Composite Signature – 02 Updates

Mike Ounsworth, John Gray, Jan Klaussner, Max Pala, Scott Fluhrer

LAMPS 120 -- Vancouver

July 2024

Changes affecting Implementation Interoperability

- Should be compatible with pre-adopted draft-ounsworth-pq-composite-sigs ver 13 except RSA-PSS based composited didn’t specify PSS parameter salt length:
  - MLDSA44-RSA2048-PSS-SHA256 now uses a 256-bit salt
  - MLDSA44-RSA3072-PSS-SHA512 now uses a 512-bit salt

ENTRUST

Composite Signature Generation Process

1. Compute a hash of the Message

   M’ = Domain || HASH(Message)

Domain and Prehash concatenation now in this step

2. Generate the n component signatures independently, according to their algorithm specifications.

   S1 := Sign(K1, A1, M’)

   S2 := Sign(K2, A2, M’)

3. Encode each component signature S1 and S2 into a BIT STRING according to its algorithm specification.

   signature ::= SEQUENCE { S1, S2 }

4. Output signature

Where:

K1, K1 are signing private keys for each component algorithm

A1. A2 are the component signing algorithms (for example ML-DSA, RSA, ECDSA)

Domain is a domain separator value which is the DER encoding of the Object ID

ENTRUST

Composite Signature Verification Process

1. Check keys, signatures, and algorithms lists for consistency.
2. Compute a Hash of the Message

   M’ = Domain || HASH(Message)

Domain and Prehash concatenation now in this step

3. Check each component signature individually, according to its algorithm specification. If any fail, then the entire signature validation fails.

   if not $Verify_{A1}$(P1, M’, S1, A1) then output "Invalid signature"

   if not $Verify_{A2}$(P2, M’, S2, A2) then output "Invalid signature"

   output "Valid signature"

“applications MUST output "Valid signature" (true) if and only if all component signatures were successfully validated, and "Invalid signature" (false) otherwise.”

ENTRUST

ML-DSA Composite Signatures

Composite Signature Algorithm	First Algorithm	Second Algorithm	Pre-Hash
id-MLDSA44-RSA2048-PSS-SHA256	MLDSA44	SHA256WithRSAPSS	SHA256
id-MLDSA44-RSA2048-PKCS15-SHA256	MLDSA44	SHA256WithRSAEncryption	SHA256
id-MLDSA44-Ed25519-SHA512	MLDSA44	Ed25519	SHA512
id-MLDSA44-ECDSA-P256-SHA256	MLDSA44	SHA256withECDSA	SHA256
id-MLDSA44-ECDSA-brainpoolP256r1-SHA256	MLDSA44	SHA256withECDSA	SHA256
id-MLDSA65-RSA3072-PSS-SHA512	MLDSA65	SHA512WithRSAPSS	SHA512
id-MLDSA65-RSA3072-PKCS15-SHA512	MLDSA65	SHA512WithRSAEncryption	SHA512
id-MLDSA65-ECDSA-P256-SHA512	MLDSA65	SHA512withECDSA	SHA512
id-MLDSA65-ECDSA-brainpoolP256r1-SHA512	MLDSA65	SHA512withECDSA	SHA512
id-MLDSA65-Ed25519-SHA512	MLDSA65	Ed25519	SHA512
id-MLDSA87-ECDSA-P384-SHA512	MLDSA87	SHA512withECDSA	SHA512
id-MLDSA87-ECDSA-brainpoolP384r1-SHA512	MLDSA87	SHA512withECDSA	SHA512
id-MLDSA87-Ed448-SHA512	MLDSA87	Ed448	SHA512

ENTRUST

Editorial Changes

- Added Scott Fluhrer as an author due to his valuable contributions and participation
- Removed references to Parallel PKI's in implementation consideration as it isn't adding value
- Resolved comments from Kris Kwiatkowski regarding FIPS
- Added a "Use in CMS" section
- Changed OID concatenation section to Domain Separators for clarity
- Expanded description of the key generation algorithm
- Added SMIME-CAPS into the sa-CompositeSignature definition in the ASN.1 module
- Added editorial changes by Jose Ignacio Escribano
- fixed nits

ENTRUST

Outstanding Issues

- A large End User wants us to add RSA-4096 bit combinations
  - Perhaps we just replace the 3072 ones with 4096, or just make new 2 new ones (PSS and PKCS1.5 flavors).
- Still need to address comments by Carl Wallace – discussed at hackathon
  - Replace
- Will address newer comments by Piotr Popis
- Discussion on making the Domain separator Hash (DER (OID)) instead of just DER(OID). This would make domain separator fixed length

Outstanding Issues

- Should we consider compacting the CompositeSignaturePrivateKey format (similar to what was done with the public key format)
- How do we carry an ML-DSA public key in the ASN.1 syntax?
- OpenPGP group draft-ehlen-openpgp-nist-bp-comp suggests using MLDSA 65 and 87 both with NIST P-384. Do we align?
  - This would change MLDSA65-ECDSA-P256-SHA512 to MLDSA65-ECDSA-P384-SHA512
  - They also don’t recommend a combination with MLDSA44 and brain pool, so maybe we can just remove MLDSA44-ECDSA-brainpool256r1-SHA256

ENTRUST

Working Implementations – Dynamic Hackathon Results

✅ = passing all verifiers
◒ = passing some verifiers
○ = not passing any verifiers

-	bc	carl-redhound	cht	corey-digicert	cryptonext	cryptonext-cnsprovider	entrust	kris	oqs-provider
ML-DSA-44-ipd	✅	✅	✅	✅	○	✅	✅	○	✅
ML-DSA-65-ipd	✅	✅	✅	✅	○	✅	✅	○	✅
ML-DSA-87-ipd	✅	✅	✅	✅	○	✅	✅	○	✅
MLDSA44-RSA2048-PSS-SHA256	○						✅		○
MLDSA65-Ed25519-SHA512	✅								✅
MLDSA87-ECDSA-P384-SHA512	✅					✅	✅		✅
MLDSA87-ECDSA-brainpoolP384r1-SHA512	✅						✅		✅
MLDSA87-Ed448-SHA512	✅								✅
MLDSA44-RSA2048-PKCS15-SHA256	✅					✅	✅		✅
MLDSA44-Ed25519-SHA512	✅								✅
MLDSA44-ECDSA-P256-SHA256	✅					✅	✅		✅
MLDSA44-ECDSA-brainpoolP256r1-SHA256	✅						✅		✅
MLDSA65-RSA3072-PSS-SHA512	○						○		✅
MLDSA65-RSA3072-PKCS15-SHA512	✅					✅	✅		✅
MLDSA65-ECDSA-P256-SHA512	✅					✅	✅		✅
MLDSA65-ECDSA-brainpoolP256r1-SHA512	✅						✅		✅

ENTRUST

Thank You

entrust.com