Semi-Offline Attack

Android Full-Disk Encryption

Who / What / Why

- Oliver Kunz
  - @olknz

- MSc. at Royal Holloway

Agenda

- Topic Introduction
- Online Attack – The Baseline
- Offline Attack – The Old-fashioned Way
- Semi-Offline Attack
- Questions

Topic Introduction

Full-Disk Encryption on Mobiles

- **Situation:** Mobile is lost or stolen
- **Objective:** Prevent access to data stored on mobile
- **Security Control:** Encryption at rest (storage encryption)
- **Limitation 1:** Data is accessible in cleartext when device is running
- **Limitation 2:** Everyone in possession of the mobile can…
  - … launch an attack on the ciphertext
  - … launch an attack on the encryption cipher
  - … launch an attack on the crypto system

Elements of Full-Disk Encryption

We need…

- Disk encryption key [Master Key]
- Disk encryption cipher [AES-CBC]
- A process to en- and decrypt data [dm-crypt]

Wait a minute…

- Do we have to remember an AES key?

We also need…

- Key Encryption Key (KEK)
- Key Derivation Function (KDF) to derive the KEK from user value

Encryption Algorithms for FDE

Advanced Encryption Standard (AES)

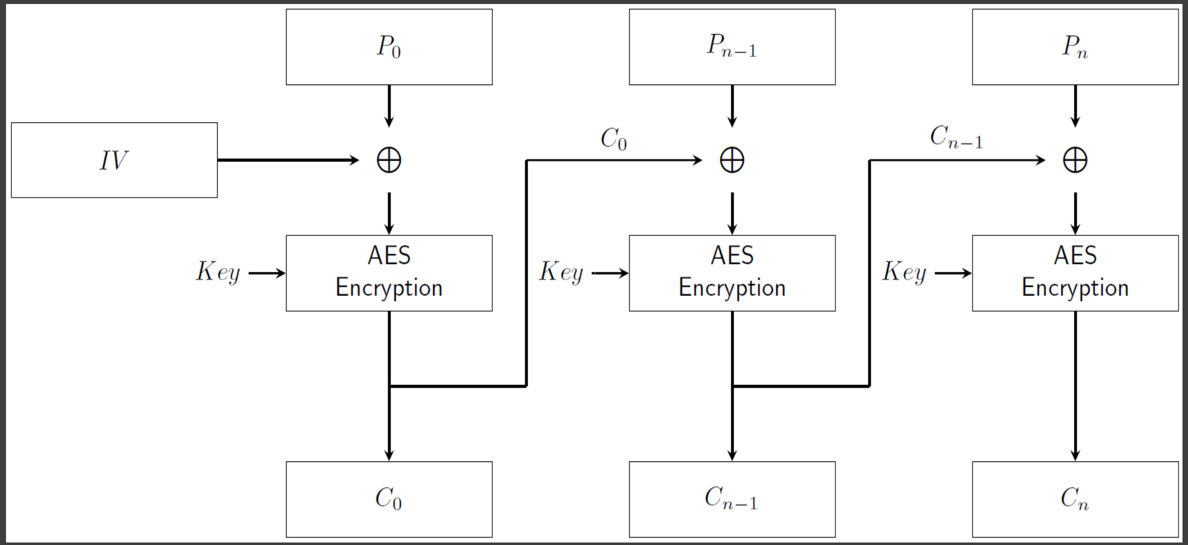
- Block Size: 128 bits
- Key Size: 128, 192, 256 bits

Block Cipher Mode

- Cipher Block Chaining (CBC)
- Alternative Mode XTS
  - XEX Encryption Mode with Tweak and Ciphertext Stealing (XTS)
  - Needs recompilation of Android source

AES-CBC

- Used for encryption and as building blocks
- Message dependency to prevent
  - Ciphertext manipulation (block swapping, insertion, or deletion)
  - Frequency analysis and dictionary attacks
- Initialisation Vector (IV) is required



Encrypted Salt-Sector Initialisation Vector

IV Properties

- Use only once with certain key
- Unpredictable value (same size as block size)
- Must be known to encrypting and decrypting party (can be public)

ESSIV

- Create a per-sector IV based on the encryption key and the sector number
- Allows to encrypt storage sector-by-sector
  - Else the whole storage needs re-encryption after an intermediate block changes

Master Key

- Created by `create_encrypted_random_key()` in cryptfs.c
- Read twice 16 Byte from `/dev/urandom` on first-time encryption
  - Master Key and Salt
  - Both values are static until smartphone is wiped!
- Protected by the KEK derived from user value
  - AES-CBC Encryption
- Stored in encrypted form in the crypto footer

Key Derivation Functions

Password-Based Key Derivation Function 2 (PBKDF2)

- Used before Android 4.4
- Only computationally expensive ➡ processing power is cheap

scrypt

- Used since Android 4.4
- Computationally expensive AND memory-hard
- Uses PBKDF2 as building block

Screen Unlock value (PIN / Password)

- Apply KDF ➡ Result = KEK

Android – What Is Encrypted?

Compatibility Definition Document (CDD)

- Encrypt the `/data` partition and non-removable internal SD card partitions
- *USERDATA* partition contains all user stored information

Crypto Footer – stores all crypto information for FDE

- Located on the *METADATA* partition
- Encrypted master key
- Salt
- KDF and cipher information

Online Attack

The Benchmark Baseline

Pre-Conditions – An Attacker Needs…

- Physical access to the device
  - For as long as the brute-force attack takes
- A brute-force script
  - In the absence of any Android Debug Bridge (ADB) security controls
- A physical attack machine
  - Justin Engler & Paul Vines Automated PIN Cracking at DC21

Online Attack Process

- Used as baseline for the research
- Enabled ADB and trusted the Attacking Host
- Attack Duration (10’000 PIN)
  - Approx. 22 hrs 40 min
  - 2000 timeout triggered

Method	Nexus 4	Nexus 6
input tap	5.1 sec	3.3 sec
input text	2.8 sec	1.8 sec

Countermeasures

- Two different prompts with two different countermeasures

	Nexus 4 (Android 4.4)		Nexus 6 (Android 5.0)	
	Start-Up	Screen-Unlock	Start-Up	Screen-Unlock
Timeout (Threshold/Duration)	10 / 30 sec	5 / 30 sec	10 / 30 sec	5 / 30 sec
Action (Threshold/Activity)	30 / Wipe	>100 / None	30 / Wipe	>100 / None

- Potential Improvements
  - Increasing penalty timeouts
  - Enforce device wipe after X^{th} failed attempt

Online Attack Conclusions

- Google has secured ADB sufficiently enough ✓
  - Disabled by default / Authentication on connect
- Complexity of physical attack devices ✓
  - Needs to support more than just PIN
  - Needs to recognize successful attempt
- Time Costs ✓
  - 10’000 PIN took almost one day!
- Missing Security Controls ✘
  - Too many failed attempts with no final penalty!

Offline Attack

The Old-fashioned Way

Pre-Conditions – An Attacker Needs…

- Physical access to device
  - Only for as long as the next two steps take
  - Mostly this will be due to device theft or loss
- A means to image a partition
  - USERDATA (encrypted data)
  - METADATA (crypto footer)

Partition Imaging

Best case: Unlocked Bootloader

- Connect the smartphone with the attack host
- `fastboot boot <path-to-recovery.img>`
- Start the imaging process with `dd` and send to the host using `nc`

Worst case: Locked bootloader, ADB disabled

- Unlocking the bootloader wipes USERDATA
  (L. Simon and R. Anderson, University of Cambridge,
  Security Analysis of Android Factory Resets)
- Image via JTAG (M. F. Breeuwsma
  Forensic imaging of embedded systems using JTAG)

Offline Attack Script

- Initially written by Thomas Cannon and Seyton Bradford from viaForensics (ships with Santoku Linux) / Nikolay Elenkov
- PoC script handling PIN numbers only
- Procedure:

  Apply KDF to candidate PIN

  ➡ Decrypt master key

  ➡ Decrypt Header file

  ➡ Check at offset 1080-1082 if ext4 magic signature (0xEF53) found

Offline Attack Analysis

- `bruteforce_stdcrypto.py [header] [footer]`
  - Header file: min. first 1088 bytes of userdata
  - Footer file: complete crypto footer
- Attack Duration (10’000 PIN)
  - Approx. 51 min

Countermeasures

- On Device – Prevent partition imaging
  - Locked Bootloader
  - Disabled ADB and ADB authentication
- Off Device – Make brute-force attack expensive
  - Use of a strong KDF
    - Is an arms race with CPU power and memory increase

Offline Attack Conclusions

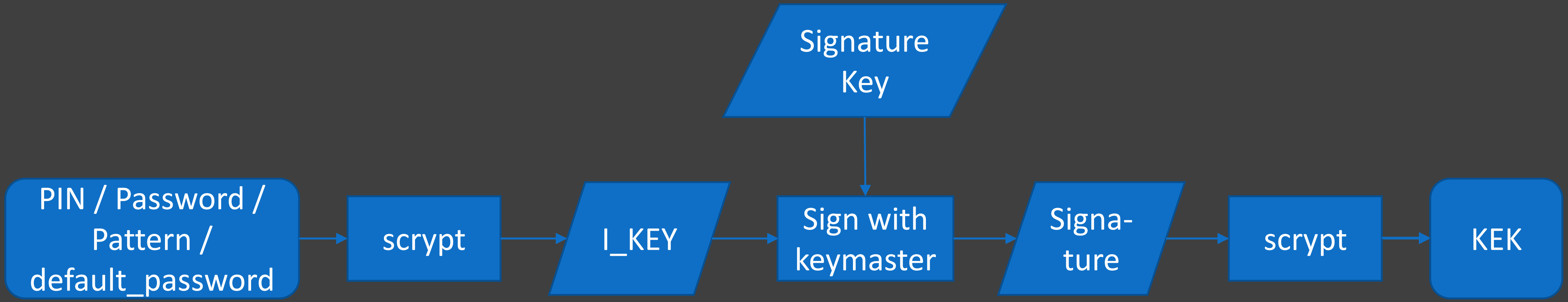
- Time Efficiency: 51 min compared to 22 hrs! ✘
- Missing Security Control ✘
  - Something to prevent offline brute-forcing
- Google responded with improvements for Android 5.0! ✓

Improvements in Android 5.0

- Google addressed the Offline Attack
  - Hardware-binding of the encryption key material to the device

> To be clear, the master key is **not** bound to the device. It is the Key Encryption Key which is cryptographically bound to the device

New KEK Generation Process



Semi-Offline Attack

“What is wanted is not the will to believe, but the will to find out, which is the exact opposite.” (Bertrand Russel)

Pre-Conditions – An Attacker Needs…

- Physical access to device
  - For as long as the brute-force attack takes
  - Mostly this will be due to device theft or loss
- A means to image a partition
  - USERDATA (encrypted data)
  - METADATA (crypto footer)

Attack Application

Client

- Runs on attack host
- Based on the Offline Attack script

Server

- Runs on target device
- Based on cryptfs.c

Tasks

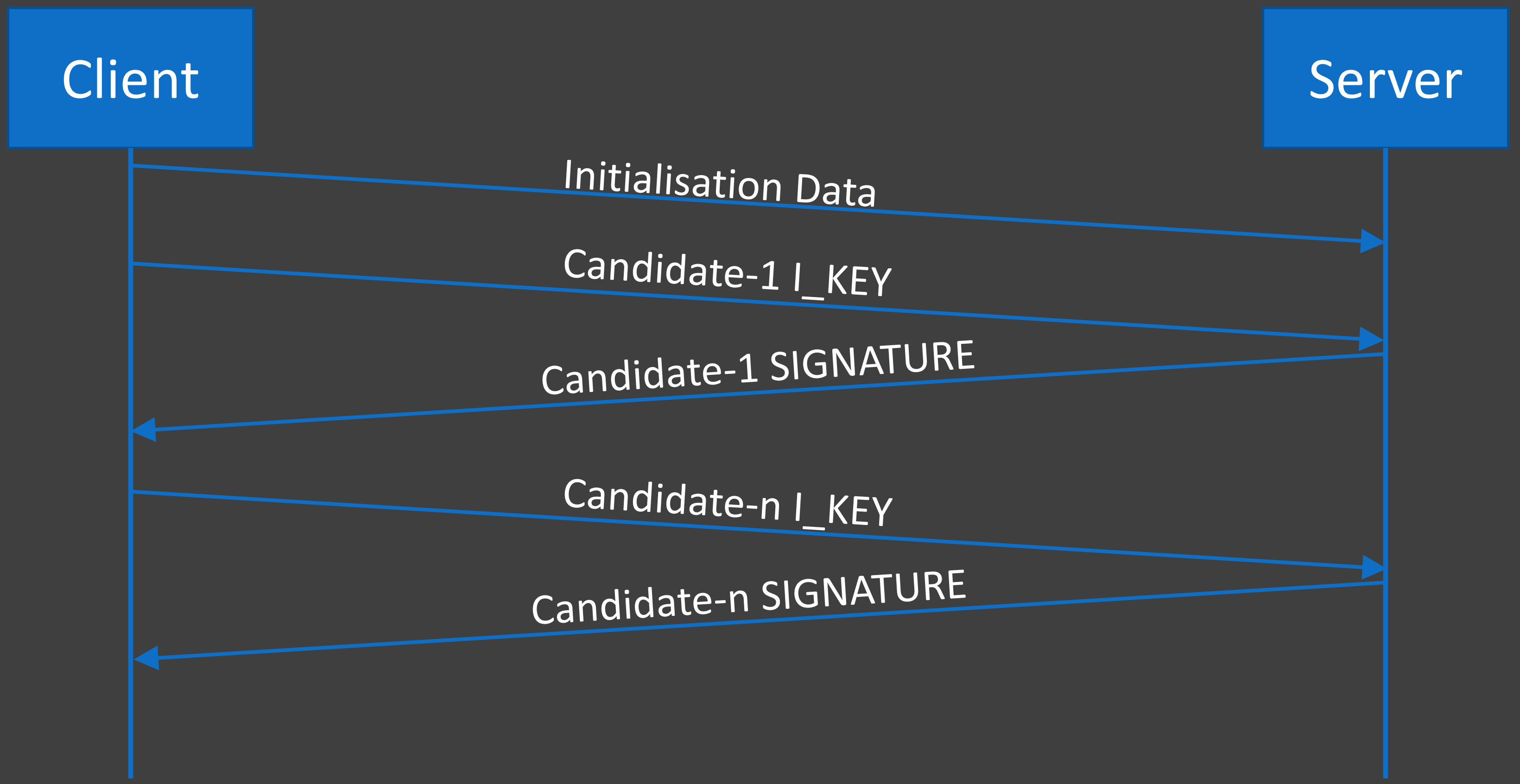
Client (attack host)

1. Process the crypto footer
2. Brute-force strategy
3. Perform KDF operations
4. Decrypt *master key*
5. Decrypt *userdata*

Server (target device)

1. Initialize *keymaster* trustlet with received key blob
2. Request RSA signature on intermediate key candidate

Communication



The Semi-Offline Attack (1/2)

- We have the device
- We have the attack application
- We have the two partitions
- We wipe the smartphone
  - After the wipe, we have a **new** master key and RSA key pair!

The Semi-Offline Attack (2/2)

1. `adb push poc-server /data/local/tmp`
2. `adb forward tcp:8888 tcp:9999`
3. `adb shell su -c '/data/local/tmp/poc-server 9999’`
4. `poc-client.py METADATA HEADER -a 127.0.0.1 -p 8888 -s --bruteforce`

- Attack Duration (10’000 PIN)
  - Approx. 2 hrs 8 min

Demo Semi-Offline Attack

```
  Received candidate from host
  Sign candidate and send result back to host
>> Test New Candidate <<
  Received candidate from host
  Sign candidate and send result back to host
>> Test New Candidate <<
  Received candidate from host
  Sign candidate and send result back to host
>> Test New Candidate <<
  Received candidate from host
  Sign candidate and send result back to host
>> Test New Candidate <<
  Received candidate from host
  Sign candidate and send result back to host
>> Test New Candidate <<
  Received candidate from host

>>> Disconnect <<<
  Close Socket: 127.0.0.1:34896
```

```
>>> PROCESSING: '0790' <<<
>>> PROCESSING: '0791' <<<
>>> PROCESSING: '0792' <<<
>>> PROCESSING: '0793' <<<
>>> PROCESSING: '0794' <<<
>>> PROCESSING: '0795' <<<
>>> PROCESSING: '0796' <<<
>>> PROCESSING: '0797' <<<
>>> PROCESSING: '0798' <<<
>>> PROCESSING: '0799' <<<
>>> PROCESSING: '0800' <<<
        ~>~>~> SUCCESS <~<~<~
        Candidate: 0800
        Decrypted Master Key:
                0xFCF79B53BF9371DE2EBDBA08937823ED



Bruteforce Duration     937.375999928
```

Bits, Please! 30.06.2016

- Gal Beniamini
  - @laginimaineb
  - http://bits-please.blogspot.com
- Read out the Keymaster key!
- Consequences for Semi-Offline Attack
  - Only one client-server interaction necessary
  - After that everything could run offline

Countermeasures

- All countermeasures of the Offline Attack still exists
- Enforce locked bootloader by all manufacturers
- Remove JTAG Interface ➡ still gives the option to unsolder chip
- Key change mechanism for keymaster key
  - Attacker has to solve the RSA problem
  - Or directly brute-force the 16 byte master key?
  - But when to destroy the keymaster key?

Semi-Offline Attack Conclusions

- After the Offline Attack ✗
  - Google said it had fixed the attack
    - Technically YES
    - Actually NO
- Time Efficiency: 2 hrs 8 min compared to 51 min / 22 hrs! ✗
- The issues ✗
  - Wrong attack conditions: How long does an attacker have access?
  - After imaging, no need to keep the mobile in it's original state
  - The internal keymaster key NEVER changes
- Google was already working on improvements! ✓

Comparison of the Attacks

Attack Method	Attack Duration (10’000 PINs)	
Online Attack	81’549 sec	22 hrs 40 min
Offline Attack	3’068 sec	51 min
Semi-Offline Attack	7’622 sec	2 hrs 8 min

Questions