First-Order Masked Kyber on ARM Cortex-M4

Work in Progress

Daniel Heinz\textsuperscript{1, 5}    Matthias J. Kannwischer\textsuperscript{2}    Georg Land\textsuperscript{3, 4}
Thomas Pöppelmann\textsuperscript{5}    Peter Schwabe\textsuperscript{2}    Daan Sprenkels\textsuperscript{6}

\textsuperscript{1}Research Institute CODE, Universität der Bundeswehr München, Germany
\textsuperscript{2}Max Planck Institute for Security and Privacy, Bochum, Germany
\textsuperscript{3}Ruhr-Universität Bochum, Germany
\textsuperscript{4}DFKI GmbH, Cyber-Physical Systems, Bremen, Germany
\textsuperscript{5}Infineon Technologies AG, Germany
\textsuperscript{6}Digital Security Group, Radboud University, Nijmegen, The Netherlands

3\textsuperscript{rd} PQC Standardization Conference, June 8th, 2021
1 Motivation and Preliminaries
2 Implementation Concept
3 Performance Evaluation
4 Conclusion
Motivation

- KyberKEM is a finalist in NIST post-quantum standardization process

Motivation

- Comparability of masked implementations between different schemes
- Gain more insights on side-channel security of proposed schemes

Side-Channel security is an important research topic

- [OSPG18] proposes first-order masked CCA2-secure Ring-Learning with errors (RLWE) scheme
- [BDK+20] presents a first-order masked implementation of Saber (Cortex-M4)
- [BGR+21] presents masked versions of Kyber on Cortex-M0
- [FBR+21] presents masked hardware accelerators using RISC-V instruction set extensions

Goal: An open-source fast first-order secure implementation on Cortex-M4
Kyber is based on Module-Learning with errors (MLWE)
RLWE and MLWE are only secure against Chosen-Plaintext Attacks
Fujisaki-Okamoto Transform: Re-encryption during decryption (CCA)
Re-encryption is dependent on the result of the decryption and therefore on the secret key
Masking of re-encryption necessary
KYBER.CCAKEM.Dec

\[ \text{KYBER.CPAPKE.Dec} \xrightarrow{c} \text{KYBER.CPAPKE.Enc} \]

Masked Comparison is non-trivial to mask
- [BPO\(^+\)20] shown to be flawed in [BDH\(^+\)21]
- [OSPG18] compares hash values ([BDH\(^+\)21] shows leakage)
- [BDK\(^+\)20] adapts [OSPG18]
Linear parts can be calculated on each share separately

$Compress_q(x, 1)$ can be calculated analogously to Masked Decode in [OSPG18]
\begin{align*}
G(m_1, h_1) & \oplus \text{PRF} \\
G(m_2, h_2) & \oplus \\
& \text{CBD} \\
& \text{Comp}(u, d_u) \\
& \text{Comp}(v, d_v) \\
& \text{Comp}(u, d_u) \\
& \text{Comp}(v, d_v) \\
& \text{Decomp}(x, 1) \\
& pk \cdot r + e_2 \\
& \end{align*}
KYBER.CPAPKE.Enc

- Masked PRF:
  - PRF is instantiated as SHAKE256
  - Efficient first-order masking approach is taken from previous work (Bertoni et al. [BDPA10])

- Masked CBD:
  - Approach from Schneider et al. (PKC2019, [SPOG19])

- Masked Decomp(x,1):
  - Approach from Oder et al. (CHES2018, [OSPG18])
  - Usage of fixed A2B conversion ([BDV21])

- No masked compression:
  - Masked comparison as proposed recently in Bos et al. ([BGR+21])
Masked CBD Sampling

- Approach from [SPOG19]
- Input: masked buffer of pseudorandom bytes (output of masked PRF)
- Basic idea:
  1. Bitsliced computation of $HW(x) - HW(y) + \eta$
  2. $B2A_q$ from [SPOG19]
  3. Subtraction of $\eta$ from each masked coefficient
- Possible for higher-order masking
Masked Comparison

- Approach from recent preprint [BGR+21]
- Basic idea:
  - No masked compression during re-encryption
  - Look up lower and upper bound for decompression of each coefficient in $u, v$ from original ct
  - For each masked coefficient in $u', v'$ from re-encryption: masked check if within possible boundaries
- A2B conversion needed to extract MSB from bound subtractions
- Alternative: use A2A conversion from [BDK+20] to extract MSB
- Possible for higher-order masking
Performance Evaluation

- Randomness generation from internal RNG (not included in the cycle counts)
- Evaluation using ARM Cortex-M4 on STM32F303 MCU (7.37 MHz)
- Table shows average cycle counts (100 executions)
- t-test in Appendix

<table>
<thead>
<tr>
<th>Operation</th>
<th>Unmasked (PQM4)</th>
<th>Masked (1st order)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KYBER.CCAKEM.KeyGen</td>
<td>751.487</td>
<td>2 520 913¹</td>
</tr>
<tr>
<td>KYBER.CCAKEM.Dec</td>
<td>847.584</td>
<td>3 596 193¹</td>
</tr>
<tr>
<td>→ KYBER.CPAPKE.Dec</td>
<td>61 505</td>
<td>134 363</td>
</tr>
<tr>
<td>→ KYBER.CPAPKE.Enc</td>
<td>683 813</td>
<td>3 122 497¹</td>
</tr>
</tbody>
</table>

¹Not final: Masked binomial sampling still shows leakage in t-test
Conclusion

- Comparison of first-order masked decapsulations (excluding randomness)
  
<table>
<thead>
<tr>
<th>Saber (Cortex-M4)</th>
<th>Kyber768 (Cortex-M0)</th>
<th>Kyber768 (Cortex-M4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 833 348</td>
<td>12 208 000</td>
<td>3 596 193¹</td>
</tr>
</tbody>
</table>

- Relative overhead factor to unmasked Cortex-M4 decapsulation of 4.2 ([FBR⁺21] with masked accelerators and RISC-V IS extension reports 3.6)

- Recent work ([NDGJ21]) shows: First-order masking is not enough

- Possible future work:
  - Improve performance on Cortex-M4 (masked binomial sampling)
  - Extend masking to higher-order on Cortex-M4
  - Combine with other countermeasures (shuffling,...)

¹Not final: Masked binomial sampling still shows leakage in t-test


Appendix: t-test Evaluation

- Evaluation on ChipWhisperer with STM32F303 target
- 100,000 traces captured
- Randomness was generated in advance (constant-time)

Figure: polysub_masked()
Figure: polyinvntt_masked()
Figure: polybasemul_masked()
Appendix: t-test Evaluation

Figure: polytomsg_masked()

Figure: polylfrommsg_masked()
Appendix: t-test Evaluation

Figure: polyreduce_masked()

Figure: polycompare_masked()