Feasibility and Performance of PQC Algorithms on Microcontrollers

Brian Hession, Jens-Peter Kaps

Second PQC Standardization Conference
August 2019
Overview

- Motivation
- Introduction to XXBX
- Getting XXBX ready for PQC
- Benchmarking
- Results
Overview

- Motivation
- Introduction to XXBX
- Getting XXBX ready for PQC
- Benchmarking
- Results
Motivation

• The move to the Internet of Things (IoT) leads to formerly “dumb” devices being connected to the Internet.
• Quantum Computers will make breaking many public key algorithms possible.
• Even IoT devices will need to run PQC algorithms.
• By 2025, over 75 billion devices will be connected to the Internet.
• 32-bit microcontrollers are projected to take lead over 8/16-bit.
• 51% of all 32-bit microcontrollers were ARM based in 2012.

PQC algorithms have to be benchmarked on microcontrollers
Overview

- Motivation
- Introduction to XXBX
- Getting XXBX ready for PQC
- Benchmarking
- Results
SUPERCOP

- Benchmarks many implementations of many primitives across multiple operations on multiple hardware platforms.
- Supports environments capable of running Linux and hosting a compiler.
- Series of shell scripts and C test harnesses, and comprehensive collection of algorithm primitive implementations.
- Verifies correct execution of implementations and times cycles required per byte processed.
- Does not measure ROM and RAM usage or power consumption.
- [http://bench.cr.yp.to/supercop.html](http://bench.cr.yp.to/supercop.html)
XBX

- eXternal Benchmarking eXtension – extends SUPERCOP
- Automated testing on real microcontrollers
- Compatibility with SUPERCOP algorithm collection ("algopacks") and output format
- Low cost hardware and software
- Our contribution to original XBX was to port it to the MSP430 platform and provide results for SHA-3 finalists.
- Measures ROM and RAM usage. Does not measure power consumption.
XXBX

• eXtended eXternal Benchmarking eXtension
• Extends XBX to
  – Support AEAD, now also PQC
  – Adds power measurement
  – Rewritten in Python 3, results in SQLite database
XXBX Components

- XBS – Benchmarking System
- XBH – Harness
- XBP – Power Shim
- XBD – Device under Test
Benchmarking Flow

- **Bootloader on XBD**
  - Writes application into ROM
  - Runs timing calibration
  - Switches to application

- **Application**
  - Loads parameters
  - Paints stack
  - Sends execution start signal
  - Executes crypto algorithm
  - Sends execution end signal
  - Sends results incl stack usage
  - Returns to bootloader
## Supported XBDs

<table>
<thead>
<tr>
<th>Board</th>
<th>by</th>
<th>CPU</th>
<th>ISA</th>
<th>Bus [bit]</th>
<th>f [MHz]</th>
<th>ROM [kByte]</th>
<th>RAM [kByte]</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSP-EXP430F5229</td>
<td>TI</td>
<td>MSP430F</td>
<td>MSP430X</td>
<td>16</td>
<td>25</td>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td>MSP-EXP430FR5994</td>
<td>TI</td>
<td>MSP430FR</td>
<td>MSP430X</td>
<td>16</td>
<td>16</td>
<td>256</td>
<td>8</td>
</tr>
<tr>
<td>MSP-EXP432P401R</td>
<td>TI</td>
<td>ARM Cortex M4F</td>
<td>ARMv7E-M</td>
<td>32</td>
<td>48</td>
<td>256</td>
<td>64</td>
</tr>
<tr>
<td>EK-TM4C123GXL</td>
<td>TI</td>
<td>ARM Cortex M4F</td>
<td>ARMv7E-M</td>
<td>32</td>
<td>80</td>
<td>256</td>
<td>32</td>
</tr>
<tr>
<td>EK-TM4C129EXL</td>
<td>TI</td>
<td>ARM Cortex M4F</td>
<td>ARMv7E-M</td>
<td>32</td>
<td>120</td>
<td>1024</td>
<td>256</td>
</tr>
<tr>
<td>NUCLEO-F091RC</td>
<td>STM</td>
<td>ARM Cortex M0</td>
<td>ARMv6-M</td>
<td>32</td>
<td>48</td>
<td>256</td>
<td>32</td>
</tr>
<tr>
<td>NUCLEO-F103RB</td>
<td>STM</td>
<td>ARM Cortex M3</td>
<td>ARMv7-M</td>
<td>32</td>
<td>72</td>
<td>128</td>
<td>20</td>
</tr>
</tbody>
</table>

- All boards are inexpensive: < $30
- RAM is very limited, below key sizes of several PQC algorithms
Overview

- Motivation
- Introduction to XXBX
- Getting XXBX ready for PQC
- Benchmarking
- Results
Getting XXBX Ready for PQC

- Save RAM
  - Moved test key (public / private) to ROM
  - Opens up stack
PQC Implementation (KEM)

- Started with SUPERCOP’s 20190110 implementations of PQ KEM algorithms.
- Added pqm4 implementations for Cortex-M4 optimizations.
  - https://github.com/mupq/pqm4
- Extended pqm4 to include some Level I and Level V versions of algorithms.
- Less than 2 weeks ago updated to SUPERCOP 20190811 which contained many more PQC algorithms.
- More than 100 variants (algorithm – parameter set combinations) are available.
- Many have more than one implementation.
PQC Signature Operations

- ek-tm4c123gxl is limited to 32kB worth of RAM.
  - Most of the Signature algorithms’ implementations exceed this memory constraint
  - Signature algorithms need to be reworked to free stack space

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Sign (B)</th>
<th>Verify (B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>dilithium</td>
<td>86,720</td>
<td>54,904</td>
</tr>
<tr>
<td>qTesla-I</td>
<td>29,336</td>
<td>23,096</td>
</tr>
<tr>
<td>qTesla-III-size</td>
<td>58,116</td>
<td>45,732</td>
</tr>
<tr>
<td>qTesla-III-speed</td>
<td>58,112</td>
<td>45,712</td>
</tr>
</tbody>
</table>

- Ways forward:
  - Limit the life of temporary variables
  - Recycle or repurpose allocated structures
Overview

- Motivation
- Introduction to XXBX
- Getting XXBX ready for PQC
- Benchmarking
- Results
Memory Constraints

- Small amount of memory available puts constraints on which algorithms can run
- Key sizes too large for RAM and/or ROM
  - E.g.: Classic McEliece, FrodoKEM, HQC, LEDAcrypt, NTS-KEM, Round5 with non-ring parameter set
Typical Problems

• Algorithms don’t compile
  – Dealing with dynamic memory allocation on system without OS.
    • Solution: Implemented simplified \_sbrk() system call
  – Algorithm depends on other libraries
    • E.g.: SHA2, Keccak, etc.

• Requires more RAM than available
  – Stack grows into .bss and .data sections
  – E.g.: Kyber 90s, Ledakem12, ntruhps4096821, firesaber
# Algorithm Selection (Levels 1 & 2)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Implementation</th>
<th>Security Level</th>
<th>Type</th>
<th>Public Key</th>
<th>Secret Key</th>
<th>Session Key</th>
<th>Cipher-text</th>
</tr>
</thead>
<tbody>
<tr>
<td>babybear</td>
<td>xbdref</td>
<td>2</td>
<td>Lattice</td>
<td>804</td>
<td>40</td>
<td>32</td>
<td>1,307</td>
</tr>
<tr>
<td>kyber512</td>
<td>m4v2</td>
<td>1</td>
<td>Lattice</td>
<td>800</td>
<td>1,632</td>
<td>32</td>
<td>736</td>
</tr>
<tr>
<td>lightsaber</td>
<td>m4v2</td>
<td>1</td>
<td>Lattice</td>
<td>672</td>
<td>1,568</td>
<td>32</td>
<td>1,088</td>
</tr>
<tr>
<td>newhope512cca</td>
<td>ref</td>
<td>1</td>
<td>Lattice</td>
<td>928</td>
<td>1,888</td>
<td>32</td>
<td>1,120</td>
</tr>
<tr>
<td>ntruhps2048509</td>
<td>m4v2</td>
<td>1</td>
<td>Lattice</td>
<td>699</td>
<td>935</td>
<td>32</td>
<td>930</td>
</tr>
<tr>
<td>r5nd1kemcca0d</td>
<td>m4v2</td>
<td>1</td>
<td>Lattice</td>
<td>634</td>
<td>666</td>
<td>16</td>
<td>698</td>
</tr>
<tr>
<td>r5nd1kemcca0dsn eik</td>
<td>m4v2</td>
<td>1</td>
<td>Lattice</td>
<td>634</td>
<td>666</td>
<td>16</td>
<td>565</td>
</tr>
<tr>
<td>r5nd1kemcca5d</td>
<td>m4v2</td>
<td>1</td>
<td>Lattice</td>
<td>445</td>
<td>477</td>
<td>16</td>
<td>565</td>
</tr>
<tr>
<td>r5nd1kemcca5dsn eik</td>
<td>m4v2</td>
<td>1</td>
<td>Lattice</td>
<td>445</td>
<td>477</td>
<td>16</td>
<td>1,005</td>
</tr>
<tr>
<td>sikep503</td>
<td>xbdref</td>
<td>2</td>
<td>Isogeny</td>
<td>378</td>
<td>434</td>
<td>16</td>
<td>402</td>
</tr>
</tbody>
</table>
# Algorithm Selection (Levels 3 & 4)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Implementation</th>
<th>Security Level</th>
<th>Type</th>
<th>Public Key</th>
<th>Secret Key</th>
<th>Session Key</th>
<th>Cipher-text</th>
</tr>
</thead>
<tbody>
<tr>
<td>kyber768</td>
<td>m4v2</td>
<td>3</td>
<td>Lattice</td>
<td>1,184</td>
<td>2,400</td>
<td>32</td>
<td>1,088</td>
</tr>
<tr>
<td>ntruhps2048677</td>
<td>m4v2</td>
<td>3</td>
<td>Lattice</td>
<td>930</td>
<td>1,234</td>
<td>32</td>
<td>1,230</td>
</tr>
<tr>
<td>ntruhrss701</td>
<td>m4v2</td>
<td>3</td>
<td>Lattice</td>
<td>1,138</td>
<td>1,418</td>
<td>32</td>
<td>1,025</td>
</tr>
<tr>
<td>mamabear</td>
<td>xbdref</td>
<td>4</td>
<td>Lattice</td>
<td>1,194</td>
<td>40</td>
<td>32</td>
<td>1,697</td>
</tr>
<tr>
<td>r5nd3kemcca0d</td>
<td>m4v2</td>
<td>3</td>
<td>Lattice</td>
<td>909</td>
<td>957</td>
<td>24</td>
<td>1,005</td>
</tr>
<tr>
<td>r5nd3kemcca0dsneik</td>
<td>m4v2</td>
<td>3</td>
<td>Lattice</td>
<td>909</td>
<td>957</td>
<td>24</td>
<td>883</td>
</tr>
<tr>
<td>r5nd3kemcca5d</td>
<td>m4v2</td>
<td>3</td>
<td>Lattice</td>
<td>780</td>
<td>828</td>
<td>24</td>
<td>883</td>
</tr>
<tr>
<td>r5nd3kemcca5dsneik</td>
<td>m4v2</td>
<td>3</td>
<td>Lattice</td>
<td>780</td>
<td>828</td>
<td>24</td>
<td>1,306</td>
</tr>
<tr>
<td>saber</td>
<td>m4v2</td>
<td>3</td>
<td>Lattice</td>
<td>992</td>
<td>2,304</td>
<td>32</td>
<td>1,088</td>
</tr>
</tbody>
</table>
## Algorithm Selection (Level 5)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Implementation</th>
<th>Security Level</th>
<th>Type</th>
<th>Public Key</th>
<th>Secret Key</th>
<th>Session Key</th>
<th>Cipher-text</th>
</tr>
</thead>
<tbody>
<tr>
<td>kyber1024</td>
<td>m4v2</td>
<td>5</td>
<td>Lattice</td>
<td>1,568</td>
<td>3,168</td>
<td>32</td>
<td>1,568</td>
</tr>
<tr>
<td>newhope1024cca</td>
<td>m4v2</td>
<td>5</td>
<td>Lattice</td>
<td>1,824</td>
<td>3,680</td>
<td>32</td>
<td>2,176</td>
</tr>
<tr>
<td>newhope1024cpa</td>
<td>m4v2</td>
<td>5</td>
<td>Lattice</td>
<td>1,824</td>
<td>1,792</td>
<td>32</td>
<td>1,120</td>
</tr>
<tr>
<td>papabear</td>
<td>xbdref</td>
<td>5</td>
<td>Lattice</td>
<td>1,584</td>
<td>40</td>
<td>32</td>
<td>1,697</td>
</tr>
<tr>
<td>r5nd5kemcca0d</td>
<td>m4v2</td>
<td>5</td>
<td>Lattice</td>
<td>1,178</td>
<td>1,242</td>
<td>32</td>
<td>1,306</td>
</tr>
<tr>
<td>r5nd5kemcca0dsn</td>
<td>m4v2</td>
<td>5</td>
<td>Lattice</td>
<td>1,178</td>
<td>1,242</td>
<td>32</td>
<td>1,095</td>
</tr>
<tr>
<td>r5nd5kemcca5d</td>
<td>m4v2</td>
<td>5</td>
<td>Lattice</td>
<td>972</td>
<td>1,036</td>
<td>32</td>
<td>1,095</td>
</tr>
<tr>
<td>r5nd5kemcca5dsneik</td>
<td>m4v2</td>
<td>5</td>
<td>Lattice</td>
<td>972</td>
<td>1,036</td>
<td>32</td>
<td>1,690</td>
</tr>
<tr>
<td>sikep751</td>
<td>pqm4ref</td>
<td>5</td>
<td>Isogeny</td>
<td>564</td>
<td>644</td>
<td>24</td>
<td>917</td>
</tr>
</tbody>
</table>
Overview

- Motivation
- Introduction to XXBX
- Getting XXBX ready for PQC
- Benchmarking
- Results
Clock Cycles

- Logarithmic Scale!
Clock Cycles till 20,000,000

- Kyber, NewHope, NTRU, Round5, and Saber complete
- Encapsulation significantly slower than decapsulation
Clock Cycles till 300,000,000

- Three Bears complete
- Reference implementation

Feasibility and Performance of PQC on Microcontrollers
Clock Cycles till 40,000,000,000

- Sike completes
- At 16 MHz, sikep503 takes 7 min, sikep751 takes 36 minutes
Our Platform only has 256 kByte of ROM
Most algorithms are significantly smaller
Our Platform only has 32 kByte of RAM
This causes several algorithms to crash, e.g., firesaber
Energy Usage

- Logarithmic scale (again)
- Mostly dependent on # of clock cycles used, power variations are minimal
XXBX Future Work

- Bigger XBD $\rightarrow$ more RAM $\rightarrow$ more algorithms
- Trying to get more algorithms to compile
  - Removing malloc
  - Removing library dependencies
- Benchmarking signature algorithms
Questions

Thank you