Fourth PQC Standardization Conference

Optimization for SPHINCS+ using Intel® Secure Hash Algorithm Extensions

Qian Wang, Santosh Ghosh, Fernando Virdia, Anne Reinders, Manoj R. Sastry, Intel Labs
Thomas Hanson, University of Maryland, College Park
Notices & Disclaimers

Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex.

No product or component can be absolutely secure. Your costs and results may vary. Results have been estimated or simulated.

Intel technologies may require enabled hardware, software or service activation.

Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy.

These materials are provided “as is.” Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

© 2022 Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.
Outline

- Background
- Objective
- Instruction Set Architecture (ISA)
- Performance Improvement using Intel® ISA
- Summary
Digital Signatures

- Used to verify authenticity
  - Necessary to validate software and firmware on the platform
  - Boot time is critical to power on the platform

- Post-Quantum Digital Signatures
  - CRYSTALS-Dilithium, FALCON
    - Faster, but rely on newer lattice-based assumptions for security
  - SPHINCS+
    - Slower, but uses established hash-based assumptions
SPHINCS+

Problem Statement

- SPHINCS+ is (~x100) times slower than lattice-based schemes
- Objective: Reduce SPHINCS+ sign and verify latency

Approach

- Use Intel® SHA-NI to speed up the SPHINCS+ algorithm

<table>
<thead>
<tr>
<th>Function</th>
<th>SHA-256 Hash Calls</th>
<th>SHA-256 % of runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key Generation</td>
<td>4 775</td>
<td>87.4%</td>
</tr>
<tr>
<td>Signing</td>
<td>111 353</td>
<td>91.5%</td>
</tr>
<tr>
<td>Verification</td>
<td>12 919</td>
<td>90.2%</td>
</tr>
</tbody>
</table>

Parameter set: sphincs-sha2-128f simple with AVX2
Intel® Instruction Set Architecture (ISA)

**SHA-NI**
- Instruction for accelerating SHA-256

**AVX2**
- 256-bit register
- Allows x8 multi-buffered SHA-256
- Included in SPHINCS+ codebase

**AVX512**
- 512-bit register
- Allows x16 multi-buffered

### Number of cycles spent for ×1 hash calls and ×16 hash calls of SHA-256

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Single buffer (1 Hash)*</th>
<th>Multi-buffer (16 Hashes)*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>901</td>
<td>4759</td>
</tr>
<tr>
<td>SHA-NI</td>
<td>270</td>
<td>2081</td>
</tr>
<tr>
<td>AVX2</td>
<td>-</td>
<td>2192</td>
</tr>
<tr>
<td>AVX512</td>
<td>-</td>
<td>878</td>
</tr>
</tbody>
</table>

* From Intel® Integrated Performance Primitives (Intel® IPP) Cryptography library
+ from Intel® crypto multi-buffer library
Experimental Setup

- Uses Intel® Core i7-1185G7
  - Has SSE instruction set including SHA-NI
  - Has AVX2, AVX512 instruction sets
- Use Intel® IPP Cryptography library for SHA-NI
- SPHINCS+-SHA2-128f-simple
  - Fastest parameter set for SPHINCS+
- Reference and AVX2 implementations are constructed from the sphincsplus-master GitHub repository (https://github.com/sphincs/sphincsplus)
Accelerating SPHINCS+ in Reference Implementation

- Calls to SHA-NI implementation of all SHA256 functions

- Reference implementation is from the sphincsplus-master/ref in GitHub
SPHINCS+ Performance for Reference Implementation

- **Key Generation Time**: 70% Speedup
- **Signing Time**: 71% Speedup
- **Verification Time**: 78% Speedup
Accelerating SPHINCS+ in AVX2 Implementation

- Incorporate SHA-NI instructions
  - Calls to an AVX2 implementation of SHA-256 for parallel hash calls
  - Calls to a SHA-NI implementation of SHA-256 for serial hash calls

- AVX2 implementation is from the sphincsplus-master/sha2-avx2 in GitHub
SHA-256 Usage in SPHINCS+

serial (x1) and parallel (x8) hash calls usage

KeyGen
- Parallel (AVX2): 99.40%
- Serial: 0.60%

Signing
- Parallel (AVX2): 99.80%
- Serial: 0.20%

Verification
- Parallel (AVX2): 74.50%
- Serial: 25.50%
SPHINCS+ Performance for AVX2 Implementation

Key Generation Time

- Reference 2 (AVX2)
- Our Implementation (SHA-NI + AVX2)

Signing Time

- 8.7% Speedup

Verification Time

- 23% Speedup
Summary

- Uses SHA-NI to improve the SPHINCS+ performance for Reference implementation in GitHub
  - Significant performance improvement more than 70%.
- Combines AVX2 and SHA-NI to improve the SPHINCS+ performance for AVX2 implementation in GitHub
  - 8.7% speedup in signing time
  - 23% speedup in verification time

Thank you!