



Extracting Seeds from (Hardware) Wallets

9th of June, 2019 - Breaking Bitcoin - Charles GUILLEMET

Ledger SAS

1, rue du Mail

75002 Paris - France

Ledger Technologies Inc.

121 2nd Street - Suite 5

94105 San Francisco - USA


About Me


- ❖ 10+ years Securing and Breaking Hardware based security systems
- ❖ Formerly Technical Manager in an ITSEF
- ❖ Cryptography, Maths, (Hardware) security



Charles GUILLEMET

CSO at Ledger

 charles-guillemet

 @P3b7_

PGP: 7DC5A359D0D5B5AB6728
1B6EF31F4219E5DC78DF

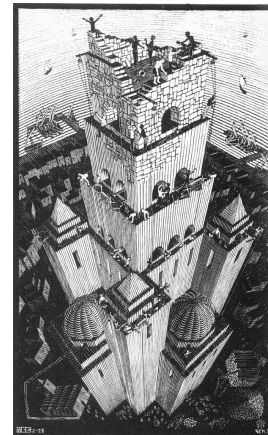


- ❖ Ledger “Red” Team - **Independent**
 - Help for a secure design
 - Improve security (HSM, Vault, Nano S/X)
 - Continuously **challenge the security** of our products
 - **Provide 3rd party security services**

- ❖ Fields of technical expertise
 - Side Channel Analysis
 - Perturbation Attacks
 - Software Attacks
 - Cryptography

- ❖ As the global leader - responsibility to enhance the security in the ecosystem
 - Help individuals and industry to protect their assets

- ❖ Open Source Attack tools: <https://github.com/Ledger-Donjon/>



Agenda

- ❖ **No Crappy attacks** - Only Primary assets: **Seeds Extractions**
 - An “air-gapped” Wallet using Trustzone
 - Open Source Hardware Wallets: PIN extraction / Seed Extraction
 - Shamir Secret Sending

- ❖ Disclaimer
 - Not finger-pointing
 - Vulnerability responsibly disclosed to vendors (through their bug bounty when available)



An “air-gapped” Wallet using Trustzone

An Android based Wallet - Yet Another Bitfi?

Interesting Security Model - From Ellipal website

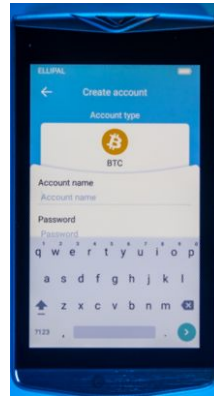
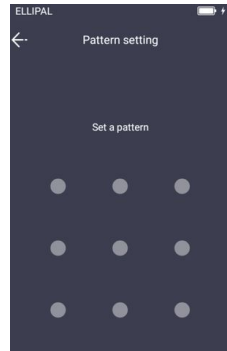
❖ Limited Interfaces

- No network capability
- QR code on screen
- Camera to scan QR code
- SD card for upgrades



❖ Pattern lock

❖ User password for encrypting xpriv

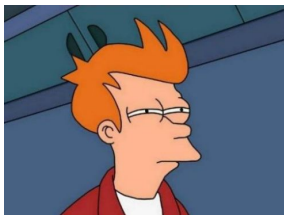


An Android based Wallet - Ellipal: Yet Another Bitfi?

Ordered our Ellipal and waited for it... Meanwhile

- ❖ Upgrade mechanism uses SDCard
- ❖ Have to put the upgrade .bin file in the Sdcard

“Binary file is encrypted and signed”



Let's check for these binaries

- ❖ Retrieve the available binaries (Bruteforce the URL)

<https://order.ellipal.com/lib/v1.7.zip>

<https://order.ellipal.com/lib/v1.8.zip>

<https://order.ellipal.com/lib/v1.8.1.zip>

<https://order.ellipal.com/lib/v1.9.zip>

<https://order.ellipal.com/lib/v1.9.3.zip>

<https://order.ellipal.com/lib/v1.9.4.zip>

<https://order.ellipal.com/lib/v2.0.zip>

Have a look to the entropy



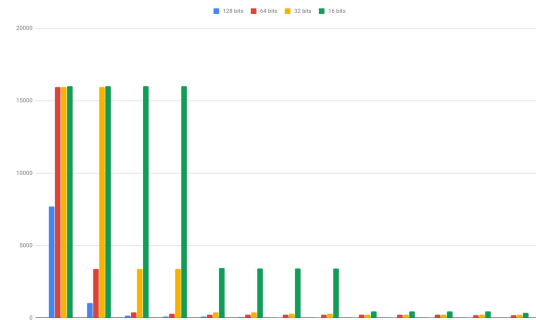
=> Does not look well encrypted

Let's do some stats

- 64 bits encryption
- ECB mode!

Is it single DES?

=> Launch

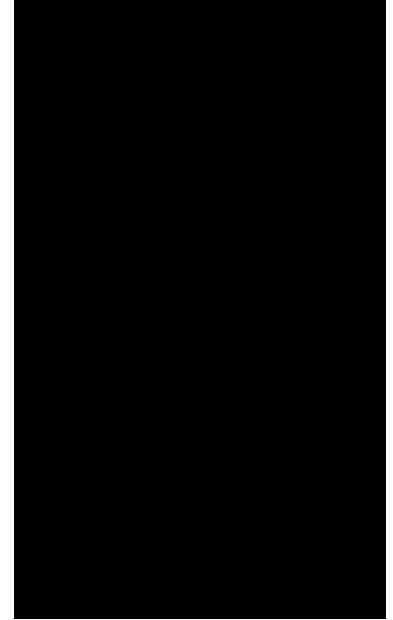
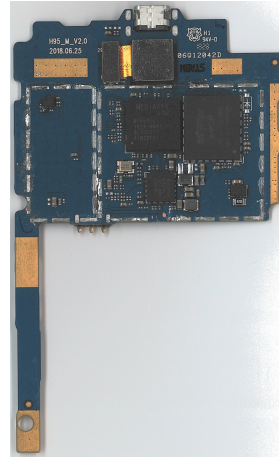
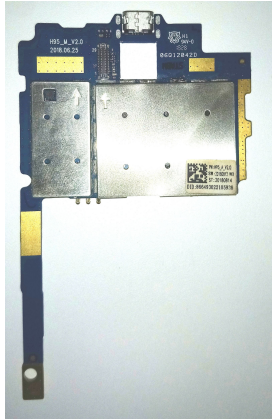


An Android based Wallet - Ellipal: Yet Another Bitfi?

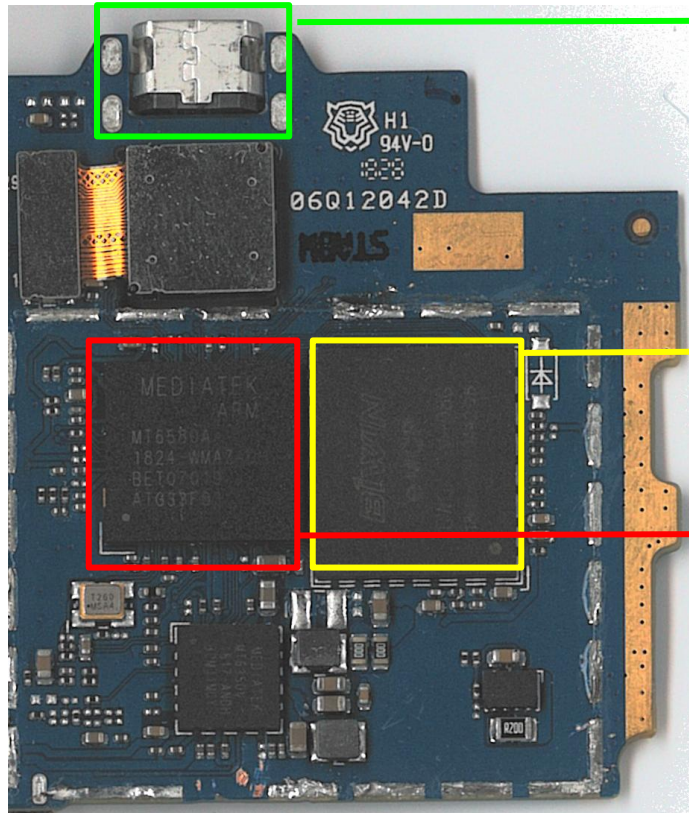
Received our Ellipal

- ❖ Played a bit with the device
- ❖ Found Android hidden menus

A few minutes later



An Android based Wallet - Ellipal: Yet Another Bitfi?



USB port - physically not connected
Only used for charging the battery

External Flash - physical dump is possible

MT6580A - Mediatek SoC

- **Core** Cortex A7
- **Camera:** 13MP ISP
- **GPU:** ARM MALI running at 500 MHz
- **Cellular Technologies:** EDGE, GPRS, HSPA +
- **General Connectivity:** Bluetooth, Wi-Fi
- **GNSS:** GPS
- **Wi-Fi:** b/g/n
- **FM Radio:** Yes

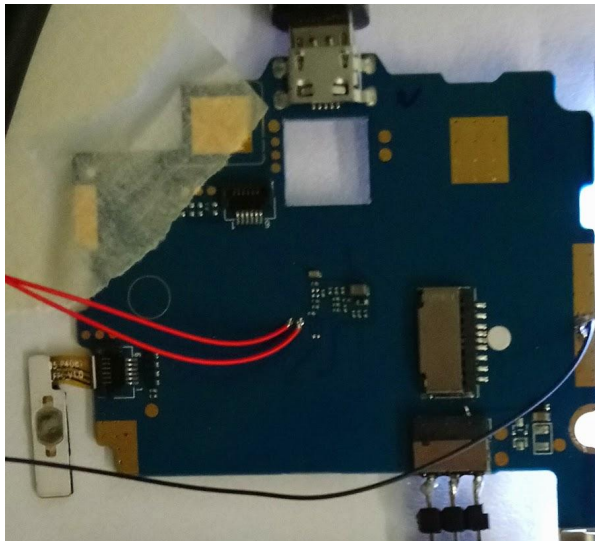


An Android based Wallet - Ellipal: Yet Another Bitfi?

UART Interface is probed

Boot Dump

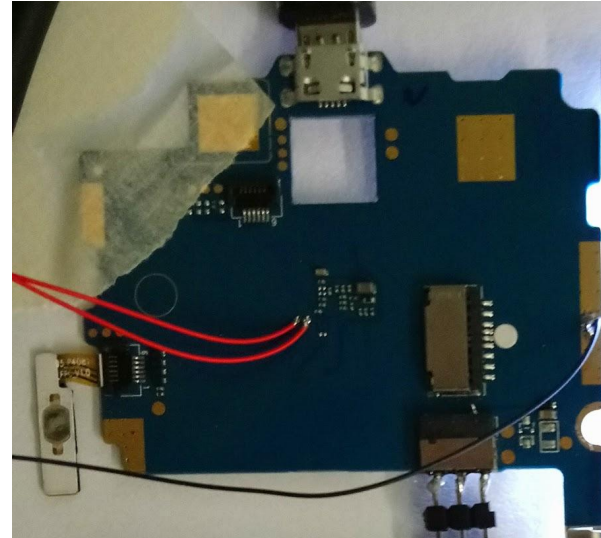
```
AP_PLL_CON1= 0x3C3C23C0
AP_PLL_CON2= 0x4
CLKSQ_STB_CON0= 0x25002100
PLL_ISO_CON0= 0x202020
ARMPLL_CON0= 0x11
ARMPLL_CON1= 0x8009A000
ARMPLL_PWR_CON0= 0x5
MPLL_CON0= 0x8000011
MPLL_CON1= 0x800E7000
MPLL_PWR_CON0= 0x5
UPLL_CON0= 0x38000001
UPLL_CON1= 0x1000060
UPLL_PWR_CON0= 0x5DISP_CG_CON0= 0xFFFFFFFFC,
DISP_CG_CON1= 0x0,
    FFE0
RGU STA:      0
RGU INTERVAL: FFF
RGU SWSYSRST: 8000
==== Dump RGU Reg End ====
RGU: g_rgu_satus:0
    mtk_wdt_mafter set KP enable: KP_SEL = 0x1C70 !
```



An Android based Wallet - Ellipal: Yet Another Bitfi?

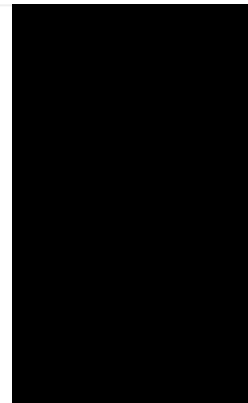
UART Interface is probed

Send FACTFACT on TX - Factory Mode



Let's play with the USB

- ❖ USB is soldered using PCB test points
- ❖ Mediatek Bootloader is activated using [saleemrashid / mediatek_flash_tool](#)
 - Success
- ❖ **Full access to the Flash memory**
 - **Can Read and Write everything**
 - **Filesystem is not encrypted**
- ❖ Enabled non-root ADB, installed third-party APK...
- ❖ Possibility to backdoor the wallet / activate WiFi, GPRS...
- ❖ Dump of the Wallet application and reverse
 - Retrieved the Firmware Signature public key
 - **Retrieved the Firmware Encryption key (3-DES)**
 - **Retrieved the encrypted wallet private key**



An Android based Wallet - Ellipal: Yet Another Bitfi?

Let's play with the USB

- ❖ The Reverse of the app shows the encryption mechanism is **weak** (sha256 based)
- ❖ Brute-force is easy -
 - 8 full random char passwd ~ a few minutes

Physical access => Seed can be extracted

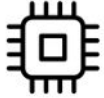


An Android based Wallet - Ellipal: Yet Another Bitfi?



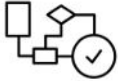
Complete Isolation - ELLIPAL does not need any physical connection
It is impossible to hack the ELLIPAL as it does not connect to any network

Wifi, BT, GPRS, USB are present and can be reactivated



ELLIPAL's encrypted storage is based on TrustZone security technology
Private key is stored securely in ARM-based TrustZone of the chip

No TrustZone on this chip



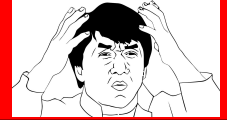
Internal data is stored using AES 128 high-intensity encryption algorithm
Even if hacking was to occur offline, it would take years to decrypt the data

AES 128 High-Intensity???
-> Bad encryption algorithm, easy Bruteforce



Dynamic system protection
ELLIPAL uses original technology to detect the operating environment and protects the system while running

??? It uses Android
Backdooring is quite easy



ELLIPAL uses hardware noises, system entropy, and user password to ensure data randomness of private key

The private keys are generated with Android randomness generation



Multiple verification steps to maximize protection
Gesture password, Account Password, Confirmation QR code, Mnemonics

Correct



An Android based Wallet - Ellipal: Yet Another Bitfi?



An Android based Wallet - Ellipal: Yet Another Bitfi?

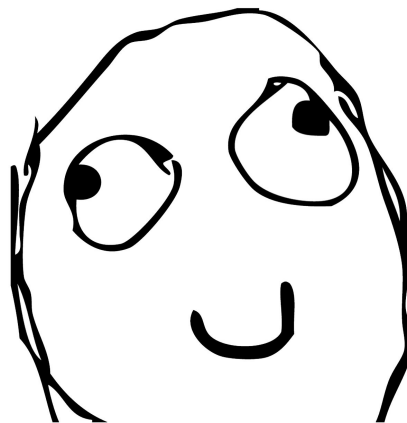
Responsibly disclosed: 2018-03

Status: Updated to v2.0 - We didn't check anything

Triggered Bounty program

They gave us a Bounty reward

They sent us an upgraded device :)



ELLIPAL Security Update

Stronger Security
Harden Your Wallet Against
Supply Chain Attacks



We would like to say thank you to Charles GUILLEMET, Chief Security Officer of Ledger for giving us remarkable advice and cooperating smoothly with ELLIPAL. We are glad to have worked together with you and looking forward to building a better product for the community.



Open Source Hardware Wallets

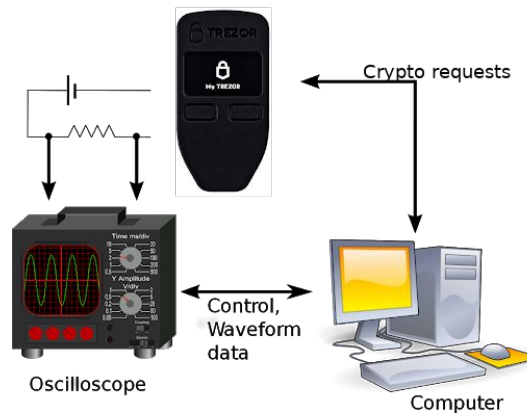
Guessing PIN

Open Source Hardware Wallets - An unexpected SCA

Power consumption, ElectroMagnetic emanations

- ❖ **Measure** the power consumption/EM during cryptographic computations
- ❖ Record traces
- ❖ **Post processing** traces
- ❖ Conduct **Side Channel Analysis**

- ❖ First attacks end 90's (except national Agencies)
 - Timing attacks 1996. (P. Kocher)
 - SPA
 - DPA 1998 (P. Kocher)
 - CPA 2004 (Brier)
 - Template Attacks 2002 (Chari)
 - Machine Learning based Attacks (2015-2016)



Side Channel Attacks

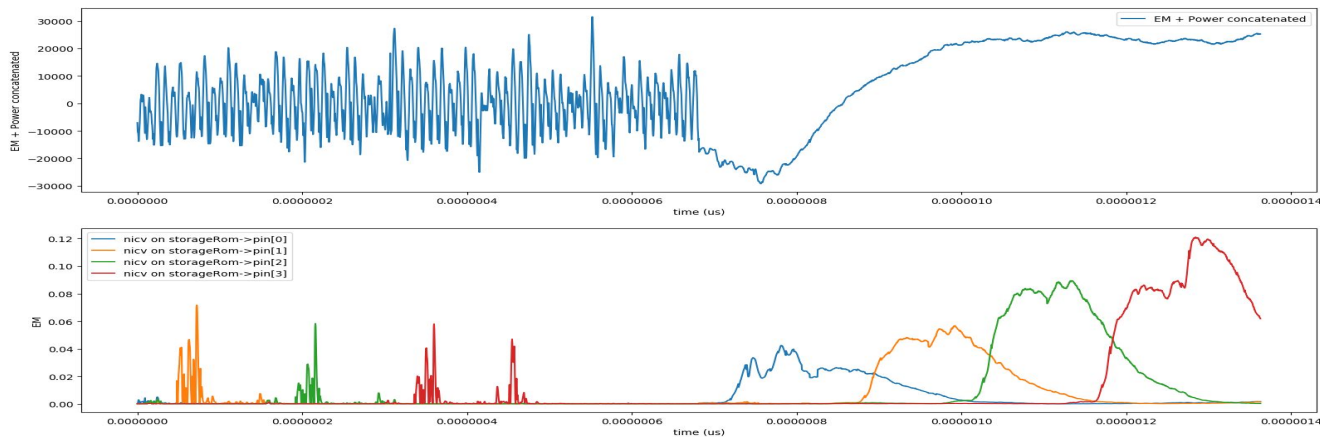
- Example on Trezor PIN

Trezor code

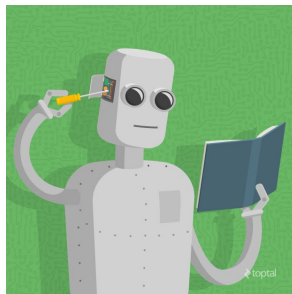
```
/* Check whether pin matches storage. The pin must be
 * a null-terminated string with at most 9 characters.
 */
bool storage_containsPin(const char *presented_pin)
{
    /* The execution time of the following code only depends on
    the
    * (public) input. This avoids timing attacks.
    */
    char diff = 0;
    uint32_t i = 0;
    while (presented_pin[i]) {
        diff |= storageRom->pin[i] - presented_pin[i];
        i++;
    }
    diff |= storageRom->pin[i];
    return diff == 0;
}
```



- ❖ Power/EM single trace
- ❖ Traces Synchronization
- ❖ POI detection depending on (*storageRom->pin[i]* - *presented_pin[i]* for $0 \leq i < 4$)

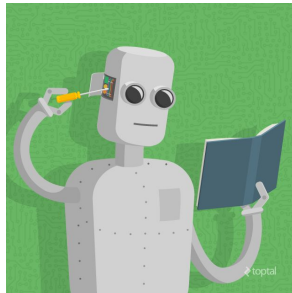
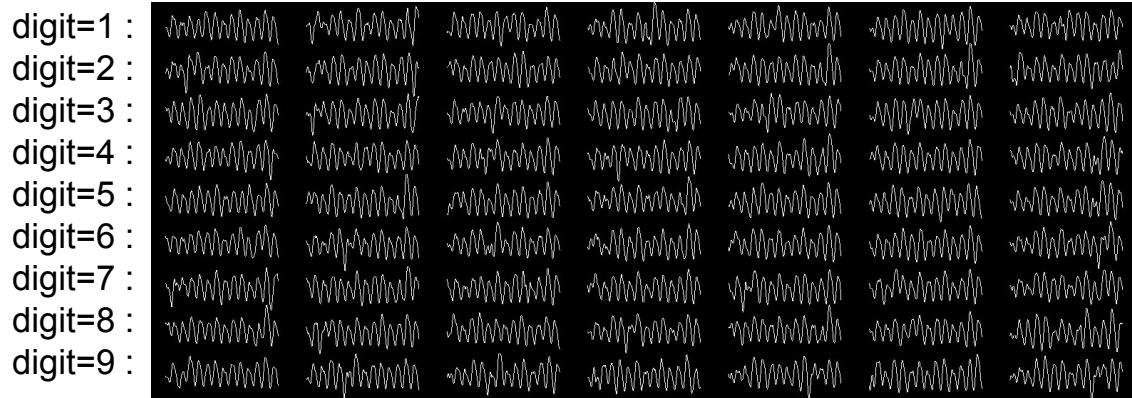


Side Channel Attacks: PIN verification function



Side Channel Attacks: PIN verification function

Pin behaviour is learnt in a very similar way...



Device A

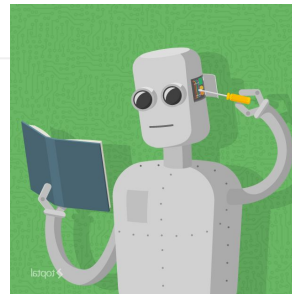
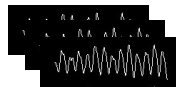


Device B



Side Channel Attacks: PIN verification function

1. Get a device A, record many traces with random PIN
2. Learn the behavior of the device

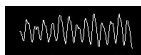


 Ledger

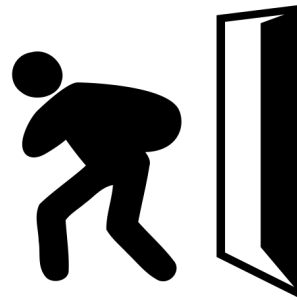
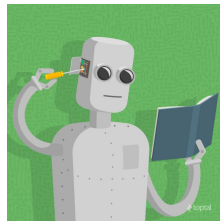


Device A

3. Get a physical access to the attacked device
4. Enter random PIN, measure the power consumption of the device, ask to the MLA
try the most likely PIN



1234



Device B

On average, **5 tries to guess the correct PIN**
(15 tries at most on Trezor)

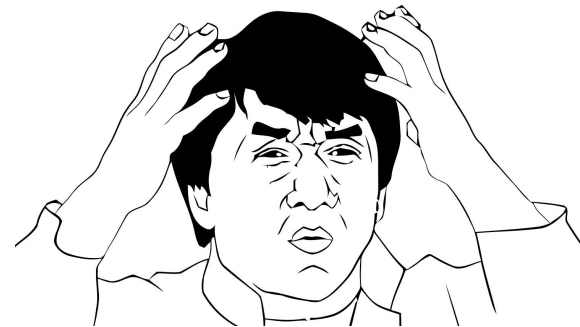
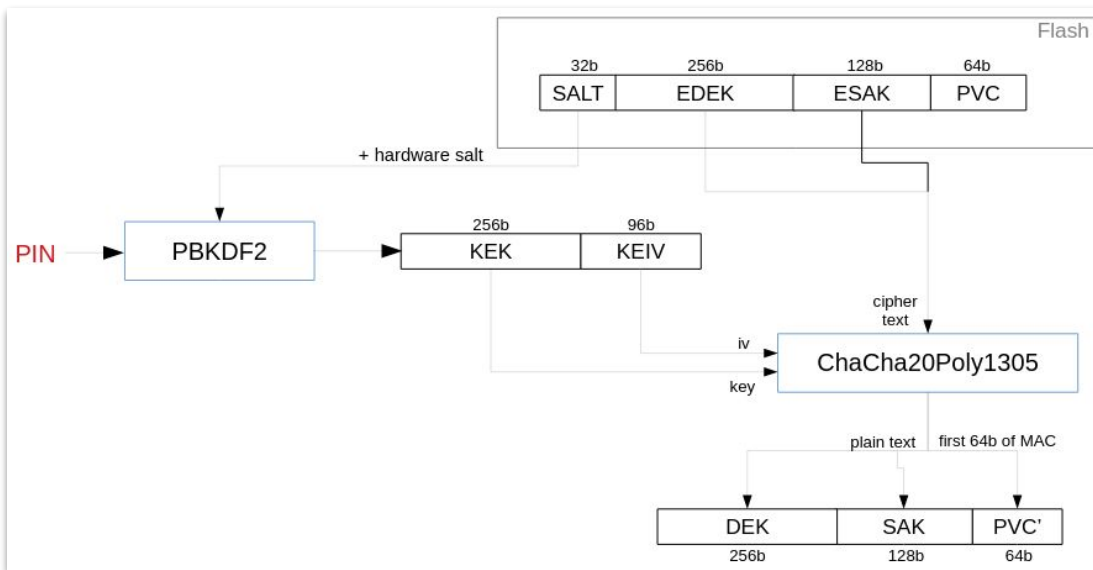
5. Enjoy



Responsibly disclosed: 2018-11-20
Status: Hardened

Issue 3— Side Channel Attack PIN

Side-channeling the PIN on Trezor One was indeed impressive and we commend Ledger's effort. At the same time, we would like to thank Ledger for responsibly disclosing the issue to us. This attack vector was closed by backporting the way to store data on Trezor Model T to Trezor One.



Open Source Hardware Wallets

Extracting seed

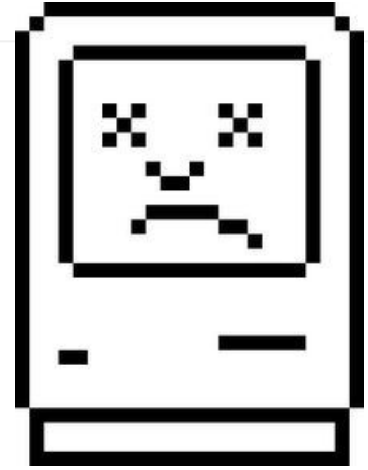
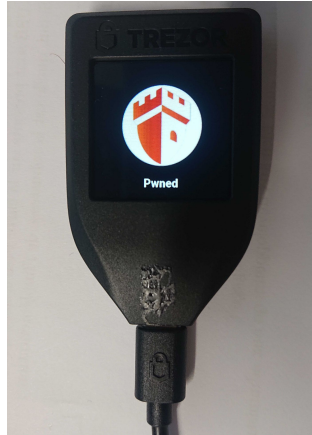
An efficient physical seed extraction attack



Found and implemented an **attack allowing**

Dump of seed

- ❖ Trezor One
- ❖ Keepkey
- ❖ B Wallet
- ❖ Trezor T



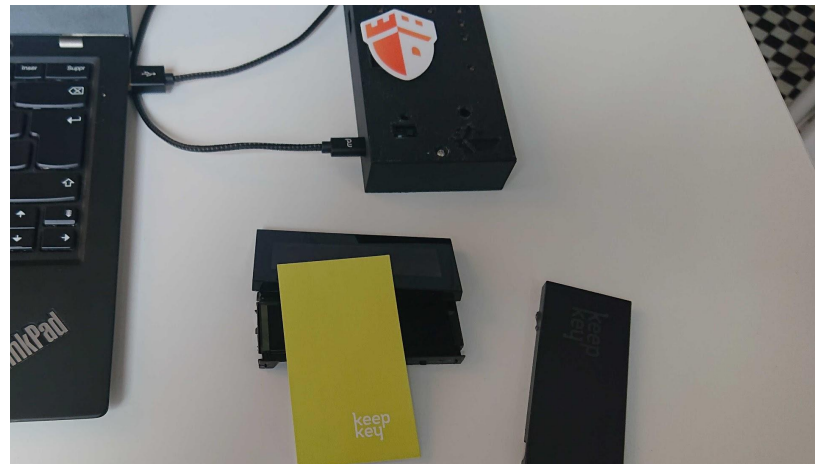
- ❖ All firmwares are (and will be) **vulnerable**

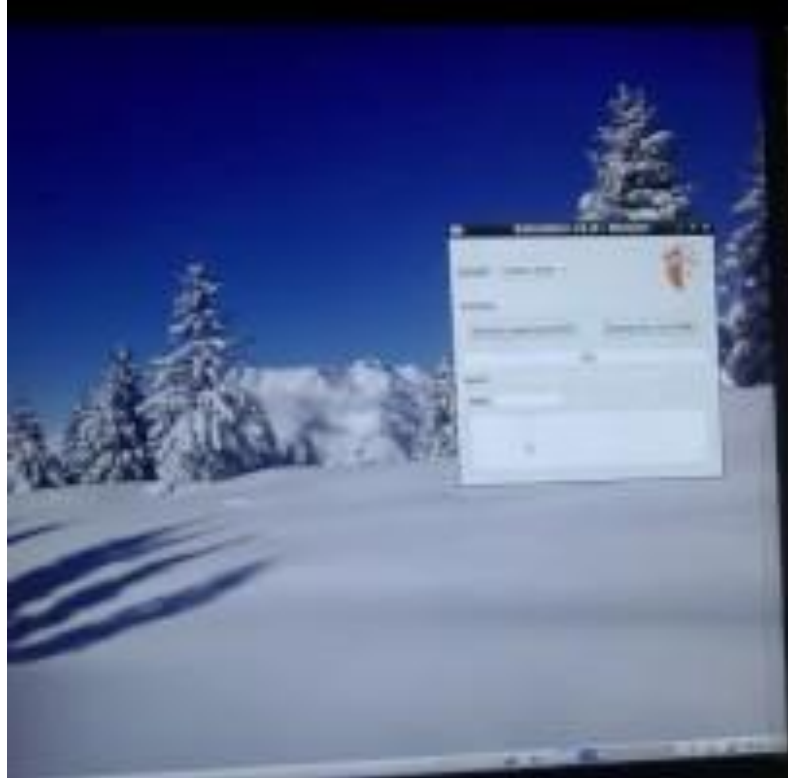
Unfortunately NOT patchable

An efficient physical seed extraction attack

Decided not to disclose the method to protect users

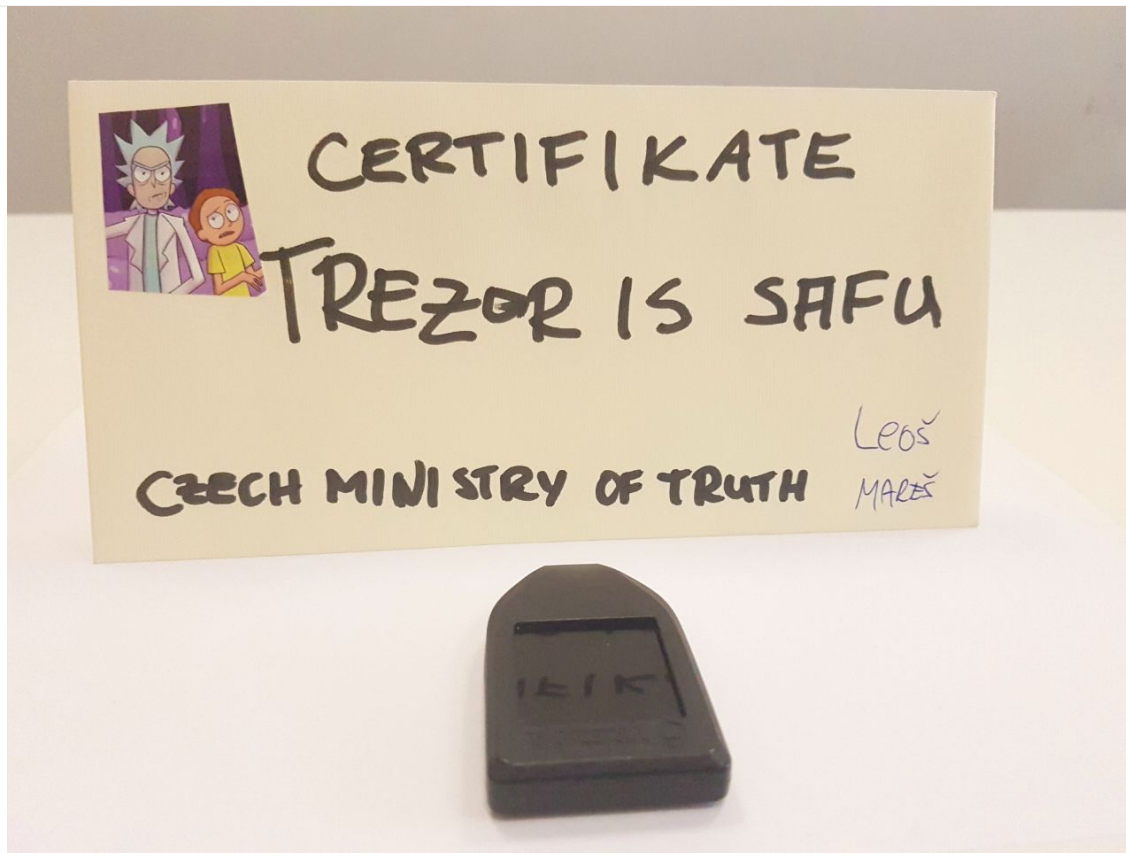
- **Been asked for details**
 - ❖ **Physical access is necessary**
 - ❖ We improved the setup
 - Setup cost is ~100\$ + computer
 - ❖ Necessary time
 - ~3 min preparation
 - < 2 min extraction
 - ❖ Works on every firmware version
 - On encrypted firmware Trezor >=1.8 or Keepkey
 - Extraction depends on the PIN length
- => A few minutes worst case





bullshit™





An efficient physical seed extraction attack



Responsibly disclosed: 2018-12-20

Status: Can not be patched

We would like to thank Ledger for practically demonstrating the attack that we have been aware of since designing Trezor.

- Suggested Physical Threat is **out of the threat model**
- Use a **long** passphrase: ~36 random characters

udP^Cs6{ZBk&ds(bTx;)\$xYWYAUv]xN`4Gq

They gave us a bounty reward



Shamir Secret Sending

“The Native Web 3.0 Blockchain Phone”

- ❖ “Hardware Wallet”
- ❖ Trusted Display
- ❖ Secure Enclave
- ❖ Social key recovery

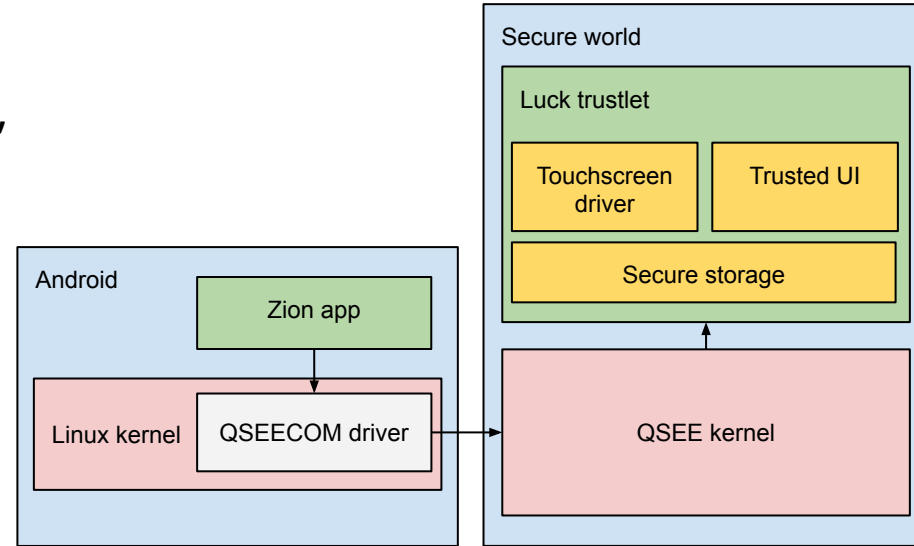
SoC Qualcomm Snapdragon 845



HTC Exodus - Hardware wallet: Zion - Security model



- ❖ Android App “Zion” + trustlet “Luck”
- ❖ Seed is stored in the secure OS
- ❖ Secure peripherals
 - Screen
 - Touchscreen for input
 - Fingerprint sensor
- ❖ Signatures are secured with PIN



The Seed is secure, even if the phone is rooted



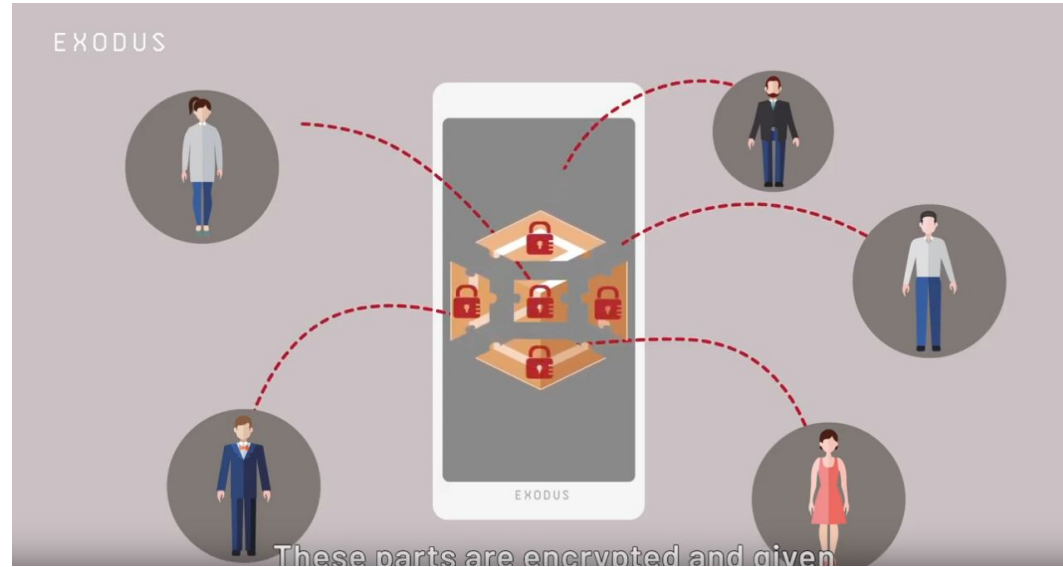
HTC Exodus - A very interesting feature: Social Key Recovery

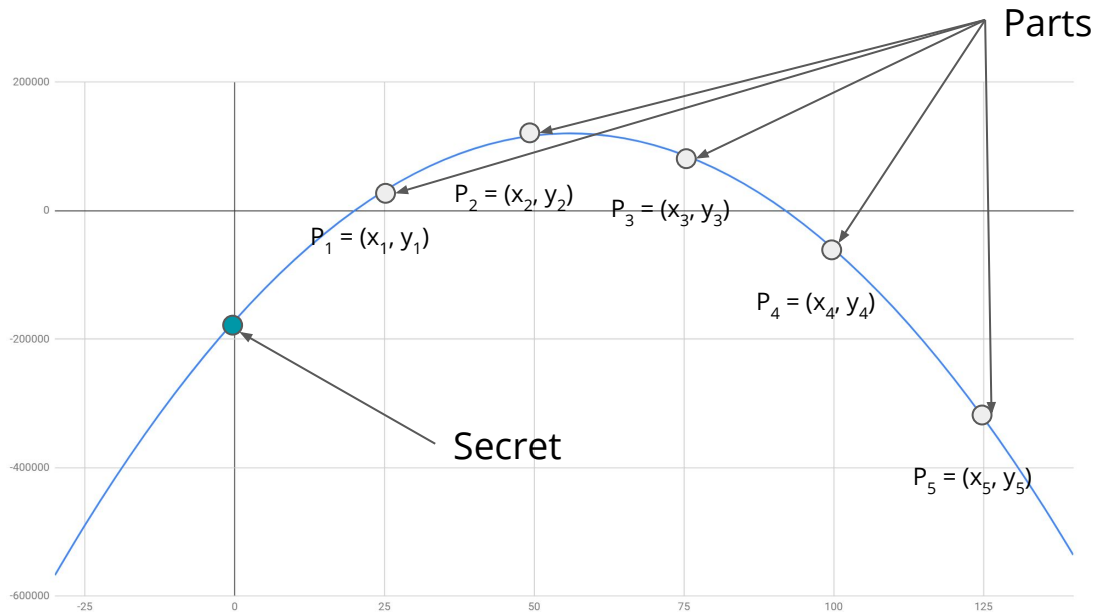
- ❖ Trusted contacts
 - install Zion app
- ❖ They receive a share

3 out 5

3 shares to reconstruct the seed

- ❖ The shares are not stored securely
But 1 or 2 shares give **no info**



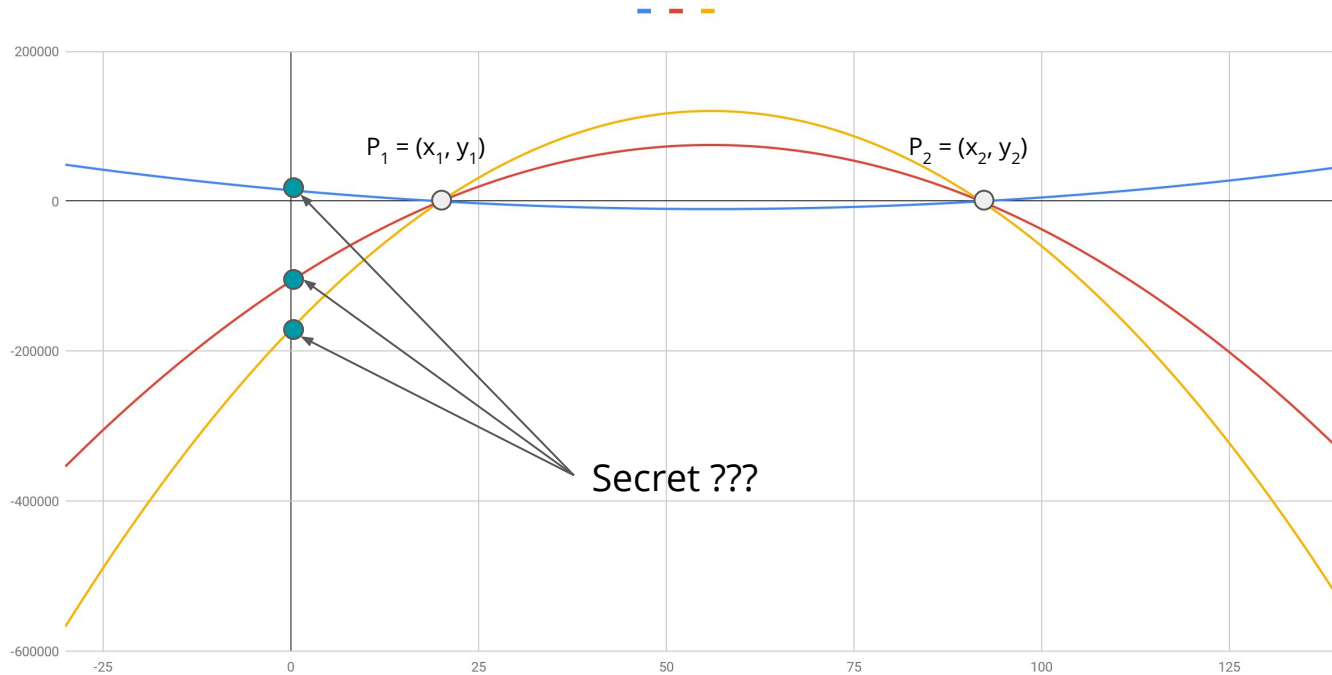


$$y = ax^2 + bx + c, \text{ secret: } c$$

- Shared secret: c
- 3 shares are necessary to reconstruct c
- a, b randomly generated and **secret**
- **Shares: (x_i, y_i)**
(Lagrange Theorem)



HTC Exodus - Use Shamir Secret Sharing



Only 2 shares: No info on the secret - As many possible secrets as possible polynomials



Android app has been reversed

- ❖ The SSS implemented shares the 256 bits seed (32 bytes)
 - with 32 polynomials of degree 2 (coeff in $GF(2^8)$)
 - Evaluates in 5 points and sends the shares
- ❖ The coefficient a,b are randomly generated with a PRNG

But the PRNG update operation is linear => a and b are linearly dependant

$$P_i(x) = a_i x^2 + b_i x + c_i$$

$$P_i(x) = L(b_0, b_1 \dots b_{31}) x^2 + b_i x + c_i$$



❖ Retrieving the secret

⇔ Solving linear system of $32 \times 3 = 96$ equations over $GF(2^8)$:

3 x 32-bytes shares -> 1 Solution for c

⇔ Solving linear system of $256 \times 8 = 768$ equations over $GF(2)$:

3 x 256-bits shares are necessary -> 1 Solution for c

❖ But the system is not linearly independent

➤ The rank of the 768-bits Matrix is **< 512**

Using 2 shares, the kernel of the Matrix is computed in less than 1 sec

=> The seed is extracted

Compromise two “Trusted contact” phones - or collusion

But there is worse

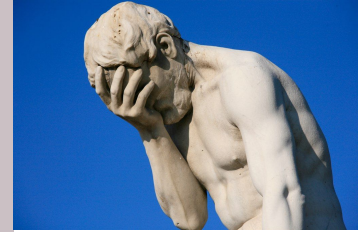
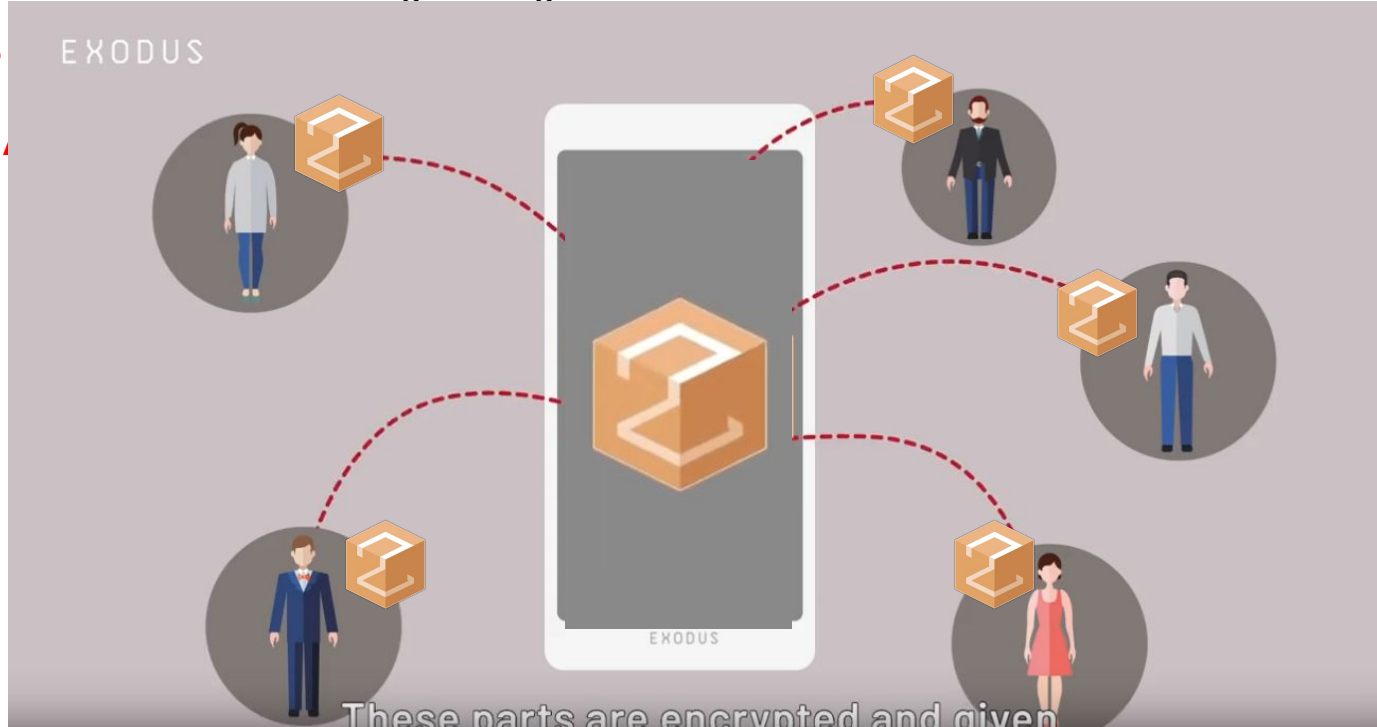


❖ In Firmware v1.54.2401.6

- The reverse engineering shows that the PRNG is seeded with a fixed value

=>

=>



HTC Exodus - Use Shamir Secret Sharing

Responsibly disclosed: 2018-03

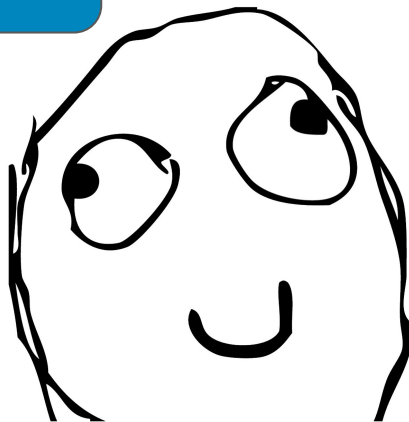
Status: Correctly patched

Users not really warned... => Did not regenerate their seed

Triggered the creation of a Bounty program

HTC EXODUS announces bounty program for the Zion Vault

The Zion Security Rewards Program follows a collaboration with Ledger, a leader in security and infrastructure solutions for cryptocurrencies and blockchain applications, in solving for a number of potential vulnerabilities.



Studied several (Hardware) wallets -

- Found critical vulnerabilities allowing to **Extract seeds**
 - **With a physical access**
 - **Ellipal, Trezor One, Trezor T, Keepkey**
 - **Remotely**
 - **HTC Exodus**

- Contribute to drastically **improve the security** of these wallets

- Triggered the creation of bounty programs
- Got small bounty rewards



Questions?

When the stakes are high
Expect Attackers with high potential

*By: Karim Abdellatif, Jean-Baptiste Bédrune, Gabriel Campana,
Olivier Hériveaux, Manuel San Pedro, Victor Servant*