

**Encrypted Storage:**

Applications:

1. Encrypted content to prevent illegal distribution.
2. Use of same secret key across restarts by Terra (or any other app running on TPM)

Method: Use TPM.

\[ S1: \text{Encrypt (Seal) key } K \text{ for me and Decrypt (unseal) it only for me.} \]

\[ S2: SS = E(PH, h || K) \quad (E \text{ represents both encryption and MAC}) \]

\[ S3: \text{Store SS on disk.} \]

This is called **seal operation**.
**Unseal Operation:**

1. **S1:** Read SS from disk.
2. **S2:** Decrypt (Unseal) SS for me.
3. **S3:** Decrypt SS using $S_H$. Let output be $h || K$. See that $h = h_1$. If yes, return $K$, return NULL otherwise.

Same method can be used by Terra to persist its secret key $S_T$.

Can be generalized to multiple applications over Terra. Messages exchanged between Applications and Terra and same as messages exchanged between System and TPM as shown above. Only thing is Terra has to use same $S_T$ across reboots and so, it needs to persist its $S_T$.

**How to handle upgrades?**

S1 interface in seal operation above can be generalized to say “Seal this Key $K$ and decrypt it only for another system whose hash is $h_1$” and then during unseal operation TPM can check the stored hash against $h_1$.

**How to prevent replay attacks using TPM?**

If TPM has a secure counter, it can use value of that to check the “recent-ness” of a key $K$ before returning it. This would make attacks maximum DoS, but will prevent integrity of data. But DoS attacks can be done by myriad of other means also.
Flicker:

1. Reduces TCB to only the code you want to run.
2. No O.S. involved.
3. Runs directly over TPM in a secure environment i.e., no interrupts, no DMA, and no debugging.
4. Supports remote attestation.
5. Typical application:
   a. User has a program that he wants to run on a cloud.
   b. He gives the cloud provider his program binary, and inputs.
   c. The cloud provider has to run user program on these inputs and return him the output.
   d. Along with it, user should be able to verify that:
      i. It was his program which was actually run.
      ii. The output returned was actually returned by his program over his inputs.
      iii. After his program finished, no change was made to the output.
6. Uses SKINIT instruction which disables all interrupts, DMA access, and debugging before executing user code called PAL.
7. TPM has a special register which can be initialized to hash of PAL only by calling SKINIT. Software can read this register and extend this register. A reboot initializes this register to -1. It’s by verifying contents of these register that the user can verify above 3 conditions.