SecretKeeper — Writeup (with commands)

Goal: Recover the plaintext flag stored by the SecretKeeper app.

1. **Run the app**
   Launch SecretKeeper and give it any input so it writes its persistent artifacts.

2. **Pull the artifacts from the device**
   Retrieve the encrypted payload and the partial key from the app's external data area:

```
adb pull /sdcard/Android/data/com.magical.secretkeeper/files/secretkeeper/flag.enc

adb pull /sdcard/Android/data/com.magical.secretkeeper/files/secretkeeper/key_part1.txt
```

3. **Get the device serial**
   Read the serial number the app uses:

```
adb shell getprop ro.serialno
```

4. **Derive the missing key part**
   Compute the SHA-256 digest of the device serial and take the first 16 hex characters (this yields the required 8-byte / 16-hex nibble fragment to complete the AES-128 key).
   (Example of a common shell pipeline — adapt to your environment if needed):

```
# example (platform-agnostic idea): compute sha256(serial) and extract first 16 hex chars

serial=$(adb shell getprop ro.serialno | tr -d '\r\n')

first16=$(echo -n "$serial" | sha256sum | awk '{print substr($1,1,16)}')
```

5. **Assemble the full key**
   Concatenate the contents of key_part1.txt with the first16 hex string to form the full AES-128 key (32 hex chars). Ensure no extra whitespace or newline characters are included when concatenating.

6. **Decrypt the encrypted flag**
   Use AES-128-ECB with the assembled key (hex) to decrypt:

```
openssl enc -d -aes-128-ecb -in flag.enc -out flag.txt -K fullkey
```

(Replace fullkey with the 32-hex-character key assembled in step 5.)

7. **Verify**
   Open flag.txt and confirm the flag is present in readable form.