Secure hardware implementations of Lattice-based cryptography

Rafael Carrera Rodriguez
Supervised by Pascal Benoit, Florent Bruguier & Emanuele Valea
Outline

• Context and Problematic
• Attack on hardware implementation of CRYStheALS-Kyber
• Implementation and analysis of protected Number Theoretic Transform
Quantum Computing vs. Classic cryptography

Quantum computing

- Superposition
- Entanglement

Classic Public-Key Cryptography

Shor’s Algorithm

New algorithms needed
(Post quantum cryptography)
A lattice is defined as the set:

\[ L(b_1, \ldots, b_n) = \left\{ \sum_{i=1}^{n} x_i b_i, x_i \in \mathbb{Z} \right\} \]

Also written as \( L(B) = \{ Bx, x \in \mathbb{Z}^n \}, B \in \mathbb{R}^{n \times n} \)

This structure can define some problems where cryptosystems can be based on:

- Shortest Vector Problem (SVP)
- Closest Vector Problem (CVP)
Implementations can leak information through

- Time
- Power
- EM radiation

Countermeasures are needed
The development of lattice-based cryptography implementations is being accelerated today.

- Most of the algorithms are not inherently secure against SCA

Question: How can the security of lattice-based cryptography implementations be improved against SCA efficiently?

Very large and open question

Algorithm?

Hardware support?

Attacks?

Countermeasures?
Correlation Power Analysis [BCO04]

- Operation to attack
  - Depends on public and secret data

- Choose random public data
- Power model of the operation
- Real trace of the operation

- Guess secret data
- Calculate correlation
  - If correlation is high enough, secret data guessed is considered correct
Polynomial multiplication in CRYSTALS-Kyber

Kyber based on variant of LWE called Module Learning with Errors

Basic elements are polynomials in the ring $R_{3329} = \mathbb{Z}_{3329}[X]/(X^{256} + 1)$

- $A \in R_q^{k \times k}$
- $s, e \in R_q^k$

- Schoolbook polynomial multiplication is expensive: $O(n^2)$
- Designers chose the Number Theoretic Transform (NTT) to accelerate it to $O(n \log n)$

After transformation, pointwise multiplication is done by:

$$\hat{h}_{2i} = \hat{f}_{2i} \hat{g}_{2i} + \hat{f}_{2i+1} \hat{g}_{2i+1} \cdot \zeta^{2br7(i)+1}$$
$$\hat{h}_{2i+1} = \hat{f}_{2i} \hat{g}_{2i+1} + \hat{f}_{2i+1} \hat{g}_{2i}$$

Direct multiplication between secret key and ciphertext in decryption routine
Related works on CPA on Kyber’s polynomial multiplication

<table>
<thead>
<tr>
<th>Software attacks</th>
<th>Works</th>
<th># Traces</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[KLdG21][MBBM+22] on ARM Cortex M4 implementations</td>
<td>200</td>
</tr>
<tr>
<td>Hardware attacks</td>
<td>None until our work</td>
<td>?</td>
</tr>
</tbody>
</table>
Goal

• Perform for the first time a CPA on a hardware implementation of CRYSTALS-Kyber and report requirements on difficulty

• Evaluate difference in difficulty w.r.t. the reported attacks on software
Setup

- Board: Digilent Basys-3 with Xilinx Artix-7 FPGA programmed with Kyber at its lowest security parameters
- Tektronix MSO64 oscilloscope, 1.25 GS/s (20 samples per clock cycle)
- EM probe Langer RF-U 5-2.
- Amplifier Femto HSA-X-2-40
Experimental results

- After using 15 sets of around 11k traces (166620 traces in total), all 512 secret key coefficients are retrieved.
- Time with Intel Core i7-11850H:
  - Trace capturing: 6 hours and 45 minutes
  - Analysis part: 2 hours and 45 minutes

Subkey 0: Maximum correlation in trace, according to number of traces sets used. In red, correct guess
Countermeasure

- Random and dummy multiplications are introduced before actual multiplications to invalidate the power model used. Only necessary before first multiplications.
- A linear-feedback shift register is used to generate the inputs to the multipliers.
- The countermeasure has an overhead of 3.80% LUTs, 6.65% flip-flops and 2-4 clock cycles.
Results of attack with countermeasure

The attack is not successful anymore, even with 10x the number of traces (1.6 million)

Limitations:

• Not valid against other possible models

• With a small modification in the model used for the first multiplications, the attack should still be possible with an increase complexity of $\approx 2^{12}$

Highest correlation of all samples for each key guess in function of the number of sets of traces used for the subkey 0 after countermeasure. In red, correct key guess.
Conclusion

• This work shows that such an attack is possible on hardware and stresses the need for countermeasures even in low power and compact hardware implementations.

• A low-cost countermeasure is presented against the proposed attack as a building block to increase security of the implementation.

• It also shows in practice the difference in difficulty between attacks on software and hardware.
Number Theoretic Transform

- One of Kyber basic operations is polynomial multiplication
- Accelerated by Number Theoretic Transform (NTT) that is a Fast Fourier Transform over the ring of integers \( \mathbb{Z}_q \), \( O(n \log n) \)

\[
X_j = \sum_{i=0}^{255} x_i \zeta^{i \cdot j} \mod q
\]

- Its efficient implementation is paramount in CRYSTALS-Kyber
Side Channel Attacks on NTT

- NTT is a highly regular algorithm
- In 2017, an attack combining algorithm information + side-channel information was published, requiring only one attack trace, called SASCA (Soft-Analytical Side-Channel Attack) [PPM17]
Related works on countermeasures

Countermeasures are focused on breaking the regularity of the algorithm:

- **Shuffling**: Randomize order of operations within an execution [ZBT19][RPBC20]
  - Two hardware implementations [ZBT19][CMJ22]
  - Shuffling may not be enough [HSST22]

- **Masking**: Randomize twiddle factor in a butterfly operation of the NTT [RPBC20]
  - Requires extra multiplications (up to 4 for certain butterfly operations)
  - No hardware implementation so far
Goal

• Explore an efficient implementation of an NTT with the masking [RPBC20] countermeasure
• Make a security analysis of this countermeasure
• Reuse hardware for polynomial multiplication
  • Use [RPBC20] countermeasure to also protect against CPA for this operation [Saa17]
• Offer configurable security at runtime, as a tradeoff with performance, by allowing the user to change the number of masks in a round at runtime
General architecture of an NTT implementation

- RAM containing coefficients
- ROM containing twiddle factors
- Control Unit
- Processing Element
Modified architecture for an NTT implementation

- RAM containing coefficients
- ROM containing twiddle factors
- Processing Element
- Control Unit
- Masking unit
Main features of NTT protected accelerator

- **Reconfigurable Processing Element:**
  - 4 non-protected butterfly ops
  - 2 or 1 protected butterfly ops
  - Point-wise multiplication
- 8 banks of memory to avoid memory collisions and dependency issues
- Masking unit to randomize twiddle factors
- Control unit to configure all routing and functional elements
Utilization results and SOTA with unprotected implementations

• Considering synthesis for an Artix-7 FPGA:

<table>
<thead>
<tr>
<th>LUTs</th>
<th>FFs</th>
<th>BRAM</th>
<th>DSPs</th>
<th>Freq (MHz)</th>
<th>ATP (for unprotected NTT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3909</td>
<td>1422</td>
<td>6</td>
<td>4</td>
<td>158</td>
<td>7.90</td>
</tr>
</tbody>
</table>

• Some unprotected implementations from the SOTA:

<table>
<thead>
<tr>
<th>Ref</th>
<th>LUTs</th>
<th>FFs</th>
<th>BRAM</th>
<th>DSPs</th>
<th>Freq (MHz)</th>
<th>ATP</th>
</tr>
</thead>
<tbody>
<tr>
<td>[LTHW22]</td>
<td>1170</td>
<td>1164</td>
<td>2</td>
<td>4</td>
<td>303</td>
<td>1.81</td>
</tr>
<tr>
<td>[BAM21]</td>
<td>801</td>
<td>717</td>
<td>2</td>
<td>4</td>
<td>222</td>
<td>2.22</td>
</tr>
<tr>
<td>[YMOS21]</td>
<td>2543</td>
<td>792</td>
<td>9</td>
<td>4</td>
<td>182</td>
<td>4.25</td>
</tr>
</tbody>
</table>
Conclusions and perspective

• The masking countermeasure of [RPBC20] can be efficiently implemented in hardware
• This module is in process of integration in a SOC made by CEA, as a proof of concept of hardware-software codesign for accelerating PQC algorithms
• Security analysis (mathematical and practical through t-tests) in progress
Thanks! Questions?

Rafael Carrera Rodriguez
rafael.carrera-rodriguez@lirmm.fr
References (1)

References (2)