Provisioning for TPM Recovery

A TPM’s storage hierarchy can be used to store encrypted copies of sensitive data, such as EVM symmetric keys (used to HMAC filesystem metadata), and LUKS symmetric keys (used to encrypt partitions). The sealed data can be tied to policies that prevent unsealing unless certain conditions are met, such as correct boot integrity measurements. In this way, sensitive decryption keys can be provided automatically at boot time, while still protecting the keys from attack.

Currently, the Trusted Computing Group’s TPM Provisioning Guidance Document recommends creating a common Storage Root Key (SRK) persistent at location 0x81000001, so that all storage applications can use this single common primary key. Unfortunately this key cannot be recovered in the event of a TPM or motherboard failure. Any applications directly using the SRK would not be able to unseal their secrets.

Currently systems place the burden of recovering such TPM protected secrets on the applications, such as EVM and LUKS, so every application has to implement their own form of recovery. A better approach for those use cases requiring recovery is to provision the TPM at install time with a Default Recoverable Storage Key (DRSK) under the default primary SRK, and then having all applications use this DRSK for sealing their secrets. Then if the TPM is replaced, recovering this DRSK transparently recovers all applications using it. One simple and secure way of provisioning and recovering a TPM’s DRSK is to duplicate it under a storage key of another TPM. In that way, the private key is never visible unencrypted outside of a TPM hardware boundary.

LUKS Key

Systemd-cryptenroll seals the LUKS slot key under a non-persistent primary storage key. Optionally, the latest upstream version can use the default SRK at the hardcoded location 0x81000001. (The ability to use a persistent handle is necessary if the owner has set an owner authorization, as creating the non-persistent primary key requires the owner authorization password, which is not available at boot time.) Neither the non-persistent primary storage key or persistent SRK can be recovered, so the LUKS slot key itself must be backed up manually. A patch is being submitted to add an option to specify “--tpm_srk=” so that systemd-cryptenroll can be directed to use a persistent DRSK, so that recovering the DRSK transparently recovers the LUKS slot key.

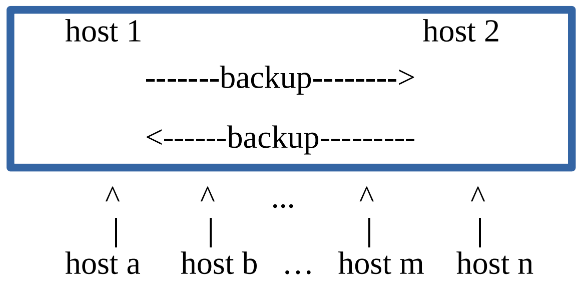
EVM Trusted Key

By default, the kernel seals new trusted keys under the SRK, which is assumed to be at persistent handle 0x81000001. This keyhandle can be specified in keyctl, so it can easily be redirected to a DRSK, and thus transparently recovered.

This package of scripts provides a simple but flexible set of tools for provisioning, backup up, and restoring a TPM’s DRSK. Each host with a TPM and these tools can act as a recovery client, a recovery server, or both. Each client can use one or more servers, and each server can support one or more clients. In the simplest use case, an owner has two systems with TPMs, and each is used to backup the other:

```
host 1                         host 2
       -------backup-------->
       <------backup---------
```

In larger systems, two hosts can back each other up, and then they can act as a centralized backup service for many client hosts:



Currently the TPM provisioning Guidance Document recommends placing the SRK at 0x81000001. Some systems come already provisioned with an SRK at 0x81000001 and an EK at 0x81000002, so this package places its SRK at 0x81000003 and DRSK at 0x81000004, so that we do not risk accidentally damaging the existing OS. For a simple two host system, with one client and one server, the data flows are roughly:

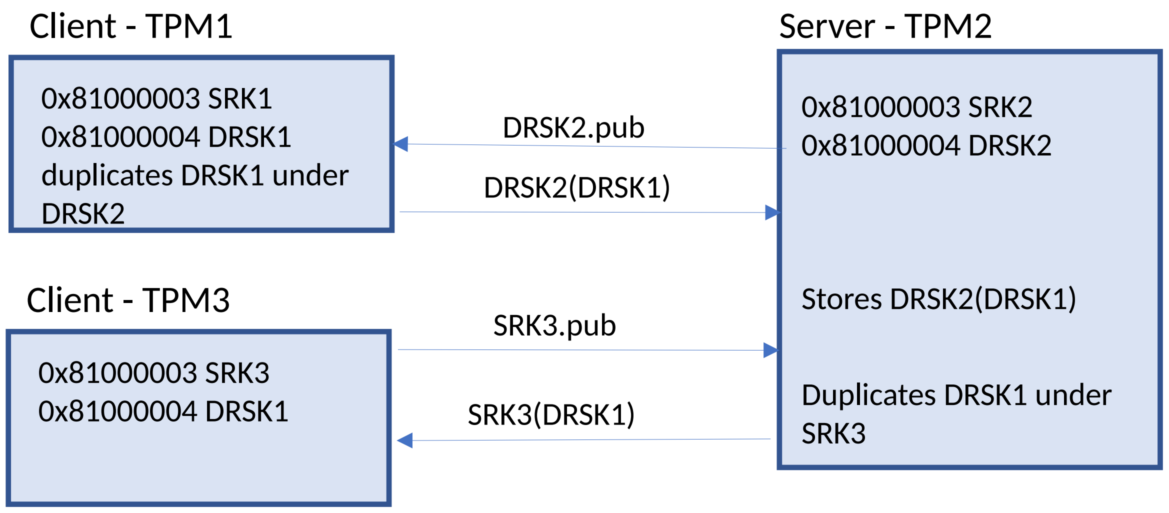


Figure 1: EVM and LUKS key blobs are encrypted under DRSK1, which is transparently recovered

Here the client initially has TPM1, backs it up to the server, and then its TPM is replaced with TPM3, (such as with a motherboard replacement) and the original DRSK1 is recovered. In this configuration, the server host could simultaneously act like a client to the first host, and back up its DRSK2 to the first host.

This package consists of six shell scripts that in turn use ssh to perform the backup and recovery between hosts. Since it uses ssh, this package is well suited for small, local use cases, but enterprise level environments will likely want to implement a more scalable web based service instead. For those use cases, this package can still server as a simple guide to the underlying TPM design and functions.

In this package, each client keeps track of one or more servers used for backup, in directories named by the server DNS name or IP address. Each server keeps track of one or more clients by the client ID, which is the hex encoded sha256 hash of the client’s DRSK public key.

In the event of a TPM failure, the client is no longer able to load its DRSK private key in the TPM, but it still has a copy of the public part of the DRSK, and it know the name or address of the backup server, so it can have the server duplicate the original DRSK under its new SRK, and thus recover the full DRSK.

Directory Structure

```
        tpm_keys
                bin
                        provision.sh, backup.sh restore.sh, server_backup.sh server_restore.sh demo.sh
                my_keys
                        my_drsk.pub, my_drsk.prv, my_drsk.ctx, primary.ctx
                servers
                        [directory named by server name or address]
                clients
                        [directory named for client ID (sha256 of their drsk.pub)]
                                dup.pub, dup.prv, dup.seed (duplicates of my_drsk, sealed by server's
                                drsk)
```

Installation and use:

First the tpm_keys directory is installed in a common location on each host. The recommended default is /boot/tpm_keys, so that boot time EVM and LUKS keys can transparently be recovered even if the root partition is encrypted. The tpm_keys directory and files must be owned by a user in the tss group (so that it can run the needed tpm2 commands), and who has public key ssh enabled on the target hosts. (The user on the client needs ssh access to the server. The server does not need ssh access to the client.)

Once the files are installed, provision.sh is run once on each host to provision the SRK and DRSK. Then "backup.sh <server name or IP address>" is run to duplicate the local DRSK and back it up on the server, under the server's DRSK.

If recovery is needed, simply run "recover.sh <server name or IP address>", which will reprovision and recover the original DRSK on the local host.

"demo.sh <server name or IP address>" will create a trusted key on the already provisioned local host, then will delete the existing SRK and DRSK, recover the system, and show that the trusted key still loads on the recovered DRSK.

A Note on Owner Authorization:
It is highly recommended that the owner set an owner authorization (password) for the TPM. In most cases, systems come with no owner authorization, so that the owner can create and persist keys as needed. Once the owner has provisioned a TPM, the owner authorization should be set. Otherwise it is much easier for attackers to evict existing keys and lock out the real owner. The scripts in this set check for the existence of the owner authorization, and ask for the existing password if it is already set, or ask for a new one (and set the new password) if it does not already exist. The owner authorizations can be different on the client and server. The client does not need to know the owner authorization on the server, and vice versa. The owner authorization is only needed on a host for the original provisioning, and restore scripts.