Mapping SKS into a TEE/SE "Combo"

An SKS (Secure Key Store) may be self-contained like in a smart card, but it may also be architected as a TEE (Trusted Execution Environment) and SE (Security Element) combination.

The primary objectives for dividing an SKS into a TEE/SE combo include:

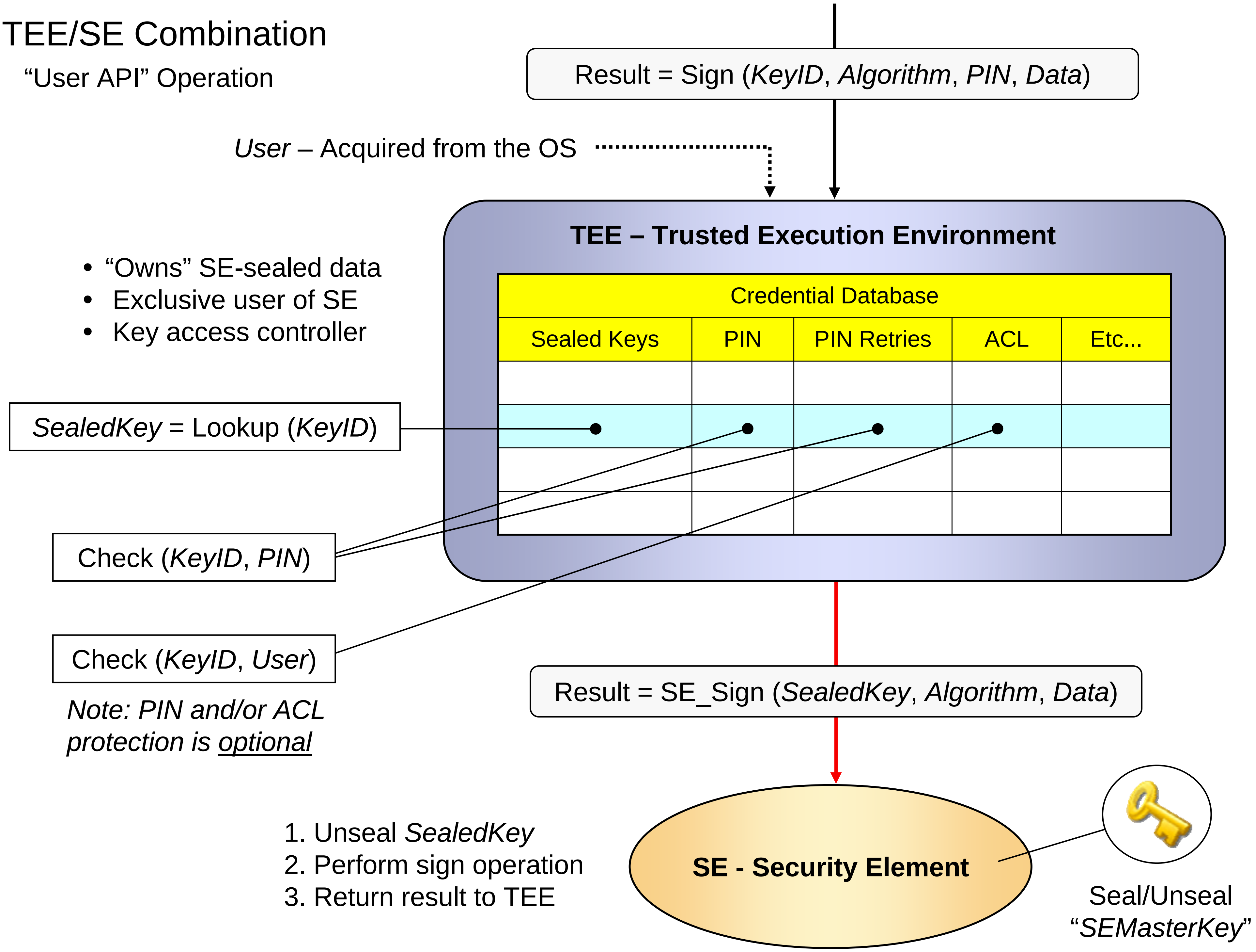
- Small SE footprint suitable for CPU integration
- Stateless SE-operation enabling simple virtualization
- Unlimited key storage
- Elimination of NVRAM
- Logical integration in modern operating systems

The described scheme is intended to work equally well in mobile phones as in high-performance servers.

The reader is supposed to be familiar with the SKS specification

TEE/SE Combination

"User API" Operation



Anatomy of a *SealedKey*

```
class SealedKey
  {
    byte[] wrappedKey;         // Encrypted PKCS #8 or symmetric key
    boolean isSymmetric;       // True if wrapped_key is symmetric
    boolean isExportable;      // True if allowed to be unsealed to the TEE
    byte[] mac;                // Integrity control
  }
```

Sealing Algorithm:

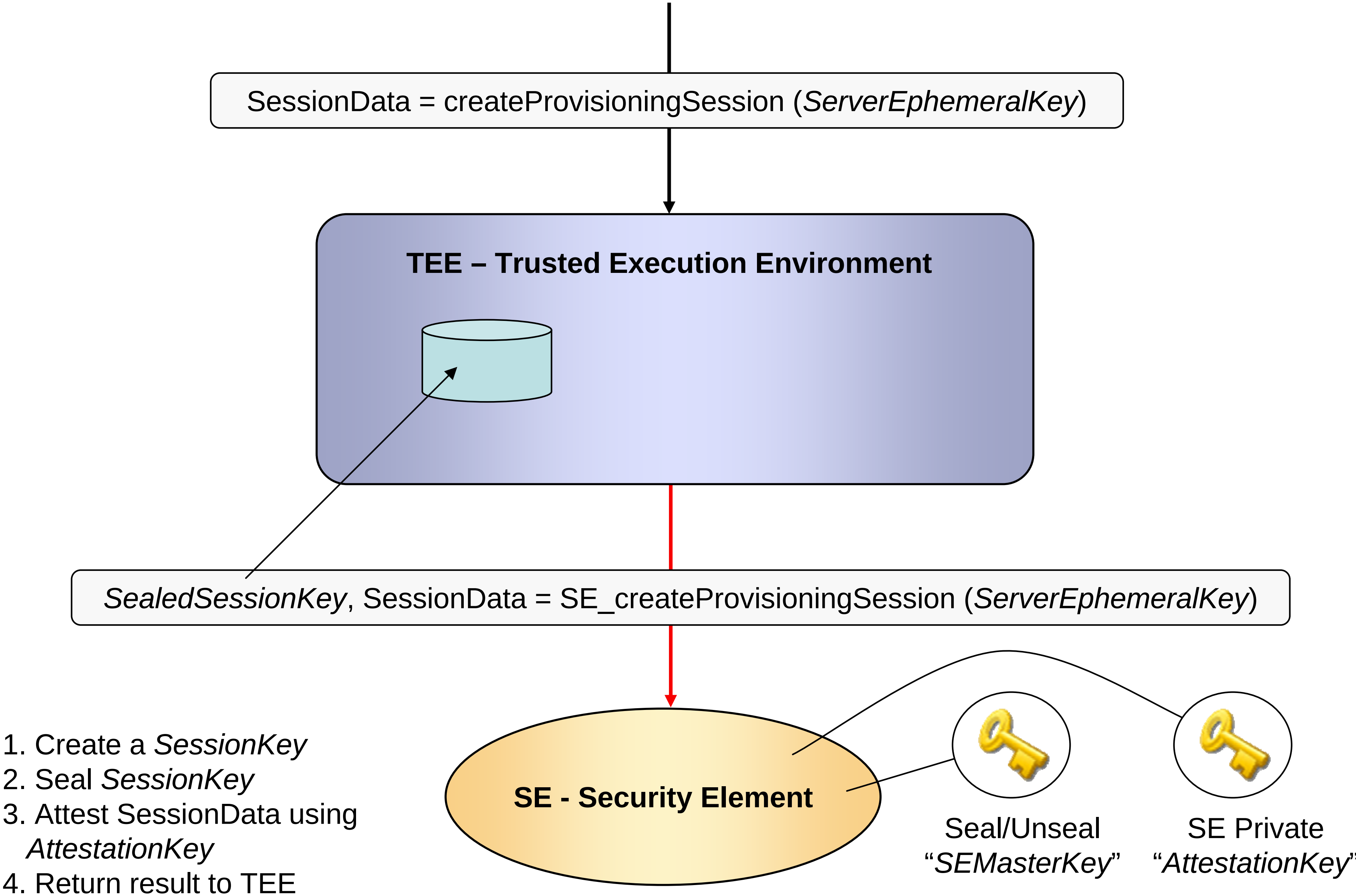
byte[] IV = randomNumber (16);
wrappedKey = IV || AES256-CBC (KDF$_{encryption}$ (*SEMasterKey*),
rawKeyValue, IV);

mac = HMAC-SHA256 (KDF$_{mac}$ (*SEMasterKey*),
isExportable || isSymmetric || wrappedKey);

TEE/SE Combination

Simplified “Provisioning API” Operation

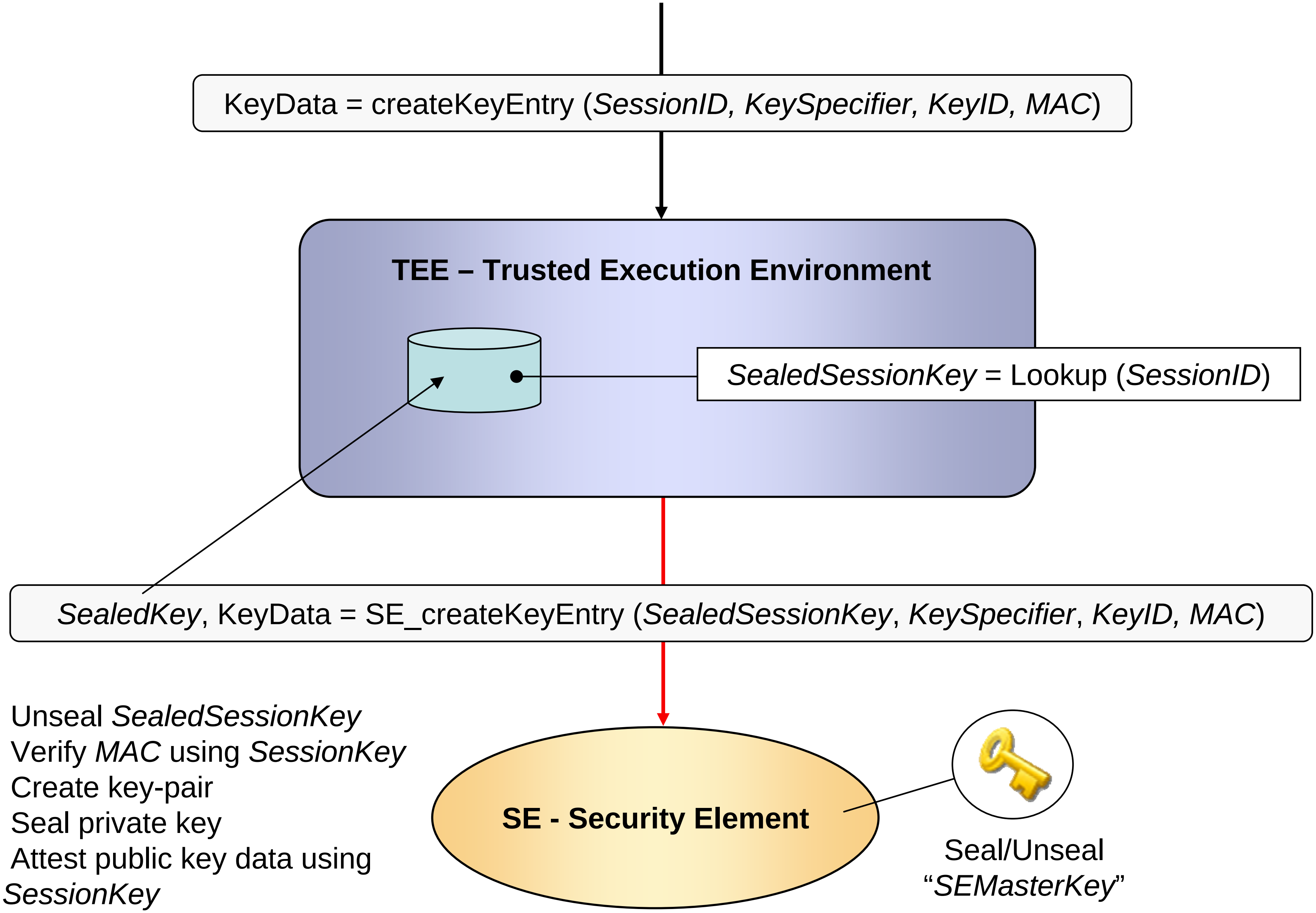
1. Create Provisioning Session



TEE/SE Combination

Simplified "Provisioning API" Operation

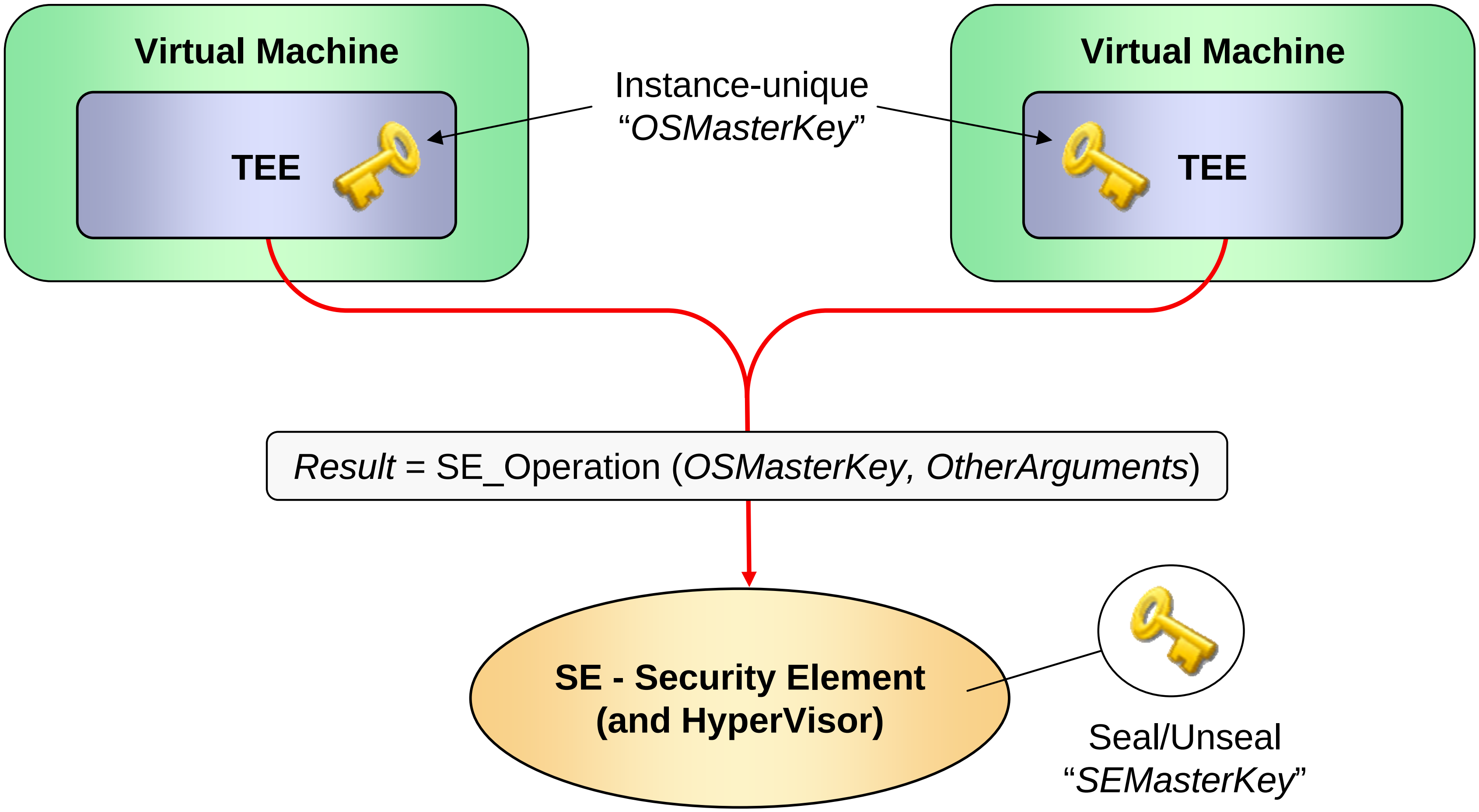
2. Create Object in Session



1. Unseal *SealedSessionKey*
2. Verify *MAC* using *SessionKey*
3. Create key-pair
4. Seal private key
5. Attest public key data using *SessionKey*
6. Return result to TEE

TEE/SE Combination

Virtualization Support – Binding keys and provisioning sessions to Virtual Machines



Actual Seal or Integrity Key: $\mathrm{KDF}_{\mathrm{operation}}$ (*SEMasterKey)* **XOR** *OSMasterKey*

Q & A

Question: Is this really secure?

Rhetoric answer: Do TEE- or application-based embedded PINs and/or obfuscated code actually bring any sustainable and provable security values to the table?

Question: Could there even be advantages of using the TEE for access control?

Answer: Yes, it enables combining various kinds of access controls like restricting keys to specific applications or users, as well as using device-wide PINs. A TEE can also provide challenge-response authentication and encrypted tunnels without burdening the SE. A TEE typically also supports a “trusted GUI” removing PIN-entry from potentially untrusted applications

Question: How does the SE protect keys from theft?

Answer: The “seal” contains an attribute which tells if the key is non-exportable. Such keys will not be exported unsealed to the TEE even it asks for it!