Vault Key Algorithm: OQS

This document describes the post-quantum key encapsulation schema implemented in VESvault (https://vesvault.com) through libOQS (Open Quantum Safe), and available through VESvault public APIs (https://ves.host). It follows the ECDH Vault Key Algorithm, with a key encapsulation mechanism (KEM) in place of the ephemeral Diffie-Hellman agreement.

OID: 1.3.6.1.4.1.53675.3.5

REST API and libVES access:

- *algo: string "OQS"*
- *publicKey: PEM encoded public key (SPKI)*
  - ML-KEM (FIPS 203), the standard encoding: AlgorithmIdentifier *id-alg-ml-kem-512*, *-768* or *-1024* (2.16.840.1.101.3.4.4.1, .2, .3), parameters absent; the public key is the encapsulation key *ek*, 800, 1184 or 1568 bytes long
  - Any other libOQS KEM: AlgorithmIdentifier 1.3.6.1.4.1.53675.3.5, with parameters an OCTET STRING holding the libOQS algorithm identifier (such as "sntrup761"); the public key as libOQS produces it
- *privateKey: PEM encoded encrypted private key (PKCS #8), under the same privateKeyAlgorithm as the public key. Recommended default symmetric algorithm for PKCS #8: AES-256-CBC*
  - ML-KEM: the privateKey is an ML-KEM-PrivateKey (IETF LAMPS, ML-KEM in X.509). libVES writes the *expandedKey* form, the decapsulation key *dk* (1632, 2400 or 3168 bytes long), and reads the *seed*, *expandedKey* and *both* forms. The encapsulation key is recovered from *dk* (FIPS 203 §7.3)
  - Any other libOQS KEM: the privateKey is the DER encoding of:

```
SEQUENCE {
    version     INTEGER (0),
    privateKey  OCTET STRING,
    publicKey   [1] EXPLICIT BIT STRING
}
```

    where *privateKey* and *publicKey* are as libOQS produces them
- Recommended default KEM: *ML-KEM-768*

Vault Entries for OQS Vault Keys deploy the following schema:

Encrypting a Vault Entry data:

- Run the KEM encapsulation with the public key pub(*V*) of the Vault Key *V*, resulting in the KEM ciphertext *E* and the shared secret *S*. *E* has the fixed length of *V*'s parameter set: 768, 1088 or 1568 bytes for ML-KEM-512, -768 or -1024

Jim Zubov <jz@vesvault.com>, 2026

- Calculate ({*K*} || {*IV*} || {*XXXX*}) = SHA384(*S*), where *K* is 32 bytes long, *IV* is 12 bytes long, *XXXX* are the unused last 4 bytes
- Produce a padded plaintext *PP*:

  `PP = {PL (1 byte)} || {P} || {ignored padding (PL bytes)}`

  where *PL* is the padding length byte (0 .. 255); the recommended value aligns *PP* to the next 32-byte boundary
- Encrypt the padded plaintext *PP* with AES-256-GCM, using the key *K* and the *IV*, resulting in the ciphertext *C* and the 16-byte GMAC value *G*
- Generate the Vault Entry structure:

  `{E} || {C} || {16-byte G}`

where "||" denotes concatenation. The result is to be passed as Base64 encoded *encData* of the Vault Entry.

Decrypting a Vault Entry data:

- Base64 decode the *encData*. Its first bytes, of the KEM ciphertext length for *V*'s parameter set, are *E*. (Throw an error if the data is not longer than that.)
- Immediately follows the ciphertext *C*, except for the last 16 bytes, which constitute the GMAC *G*
- Run the KEM decapsulation with *E* and the unlocked private Vault Key priv(*V*), resulting in the shared secret *S*. (For ML-KEM, an altered *E* yields an unrelated *S* by implicit rejection, and the GMAC check below fails.)
- Calculate ({*K*} || {*IV*} || {*XXXX*}) = SHA384(*S*), where *K* is 32 bytes long, *IV* is 12 bytes long, *XXXX* are the unused last 4 bytes
- Decrypt *C* using AES-256-GCM with the key *K* and the *IV*, resulting in the padded plaintext *PP*
- Validate the GMAC value *G*, return an error if not valid
- Restore the plaintext *P* by stripping the padding from *PP*:

  `{PL (1 byte)} || {P} || {ignored padding (PL bytes)}`

- Return *P*

An OQS Vault Key is not a hybrid: the confidentiality of its Vault Entries rests on the KEM alone. libVES implements this schema in libVES.c (ML-KEM since 1.33, the default for new Vault Keys since 1.35) and in libVES.js.

Jim Zubov <jz@vesvault.com>, 2026