

Max Planck Institute | Security and Privacy

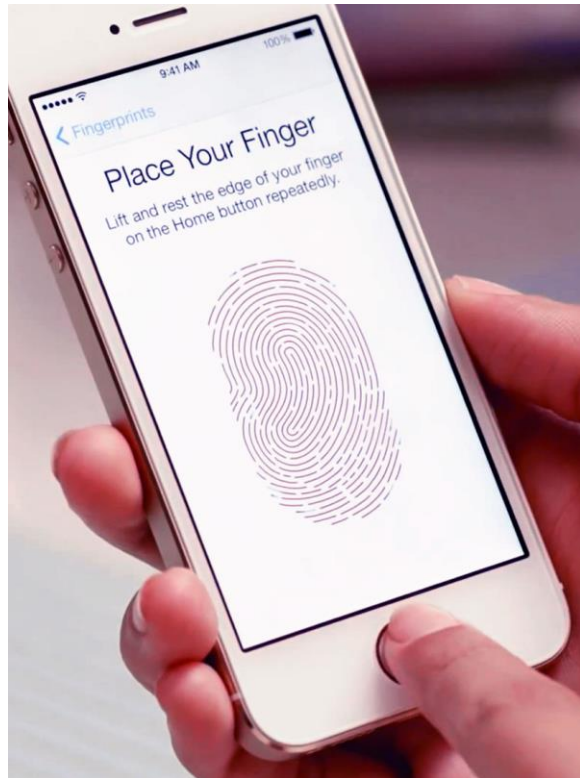
Extracting iOS' Passcode Blacklist

Maximilian Golla

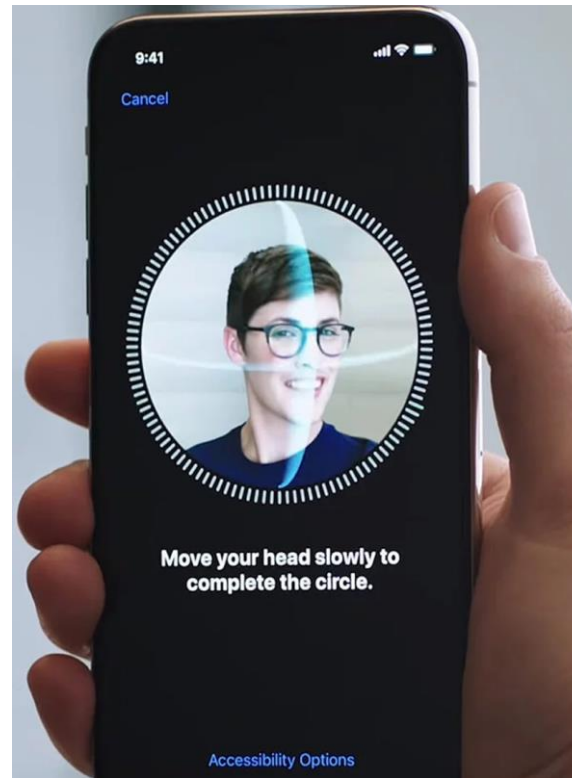
March 2020 | Bochum, Germany

Biometric-Based Reauthentication

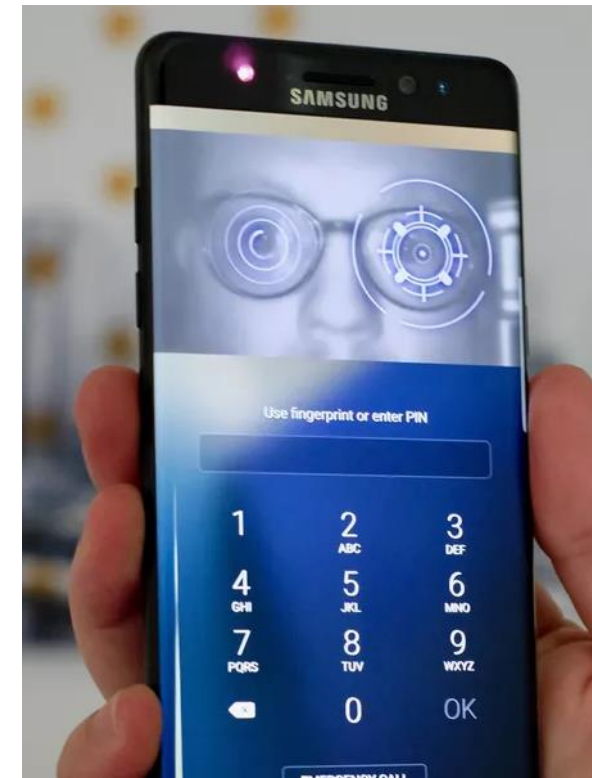
“Face unlock feels almost like not having any lock screen security.”



Fingerprint



Face

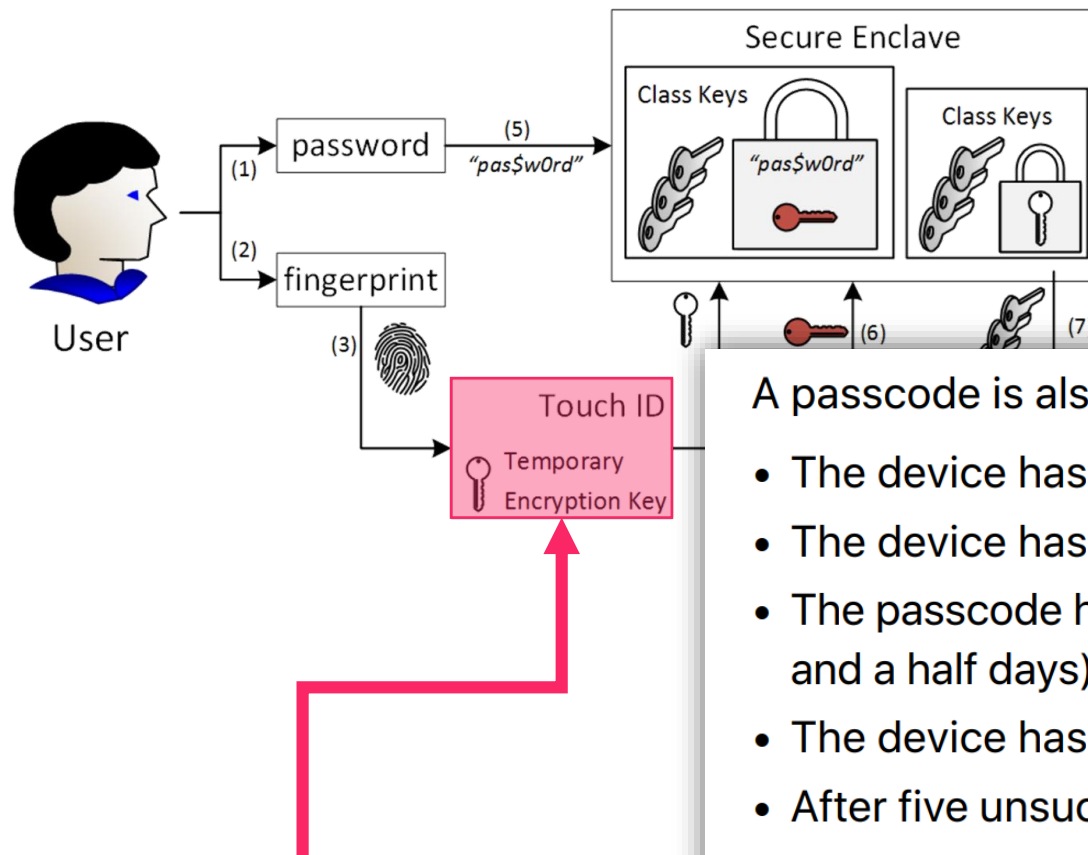


Iris

“Intelligent Scan”



Biometrics are a Convenience Feature!



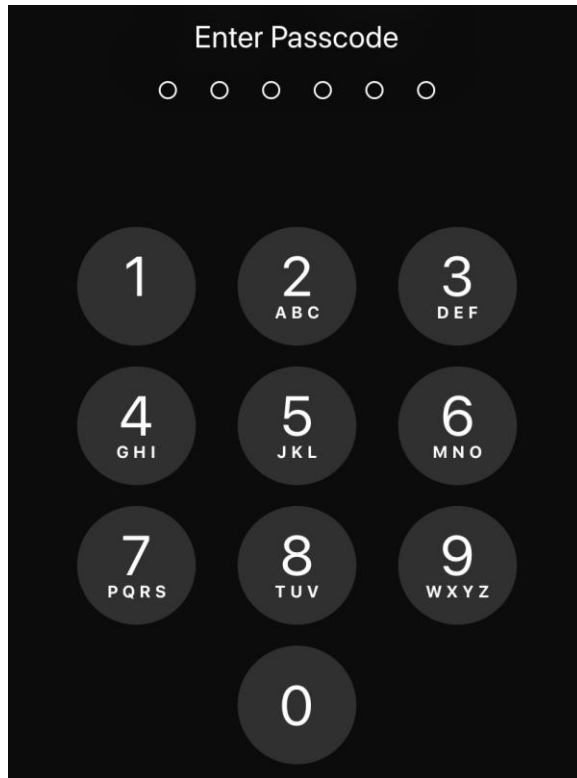
TEK is flushed periodically
and on device reboot.

A passcode is also required if your device is in the following states:

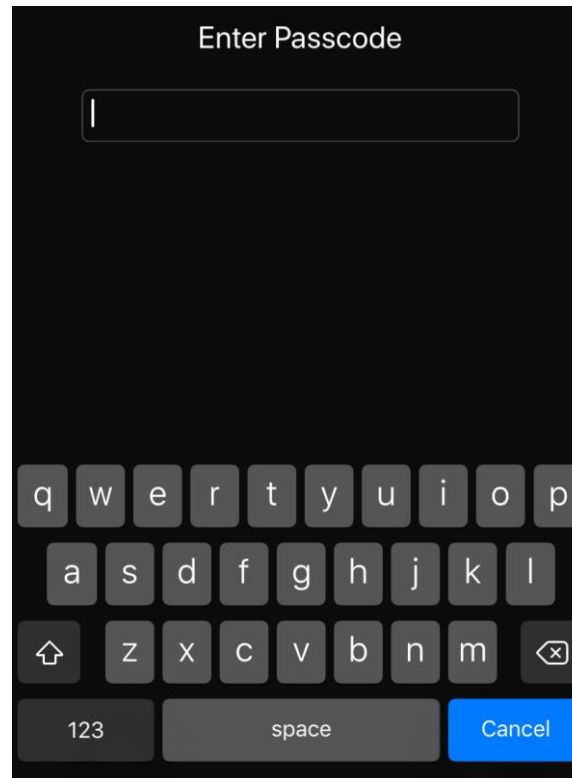
- The device has just been turned on or restarted.
- The device hasn't been unlocked for more than 48 hours.
- The passcode hasn't been used to unlock the device in the last 156 hours (six and a half days) and a biometric hasn't unlocked the device in the last 4 hours.
- The device has received a remote lock command.
- After five unsuccessful biometric match attempts.
- After initiating power off/Emergency SOS.



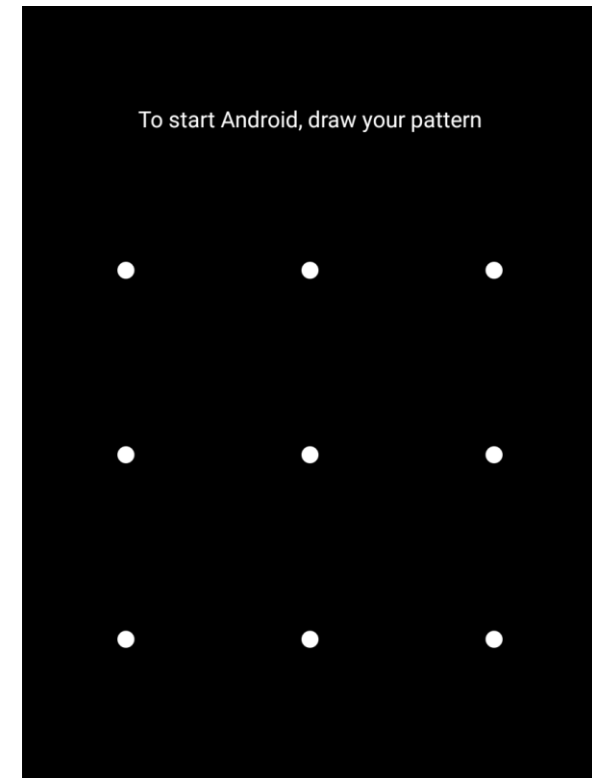
Knowledge-Based Authentication



4/6-digit PINs



Passwords



Pattern



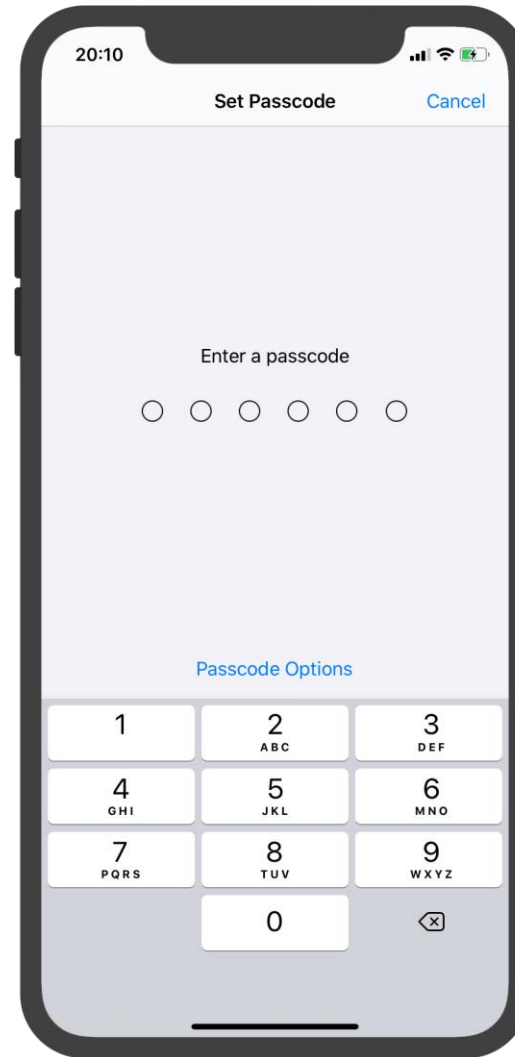
iOS Passcode

Knowledge-based auth. scheme

- **6-Digit Numeric Code (default)**
- 4-Digit Numeric Code
- Custom Numeric Code
- Custom Alphanumeric Code

Strict rate-limiting (10 guesses)

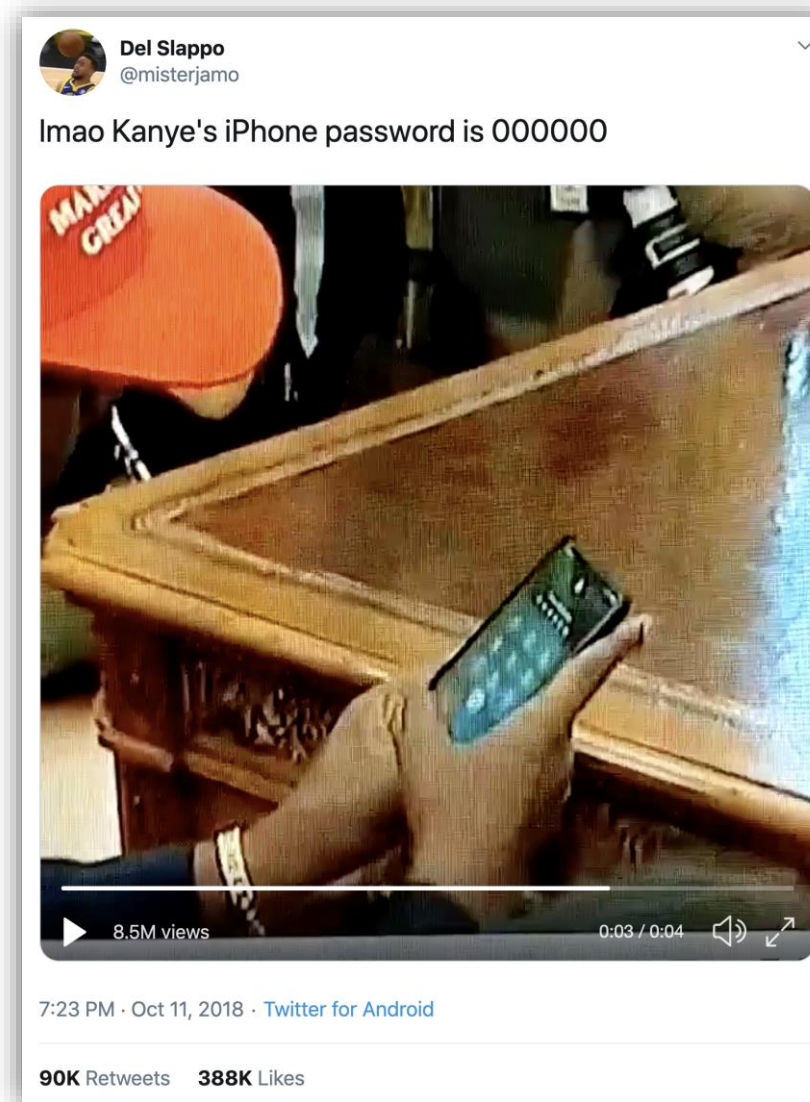
Enrollment



Authentication



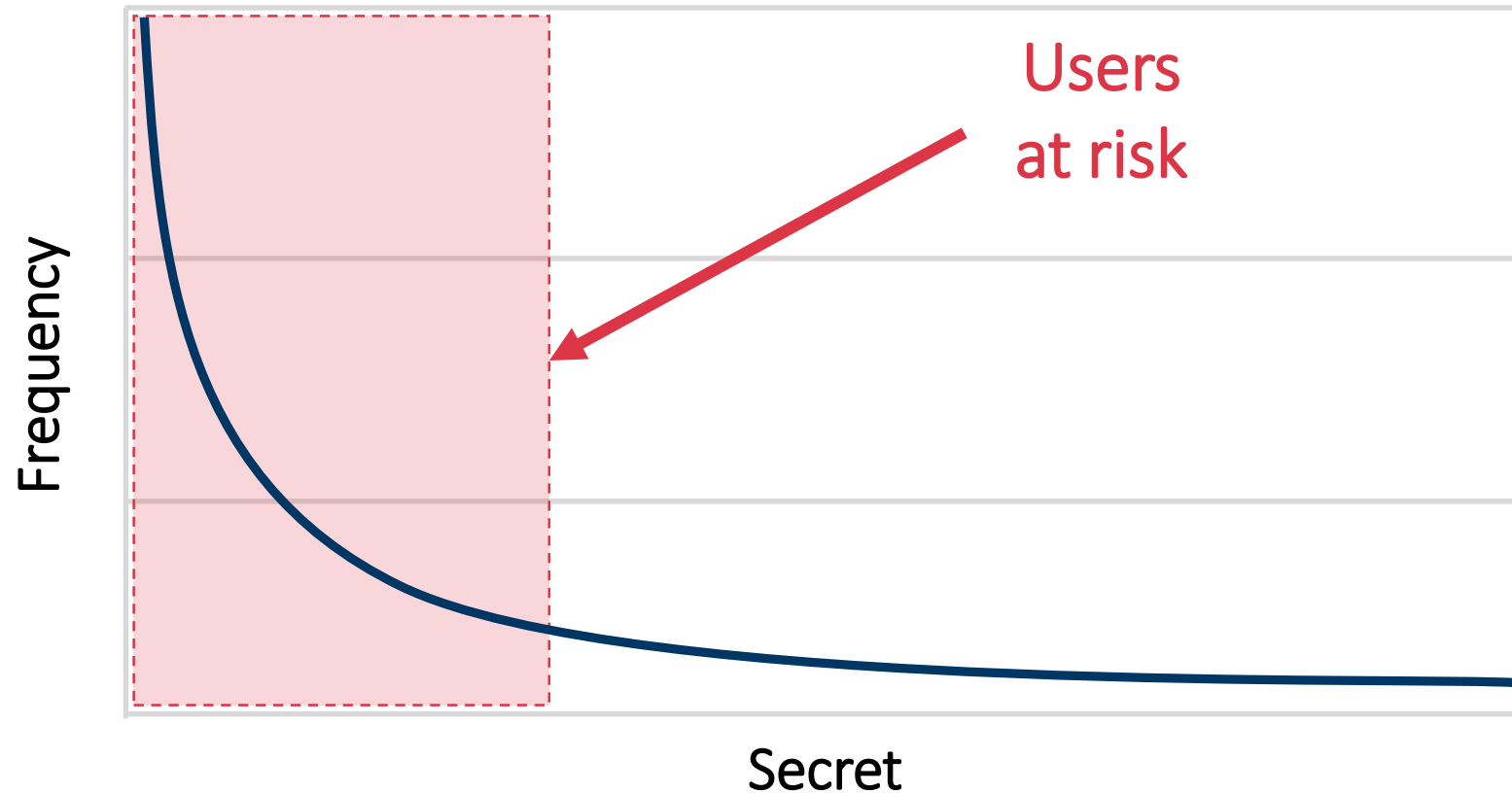
User-Chosen PINs



Selection Bias



User-choice heavily biased.

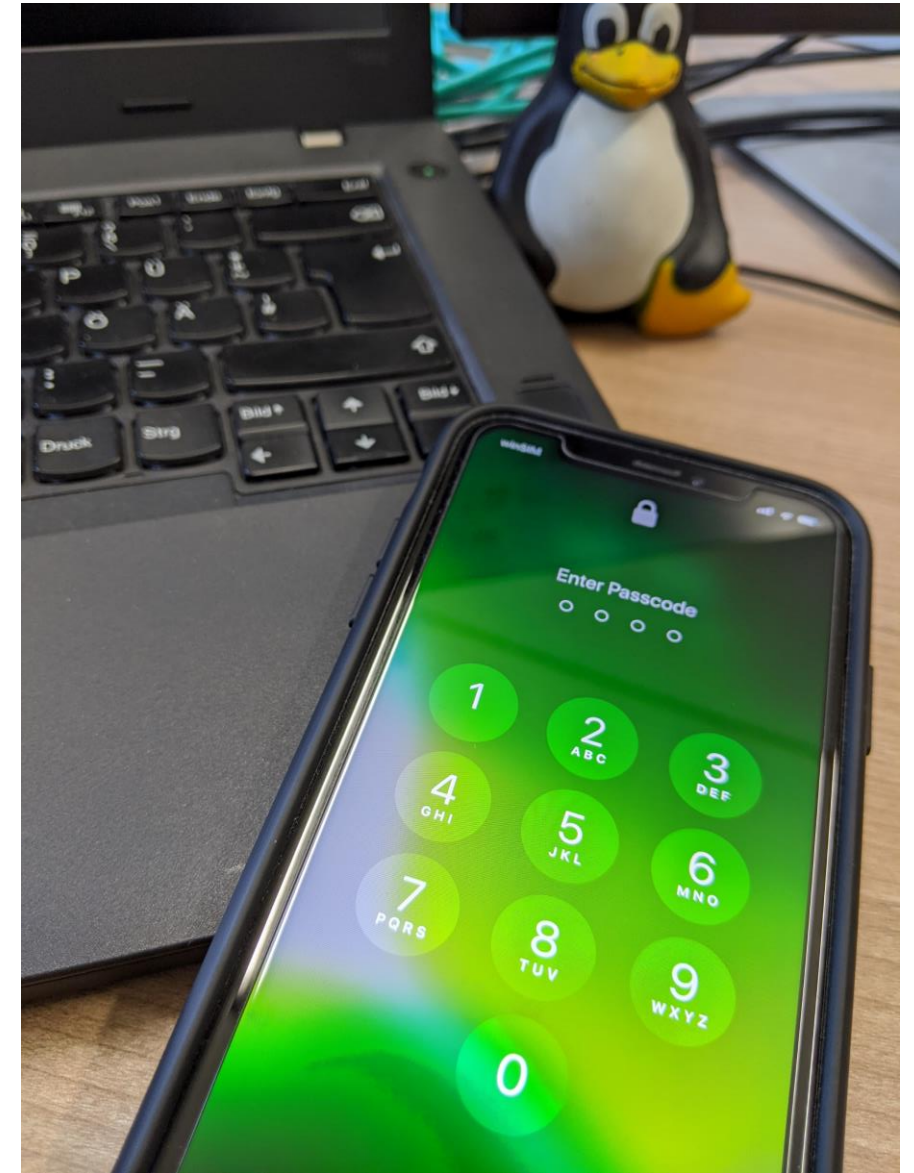


Threat Model

Attacker guesses the n most common secrets in decreasing order of success.

Throttled Guessing Attack:

	iOS 9-13	Android 7-10
3 Guesses	00s	00s
10 Guesses	1h 36m 00s	30s
30 Guesses	Disabled	10m 30s
100 Guesses	Disabled	10h 45m 30s



What This Talk is Not About!



~~iPhone unlocking as used by law enforcement.~~

→ We only extract the iOS Passcode blacklist!



iOS Passcode Blacklist

- “Blacklist” consisting of weak PINs
- Not documented
- Allows users to click-through (“Use Anyway”)

Examples:

“000000”

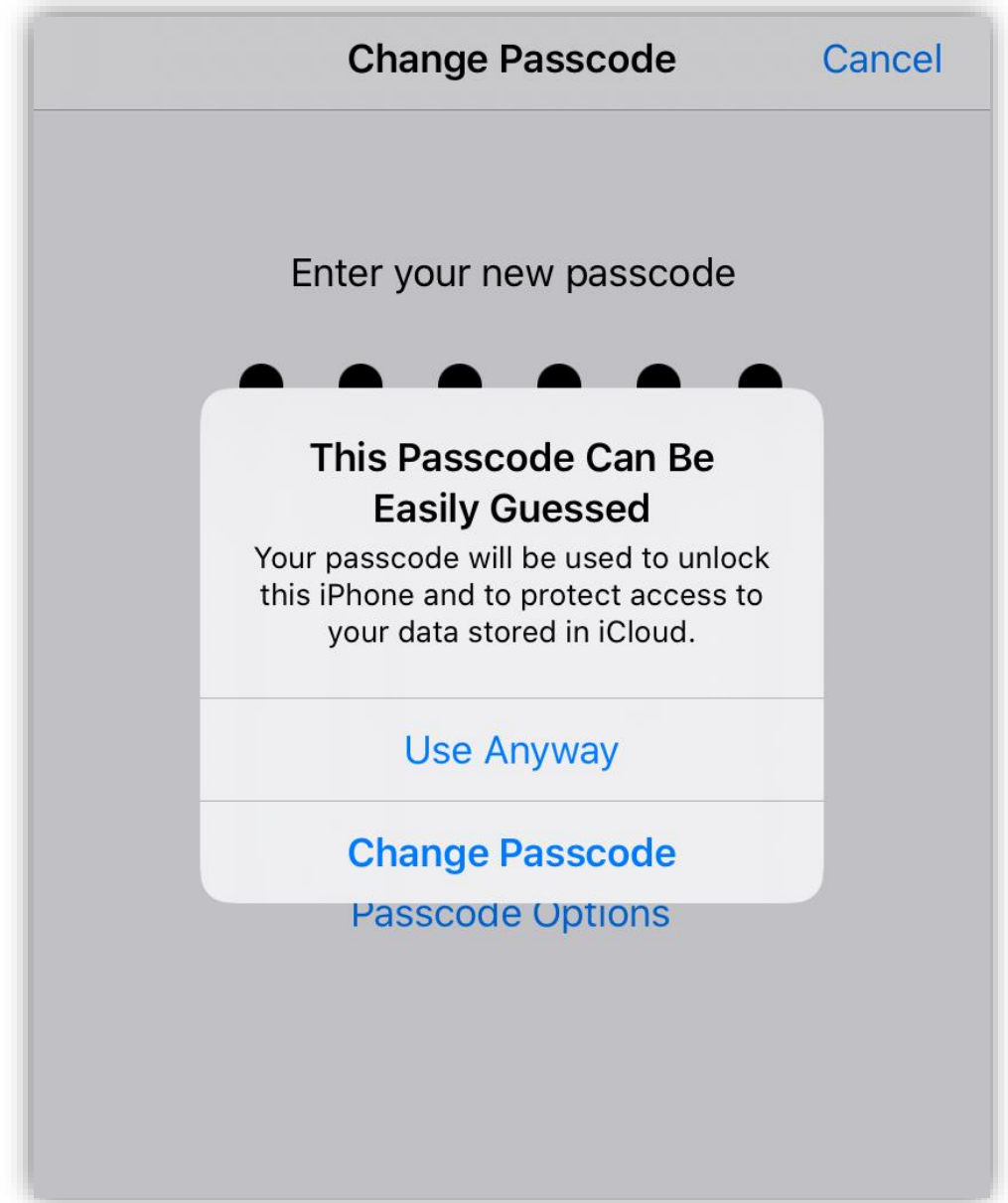
“123456”

or

“2580”

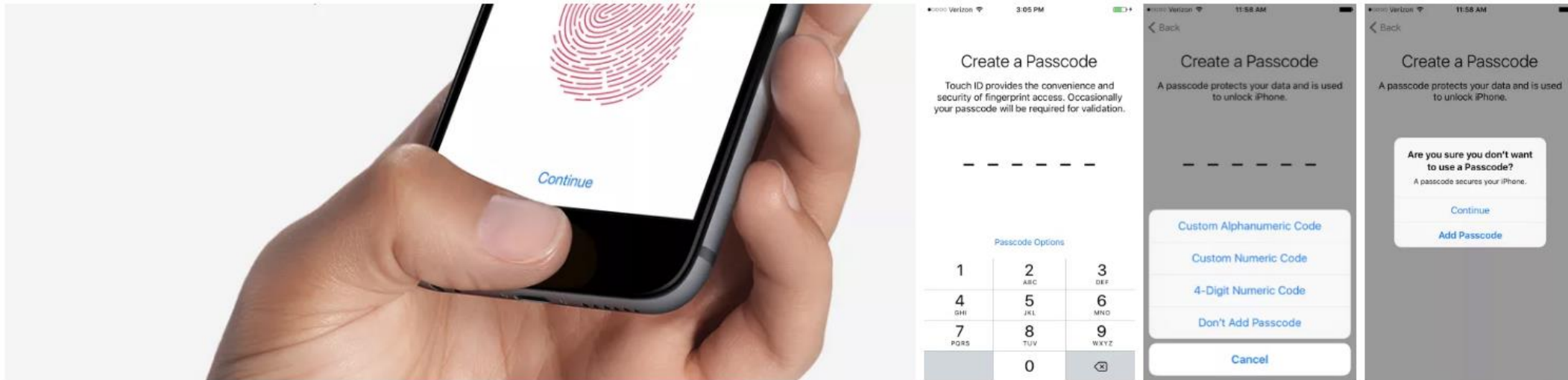
“1956”

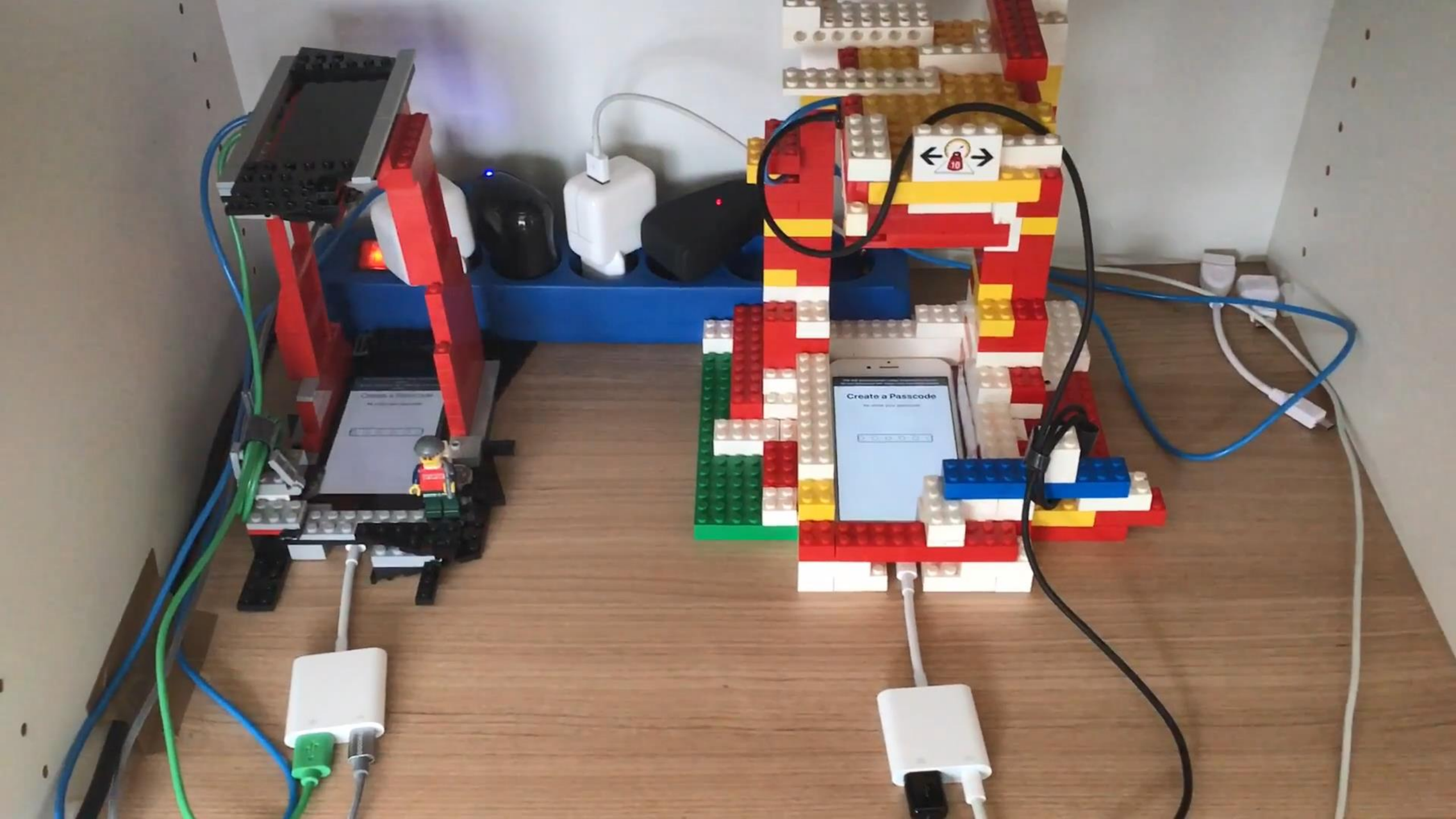
How to obtain the only “available” PIN blacklist on the market?



Documentation?

When creating a Passcode, the system is smart and does detect if you are wanting to use a simple, easy to guess, commonly used passcode and it has you confirm whether or not you want to do it. Examples of easy to guess, commonly used passcodes are 000000, 111111, 222222, 123456, etc. Before iOS 7, if you use a simple, easy to guess, commonly used passcode it did allow you to proceed and create the passcode without questioning you. Having a passcode is not required, and you can choose to not have a passcode by tapping on the Passcode Option. However, if you want to use [iCloud Keychain](#) and Touch ID, it is required to have a passcode.



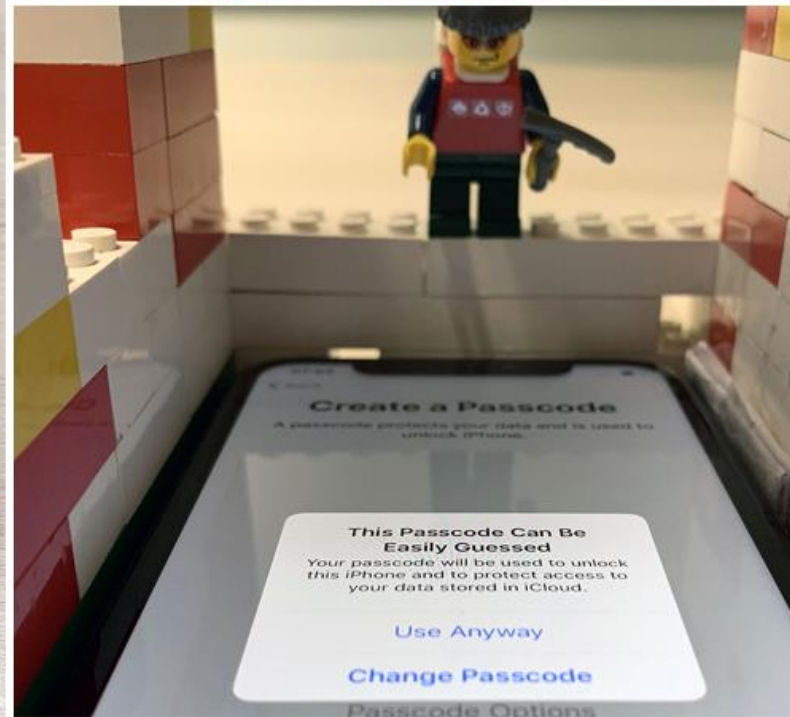
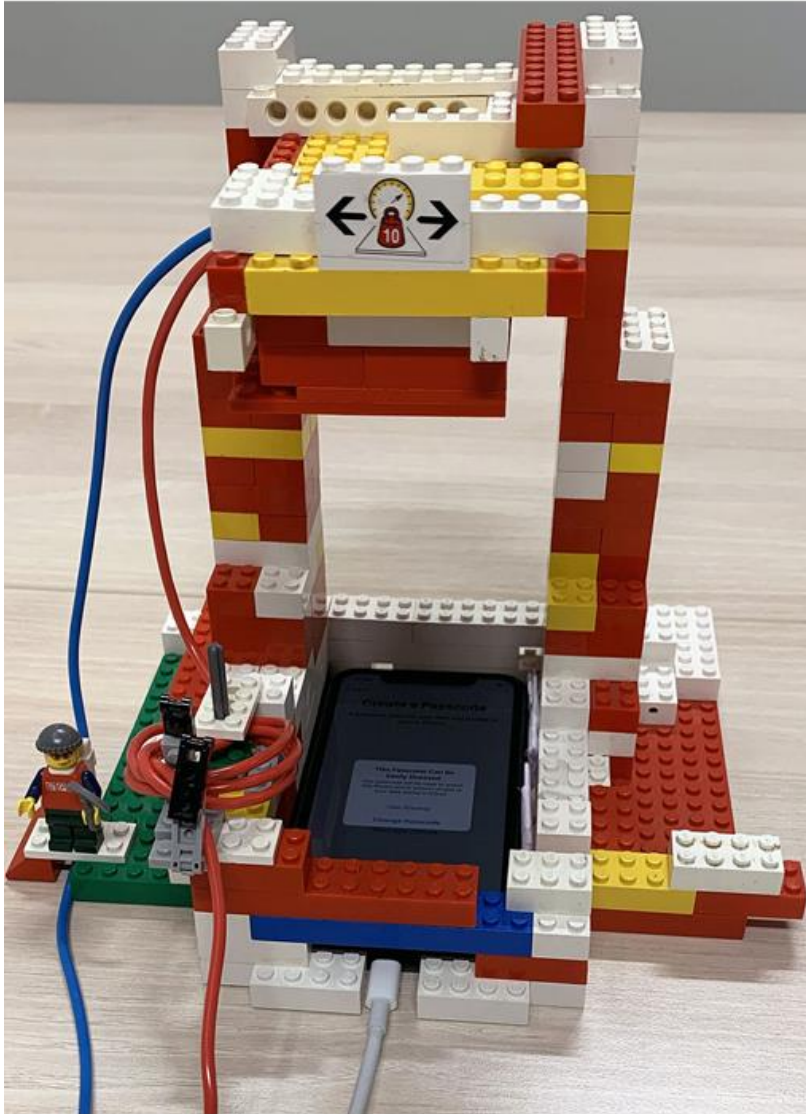


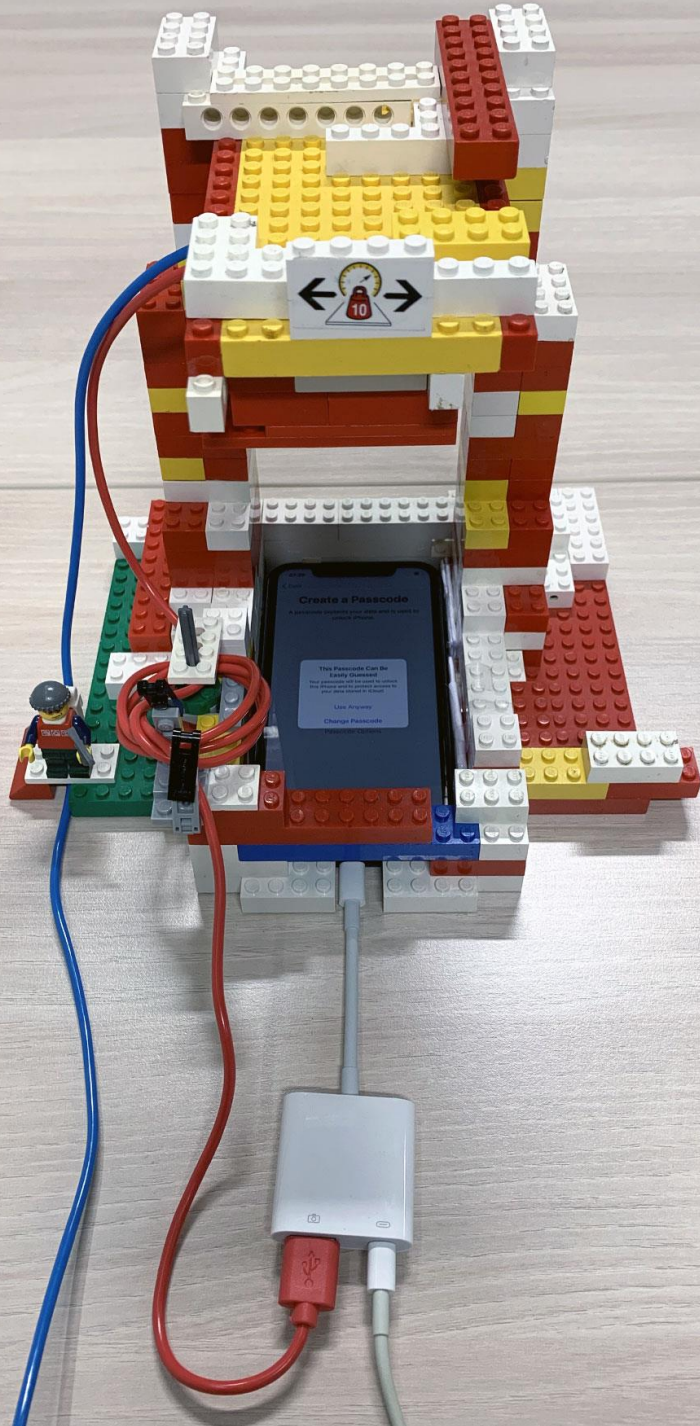


The iOS screen reader, called VoiceOver, is active.
To turn VoiceOver off, triple-click the Home button.

Create a Passcode

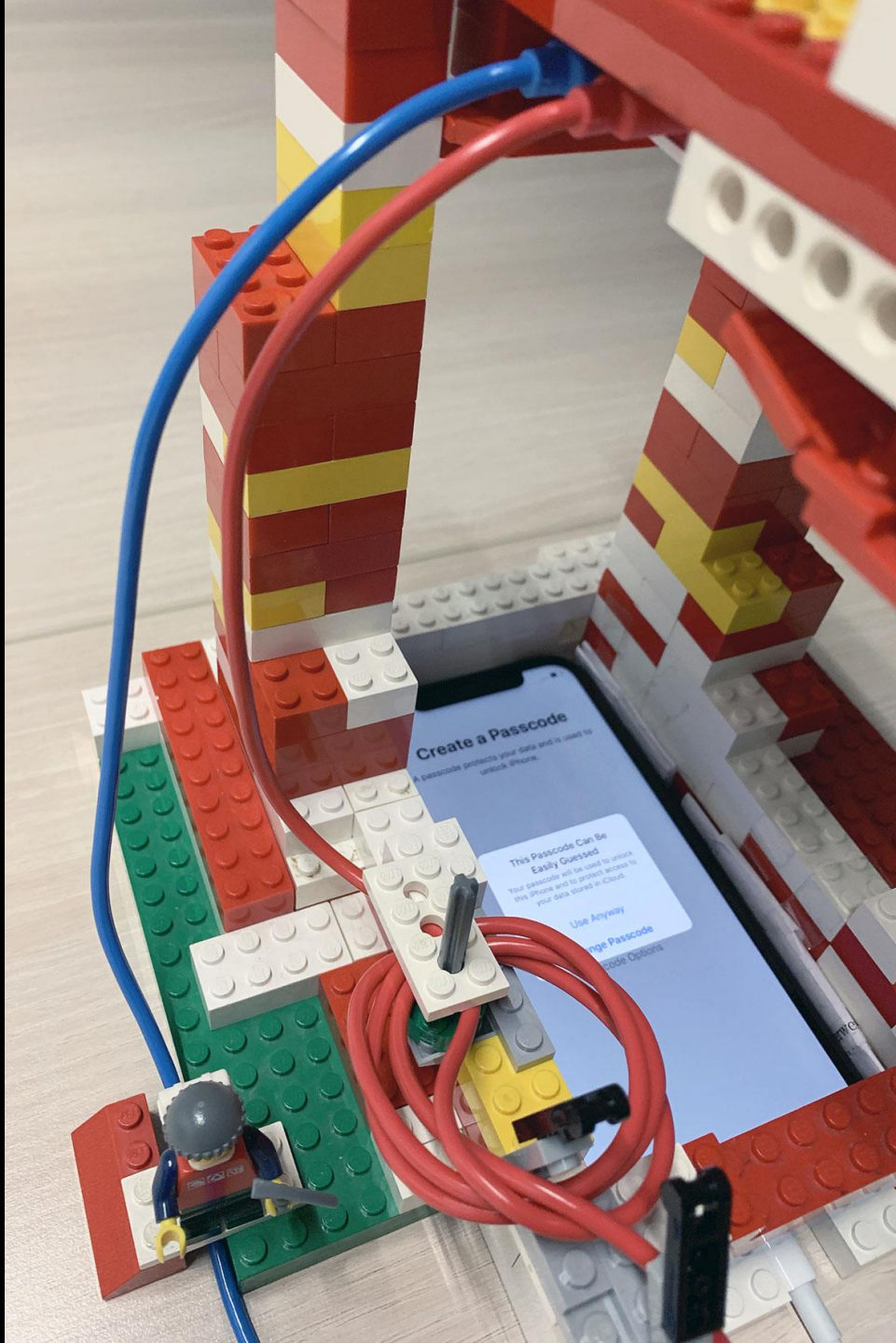
Interlocking Plastic Bricks Robot

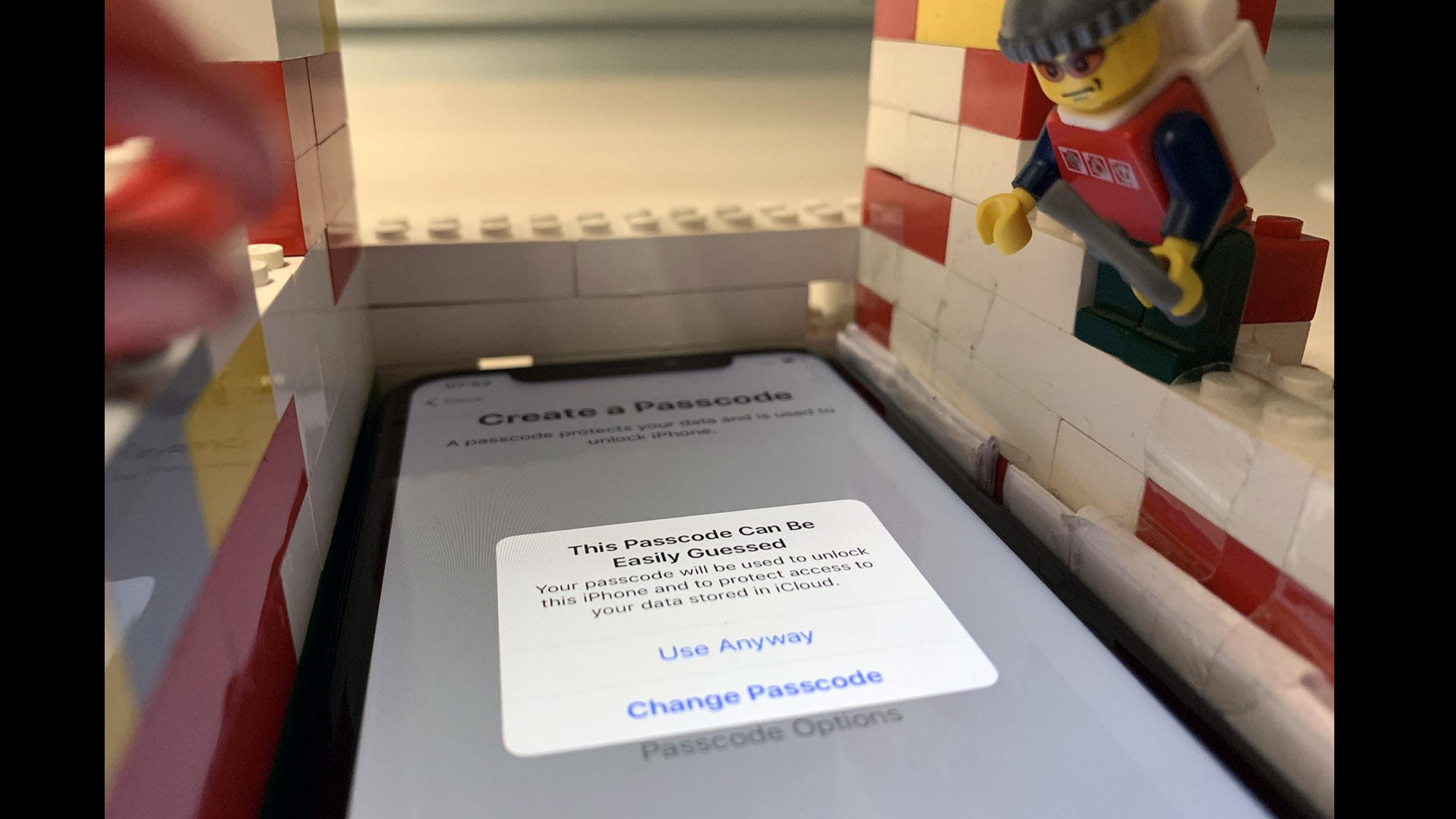












Create a Passcode

A passcode protects your data and is used to unlock iPhone.

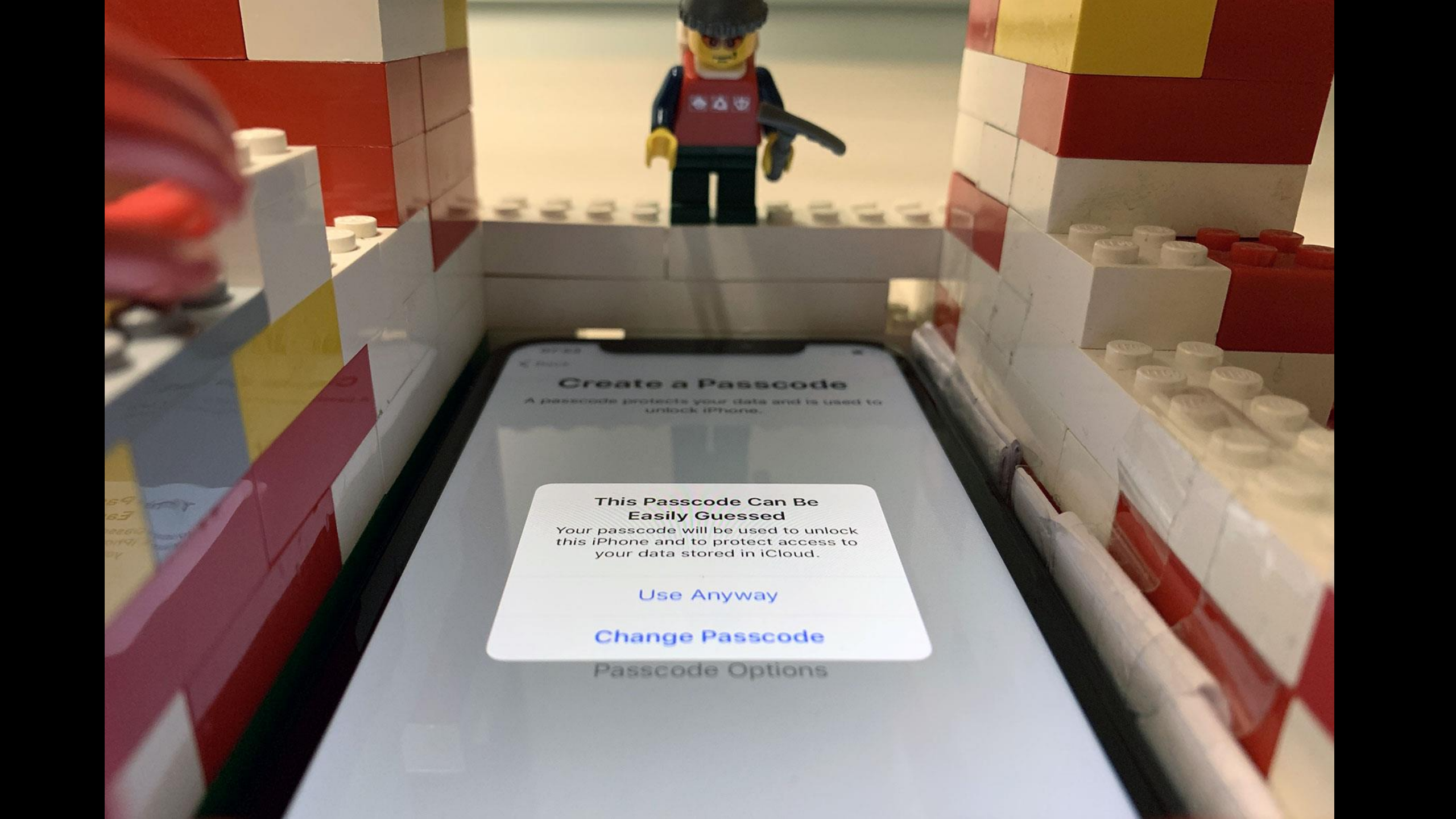
This Passcode Can Be Easily Guessed

Your passcode will be used to unlock this iPhone and to protect access to your data stored in iCloud.

Use Anyway

Change Passcode

Passcode Options



Create a Passcode

A passcode protects your data and is used to unlock iPhone.

This Passcode Can Be Easily Guessed

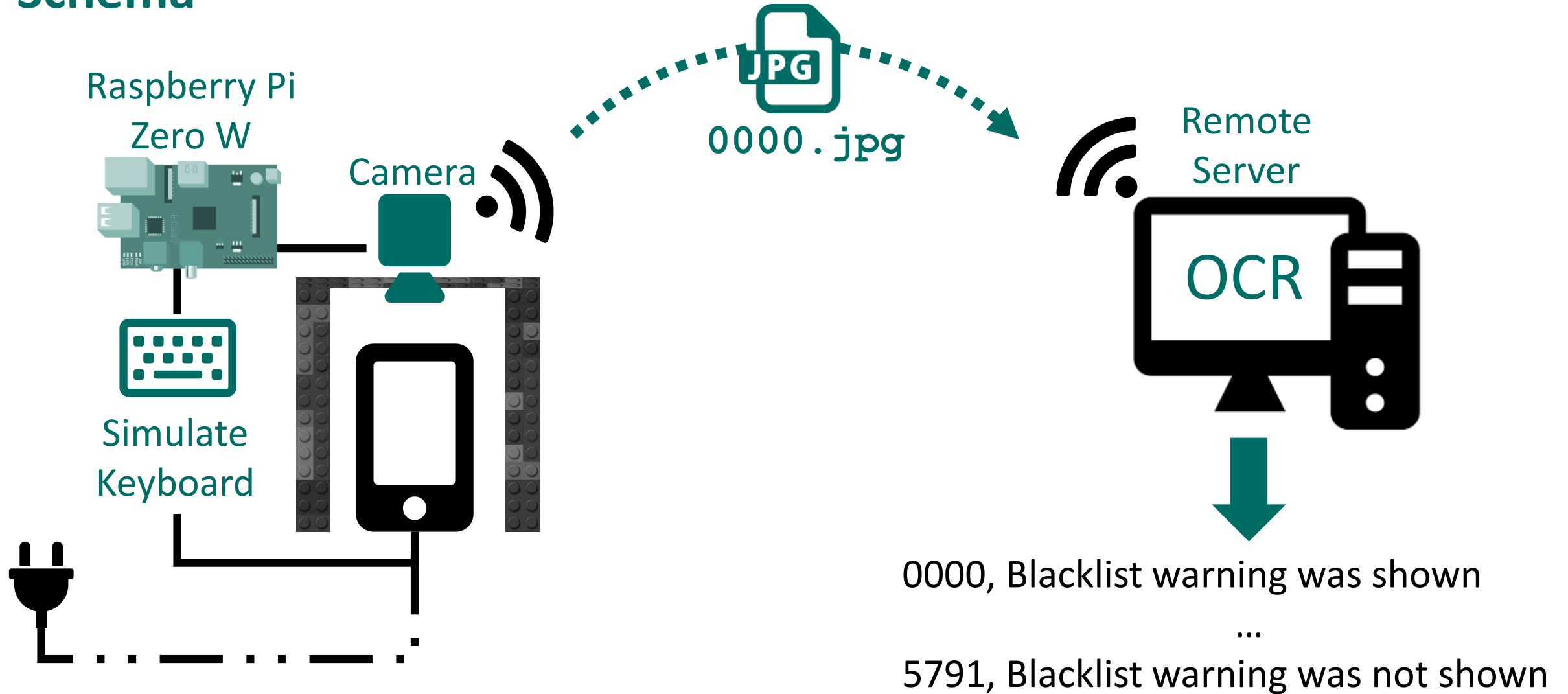
Your passcode will be used to unlock this iPhone and to protect access to your data stored in iCloud.

[Use Anyway](#)

[Change Passcode](#)

Passcode Options

Schema



Parts List (~\$100 + Phone)



- 1x Raspberry Pi Zero W – \$30
 - ✓ Case
 - ✓ Power supply
 - ✓ Micro SD card
- 1x Raspberry Pi Camera Module v2 – \$25
- 1x Micro USB cable – \$5
- 1x Lightning to USB 3 Camera Adapter – \$39
- 1x Apple iPhone 6s (or newer) – ~\$150
- Some interlocking plastic bricks



Turn a Raspberry Pi Zero W Into a Keyboard

The screenshot shows the GitHub repository page for `c4software/pi-as-keyboard`. The repository is described as "Make your Raspberry act as a Keyboard" and includes tags for `raspberry-pi`, `keyboard`, and `linux`. It has 34 commits, 1 branch, 0 packages, 0 releases, 2 contributors, and is licensed under Apache-2.0. The commit history shows updates to the LICENSE, README.md, and scripts for enabling the hid service and adding mouse/keyboard samples.

GitHub - c4software/pi-as-keyboard

github.com/c4software/pi-as-keyboard

Why GitHub? Enterprise Explore Marketplace Pricing Search Sign in Sign up

c4software / pi-as-keyboard

Watch 7 Star 78 Fork 22

Code Issues 3 Pull requests 0 Projects 0 Security Insights

Make your Raspberry act as a Keyboard

raspberry-pi keyboard linux

34 commits 1 branch 0 packages 0 releases 2 contributors Apache-2.0

Branch: master New pull request Find file Clone or download

c4software Update test.sh Latest commit 04776bf on Oct 25

LICENSE	Update LICENSE	3 years ago
README.md	Update README.md	last month
enable_hid.service	Restart service on failure.	4 months ago
enable_hid.sh	Add Mouse and Keyboard sample	4 months ago



Turn a Raspberry Pi Zero W Into a Keyboard

```
50 # OS descriptors
51 echo 1 > os_desc/use
52 echo 0xcd > os_desc/b_vendor
53 echo MSFT100 > os_desc/qw_sign
54 ln -s configs/c.1 os_desc
55
56 mkdir -p configs/c.1/strings/0x
57 echo "Config 1: Keyboard" > con
58 echo 100 > configs/c.1/MaxPower
59 ls /sys/class/udc > UDC
```



~~250~~ -> 100 (milliwatts)



Idea

Test all possible PINs, e.g., via a keyboard

Problems:

Keyboard navigation

Rate-Limiting

Battery



How to navigate in iOS using a keyboard?



Idea

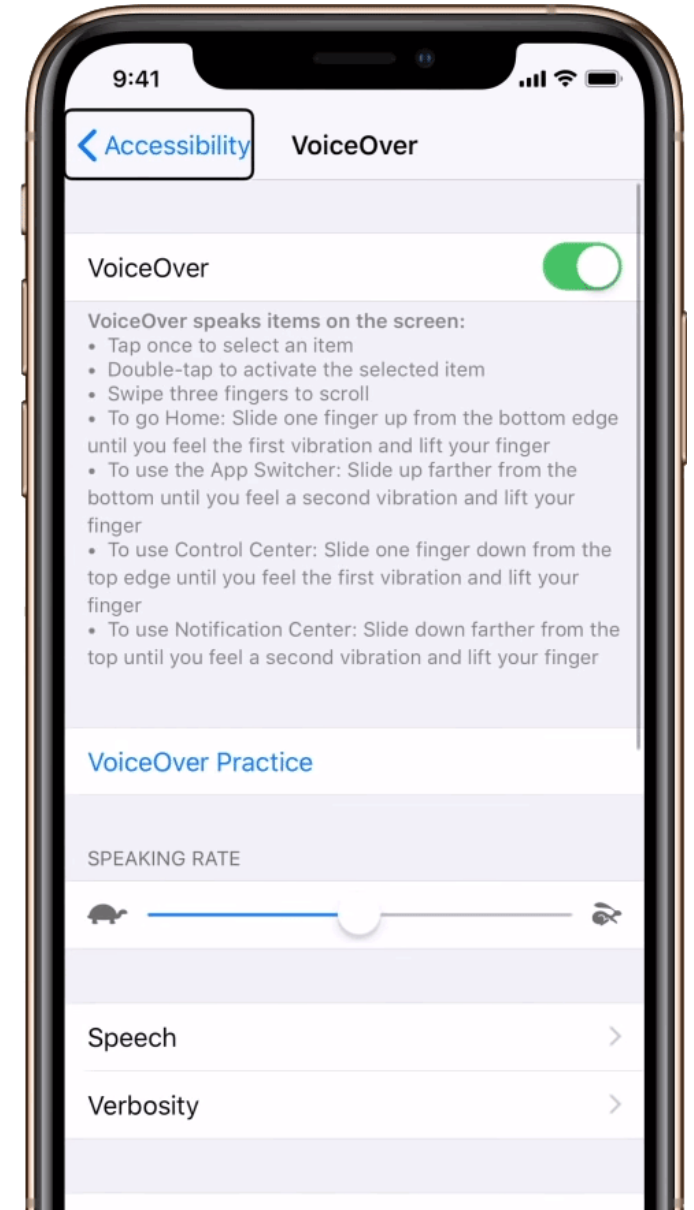
Test all possible PINs, e.g., via a keyboard

Problems:

Keyboard navigation -> **Activate “VoiceOver”**

Rate-Limiting

Battery



Idea

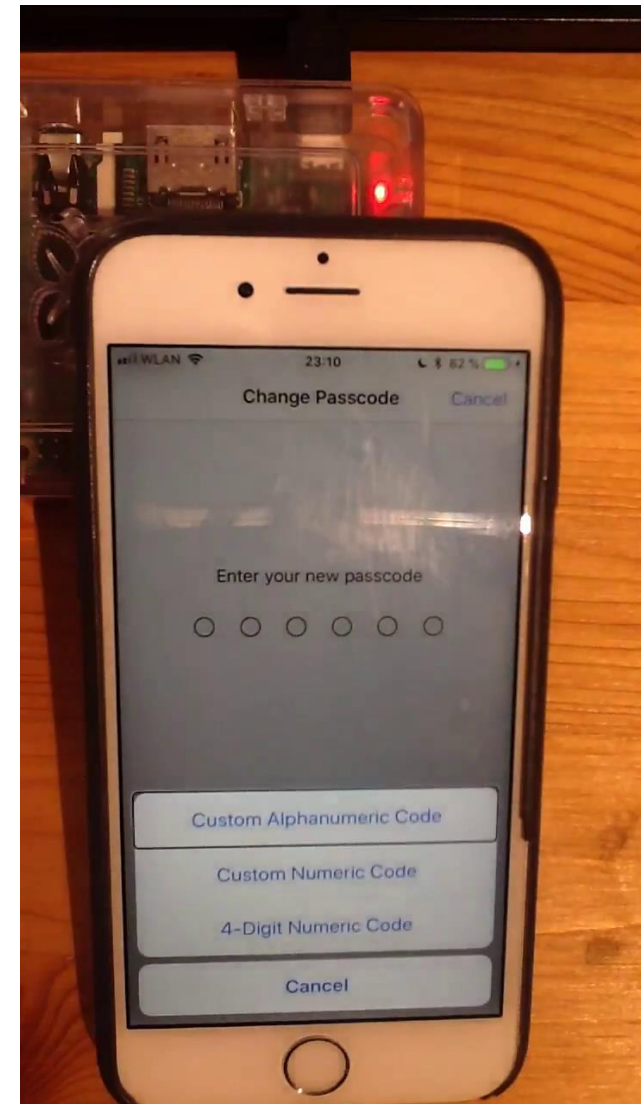
Test all possible PINs, e.g., via a keyboard

Problems:

Keyboard navigation

Rate-Limiting

Battery



Bluetooth Prototype



Idea

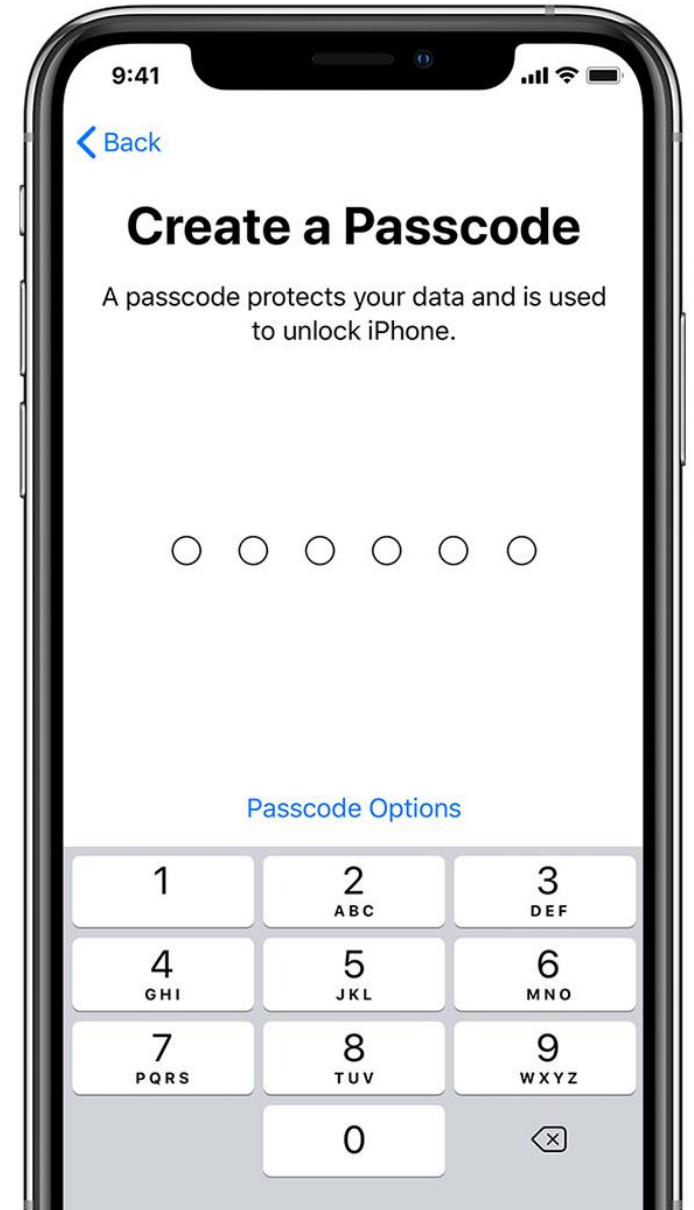
Test all possible PINs, e.g., via a keyboard

Problems:

Keyboard navigation

Rate-Limiting -> Exploit initial setup

Battery



Idea

Test all possible PINs, e.g., via a keyboard

Problems:

Keyboard navigation

Rate-Limiting

Battery



iPad 2 Prototype





Drew Breunig

@dbreunig

 Follow

Apple's fastest growing product category.



Idea

Test all possible PINs, e.g., via a keyboard

Problems:

- Keyboard navigation

- Rate-Limiting

- Battery** -> Use “Lightning to USB 3 Camera Adapter”



Enter PIN + Take a Photo (Raspi Part)

```
Bash
for pin in pins:
    for digit in pin:                # Enter a PIN
        send_key(digit)
    time.sleep()

    take_picture(camera_pid, pin) # Tell Pi to take a photo

    discard_warning()                # Navigate down
    time.sleep()

    enter_stop_pin()                 # Intentionally fail to
    time.sleep()                     # re-enter the PIN
```



Optical Character Recognition (Server Part)

```
Bash
import pytesseract                # Tesseract OCR Engine
import cv2                        # OpenCV (Computer Vision)

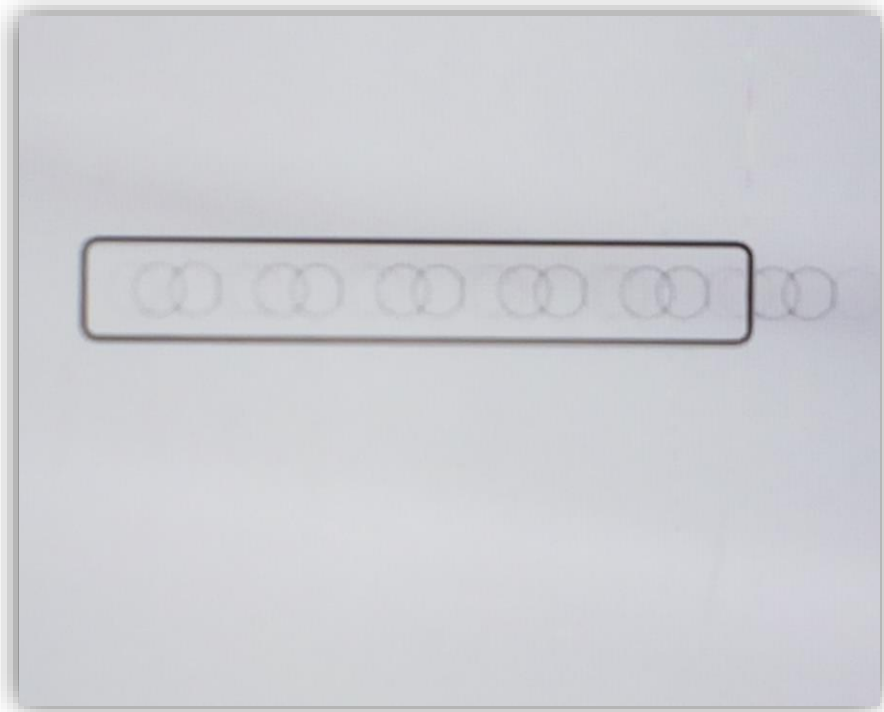
for file in files:
    cv2.imread()                  # Read img. from disk
    cv2.threshold()               # Convert img. to grayscale
    pytesseract.image_to_string() # Convert img. to text

    if len(text) > 14:
        output = "PIN is blacklisted"
    else:
        output = "No warning was shown"
```

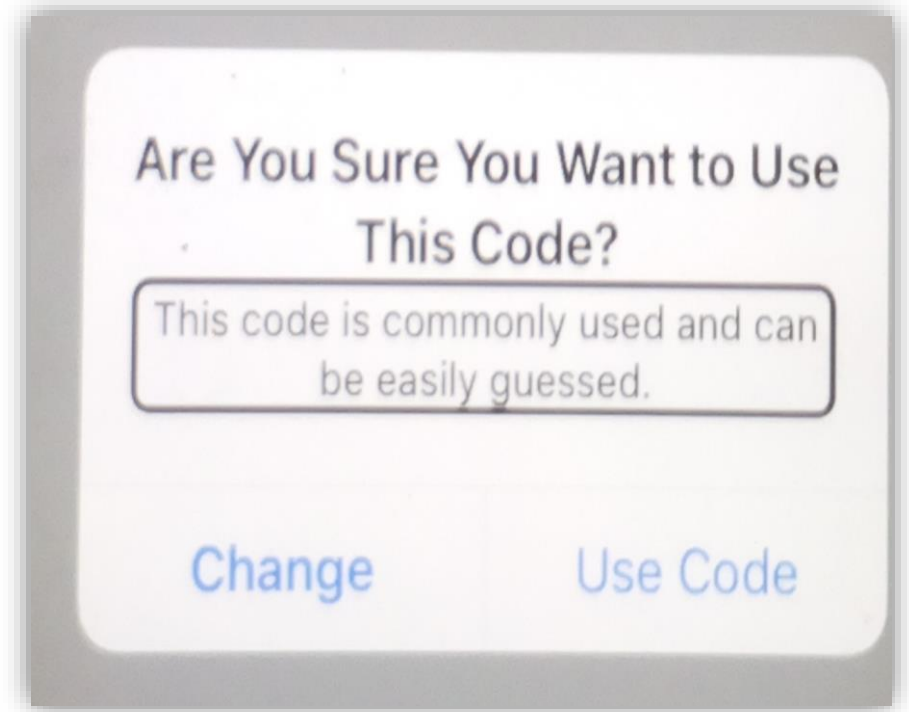


Raspberry Pi Camera Module v2 (8MP)

```
raspistill -hf -vf -roi 0.4,0.37,0.24,0.24 --width 1280 --height 1024  
--nopreview --quality 25 --timeout 0 --signal -o /dev/shm/%06d.jpg
```



999000.jpg

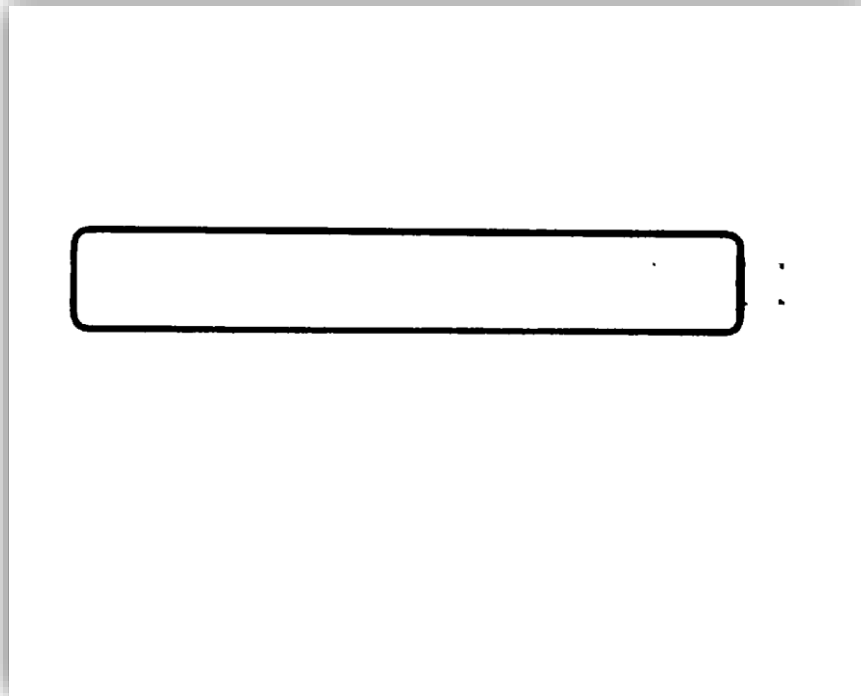


159753.jpg

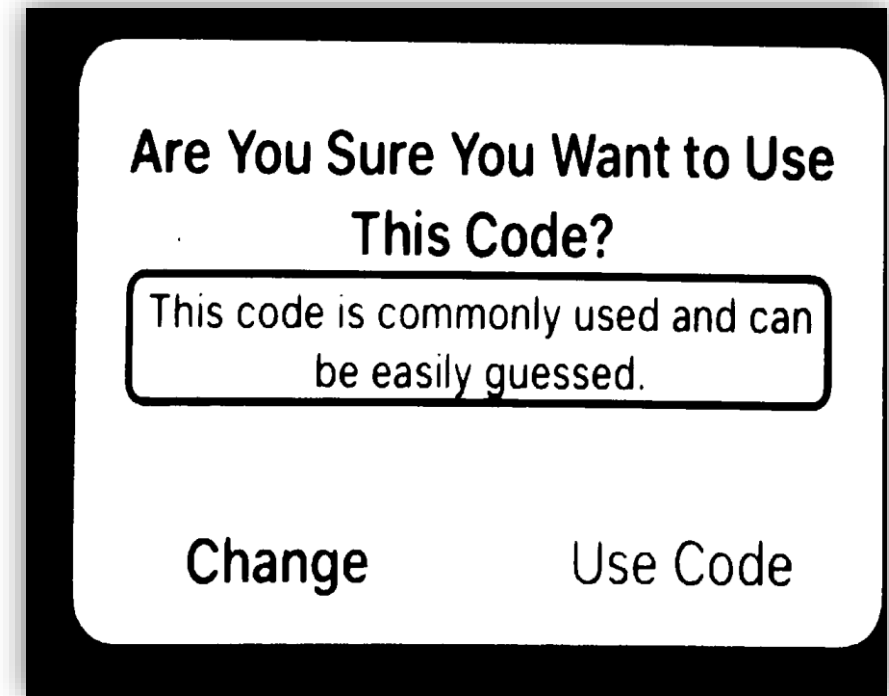


OpenCV – RGB ↔ GRAY and Thresholding

```
cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)  
cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU) [1]
```



999000.jpg



159753.jpg

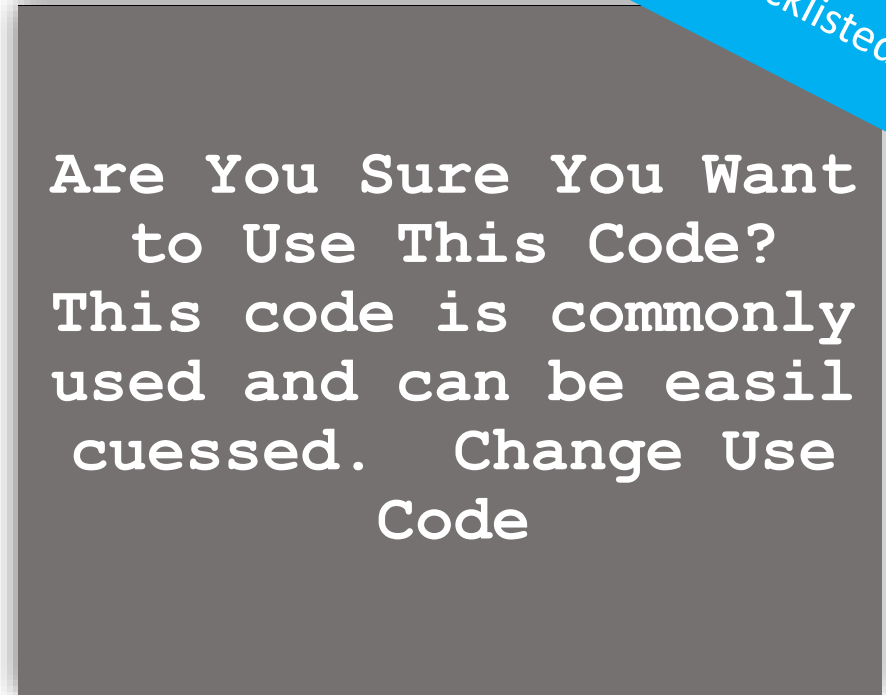


Tesseract Open Source OCR Engine

```
pytesseract.image_to_string(Image.open(filename))
```



999000.jpg



159753.jpg



The iOS Passcode Blacklist

4-digit Passcodes:

Search: ~9h (1x Setup)

Key space: 10,000 PINs

Blacklisted: 274 PINs (2.74%)

- **Common PINs:** Bonneau et al. [1]
- **Years:** 1956-2015
- **Patterns:**
 - ✓ aaaa
 - ✓ abab
 - ✓ aabb

6-digit Passcodes:

Search: ~30 days (2x Setups)

Key space: 1,000,000 PINs

Blacklisted: 2910 PINs (0.29%)

- **Common PINs:** Wang et al. [2]
- **Ascending/Descending:** “543210”
- **Patterns:**
 - ✓ aaaaaa
 - ✓ abcabc
 - ✓ abccba



Lessons Learned

- Be prepared to reset the devices
- iOS 9.3.5 blacklist != iOS 10.3.3 blacklist
- Mute the phone, because it is called “VoiceOver”!
- We reduced the brightness to a minimum, still, 2x iPhones were harmed in the process



Responsible Disclosure

Subject: ...

From: Maximilian Golla <maximilian.golla@rub.de>

Date: 04.06.19, 15:13

To: product-security@apple.com

Hello Apple Product Security,

We would like to responsibly disclose some vulnerabilities we discovered while conducting research into iOS numeric 4- and 6-digit Passcodes.

First, during the initial setup of an iOS device where a user can create a Passcode, we found that there is **no rate-limiting in place**. While the initial setup phase has no data worth protecting, this flaw allows a user to enumerate all Passcodes that are deemed not strong enough, or "blacklisted" by iOS 9-12, and we did so for 4- and 6-digit numeric Passcodes for the purposes of academic research. We enumerated the blacklist by detecting when iOS prompts the user with the warning message:



Great Robot, But Isn't There an Easier Way?

The collage illustrates a complex task in reverse engineering, likely related to Apple Pay. It includes several components:

- Terminal Window:** Shows repetitive commands like `ch file or directory` and `such file or directory`.
- File Browser:** Displays a directory tree with folders like `Library`, `Backup`, `BulletinBoardPlugins`, `CacheDelete`, `Caches`, `Carrier Bundles`, `ColorSync`, `ControlCenter`, and `CoreAccessories`.
- Code Editor:** Contains Python code for PIN validation. The code defines a list of common PINs and a function `check_pin` that checks if a given PIN is in the common list.
- Yellow Box:** A large yellow box highlights a list of Apple Pay issuer IDs. The list includes `123123`, `123321`, `123654`, `147258`, `159753`, `321654`, and `520131`. The ID `159753` is highlighted in green.
- Debugger Window:** Shows a crash report with the message `3746: pcsc-pcs-com.apple.issuers-m_EmailAddressIfPresent-m_SubjectContains-123123-123321-123654-147258-159753-321654-520131-ApplePayIssuerEncryption-AppleTVVPNProfileS`. The ID `159753` is highlighted in green. Below the crash report, a table lists frameworks and their versions.

Type	Value
Signed Byte	49
Unsigned Byte	49
Signed Short	13617
Unsigned Short	13617



Extracting the “Common” Passcodes Directly From the IPSW

1. Download and install Malus-Security/**iExtractor** by Răzvan Deaconescu
(A tool for macOS to automate the extraction of data from iOS firmware files.)
2. Download and decrypt the latest iOS version using iExtractor
3. Use the following code to extract the PINs directly from the dyld_shared_cache:

```
Bash
VERSION="13.3.1_17D50"
CODENAME="YukonD17D50.D22D221OS"
FILE="dyld_shared_cache_arm64" # or "dyld_shared_cache_armv7s" for iOS 7 to 10.3

hdiutil attach decrypted.dmg

strings /Volumes/$CODENAME/System/Library/Caches/$FILE | \
grep "\bSecPasswordSeparator\b" -A 120 > blacklist_iOS_$VERSION.txt

hdiutil unmount $CODENAME
```

