One Key to Rule Them All: Recovering the Master Key from RAM to break Android's File-Based Encryption

By:
Tobias Groß, Marcel Busch and Tilo Müller

From the proceedings of
The Digital Forensic Research Conference
DFRWS EU 2021
March 29 - April 1, 2021

One Key to Rule Them All: Recovering the Master Key from RAM to Break Android's File-Based Encryption

DFRWS EU 2021

Tobias Groß, Marcel Busch, Tilo Müller

March 29th - April 1st, 2021

IT Security Infrastructures Lab
Department of Computer Science
Friedrich-Alexander University Erlangen-Nuremberg (FAU)

Introduction

ANDROID DEVICES

FULL DISK ENCRYPTION (FDE)

FILE BASED ENCRYPTION (FBE)

Our Contributions

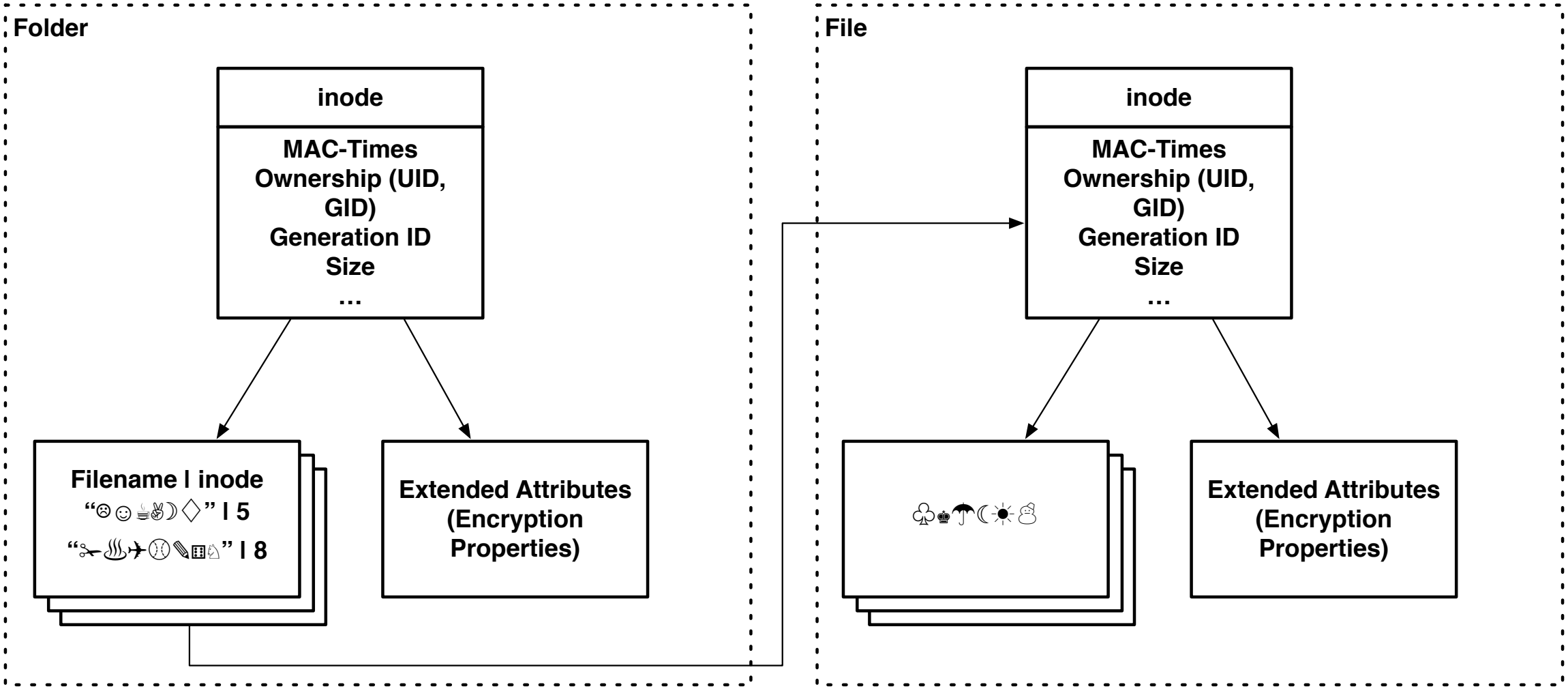
- We developed a method that recovers the ext4 FBE master keys from file keys present on a raw memory image of an Android device
- We extended *The Sleuth Kit to*
  - Output FBE attributes of metadata
  - Decrypt file names and content when FBE master key is provided
- We extended the *Plaso* framework to extract events from FBE encrypted partitions
- Evaluation of 13 Android smartphones, in respect of their used disk encryption schema
  - 7 out of them use a vulnerable file-based encryption key derivation function

Outline

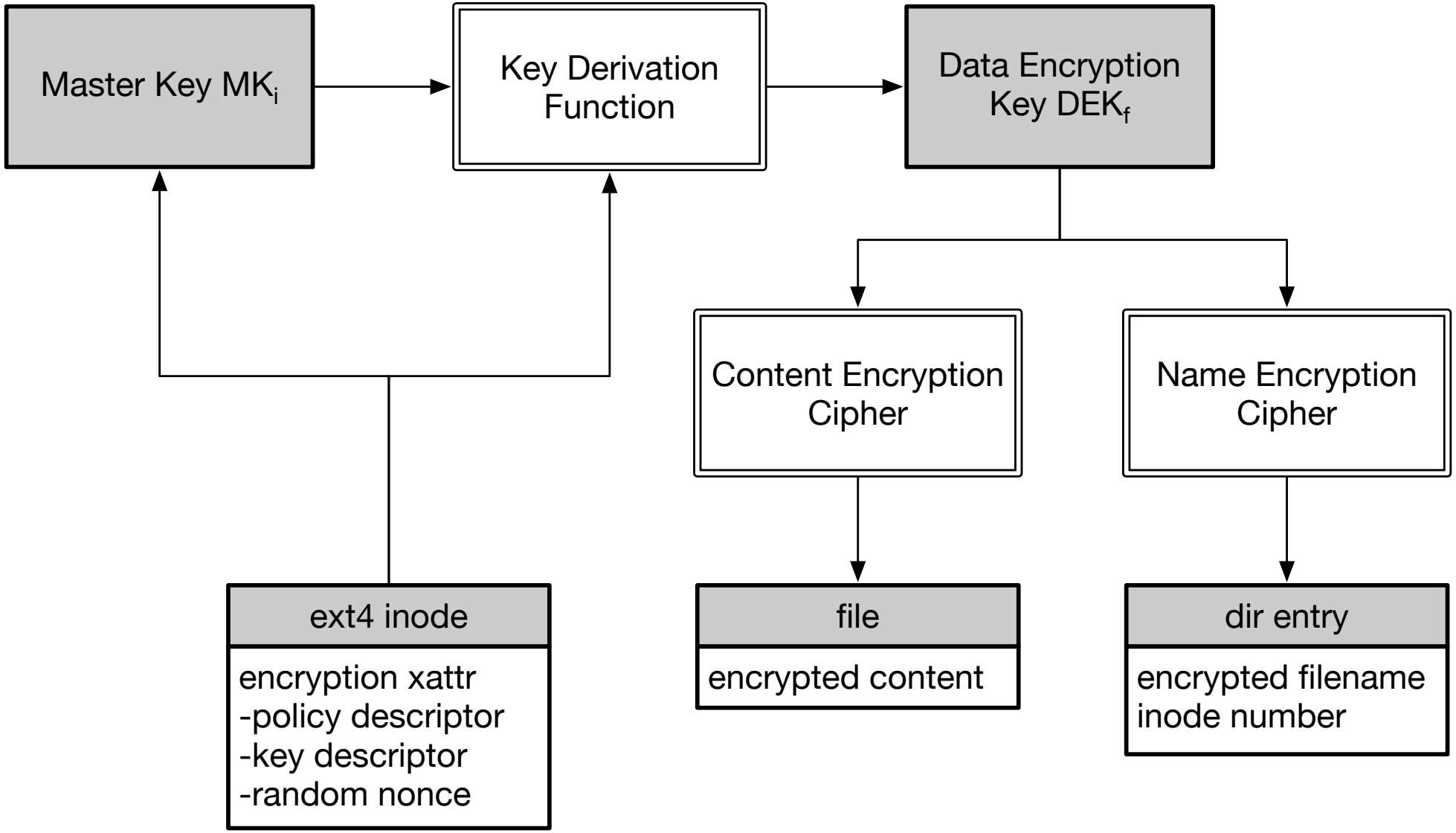
- Background
- File-Based Encryption Attack
- Implementation
- Evaluation
- Limitations

Background

Ext4 File Based Encryption



Android File Based Encryption - Overview



Used Ciphers

- Used function for key derivation: AES 128 ECB
- Encryption modes for content data and names:
  - AES 256 XTS
  - AES 256 GCM
  - AES 256 CBC
  - AES 256 CTS (used for name encryption)
  - AES 256 HEH (used for name encryption)
  - “private”

File-Based Encryption Attack

Key Derivation Version 1

```
1  static int derive_key_aes(u8 deriving_key[FS_AES_128_ECB_KEY_SIZE],
2                            const struct fscrypt_key *source_key,
3                            u8 derived_raw_key[FS_MAX_KEY_SIZE])
4  {
5      /* ... */
6      struct crypto_skcipher *tfm = crypto_alloc_skcipher("ecb(aes)", 0, 0);
7      /* ... */
8      res = crypto_skcipher_setkey(tfm, deriving_key,
9                                   FS_AES_128_ECB_KEY_SIZE);
10     /* ... */
11     sg_init_one(&src_sg, source_key->raw, source_key->size);
12     sg_init_one(&dst_sg, derived_raw_key, source_key->size);
13     skcipher_request_set_crypt(req, &src_sg, &dst_sg, source_key->size,
14                                NULL);
15     res = crypto_wait_req(crypto_skcipher_encrypt(req), &wait);
16     /* ... */
17     return res;
18 }
```

file nonce

master key

file specific data encryption key

file: fs/crypto/keyinfo.c
from Android kernel repository
commit: ASB-2018-12-05_4.14-p-release

Key Derivation Version 1 cont.

- Key derivation function:

$$DEK_f = AES^{ECB}_{nonce_f}(MK)$$

- Trivially calculate master key from publicly accessible nonce:

$$MK = AES^{ECB}_{nonce_f}(DEK_f)$$

- Data encryption keys are stored in kernel space memory → cold boot attack

- Problem: we can not easily link an extracted DEK_f to a specific file (nonce)

Solution: Calculate all possible Master Keys

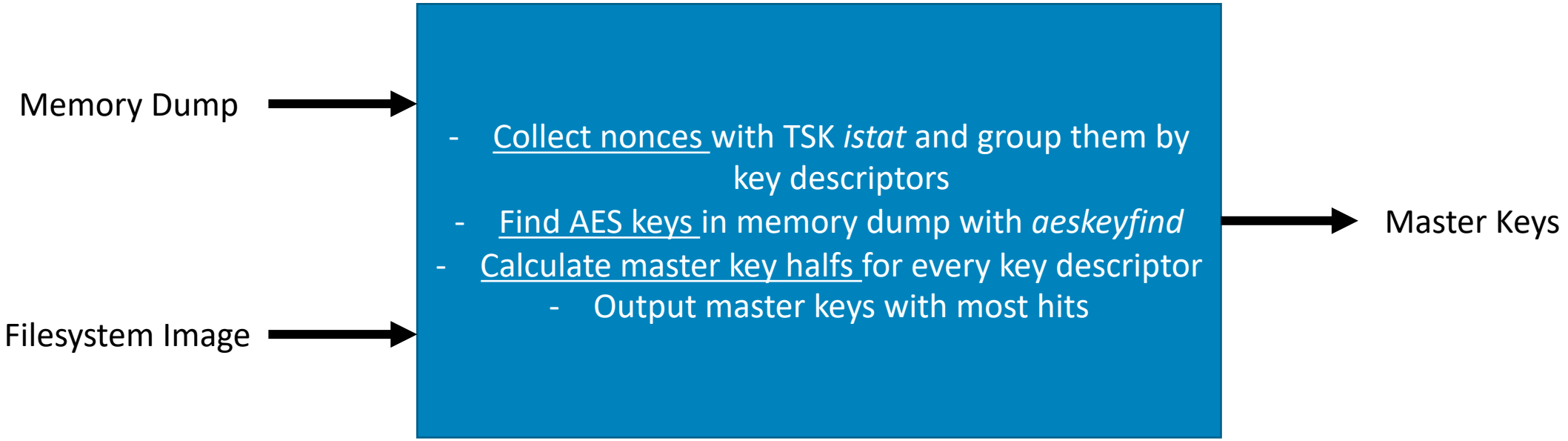
- Extract all used nonces from file system: $N = \{nonce_1, nonce_2, nonce_n\}$
- Extract all encryption keys from memory dump: $FK = \{DEK_1, DEK_2, DEK_n\}$
- Calculate the set of potential master keys $M = \{MK_1, MK_2, MK_n\}$ for all combinations of $n \in N$ and $\mathrm{fk} \in FK$
- **Master key candidates which are present more then once are the used master keys**
- On more recent Android kernel versions, a fixed key derivation function is used when AES 256 HEH is selected as name encryption mode

Implementation

Implementation of Tools

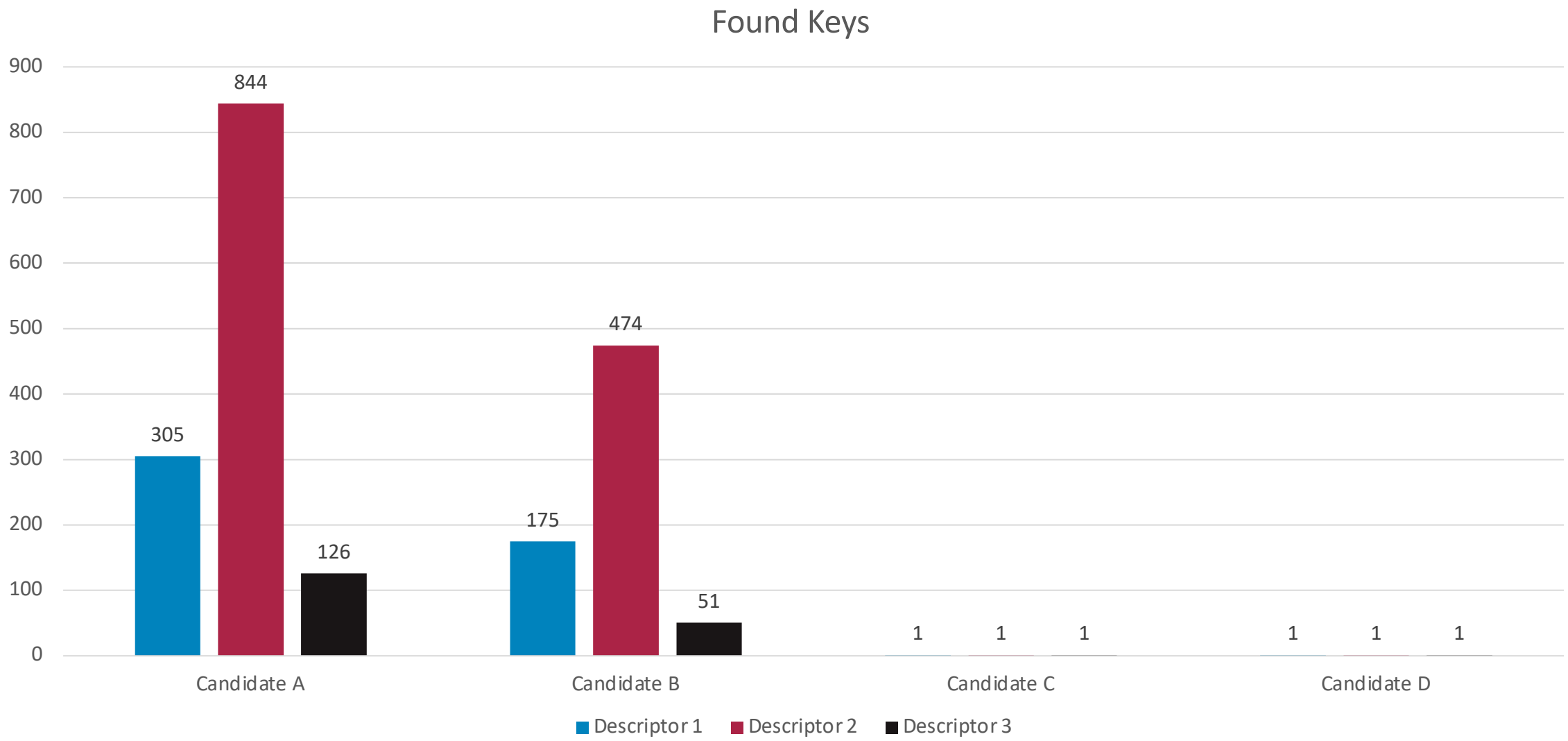
- *The Sleuth Kit* extension
  - *istat* outputs FBE related metadata (nonce, key descriptor) for given inode
  - *fls, fcat, icat and ifind* can decrypt file names and content when master keys are provided via an argument
- *Plaso* extension
  - Added possibility to provide master keys of image
  - Uses our *The Sleuth Kit* implementation via *pytsk3* to extract events from FBE encrypted images

Master Key Restorer



Evaluation

Restored Keys Nexus 5X



Full Evaluation of Implementation

Release	Device	OS Version	Content Enc. Mode	Name Enc. Mode
2015	Google Nexus 5X	8.1.0	AES XTS	AES CBC CTS
2016	Google Pixel XL	10.0.0.	private	AES CBC CTS
(2019)	Virtual Device	10	AES XTS	AES CBC CTS

Evaluation Based on Metadata

Release	Device	OS Version	Content Enc. Mode	Name Enc. Mode	old KDF	Metadata Enc.
2015	Samsung Galaxy S6	7.0	Full-Disk Encryption		-	-
2015	Google Nexus 6P	8.1.0	AES XTS	AES CBC CTS	✓	X
2016	Huawei P9 lite	7.0	Full-Disk Encryption		-	-
2017	Google Pixel 2	10	private	AES HEH	X	X
2017	BQ Aquaris X	8.1.0	Full-Disk Encryption		-	-
2018	Google Pixel 3	9	private	AES CBC CTS	✓	✓
2018	Xiaomi Mi 8	8.1.0	private	AES CBC CTS	✓	X
2018	Huawei P20 lite	8.0.0	AES XTS	AES CBC CTS	✓	X
2019	Google Pixel 4	10	private	AES CBC CTS	✓	✓
2019	Samsung Galaxy S10	10	❋	❋	❋	X
2020	Huawei P40 Pro	10.1.0	❋	❋	❋	✓

Limitations

- New key derivation function renders our approach ineffective
  - But this new function gets only used together with name encryption mode AES 256 HEH
  - Already shipped devices will not be updated, because this needs re-encryption of the user-data partition
- Metadata encryption hinders us from accessing the decrypted FBE encrypted partition
  - But every encryption layer should be implemented properly on its own to protect data best

Thank you!