Encrypted keys for the eCryptfs filesystem

ECryptfs is a stacked filesystem which transparently encrypts and decrypts each file using a randomly generated File Encryption Key (FEK).

Each FEK is in turn encrypted with a File Encryption Key Encryption Key (FEKEK) either in kernel space or in user space with a daemon called 'ecryptfsd'. In the former case the operation is performed directly by the kernel CryptoAPI using a key, the FEKEK, derived from a user prompted passphrase; in the latter the FEK is encrypted by 'ecryptfsd' with the help of external libraries in order to support other mechanisms like public key cryptography, PKCS#11 and TPM based operations.

The data structure defined by eCryptfs to contain information required for the FEK decryption is called authentication token and, currently, can be stored in a kernel key of the 'user' type, inserted in the user's session specific keyring by the userspace utility 'mount.ecryptfs' shipped with the package 'ecryptfs-utils'.

The 'encrypted' key type has been extended with the introduction of the new format 'ecryptfs' in order to be used in conjunction with the eCryptfs filesystem. Encrypted keys of the newly introduced format store an authentication token in its payload with a FEKEK randomly generated by the kernel and protected by the parent master key.

In order to avoid known-plaintext attacks, the datablob obtained through commands 'keyctl print' or 'keyctl pipe' does not contain the overall authentication token, which content is well known, but only the FEKEK in encrypted form.

The eCryptfs filesystem may really benefit from using encrypted keys in that the required key can be securely generated by an Administrator and provided at boot time after the unsealing of a 'trusted' key in order to perform the mount in a controlled environment. Another advantage is that the key is not exposed to threats of malicious software, because it is available in clear form only at kernel level.

Usage:

```
keyctl add encrypted name "new ecryptfs key-type:master-key-name keylen" ring
keyctl add encrypted name "load hex_blob" ring
keyctl update keyid "update key-type:master-key-name"
```

Where:

```
name:= '<16 hexadecimal characters>'
key-type:= 'trusted' | 'user'
keylen:= 64
```

Example of encrypted key usage with the eCryptfs filesystem:

Create an encrypted key "1000100010001000" of length 64 bytes with format 'ecryptfs' and save it using a previously loaded user key "test":

```
$ keyctl add encrypted 1000100010001000 "new ecryptfs user:test 64" @u
19184530

$ keyctl print 19184530
ecryptfs user:test 64 490045d4bfe48c99f0d465fbbbb79e7500da954178e2de0697
dd85091f5450a0511219e9f7cd70dcd498038181466f78ac8d4c19504fcc72402bfc41c2
f253a41b7507ccaa4b2b03fff19a69d1cc0b16e71746473f023a95488b6edfd86f7fdd40
9d292e4bacded1258880122dd553a661

$ keyctl pipe 19184530 > ecryptfs.blob
```

Mount an eCryptfs filesystem using the created encrypted key "1000100010001000" into the '/secret' directory:

```
$ mount -i -t ecryptfs -oecryptfs_sig=1000100010001000,\
  ecryptfs_cipher=aes,ecryptfs_key_bytes=32 /secret /secret
```