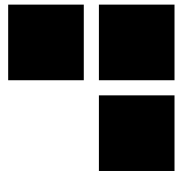
SYNACKTIV
DIGITAL SECURITY

# Déchiffrement du keystore logiciel d'Android

Présenté 02/04/2019

Par Julien LEGRAS, Thomas ETRILLARD

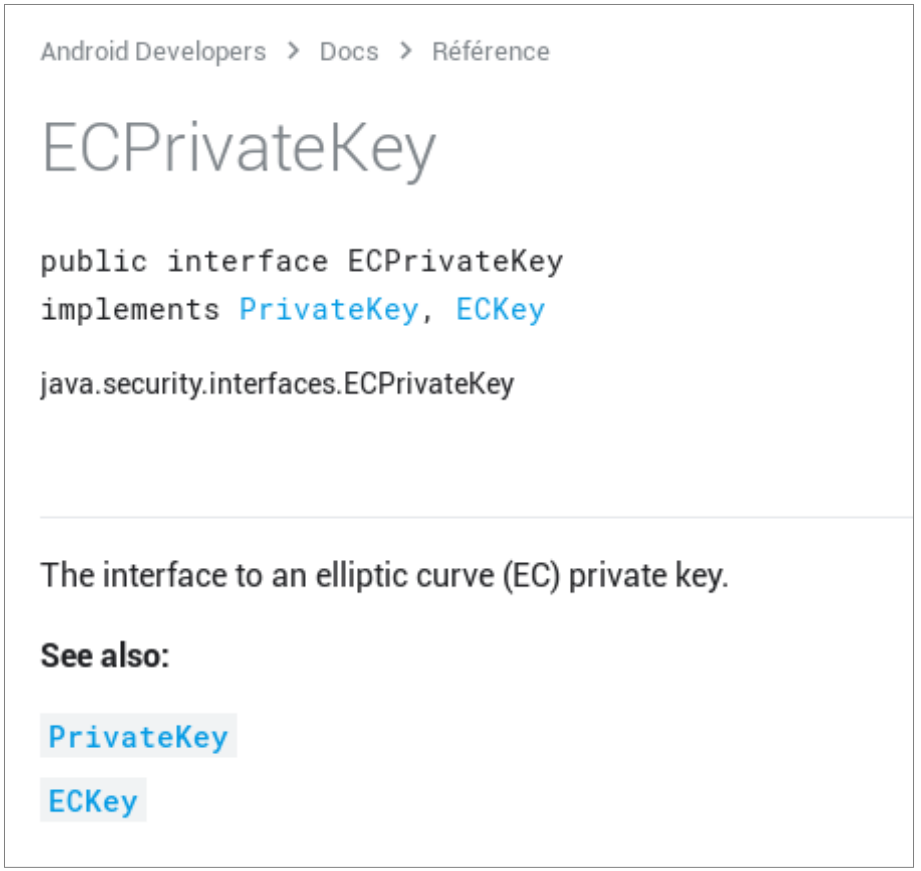
# Contexte



- **Pré-évaluation d'une application Android**
  - Utilisation de l'émulateur Android pour les tests
- **L'application utilise du chiffrement à l'aide de courbes elliptiques**
  - Le certificat, et les clefs sont générées par l'appareil
  - Authentification du client par le serveur Web
- **Objectif : utiliser *Burp* pour intercepter les communications**

# Extraction

- **Avec FRIDA**
  - Impossible dans le cas de clefs EC
  - Pourtant des méthodes existent
  - Mais ne sont pas applicables aux instances récupérées à partir du KeyStore
- **En offline**
  - Dump & déchiffrement du keystore logiciel (/data/misc/keystore/user_#)



Android Developers > Docs > Référence

ECPrivateKey

```
public interface ECPrivateKey
implements PrivateKey, ECKey
```

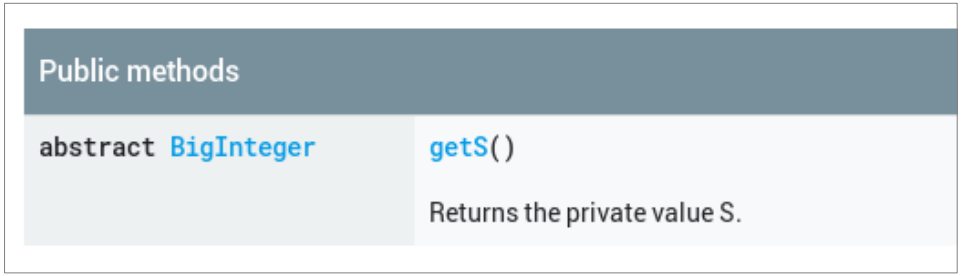
java.security.interfaces.ECPrivateKey

The interface to an elliptic curve (EC) private key.

**See also:**

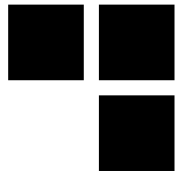
`PrivateKey`

`ECKey`



| Public methods | |
|---|---|
| `abstract BigInteger` | `getS()`<br>Returns the private value S. |

# Android KeyStore Decryptor



- **https://github.com/nelenkov/keystore-decryptor**
- **Logiciel en Java développé par Nikolay Elenkov**
  - Auteur de « Android Security Internals »
- **Le logiciel crash dans le cas de clefs EC**
  - *« The tool supports Android M keymaster v1.0 blobs, encrypted with the default (all zero) key. Blobs from later versions may not be supported. »*
  - N’a pas été modifié depuis ~2 ans

# (python) Android keystore decryptor

- **Temps contraint + blob-ception + 1000 lignes de java + historique du format**
  - Pourquoi pas recréer l'outil en python (~250 lignes) par curiosité et pour mieux comprendre pourquoi ça casse

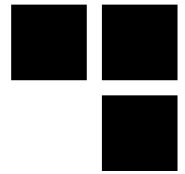
```
$ ./android-ks-decryptor.py  -h
usage: android-ks-decryptor.py [-h] [--master-key MASTER_KEY]
                               [--password PASSWORD] [--dump-pem]
                               keyfile

Parses Android keystores

positional arguments:
  keyfile               user_X/uid_USRPKEY_keyname or just user_X/ to parse
                        the whole directory

optional arguments:
  -h, --help            show this help message and exit
  --master-key MASTER_KEY
                        user_X/.masterkey
  --password PASSWORD   The password protecting the lockscreen
  --dump-pem            Dump the decoded keys and certificates in
                        <keyfile>.pem
```

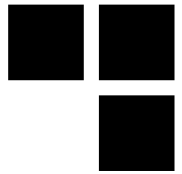
# (python) Android keystore decryptor



- **Ça fonctionne pour la mission**

```
$ ./android-ks-decryptor.py --password=1234 user_0_android_6_ec/
[+] Parsing 10056_USRCERT_myKey
-----BEGIN CERTIFICATE-----
MIIBDjCBtKADAgECAgIFOTAKBggqhkjOPQQDAjAQMQ4wDAYDVQQDEwVteUtleTAe
Fw0xODExMTUxNjU1MTZaFw0xOTExMTUxNjU1MTZaMBAxDjAMBgNVBAMTBW15S2V5
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEgIiT96FzbpxfBlotelrPUv9s6j1C
9L+pGeU4t77FGPYzY/NaGhnwxHNGunSxzFGrChNMx1hvGDEqt3DV686NHjAKBggq
hkjOPQQDAgNJADBGAiEAhtK2GSyjkihURF3nKFyjiyvaR2Yuhp8p0xcFtZfyBm8C
IQCtKZf1Yph1kGmeC7zfkISKT8pbDEBlJbsKm9mCMzs3Sw==
-----END CERTIFICATE-----
[+] Parsing 10056_USRPKEY_myKey
-----BEGIN EC PRIVATE KEY-----
MHcCAQEEIMKArpoZbl22SALGa8LFODfeeN/525j/MV+fAJC22rV5oAoGCCqGSM49
AwEHoUQDQgAEgIiT96FzbpxfBlotelrPUv9s6j1C9L+pGeU4t77FGPYzY/NaGhnw
xHNGunSxzFGrChNMx1hvGDEqt3DV686NHg==
-----END EC PRIVATE KEY-----
```

# To-do



- **Gestion du format « Super-Encrypted »**
- **PR (java) sur Android Keystore**

# AVEZ-VOUS DES QUESTIONS ?

MERCI DE VOTRE ATTENTION,

SYNACKTIV
DIGITAL SECURITY

# Bibliographie


- **https://nelenkov.blogspot.com/2015/06/keystore-redesign-in-android-m.html**
- **http://www.cs.ru.nl/~joeri/papers/spsm14.pdf**
- **http://androidxref.com/6.0.1_r10/xref/system/security/keystore/keystore.cpp#587**
- **https://github.com/googlesamples/android-BasicAndroidKeyStore/**
- **https://developer.android.com/training/articles/keystore**