Technology Guide

Post-Quantum Cryptography: Accelerating Open Quantum Safe Library with Intel® AVX-512 Keccak 1600 Implementation

Authors

Erdinc Ozturk

Dan Zimmerman

Kirk Yap

Marcel Cornu

Tomasz Kantecki

Intel Corporation

1 Introduction

This document is intended for organizations and individuals who have incorporated Post-Quantum Cryptography into their solutions through the Open Quantum Safe liboqs library. We will review the benefits of Intel led liboqs software optimizations on Intel® Xeon® processors.

Post-Quantum Cryptography (PQC) is crucial for protecting data against future quantum computers, with the U.S. National Institute of Standards and Technology (NIST) leading the standardization of new algorithms like ML-KEM (key encapsulation) and ML-DSA (digital signatures).

The Open Quantum Safe (OQS) project and its liboqs library provide implementations of these algorithms, optimized for Intel processors with instruction sets such as Intel® Advanced Vector Extensions 2 (Intel® AVX2).

Furthermore, Intel led optimizations using Intel® Advanced Vector Extensions 512 (Intel® AVX-512) have demonstrated a significant performance boost for cryptographic operations, especially for SHA3, SHAKE, ML-KEM, and ML-DSA algorithms, with observed speed gains up to 1.64x over that of Intel AVX2.

These performance improvements are expected to enhance the efficiency of applications such as secure communications and web servers as more of the software stacks adopt PQC standards.

Table of Contents

Tables

1.1 Terminology

Table 1. Terminology

Abbreviation	Description
PQC	Post-Quantum Cryptography
OQS	Open Quantum Safe
ISA	Instruction Set Architecture
AVX	Advanced Vector Extensions
NIST	National Institute of Standards and Technology

1.2 Reference Documentation

Table 2. Reference Documents

Reference	Source
Intel® Xeon® Scalable Platform Built for Most Sensitive Workloads	https://www.intc.com/news-events/press-releases/detail/1423/intel-xeon-scalable-platform-built-for-most-sensitive
Open Quantum Safe	https://openquantumsafe.org/
FIPS203	https://csrc.nist.gov/pubs/fips/203/final
FIPS204	https://csrc.nist.gov/pubs/fips/204/final
Intel® AVX-512 - Instruction Set for Packet Processing	https://builders.intel.com/docs/networkbuilders/intel-avx-512-instruction-set-for-packet-processing-technology-guide-1645717553.pdf
liboqs 0.14.0 CAVP	https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program/details?product=20098
BoringSSL	https://boringssl.googlesource.com/boringssl
OpenSSL	https://www.openssl.org/
PQClean	https://github.com/PQClean

2 Post-Quantum Cryptography

Post-Quantum Cryptography (PQC) is a rapidly evolving field aimed at developing cryptographic algorithms that remain secure in the presence of quantum computers. Quantum algorithms, most notably Shor's algorithm, can efficiently solve problems like integer factorization and discrete logarithms, which underpin the security of widely used public-key systems such as Rivest-Shamir-Adleman (RSA), Digital Signature Algorithm (DSA), and Elliptic Curve Cryptography (ECC). As quantum computing advances, these traditional systems are at risk of being rendered obsolete, potentially compromising the confidentiality and integrity of digital communications and stored data. PQC is essential to support the transition away from these traditional public-key systems and for ensuring the confidentiality and integrity of data both at rest, and in transit across many sectors including telecommunications, finance, government, healthcare, and critical infrastructure.

The U.S. National Institute of Standards and Technology (NIST) has been leading a global standardization effort since 2016, culminating in the selection of several algorithms, such as Module-Lattice-Based Key Exchange Mechanism (ML-KEM) for key encapsulation and Module-Lattice-Based Digital Signature Algorithm (ML-DSA) for digital signatures, as part of its first set of post-quantum cryptographic standards. These efforts are paving the way for a secure transition to quantum-resilient cryptographic systems.

An increasing number of software libraries, including liboqs from Open Quantum Safe, OpenSSL, and BoringSSL, are incorporating quantum-resistant algorithms to facilitate the transition to quantum-resilient cryptography. These tools allow developers to implement and test secure protocols such as TLS 1.3 with post-quantum capabilities. Projects like PQClean offer portable C implementations of NIST candidate algorithms for integration into other projects.

3 Open Quantum Safe Library

The liboqs library, developed by the Open Quantum Safe (OQS) project, is a widely used software library that combines implementations of NIST-selected post-quantum cryptographic algorithms with ongoing academic and industry research. It is designed to be portable and flexible, offering multi-architecture support through unoptimized scalar implementations that can run on a wide range of hardware platforms. For performance-critical applications, Intel AVX2-optimized implementations for select algorithms are also provided for the Intel® architecture, enabling improved performance on modern processors.

To facilitate integration into existing cryptographic infrastructure, the OQS project also maintains the OQS provider, which allows seamless use of post-quantum algorithms within OpenSSL 3.x. This makes it possible for developers and organizations to more easily integrate PQC support into their software stacks and speed up the migration to a quantum-resilient future.

Version 0.14.0 of the liboqs library includes optimized low-level implementations of the SHA3 and SHAKE cryptographic hash functions. These implementations utilize the Intel AVX-512 instruction set extensions, present in current Intel processors, to enhance the efficiency of performance-critical Keccak operations. As a result, substantial performance gains have been achieved for these hash functions and for algorithms that depend on them, such as ML-KEM and ML-DSA.

4 Module-Lattice-Based Key-Encapsulation Mechanism Standard (ML-KEM)

ML-KEM is a post-quantum cryptographic algorithm designed to provide secure key encapsulation (see FIPS203). It is based on the CRYSTALS-Kyber algorithm and is known for its efficiency and security. The algorithm involves three main operations: key generation, encapsulation, and decapsulation.

- **Key Generation:** This process generates a public and private key pair. The public key is used for encapsulation, while the private key is used for decapsulation.
- **Encapsulation:** This operation takes the public key and generates a ciphertext and a shared secret. The ciphertext is sent to the recipient, and the shared secret is used for secure communication.
- **Decapsulation:** The recipient uses their private key to decrypt the ciphertext and recover the shared secret.

5 Module-Lattice-Based Digital Signature Algorithm (ML-DSA)

ML-DSA is a post-quantum digital signature algorithm based on the CRYSTALS-Dilithium algorithm (see FIPS204). It is designed to provide secure and efficient digital signatures, and its main operations are:

- **Key Generation:** This process generates a public and private key pair. The public key is used for signature verification, while the private key is used for signing.
- **Signing:** This operation takes the private key and a message to generate a digital signature. The signature ensures the authenticity and integrity of the message.
- **Verification:** The recipient uses the public key to verify the digital signature. If the signature is valid, the message is considered authentic and unaltered.

6 Secure Hash Algorithm 3 (SHA3) and Secure Hash Algorithm Keccak (SHAKE)

SHA3 is a cryptographic hash function (see NIST SP 800-185) based on the Keccak algorithm. It uses a sponge construction, which involves two main phases: absorption and squeezing.

- **Absorption Phase:** The input message is divided into blocks, and each block is absorbed into the internal state of the sponge function. This involves XORing the message blocks with the state and applying the Keccak permutation.
- **Squeezing Phase:** After all message blocks are absorbed, the sponge function produces the hash output by squeezing the state. This involves applying the Keccak permutation and extracting the output blocks.

SHAKE is an extendable-output function (XOF) based on the Keccak algorithm (see NIST SP 800-185). It can produce variable-length outputs and is used for various

cryptographic purposes, including key derivation and pseudorandom number generation.

- **SHAKE128 and SHAKE256** are two common variants of SHAKE, with different security strengths. They follow the same sponge construction as SHA3, with absorption and squeezing phases.

6.1 ML-DSA and ML-KEM Dependency on SHA3 and SHAKE

ML-DSA (FIPS204) and ML-KEM (FIPS203), rely on the SHA3 and SHAKE algorithms (NIST SP 800-185) for various cryptographic operations. The SHA3 algorithm is used for hashing and SHAKE algorithm is used for generating pseudo-random values:

- **Hashing:**
  - Before signing a message, ML-DSA uses SHA3 to hash the message. This hash value is then used in the signing algorithm to generate the digital signature.
  - During the encapsulation process, ML-KEM uses SHA3 to hash the public key and other input values. This hash value is then used to generate the ciphertext and the shared secret.
- **Pseudo-Random Value Generation:**
  - During key generation and signing, ML-DSA uses SHAKE to generate pseudo-random values.
  - ML-KEM uses SHAKE to generate pseudo-random values during key generation and encapsulation.

Subsequently, performance of ML-KEM and ML-DSA operations depend on the performance of SHA3 and SHAKE compute components.

6.2 Keccak Permute Optimization

The Keccak permute operation, specifically the Keccak-f variant, is a core component of the SHA3 and SHAKE cryptographic hash functions.

- **SHA3:** Secure Hash Algorithm 3, which produces fixed-length hash values.
- **SHAKE:** Secure Hash Algorithm Keccak, which produces variable-length extendable outputs.

The Keccak-f permute operation ensures the security and efficiency of these cryptographic functions by providing strong diffusion and mixing of the input data.

The Keccak-f permute operation consists of a sequence of 24 rounds. Each round applies a series of transformations to the 1600-bit state, ensuring the security and randomness of the permutation.

Steps of the Keccak-f Permute Operation include the following transformations:

1. **Initialization:** The internal state of the Keccak algorithm is initialized with the input data. The state is represented as a 3-dimensional array of bits.
2. **Round Constants:** Each round of the Keccak-f permute operation uses a unique round constant to ensure the security and randomness of the permutation. These constants are pre-defined and vary for each round.
3. **Round Function:** The round function consists of five main steps, which are applied sequentially to the state array:
   - **Theta:** This step ensures the diffusion of bits across the state array by XORing each bit with a parity value computed from neighbouring bits.
   - **Rho:** This step rotates the bits within the state array to further mix the data.
   - **Pi:** This step permutes the bits within the state array by rearranging their positions.
   - **Chi:** This step applies a non-linear transformation to the bits within the state array, ensuring the complexity and security of the permutation.
   - **Iota:** This step XORs the state array with the round constant, adding an additional layer of randomness.
4. **Final State:** After all 24 rounds are completed, the final state of the Keccak algorithm is produced. This state is then used to generate the hash output or extendable output, depending on the specific function (SHA-3 or SHAKE).

6.3 Intel® AVX-512 Optimization

In our advanced implementation of the Keccak algorithm, Intel AVX-512 instruction set architecture (ISA) was leveraged to maximize throughput and parallelism, using modern Intel processors. Each logical state register within the Keccak permutation is directly mapped to a native single instruction, multiple data (SIMD) register, ensuring that computation remains tightly bound to the hardware for minimal latency. Specifically, with the extended vector length of Intel AVX-512 enabled, we assign each state register to a YMM register, preserving compatibility from the first generation of Intel® Xeon® Scalable processor onward.

To enhance computational density, we make extensive use of ternary logic instructions - an advanced feature introduced with Intel AVX-512. Unlike traditional binary logic operations that operate on two inputs, ternary logic instructions can simultaneously process three input values within a single operation. This capability significantly broadens the range of logical combinations achievable in hardware, streamlining complex sequences that would otherwise require multiple instructions. See Intel® AVX-512 - Instruction Set for Packet Processing for more details about ternary logic instructions.

In the context of the Keccak algorithm, this means operations like those found in the Chi and Theta steps - where intricate bitwise manipulations and conditional logic are essential - can be executed more efficiently by condensing them into fewer instructions reducing both instruction count and execution time. This not only improves the speed of single-state hashing but also scales efficiently when handling multiple states in parallel, making our implementation highly suitable for modern cryptographic workloads.

A notable aspect of our optimization is the multi-state capability: the design allows the simultaneous processing of up to four independent Keccak states in parallel. This batch processing approach is crucial for applications requiring high-throughput hashing, i.e. multi-buffer SHAKE128 or SHAKE256.

7 Software Configuration

libOQS location: https://github.com/open-quantum-safe/liboqs

To obtain tested software version please follow the steps below:

```
> git clone https://github.com/open-quantum-safe/liboqs.git
> git checkout 0.14.0
```

Note: 8f926065ebd90591106e121a847f586488e6071f is the liboqs commit ID from which Intel AVX-512 optimized code is available. This corresponds to version 0.13.1-dev (major: 0, minor: 13, patch: 1, pre-release: -dev).

The steps below show how to configure and compile the software to obtain Intel AVX-512 and Intel AVX2 optimized versions. Intel AVX-512 optimizations are available by default on Intel architecture processors. If the hardware platform doesn't support Intel AVX-512 instructions, then the code falls back to implementation suitable for the hardware.

7.1 Intel AVX-512 Optimized Configuration

To compile Intel AVX-512 optimized software configuration please follow the following instructions:

```
> mkdir build_avx512
> cd build_avx512
> cmake ..
> make -j
```

After successful build, "speed_common", "speed_kem" and "speed_sig" applications can be found in "build_avx512/tests" directory.

7.2 Intel AVX2-Only Optimized Configuration

To compile Intel AVX2 optimized software configuration please follow the following instructions:

```
> mkdir build_avx2
> cd build_avx2
> cmake -DOQS_USE_SHA3_AVX512VL=OFF ..
> make -j
```

After successful build, "speed_common", "speed_kem" and "speed_sig" applications can be found in "build_avx2/tests" directory.

8 Performance Results

The liboqs library comes with several performance tests applications. "speed_common", "speed_kem" and "speed_sig" are used to measure gains from Intel AVX-512 optimization for SHA3 and SHAKE, as well as ML-KEM and ML-DSA.

See below how the tools were used and performance results captured on the test system (see 9 for more details).

8.1 Single Buffer SHA3 and SHAKE Results

The liboqs "speed_common" tool doesn't benchmark multi buffer SHAKE implementations which are available in the library. However, it provides comprehensive tests for single buffer algorithm implementations which is the point of focus in this section.

Table 3. SHA3 and SHAKE Performance Results with speed_common Tool

Algorithm	Intel® Advanced Vector Extensions 2 (Intel® AVX2) [CPU Cycles Mean]	Intel® Advanced Vector Extensions 512 (Intel® AVX-512) [CPU Cycles Mean]	Intel® AVX-512 Gain
OQS_SHA3_sha3_256	72,879	46,374	1.57
OQS_SHA3_sha3_384	93,990	59,823	1.57
OQS_SHA3_sha3_512	135,044	86,221	1.57
OQS_SHA3_shake128	114,866	73,485	1.56
OQS_SHA3_shake256	142,704	91,488	1.56

This is the command used to get throughput and cycle cost of single buffer SHA3-256, SHA3-384 and SHA3-512 operations on for a 8,192 byte message:

```
> taskset -c 4 ./speed_common -d 3 --msglen 8192 sha3
```

Note that "taskset -c 4" schedules execution of the subsequent command on CPU 4. "-d 3" speed common option executes the test for 3 seconds, see "./speed_common -h" for more options.

These are two commands used to get throughput and cycle cost of SHAKE128 and SHAKE256 operations absorbing and squeezing 8,192 bytes:

```
> taskset -c 4 ./speed_common -d 3 --outlen 8192 --msglen 8192 shake128
> taskset -c 4 ./speed_common -d 3 --outlen 8192 --msglen 8192 shake256
```

The Intel AVX-512 optimized code gain over previously used Intel AVX2 optimized code gravitates around 1.56x for the 8,192 byte message size.

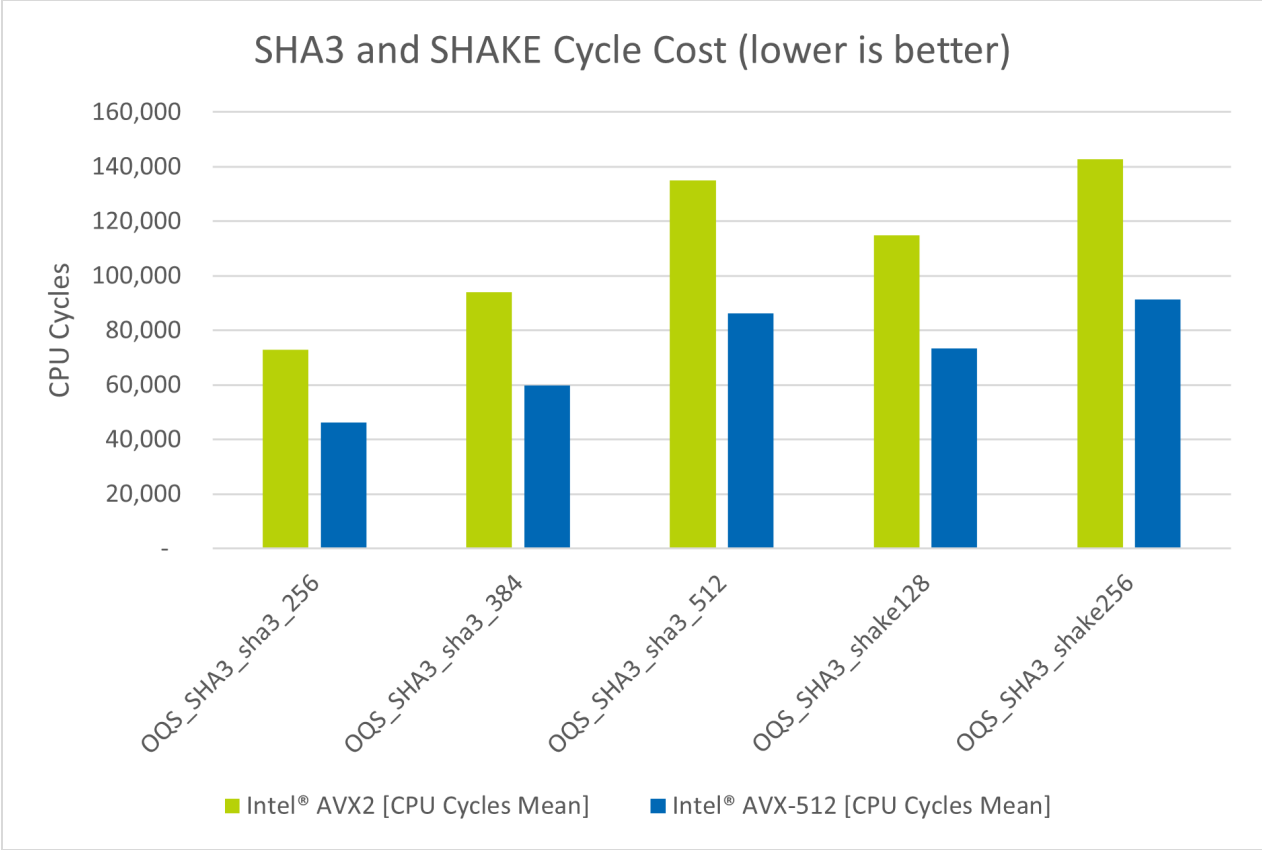

Figure 1. SHA3 and SHAKE Performance results with speed_common Tool on 8192 Byte Message (lower is better)

8.2 ML-KEM Results

These three commands were used to get throughput and CPU cycle cost of ML-KEM-512, ML-KEM-768 and ML-KEM-1024:

```
> taskset -c 4 ./speed_kem ML-KEM-512
> taskset -c 4 ./speed_kem ML-KEM-768
> taskset -c 4 ./speed_kem ML-KEM-1024
```

Note that "taskset -c 4" schedules execution of the subsequent command on CPU 4.

Table 4. ML-KEM Performance Results with speed_kem Tool (lower cycles is better)

Algorithm	Operation	Intel® AVX2 [CPU Cycles Mean]	Intel® AVX-512 [CPU Cycles Mean]	Intel® AVX-512 Gain
ML-KEM-512	decaps	33,462	22,492	1.49
ML-KEM-512	encaps	27,651	19,493	1.42
ML-KEM-512	keygen	25,179	17,401	1.45
ML-KEM-768	decaps	51,746	34,376	1.51
ML-KEM-768	encaps	42,047	28,593	1.47
ML-KEM-768	keygen	41,002	27,548	1.49
ML-KEM-1024	decaps	71,117	46,145	1.54
ML-KEM-1024	encaps	57,460	37,865	1.52
ML-KEM-1024	keygen	55,382	36,224	1.53

The Intel AVX-512 optimized SHA3 and SHAKE implementation helps improve ML-KEM operations from 1.42x to 1.54x. Most performance gain is observed for the highest ML-KEM security level (ML-KEM-1024).

8.3 ML-DSA Results

These three commands were used to get throughput and CPU cycle cost of ML-DSA-44, ML-DSA-65 and ML-DSA-87:

```
> taskset -c 4 ./tests/speed_sig ML-DSA-44
> taskset -c 4 ./tests/speed_sig ML-DSA-65
> taskset -c 4 ./tests/speed_sig ML-DSA-87
```

Note that "taskset -c 4" schedules execution of the subsequent command on CPU 4.

Table 5. ML-DSA Performance Results with speed_sig Tool (lower cycles is better)

Algorithm	Operation	Intel® AVX2 [CPU Cycles Mean]	Intel® AVX-512 [CPU Cycles Mean]	Intel® AVX-512 Gain
ML-DSA-44	keypair	80,918	53,562	1.51
ML-DSA-44	sign	224,482	167,985	1.33
ML-DSA-44	verify	80,227	52,746	1.52
ML-DSA-65	keypair	134,847	84,628	1.59
ML-DSA-65	sign	360,848	267,424	1.34
ML-DSA-65	verify	131,733	82,849	1.59
ML-DSA-87	keypair	211,912	130,721	1.62
ML-DSA-87	sign	439,121	314,118	1.39
ML-DSA-87	verify	206,553	125,279	1.64

The Intel AVX-512 optimized SHA3 and SHAKE also helps improve ML-DSA performance and gains depend on security level and type of operation. Key pair operation gains range from 1.51x for ML-DSA-44 to 1.62x for ML-DSA-87. Verify operation gains range from 1.52x for ML-DSA-44 to 1.64x for ML-DSA-87. Sign operation gains range from 1.33x for ML-DSA-44 to 1.39x for ML-DSA-87.

9 Summary

Optimized Intel AVX-512 Keccak-1600 provides substantial performance enhancements for SHA-3 and SHAKE algorithms, especially the multi-buffer implementation. As fundamental hash functions in NIST FIPS203 ML-KEM and FIPS204 ML-DSA PQC standards, this optimization delivers up to 1.64x speedup compared to Intel AVX2. Specifically, it enhances ML-KEM performance up to 1.54x and ML-DSA up to 1.64x. Such algorithmic speed-ups are expected to proportionally improve application-level efficiency, notably in web server connections and other use cases where cryptographic processing is a bottleneck.

With the growing adoption of post-quantum cryptography, Intel AVX-512-optimized implementation, integrated through the liboqs library, delivers enhanced security and efficiency on Intel processors. This advancement supports the development of resilient and high-performance digital infrastructure for the quantum era. Notably, Intel AVX-512 optimized liboqs v0.14.0 has earned CAVP certification from NIST on a range of Intel Xeon processors and Intel Xeon SoCs; for further details, refer to Table 2: liboqs 0.14.0 CAVP.

10 System Configuration

CPU Model	Intel® Xeon® 6767P processor
Microarchitecture	GNR_X2
Sockets	2
Cores per Socket	64
Hyperthreading	Enabled
CPUs	256
Intel Turbo Boost	Enabled (***Note***: Intel Turbo Boost was ***disabled*** through software during the tests.)
Base Frequency	2.4GHz
All-core Maximum Frequency	3.6GHz
Maximum Frequency	3.9GHz
NUMA Nodes	4
Prefetchers	L2 HW, L2 Adj., DCU HW, DCU IP
PPINs	6a706332a9a77d92,08023a6fd13095a5
Accelerators Available [used]	DLB 0 [0], DSA 8 [0], IAA 8 [0], QAT 0 [0]
Installed Memory	256GB (16x16GB DDR5 5600 MT/s [5600 MT/s])
Hugepagesize	2048 kB
Transparent Huge Pages	madvise
Automatic NUMA Balancing	Enabled
NIC	1x Intel(R) Ethernet Controller I210 Gigabit Network Connection, 2x BCM57416 NetXtreme-E Dual-Media 10G RDMA Ethernet Controller
Disk	1x 894.3G Micron_7450_MTFDKBG960TFR
BIOS	BHSDCRB1.IPC.3544.O01.2503110235
Microcode	0x10003a5
OS	Ubuntu 24.04.1 LTS
Kernel	6.8.0-50-generic
TDP	350 watts
Power & Perf Policy	Normal (6)
Frequency Governor	performance
Frequency Driver	intel_pstate
Max C-State	9
Vulnerability	CVE-2017-5715:OK, CVE-2017-5753:OK, CVE-2017-5754:OK, CVE-2018-12126:OK, CVE-2018-12127:OK, CVE-2018-12130:OK, CVE-2018-12207:OK, CVE-2018-3615:OK, CVE-2018-3620:OK, CVE-2018-3639:OK, CVE-2018-3640:OK, CVE-2018-3646:OK, CVE-2019-11091:OK, CVE-2019-11135:OK, CVE-2020-0543:OK, CVE-2022-40982:OK, CVE-2023-20569:OK, CVE-2023-20593:OK

1025/TK/PDF 367250-001US Please Recycle (if printed)