022). https://doi.org/10.46586/tches.v2022.i2.1-40, https://tches.iacr.org/index.php/TCHES/article/view/9479
23. Coron, J.S., Gérard, F., Montoya, S., Zeitoun, R.: High-order table-based conversion algorithms and masking lattice-based encryption. IACR Transactions on Cryptographic Hardware and Embedded Systems **2022**, 1–40 (02 2022). https://doi.org/10.46586/tches.v2022.i2.1-40

24. Coron, J.S., Gérard, F., Trannoy, M., Zeitoun, R.: High-order masking of NTRU. IACR Transactions on Cryptographic Hardware and Embedded Systems **2023**(2), 180–211 (Mar 2023). https://doi.org/10.46586/tches.v2023.i2.180-211, https://tches.iacr.org/index.php/TCHES/article/view/10281
25. D'Anvers, J.P., Tiepelt, M., Vercauteren, F., Verbauwhede, I.: Timing attacks on error correcting codes in post-quantum schemes. In: Proceedings of ACM Workshop on Theory of Implementation Security Workshop. p. 2–9. TIS'19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3338467.3358948, https://doi.org/10.1145/3338467.3358948
26. De Cnudde, T., Bilgin, B., Gierlichs, B., Nikov, V., Nikova, S., Rijmen, V.: Does coupling affect the security of masked implementations? In: Guilley, S. (ed.) COSADE 2017. LNCS, vol. 10348, pp. 1–18. Springer, Heidelberg (Apr 2017). https://doi.org/10.1007/978-3-319-64647-3_1
27. De Meyer, L., Reparaz, O., Bilgin, B.: Multiplicative masking for aes in hardware. IACR Transactions on Cryptographic Hardware and Embedded Systems **2018**(3), 431–468 (Aug 2018). https://doi.org/10.13154/tches.v2018.i3.431-468, https://tches.iacr.org/index.php/TCHES/article/view/7282
28. Division, N.C.S.: Post-quantum cryptography standardization. https://csrc.nist.gov/projects/post-quantum-cryptography (2016), [Online; accessed 17-August-2023]
29. Division, N.C.S.: Comments requested on three draft fips for post-quantum cryptography. https://csrc.nist.gov/news/2023/three-draft-fips-for-post-quantum-cryptography (2023), [Online; accessed 30-October-2023]
30. Division, N.C.S.: Fips 203 (draft): Module-lattice-based key-encapsulation mechanism standard. https://csrc.nist.gov/pubs/fips/203/ipd (2023). https://doi.org/https://doi.org/10.6028/NIST.FIPS.203.ipd, [Online; accessed 30-October-2023]
31. Division, N.C.S.: Fips 204 (draft): Module-lattice-based digital signature standard. https://csrc.nist.gov/pubs/fips/204/ipd (2023). https://doi.org/https://doi.org/10.6028/NIST.FIPS.204.ipd, [Online; accessed 30-October-2023]
32. Division, N.C.S.: Post-quantum cryptography: Digital signature schemes. https://csrc.nist.gov/projects/pqc-dig-sig/round-1-additional-signatures (2023), [Online; accessed 7-September-2023]
33. D'Anvers, J.P.: One-hot conversion: Towards faster table-based a2b conversion. In: Advances in Cryptology – EUROCRYPT 2023: 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part IV. p. 628–657. Springer-Verlag, Berlin, Heidelberg (2023). https://doi.org/10.1007/978-3-031-30634-1_21, https://doi.org/10.1007/978-3-031-30634-1_21
34. Faust, S., Grosso, V., Merino Del Pozo, S., Paglialonga, C., Standaert, F.X.: Composable masking schemes in the presence of physical defaults & the robust probing model. IACR Transactions on Cryptographic Hardware and Embedded Systems **2018**(3), 89–120 (Aug 2018). https://doi.org/10.13154/tches.v2018.i3.89-120, https://tches.iacr.org/index.php/TCHES/article/view/7270
35. Fritzmann, T., Van Beirendonck, M., Basu Roy, D., Karl, P., Schamberger, T., Verbauwhede, I., Sigl, G.: Masked accelerators and instruction set extensions for post-quantum cryptography. IACR Transactions on Cryptographic Hardware and Embedded Systems **2022**(1), 414–460 (Nov 2021). https://doi.org/10.46586/tches.v2022.i1.414-460, https://tches.iacr.org/index.php/TCHES/article/view/9303
36. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic analysis: Concrete results. In: Koç, Çetin Kaya., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 251–261. Springer, Heidelberg (May 2001). https://doi.org/10.1007/3-540-44709-1_21

37. Goodwill, G., Jun, B., Jaffe, J., Rohatgi, P.: A testing methodology for side channel resistance. https://csrc.nist.gov/csrc/media/events/non-invasive-attack-testing-workshop/documents/08_goodwill.pdf (2011), [Online; accessed 6-November-2023]
38. Goubin, L.: A sound method for switching between Boolean and arithmetic masking. In: Koç, Çetin Kaya., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 3–15. Springer, Heidelberg (May 2001). https://doi.org/10.1007/3-540-44709-1_2
39. Groß, H., Mangard, S.: Reconciling d+1 masking in hardware and software. In: Fischer, W., Homma, N. (eds.) CHES 2017. LNCS, vol. 10529, pp. 115–136. Springer, Heidelberg (Sep 2017). https://doi.org/10.1007/978-3-319-66787-4_6
40. Gross, H., Mangard, S., Korak, T.: Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order. In: Proceedings of the 2016 ACM Workshop on Theory of Implementation Security. p. 3. TIS '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2996366.2996426, https://doi.org/10.1145/2996366.2996426
41. Gross, H., Schaffenrath, D., Mangard, S.: Higher-order side-channel protected implementations of KECCAK. In: 2017 Euromicro Conference on Digital System Design (DSD). pp. 205–212 (2017). https://doi.org/10.1109/DSD.2017.21
42. Guo, Q., Johansson, T., Nilsson, A.: A key-recovery timing attack on post-quantum primitives using the Fujisaki-Okamoto transformation and its application on FrodoKEM. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part II. LNCS, vol. 12171, pp. 359–386. Springer, Heidelberg (Aug 2020). https://doi.org/10.1007/978-3-030-56880-1_13
43. Heinz, D., Kannwischer, M.J., Land, G., Pöppelmann, T., Schwabe, P., Sprenkels, D.: First-order masked kyber on ARM cortex-m4. Cryptology ePrint Archive, Paper 2022/058 (2022), https://eprint.iacr.org/2022/058, https://eprint.iacr.org/2022/058
44. Hutter, M., Schmidt, J.M.: The temperature side channel and heating fault attacks. In: Smart Card Research and Advanced Applications: 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers. p. 219–235. Springer-Verlag, Berlin, Heidelberg (2014). https://doi.org/10.1007/978-3-319-08302-5_15
45. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: Securing hardware against probing attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Heidelberg (Aug 2003). https://doi.org/10.1007/978-3-540-45146-4_27
46. Knichel, D., Moradi, A.: Low-latency hardware private circuits. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022. pp. 1799–1812. ACM Press (Nov 2022). https://doi.org/10.1145/3548606.3559362
47. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO'96. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (Aug 1996). https://doi.org/10.1007/3-540-68697-5_9
48. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M.J. (ed.) CRYPTO'99. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (Aug 1999). https://doi.org/10.1007/3-540-48405-1_25
49. Krausz, M., Land, G., Stolz, F., Naujoks, D., Richter-Brockmann, J., Güneysu, T., Kogelheide, L.: Generic accelerators for costly-to-mask pqc components. Cryptology ePrint Archive, Paper 2023/1287 (2023), https://eprint.iacr.org/2023/1287, https://eprint.iacr.org/2023/1287
50. Kundu, S., D'Anvers, J.P., Van Beirendonck, M., Karmakar, A., Verbauwhede, I.: Higher-order masked saber. In: Galdi, C., Jarecki, S. (eds.) Security and Cryptography for Networks. pp. 93–116. Springer International Publishing, Cham (2022)
51. Mangard, S., Popp, T., Gammel, B.M.: Side-channel leakage of masked cmos gates. In: Menezes, A. (ed.) Topics in Cryptology – CT-RSA 2005. pp. 351–365. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)

52. Mangard, S., Schramm, K.: Pinpointing the side-channel leakage of masked AES hardware implementations. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 76–90. Springer, Heidelberg (Oct 2006). https://doi.org/10.1007/11894063_7
53. Migliore, V., Gérard, B., Tibouchi, M., Fouque, P.A.: Masking Dilithium - efficient implementation and side-channel evaluation. In: Deng, R.H., Gauthier-Umaña, V., Ochoa, M., Yung, M. (eds.) ACNS 19. LNCS, vol. 11464, pp. 344–362. Springer, Heidelberg (Jun 2019). https://doi.org/10.1007/978-3-030-21568-2_17
54. Moradi, A., Wild, A.: Assessment of hiding the higher-order leakages in hardware - what are the achievements versus overheads? In: Güneysu, T., Handschuh, H. (eds.) CHES 2015. LNCS, vol. 9293, pp. 453–474. Springer, Heidelberg (Sep 2015). https://doi.org/10.1007/978-3-662-48324-4_23
55. Müller, N., Knichel, D., Sasdrich, P., Moradi, A.: Transitional leakage in theory and practice: Unveiling security flaws in masked circuits. IACR Transactions on Cryptographic Hardware and Embedded Systems **2022**(2), 266–288 (Feb 2022). https://doi.org/10.46586/tches.v2022.i2.266-288, https://tches.iacr.org/index.php/TCHES/article/view/9488
56. Nikova, S., Rechberger, C., Rijmen, V.: Threshold implementations against side-channel attacks and glitches. In: Ning, P., Qing, S., Li, N. (eds.) Information and Communications Security. pp. 529–545. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)
57. Prest, T., et al.: Falcon: Technical report, national institute of standards and technology. https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions (2020), [Online; accessed 17-August-2023]
58. Primas, R., Pessl, P., Mangard, S.: Single-trace side-channel attacks on masked lattice-based encryption. In: Fischer, W., Homma, N. (eds.) CHES 2017. LNCS, vol. 10529, pp. 513–533. Springer, Heidelberg (Sep 2017). https://doi.org/10.1007/978-3-319-66787-4_25
59. Ravi, P., Sinha Roy, S., Chattopadhyay, A., Bhasin, S.: Generic side-channel attacks on CCA-secure lattice-based PKE and KEMs. IACR Transactions on Cryptographic Hardware and Embedded Systems **2020**(3), 307–335 (Jun 2020). https://doi.org/10.13154/tches.v2020.i3.307-335, https://tches.iacr.org/index.php/TCHES/article/view/8592
60. Reparaz, O., Bilgin, B., Nikova, S., Gierlichs, B., Verbauwhede, I.: Consolidating masking schemes. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part I. LNCS, vol. 9215, pp. 764–783. Springer, Heidelberg (Aug 2015). https://doi.org/10.1007/978-3-662-47989-6_37
61. Schneider, T., Paglialonga, C., Oder, T., Güneysu, T.: Efficiently masking binomial sampling at arbitrary orders for lattice-based crypto. In: Lin, D., Sako, K. (eds.) Public-Key Cryptography - PKC 2019 - 22nd IACR International Conference on Practice and Theory of Public-Key Cryptography, Beijing, China, April 14-17, 2019, Proceedings, Part II. Lecture Notes in Computer Science, vol. 11443, pp. 534–564. Springer (2019). https://doi.org/10.1007/978-3-030-17259-6_18, https://doi.org/10.1007/978-3-030-17259-6_18
62. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (nov 1979). https://doi.org/10.1145/359168.359176, https://doi.org/10.1145/359168.359176
63. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Journal on Computing **26**(5), 1484–1509 (1997). https://doi.org/10.1137/S0097539795293172, https://doi.org/10.1137/S0097539795293172
64. Van Beirendonck, M., D'Anvers, J.P., Verbauwhede, I.: Analysis and comparison of table-based arithmetic to boolean masking. IACR Transactions on Cryptographic Hardware and Embedded Systems **2021**(3), 275–297 (Jul 2021). https://doi.org/10.46586/tches.v2021.i3.275-297, https://tches.iacr.org/index.php/TCHES/article/view/8975

65. Xu, Z., Pemberton, O., Roy, S.S., Oswald, D.F., Yao, W., Zheng, Z.: Magnifying side-channel leakage of lattice-based cryptosystems with chosen ciphertexts: The case study of kyber. IEEE Transactions on Computers **71**, 2163–2176 (2022), https://api.semanticscholar.org/CorpusID:220794801