Vault Key Algorithm: ECDH

This document describes the elliptic curve integrated encryption schema (ECIES) implemented in VESvault (https://vesvault.com), and available through VESvault public APIs (https://ves.host).

REST API and libVES access:

- *algo*: string "ECDH"
- *publicKey*: PEM encoded EC public key (SPKI)
- *privateKey*: PEM encoded encrypted EC private key (PKCS #8). Recommended default symmetric algorithm for PKCS #8: AES-256-CBC
- Recommended default EC domain parameters: *secp521r1*

Vault Entries for ECDH Vault Keys deploy the following implementation of ECIES:

Encrypting a Vault Entry data:

- Generate an ephemeral key *E* with the same parameters as the Vault Key *V*
- Calculate the ECDH secret agreement *S* using pub(*V*) and priv(*E*)
- Calculate ({*K*} || {*IV*} || {*XXXX*}) = SHA384(*S*), where *K* is 32 byte long, *IV* is 12 byte long, *XXXX* are 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), recommended value is to align *PP* to the next 32-byte boundary
- Encrypt the padded plaintext *PP* with AES-256-GCM, using the key *K* and the *IV*, result in ciphertext *C* and 16-byte GMAC value *G*
- Generate the Vault Entry structure:

  ```
  {DER SPKI pub(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*, identify the length of pub(*E*) from DER framing, extract the public key pub(*E*). (Throw an error if DER framing is not consistent or if pub(*E*) is not a valid public EC key in the same domain as the Vault Key *V*)
- Immediately follows the ciphertext *C*, except for the last 16 bytes which constitute GMAC *G*

Jim Zubov <jz@vesvault.com>, 2018

- Calculate the ECDH secret agreement *S* using pub(*E*) and the unlocked private Vault Key priv(*V*)
- Calculate ({*K*} || {*IV*} || {*XXXX*}) = SHA384(*S*), where *K* is 32 byte long, *IV* is 12 byte long, *XXXX* are unused last 4 bytes
- Decrypt *C* using AES-256-GCM with the key *K*, and *IV*, result in the padded plaintext *PP*
- Validate the GMAC value *G*, return an error if not valid
- Restore the plaintext P by stripping padding from *PP*:

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

- Return *P*

Jim Zubov <jz@vesvault.com>, 2018