string every 28 cycles during the extraction-phase. Furthermore, one instance of the Keccak core consumes a large area, thus indicating that implementing a vectorized Keccak core would make the implementation both area-expensive and more complex. Additionally, as the Keccak core is already very fast in hardware, the use of parallel cores would be of little help in improving the speed in hardware.

Table 7: Performance of CCA-secure Saber (module dimension 3) on hardware platforms. Entries denoted by a * are estimated based on the timings of PKE functions.

Ref.	Platform	Time in μs (KeyGen./Encaps./ Decaps.)	Frequency (MHz)	Area (LUT/FF/DSP/BRAM) (or mm^2 for ASIC)
[10]	Artix-7 (HW/SW)	3.2K/4.1K/3.8K	125	7.4K/7.3K/28/2
[17]	UltraScale+ (HW/SW)	-/60/65	322	12.5K/11.6K/256/4
[41]	UltraScale+	21.8/26.5/32.1	250	23.6K/9.8K/0/2
[46]*	UltraScale+	-/14.0/16.8	100	34.9k/9.9K/85/6
[46]*	ASIC	2.60/3.49/4.21	400	0.35 mm^2

5.4 Saber on RSA coprocessor

Secure smart cards, microcontrollers, and trusted platform modules or hardware security modules come with dedicated coprocessors for accelerating RSA. Such coprocessors contain long integer multipliers for computing modular multiplications in the RSA algorithm. [44] accelerates polynomial multiplications in Saber using long integer multipliers. The authors report that, for polynomials with 256 coefficients, one polynomial multiplication takes 97K and 85K clock cycles for powers of 2 moduli 8192 and 1024 respectively on a ESP32 platform. Whereas, on the same platform NTT-based polynomial multiplications takes 244K clock cycles when the modulus is a prime 7681. Their work shows that Saber can benefit from long integer multipliers present in secured devices with RSA coprocessors.

6 Expected strength (2.B.4) in general

6.1 Security

The IND-CPA security of Saber.PKE can be reduced from the decisional Mod-LWR problem as shown by the following theorem:

Theorem 6.1. *In the random oracle model, where* `gen` *is assumed to be a random oracle, for any adversary A, there exist three adversaries B_0, B_1 and B_2 with roughly the same running time as A, such that* $Adv^{ind\text{-}cpa}_{Saber.PKE}(A) \leqslant Adv^{prf}_{\mathtt{gen}()}(B_0) + Adv^{mod\text{-}lwr}_{l,l,\nu,q,p}(B_1) + Adv^{mod\text{-}lwr}_{l+1,l,\nu,q,q/\zeta}(B_2)$, *where* $\zeta = \min(\frac{q}{p}, \frac{p}{T})$.

The correctness of Saber.PKE can be calculated using the python scripts included in the submission, following theorem 6.2:

Theorem 6.2. *Let* $\boldsymbol{A}$ *be a matrix in* $R_q^{l\times l}$ *and* $\boldsymbol{s}, \boldsymbol{s}'$ *two vectors in* $R_q^{l\times 1}$ *sampled as in Saber.PKE. Define* $\boldsymbol{e}$ *and* $\boldsymbol{e}'$ *as the rounding errors introduced by scaling and rounding* $\boldsymbol{A}^T\boldsymbol{s}$ *and* $\boldsymbol{A}\boldsymbol{s}'$*, i.e.* $((\boldsymbol{A}^T\boldsymbol{s}+\boldsymbol{h}) \bmod q) \gg (\epsilon_q-\epsilon_p) = \frac{p}{q}\boldsymbol{A}^T\boldsymbol{s}+\boldsymbol{e}$ *and* $((\boldsymbol{A}\boldsymbol{s}'+\boldsymbol{h}) \bmod q) \gg (\epsilon_q-\epsilon_p) = \frac{p}{q}\boldsymbol{A}\boldsymbol{s}'+\boldsymbol{e}'$*. Let* $e_r \in R_q$ *be a polynomial with uniformly distributed coefficients with range* $[-p/2T, p/2T]$*. If we set*

$$\delta = Pr[||(\boldsymbol{s}'^T\boldsymbol{e} - \boldsymbol{e}'^T\boldsymbol{s} + e_r) \mod p||_\infty > p/4]$$

then after executing the Saber.PKE protocol, both communicating parties agree on a n*-bit key with probability* $1-\delta$*.*

For these calculations, the failure probabilities of the different coefficients of $(\boldsymbol{s}'^T\boldsymbol{e}-\boldsymbol{e}'^T\boldsymbol{s}+e_r)$ can be assumed independent, as discussed in [21].

This IND-CPA secure encryption scheme is the basis for the IND-CCA secure KEM Saber.KEM=(`Encaps, Decaps`), which is obtained by using an appropriate transformation. Recently, several post-quantum versions [23, 42, 38, 24] of the Fujisaki-Okamoto transform with corresponding security reductions have been developed. At this point, the FO$^{\not\perp}$ transformation in [23] with post-quantum reduction from Jiang et al. [24] gives the tightest reduction for schemes with non-perfect correctness. However, other transformation could be used to turn Saber.PKE into a CCA secure KEM.

6.1.1 Security in the Random Oracle Model

By modeling the hash functions $\mathcal{G}$ and $\mathcal{H}$ as random oracles, a lower bound on the CCA security can be proven. We use the security bound of Hofheinz et al. [23], which considers a KEM variant of the Fujisaki-Okamoto transform that can also handle a small failure probability δ of the encryption scheme. This failure probability should be cryptographically negligibly small for the security to hold. Using Theorem 3.2 and Theorem 3.4 from [23], we get the following theorems for the security and correctness of our KEM in the random oracle model:

Theorem 6.3. *In the random oracle model, where* $\mathcal{G}$ *and* $\mathcal{H}$ *are assumed to be random oracles, for a IND-CCA adversary* B*, making at most* $q_\mathcal{H}$ *and* $q_\mathcal{G}$ *queries to respectively the random oracle* $\mathcal{G}$ *and* $\mathcal{H}$*, and* q_D *queries to the decryption oracle, there exists an IND-CPA adversary* A *with approximately the same running time as adversary* B*, such that:*

$$Adv^{ind\text{-}cca}_{Saber.KEM}(B) \leqslant 3Adv^{ind\text{-}cpa}_{Saber.PKE}(A) + q_\mathcal{G}\delta + \frac{2q_\mathcal{G}+q_\mathcal{H}+1}{2^{256}}.$$

6.1.2 Security in the Quantum Random Oracle Model

Jiang et al. [24] provide a security reduction against a quantum adversary in the quantum random oracle model from IND-CCA security to OW-CPA security. IND-CPA with a

sufficiently large message space M implies OW-CPA [23, 13], as is given by following lemma:

Theorem 6.4. *For an OW-CPA adversary B, there exists an IND-CPA adversary A with approximately the same running time as adversary B, such that:*

$$Adv^{ow\text{-}cpa}_{Saber.PKE}(B) \leqslant Adv^{ind\text{-}cpa}_{Saber.PKE}(A) + 1/|M|$$

Therefore, we can reduce the IND-CCA security of Saber.KEM from the IND.CPA security of the underlying public key encryption:

Theorem 6.5. *In the quantum random oracle model, where $\mathcal{G}$ and $\mathcal{H}$ are assumed to be random oracles, for any IND-CCA quantum adversary B, making at most $q_{\mathcal{H}}$ and $q_{\mathcal{G}}$ queries to respectively the random quantum oracle $\mathcal{G}$ and $\mathcal{H}$, and q_D many (classical) queries to the decryption oracle, there exists an adversary A, with approximately the same running time as B, such that:*

$$Adv^{ind\text{-}cca}_{Saber.KEM}(B) \leqslant 2q_{\mathcal{H}}\frac{1}{\sqrt{2^{256}}} + 4q_{\mathcal{G}}\sqrt{\delta} + 2(q_{\mathcal{G}} + q_{\mathcal{H}})\sqrt{Adv^{ind\text{-}cpa}_{Saber.PKE}(A) + 1/|M|}$$

In all attack scenarios we assume that the depth of quantum computation is limited to 2^{64} quantum gates.

Using recent results [26], it is likely that the square root loss can be avoided, but more research is required to evaluate whether Saber satisfies the conditions required by [26].

6.2 Multi-target protection

As described in [14], hashing the public key into $\hat{K}$ has two beneficial effects: it makes sure that K depends on the input of both parties, and it offers multi-target protection. Hashing pk into $\hat{K}$ ensures that an attacker is not able to use precomputed 'weak' values of m on multiple targets when searching for decryption failures as will be addressed in next section.

6.3 Decryption failure attack

Instead of solving the Mod-LWR problem, an attacker can mount an attack that uses decryption failures. In this scenario, the adversary uses Grover's algorithm to precompute m that have a relatively high failure probability. Once messages m are found that trigger a decryption failure, they can be used to estimate the secret. This attack strategy is covered by the $4q_{\mathcal{G}}\sqrt{\delta}$ term of the quantum IND-CCA security reduction.

The best known attack was described in [20] and the follow up work [19]. In a single target setting this attack requires an impractical number of decryption queries far above 2^{64}. It is possible to reduce the number of decryption queries required by the attack by doing additional preprocessing or by mounting a multi-target attack where a number of victims T

are targeted of which only one will be broken, in which case the attacker can perform $2^{64} \cdot T$ queries to find the first failing ciphertext. Note that due to the multi-target protection an attacker can not reuse ciphertexts over multiple victims and needs to generate each ciphertext independently. Both attacks are more costly than breaking the Mod-LWR problem of the public keys.

If future cryptanalysis in this direction would threaten the security of Saber, one could increase the parameter T to reduce the failure probability and thus increase the security at the cost of additional bandwidth.

6.4 Side-channel attacks

Timing analysis Information about the timing of certain calculations can be used to extract information about the long-term secret key. Constant-time execution is a well-known countermeasure against such attacks. As mentioned before, the design choices of Saber allow simple constant-time implementations. The use of power-of-two moduli avoids variable-time operations, such as rejection sampling or modular reduction with a prime modulus. All implementations of Saber described in Section 5 are constant-time.

Differential analysis Differential power or electromagnetic radiation analysis allows an attacker to infer the intermediate data that a device is processing through statistical analysis. These attacks are particularly powerful, requiring only minimal signal-to-noise ratios and limited knowledge of the underlying device. Such attacks are possible on all operations that are influenced by the long-term secret key (e.g., [33, 45]), notably also including the applications of error correcting codes and CCA-transform [34]. In a scenario where such attacks are possible, i.e. the attacker has physical access to the device, masked implementations are a popular countermeasure. Since Saber does not use any error correcting code, it avoids any complications related to masking an error correcting block.

We have described a first-order masked implementation of Saber decapsulation in Section 5.2.1, based on our work in [7]. The implementation is shown secure against first-order DPA on an STM32 microcontroller featuring the popular ARM Cortex-M4. Again, Saber is shown very efficient to protect due to its choice for power-of-two moduli. Additionally, for a masked implementation, the choice for rounding (LWR) instead of error sampling (LWE) offers additional efficiency.

In [7], we also briefly discuss extending the implementation to higher-order masking, since the first-order design is still vulnerable to higher-order differential analysis. We expect a higher-order masked implementation of Saber to benefit from its design choices in the same way, and to significantly outperform LWE-based schemes with prime moduli as well.

Single-trace attacks Implementations secure against differential analysis can still be broken using other side-channel attack methods. In particular, single-trace attacks can be used

to break masked implementations. These attacks process a power or EM trace horizontally, and the horizontal information of a single trace still contains information on both of the shares. Such attacks have recently been shown to be feasible on implementations of NewHope [5], Frodo [15], Kyber [32], and Saber [40]. Polynomial or matrix multiplication with the secret key are typical targets for these attacks. A popular countermeasure is randomized shuffling. Randomness is used to shuffle the order of operations or introduce dummy operations, which can be applied on top of masked implementations. These techniques increase the noise level to harden an implementation against higher-order DPA attacks as well. Shuffling was included in the linear operations of a masked implementation of NewHope KEM [31] at an extra overhead cost of only $1.01\times$, but this does not yet protect against attacks that target the non-linear NTT [32]. For Saber, which uses a combination of Toom-Cook, Karatsuba and schoolbook multiplication, further research is necessary to assess the exact granularity and overall cost of including shuffling operations.

7 Advantages and limitations (2.B.6)

Advantages:

- No modular reduction: since all moduli are powers of 2 we do not require explicit modular reduction. Furthermore, sampling a uniform number modulo a power of 2 is trivial in that it does not require any rejection sampling or more complicated sampling routines. This is especially important when considering constant time implementations.
- Modular structure and flexibility: the core component consists of arithmetic in the fixed polynomial ring $\mathbb{Z}_{2^{13}}[X]/(X^{256}+1)$ for all security levels. To change security, one simply uses a module of higher rank.
- Less pseudorandomness required: due to the use of Mod-LWR, our algorithm requires less pseudorandomness since no error sampling is required as in (Mod-)LWE. More specifically, instead of generating $2k+1$ secret polynomials as would be the case in (Mod-)LWE schemes, our design only needs to generate k secret polynomials. This is especially beneficial in masked implementations, where due to the non-linearity the cost of the secret polynomial sampling increases sharply with the masking order.
- Use of powers of 2 moduli p and q, and T simplifies scaling and rounding operations significantly.
- Low bandwidth: again due to the use of Mod-LWR, the bandwidth required is lower than similar systems based on (Mod-)LWE.
- Generic polynomial multiplication: since Saber does not make any specific algorithm an integral part of the protocol, implementers can choose the best polynomial multiplication algorithm depending on the platform and application constraints. This has allowed efficient implementations of Saber on a wide range of platforms: high-end