Software Assurance Tips

A product of the Software Assurance Tips Team[2]

Jon Hood

Monday 19th September, 2022

CNSA 1.0	CNSA 2.0
Elliptic Curve Digital Signature Algorithm (ECDSA) using P-384 Rivest-Shamir-Adleman (RSA) using minimum 3072-bit modulus	Leighton-Micali Signature (LMS) Xtended Merkle Signature Scheme (XMSS)

Table 1: Software and Firmware Signing Algorithms: CNSA 1.0 vs 2.0

1 Commercial National Security Algorithm (CNSA) Suite 2.0

Updated Friday 23rd September, 2022

As the government switches to quantum-resistant and post-quantum cryptographic algorithms, the CNSA suite (formerly Suite-B algorithms) is undergoing some changes with a new release: version 2.0. This release of the CNSA suite includes the following information:[3]

- a requirement to be compliant and support these algorithms by 2035
- a requirement to use NIST 800-208 for application signing which includes:
  - a recommendation to use at least LMS with the SHA-256/192 parameters or XMSS with any parameters
  - deprecation of all other algorithms by 2025 (2030 for current deployments)
- a requirement to use FIPS PUB 197 for symmetric block cipher encryption, requiring 256-bit keys (systems should already be doing this)
- a requirement to use FIPS PUB 180-4 for hashing, requiring either the SHA-384 or SHA-512 algorithms (systems should already be doing this)
- a requirement to use CRYSTALS-Kyber for public-key encryption, requiring Level V parameters and a deprecation of RSA, DH, and ECC once implemented by 2035
- a requirement to use CRYSTALS-Dilithium for digital signatures by 2035

Navigating these new requirements imposes a technical hurdle, and acquisition managers should prepare for this move in their timelines. This article attempts to help explain the requirements at a technical level to help drive the transition. At the time of this writing, the algorithms in CNSA 2.0 are not included in any mainline cryptography libraries. For the examples, the Open Quantum Safe (OQS) Project OpenSSL 1.1.1-stable snapshot 2022-08 (`https://github.com/open-quantum-safe/openssl/releases/tag/OQS-OpenSSL-1_1_1-stable-snapshot-2022-08`) is used on a vanilla Ubuntu 22.04 installation, and the reference implementations for all algorithms are used for key implementation.

1.1 Software and Firmware Signing

Software and firmware signing can begin transitioning to CNSA 2.0 immediately with plans to retire CNSA 1.0 support.

Current PKI infrastructure in the DoD only supports using RSA for digital signatures. Nevertheless, the LMS reference implementation includes LMS key generation and signature verification for LMS with SHA-256, one of the permitted signature algorithms in NIST SP 800-208.[1, § 4.1] Using the demo application that is available with the reference from `https://github.com/cisco/hash-sigs`, an application/firmware hashing mechanism can be implemented:

```
1 $ ./demo genkey mykey 10/8,5/1
2 $ ./demo sign mykey myapp.exe
3 $ ./demo verify mykey myapp.exe
```

Listing 1: LMS Signature of myapp.exe

CNSA 1.0	CNSA 2.0
Advanced Encryption Standard (AES) using 256-bit keys	Advanced Encryption Standard (AES) using 256-bit keys

Table 2: Symmetric Block Ciphers: CNSA 1.0 vs 2.0

CNSA 1.0	CNSA 2.0
Secure Hash Algorithm 2 (SHA-2) with 384-bit digests (SHA-384)	Secure Hash Algorithm 2 (SHA-2) with 384-bit or 512-bit digests (SHA-384 or SHA-512)

Table 3: Hashing: CNSA 1.0 vs 2.0

Line 1 will generate a `mykey.prv` and a `mykey.pub` file using 2 levels of Merkle trees. The first level has a height of 10 and Winternitz parameter of 8, and the second level a height of 5 and Winternitz parameter of 1. The NIST 800-208 standard permits Merkle tree heights of 5, 10, 15, 20, or 25, and RFC 8554 specifies that the Winternitz parameter may be 1, 2, 4, or 8. There are limits to the number of signatures an LMS key can be used for, and higher Merkle tree heights can take a long time to generate a key (it took nearly 2 hours for me to generate a key with $h = 25$). At $h = 5$, the key is only good for a handful of signatures. Generally speaking, larger Merkle tree heights take longer to generate but can be safely used for more signatures. Higher Winternitz parameters minimize the signature size while increasing the time. RFC 8554 documents that a single Merkle tree level generated with `15/8` will be good for about 30,000 signatures, while a 2-level Merkle tree generated with `25/8,15/8` would be good for more than 1 trillion signatures.

Line 2 signs a file (in this case, an executable) using `mykey.prv`. This generates a `myapp.exe.sig` file with the signature. An additional note of caution should be made: each signature advances the private key. `mykey.prv` changes after every valid signature. If signatures are generated using the same key information more than once, the security of those hashes may leak information about the private key. Backing up the private key is not sufficient: a backup solution must back up the key *with the number of signatures that have been made using it*, and a restoration process must advance the key by that many signatures.

Line 3 only needs to have `mykey.pub` present to verify the signature on another machine.

1.2 Symmetric Key Algorithms

No change is made between CNSA 1.0 and CNSA 2.0 for symmetric block ciphers. AES with 256-bit keys is the only algorithm permitted.

1.3 Hashing

CNSA 2.0 officially adds SHA-512 as an approved hash, but continues to permit SHA-384 as an approved hashing algorithm.

Some may notice that internal hashes in the software and firmware signing algorithms use smaller hash sizes or different hashing algorithms. For example, LMS permits the use of SHAKE-256, one of the unapproved SHA-3 algorithms from FIPS 202, internally. “NSA has no concerns about this, but does not anticipate approving SHA-3 algorithms for general-purpose use at this time.”[4]

1.4 Asymmetric Digital Signatures and Key Establishment

CNSA 2.0 deprecates all of the algorithms of CNSA 1.0, requiring the use of the CRYSTALS-Dilithium algorithm.

To achieve NIST Level V functionality for the CRYSTALS-Kyber algorithm, the CRYSTALS-Kyber parameters must be set to use KYBER1024 or KYBER1024-90s. While both parameter sets are permitted

CNSA 1.0	CNSA 2.0
Rivest-Shamir-Adleman (RSA) with minimum 3072-bit modulus Elliptic Curve Digital Signature Algorithm (ECDSA) using curve P-384 Diffie-Hellman (DH) Key Exchange with minimum 3072-bit modulus Elliptic Curve Diffie-Hellman (ECDH) Key Exchange using curve P-384	CRYSTALS-Kyber at Level V CRYSTALS-Dilithium at Level V

Table 4: Asymmetric Digital Signature and Key Establishment: CNSA 1.0 vs 2.0

by CNSA 2.0, KYBER1024 uses SHAKE and SHA-3 internally, both of which, on their own, are unapproved algorithms. Instead, KYBER1024-90s uses AES-256-CTR and SHA-2 internally, using SHAKE-256 only for the Extendable-Output Function (XOF).

Amazon Web Services (AWS) and the Open Quantum Safe (OQS) project implement CRYSTALS. While production implementations are not recommended yet, configuring the OQS port of OpenSSL so that it uses CRYSTALS-Kyber can be done with the command in Listing 2. This will configure OpenSSL to use the CNSA 2.0 implementations of KYBER1024.

```
$ ./Configure -DOQS_DEFAULT_GROUPS="kyber1024,p521_kyber1024,kyber90s1024,p521_kyber90s102
```

Listing 2: Configuring OpenSSL TLS 1.3 for using CRYSTALS-Kyber

You may notice that the implementation in OQS OpenSSL includes key exchange using ECDH with the P-521 curve, a curve not permitted under CNSA 1.0 and 2.0. There is not yet a reference implementation for OQS using CRYSTALS-Dilithium; however, we should see updates to the standard to implement this soon. Currently, only the Dilthium5 parameter sets achieve NIST Level V functionality. Currently, both Dilithium5 and Dilithium5-AES (where AES-256-CTR and SHA-2 are used instead of SHAKE-256) achieve the CNSA 2.0 parameter goals. This creates a public key of 2592-bytes and a secret key of 4864-bytes.

During the CNSA 1.0 to 2.0 transition, there will likely be several years where both implementations continue to exist. During this time period, expect to see several hybrid implementations (such as KYBER's "90s" parameter sets and Dilthium's AES algorithm options. These will likely begin to fade out of use as the 1.0 suite continues its slow march to obsolescence.

1.5 Conclusion

We are likely more than a decade away from seeing CNSA-2.0-only implementations; however, for project acquisitions and contract writers, it may be necessary to plan for these architecture changes before the next contract awards. The CNSA 2.0 implementation timeline indicates that the hybrid approaches will start rolling out in the 2025 timeframe with preference given to the 2.0 algorithms over the 1.0 algorithms by 2030. Finally, we should see the obsolescence of 1.0 as it is replaced with 2.0-only implementations in the 2033-2035 timeframe. As with most government mandates, expect delays, but prepare for the best. We'll know if the timeline is on-track if the final CRYSTALS parameter sets are published by the end of 2024.

References

[1] David Cooper et al. Recommendation for Stateful Hash-Based Signature Schemes. Tech. rep. Special Publication (SP) 800-208. Washington, D.C.: National Institute of Standards and Technology, 2020. DOI: 10.6028/NIST.SP.800-208. URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-208.pdf.

[2] Jon Hood, ed. SwATips. https://www.SwATips.com/.

[3] National Security Agency. "Announcing the Commercial National Security Algorithm Suite 2.0". In: (2022). URL: https://media.defense.gov/2022/Sep/07/2003071834/-1/-1/0/CSA_CNSA_2.0_ALGORITHMS_.PDF.

[4] National Security Agency. "The Commercial National Security Algorithm Suite 2.0 and Quantum Computing FAQ". In: (2022). URL: https://media.defense.gov/2022/Sep/07/2003071836/-1/-1/0/CSI_CNSA_2.0_FAQ_.PDF.