Efficient and Secure Hashing with SHA3

Memory, Throughput, and Security Software Optimizations on ARMv7m Cortex-M4

REU Scholar: Maggie Simmons
REU Mentor: Dr. Reza

Home University: Rice University

August 1, 2024

IoT Device Security

Challenges with IoT Security

- IoT Devices $\rightarrow$ low computational resources
- Side-Channel Attacks: extract secrets through inadvertently leaked information
- Hardware solutions
  to cryptographic problems can be expensive

Quantum Age: Coming Soon

"Now is the Time to Worry"

- $2n$ qubits to factor n-bit integer with Shor's algorithm
- In reality, many more qubits needed for error correction
- "We have time, but take action now"

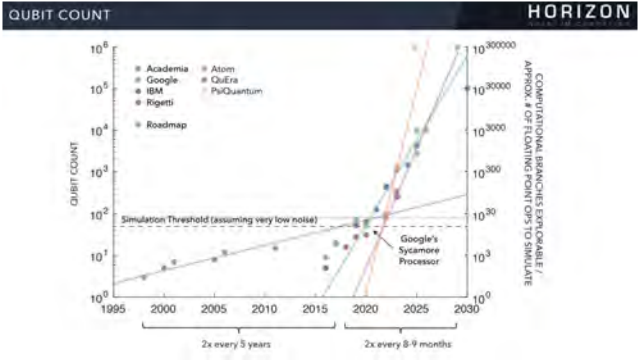


Figure. 1: Predictions on Quantum Computing Advancements from Horizon Quantum Computing

What is Post Quantum Cryptography?

Objective

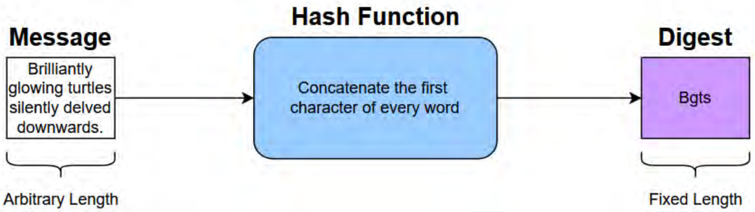
- Secure systems against both quantum and classical computers, without changing existing communications protocols

Structure

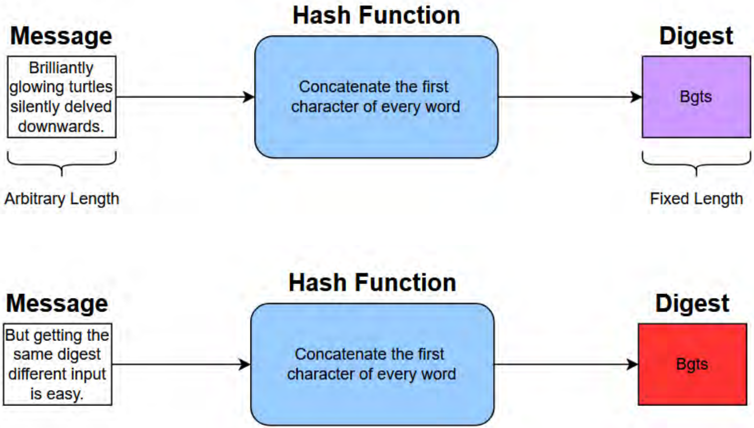
- Cryptographic algorithms built from common set of building blocks called "primitives"

Right: https://quantumai.google/discover/whatisqc

An Overview of Hash Functions



An Overview of Hash Functions



SHA3 Overview

SHA3 and Sponge Construction

- Built around Keccak,
  a permutation of the state [Div14]
- Introduction of Sponge Construction

SHA3 State

- Bitrate r + Capacity $c = 1600$
- Security
  Level depends solely on c [PA11]

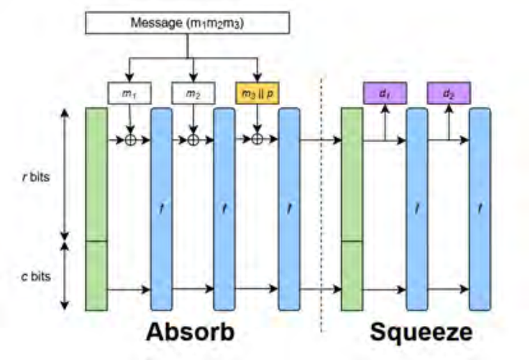
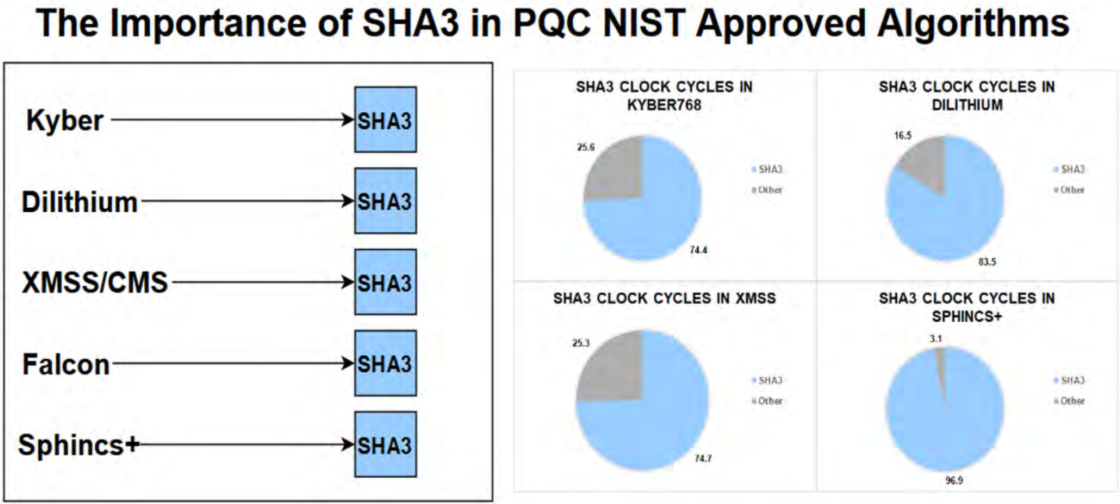


Figure. 2: Visualization of Sponge Construction

Applications of SHA3



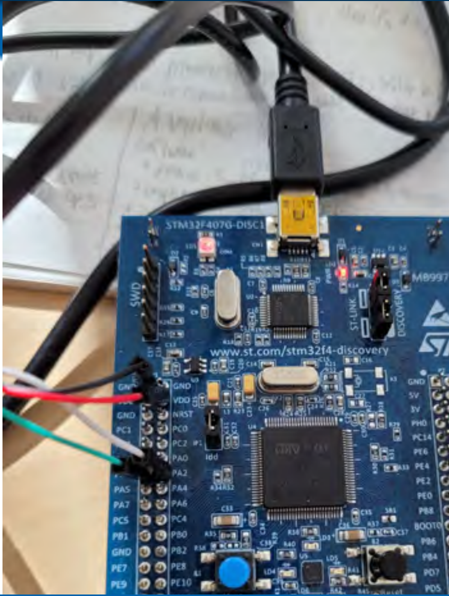
Methods and Environment

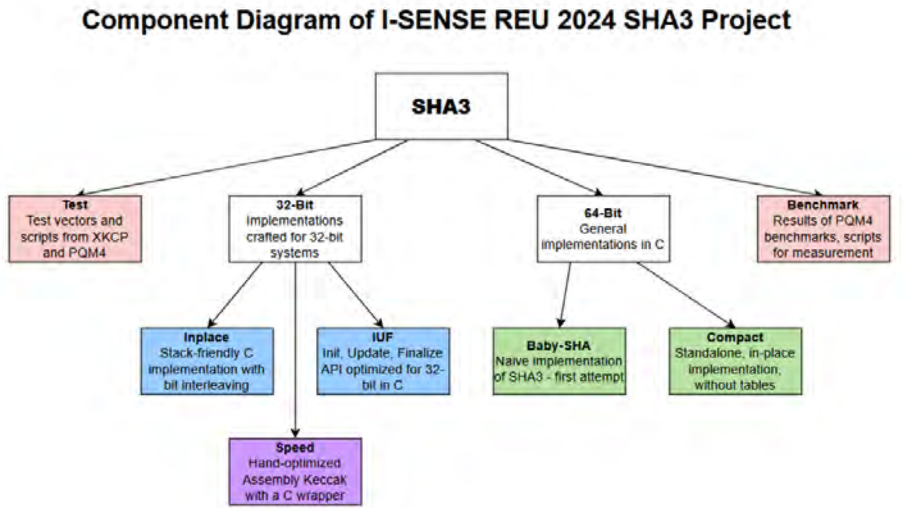
Debugging

- Cpulator: Debugging and brainstorming for ARMv7m architectures
- OpenOCD with ARM Embedded Toolchain in Visual Studio Code

Main Environment

- Fedora 40 VM
- STM32F4Discovery: measuring clock cycles (see right)
- **PQM4** and **XKCP** Cryptographic Libraries: benchmarking




Component Diagram of I-SENSE REU 2024 SHA3 Project
SHA3
Test
Test vectors and scripts from XKCP and PQM4
32-Bit
Implementations crafted for 32-bit systems
64-Bit
General implementations in C
Benchmark
Results of PQM4 benchmarks, scripts for measurement
Inplace
Stack-friendly C implementation with bit interleaving
IUF
Init, Update, Finalize API optimized for 32-bit in C
Baby-SHA
Naive implementation of SHA3 - first attempt
Compact
Standalone, in-place implementation, without tables
Speed
Hand-optimized Assembly Keccak with a C wrapper

SHA3 Assembly Implementations (Ongoing)

Keccak Assembly Implementation

- First attempt:
  20% slower than PQM4 [Ado23], but significantly more readable
- Currently on second attempt - no bit interleaving and fast explicit rotations [Ber+12]
- Will hopefully result in 1-2% throughput increase
- Readability and ease of side-channel protections vastly improved

```
.global ROL64
.thumb_func
.type ROL64, %function
.align
ROL64:
    lsr     r2, r0, #8              @ 32 - offse
    lsl     r0, r0, #24             @ offset
    orr     r0, r0, r1, lsr #8      @ 32 - offse
    orr     r1, r2, r1, lsl #24     @ offset
    bx lr
```

```
for(int i = 0; i < NUM_TESTS; i++) {
    // 4 cycles
    // 100 cycles for all 25 lanes
    t0 = hal_get_time();
    ROL64();
    t1 = hal_get_time();
    t_ROL64 += t1 - t0 - t_Overhead;

    // 1 cycle
    t0 = hal_get_time();
    ROL32();
    t1 = hal_get_time();
    t_ROL32 += t1 - t0 - t_Overhead;

// 6000-ish total for the bit interleaving (1,500 for BL,
// 3000 for 64-bit rotate
```

Conclusion

Future Exploration

- Formal verification and side-channel analysis of new SHA3 implementation
- Exploration in performance increase in context of larger algorithms
- Optimization on the newly released ARM M52 Cortex

Conclusions

- PQC algorithms increasingly important due to advancements in quantum computers
- Small optimizations to key primitives like SHA3 have huge effect on PQC implementations

Acknowledgements

Thanks to the I-SENSE Program for making this summer experience possible.

Special thanks to Dr. Reza, Maryam Taghi Zadeh, Merve Karabulut, and Daniel Owens for guiding me throughout the summer.

References

[PA11] G. Bertoni, J. Daemen, M. Peeters and G. Van Assche. *The Keccak reference*. Round 3 submission to NIST SHA-3. 2011. URL: http://keccak.noekeon.org/Keccak-reference-3.0.pdf.

[Ber+12] Gianpiero Bertoni et al. *Keccak Implementation Overview*. Round 3 Report. SHA-3 Competition, Round 3. SHA-3 Competition, 2012.

[Div14] NIST Computer Security Division. *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. FIPS Publication 202. National Institute of Standards and Technology, U.S. Department of Commerce, May 2014. URL: http://csrc.nist.gov/publications/drafts/fips-202/fips_202_draft.pdf.

[Ado23] Alexandre Adomnicai. "An update on Keccak performance on ARMv7-M". In: *Cryptology ePrint Archive* (2023).