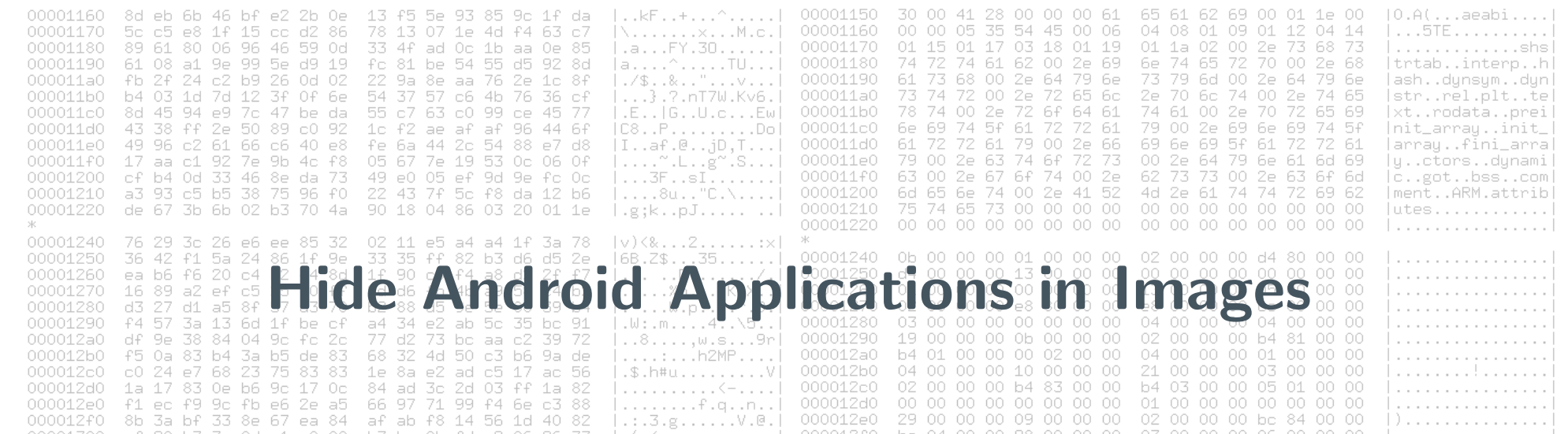


Hide Android Applications in Images

Axelle Apvrille - FortiGuard Labs, Fortinet
Ange Albertini, Corkami

BlackHat Europe, Amsterdam, NH
October 2014

FORTINET®

Agenda

- What is this all about? (quick)
- Who are we? (quick)
- Demo
- Details!

What is this all about?

Read the title! ;)

What is this all about?

Read the title! ;)

Hiding

What is this all about?

Read the title! ;)

Hiding Android Applications

What is this all about?

Read the title! ;)

Hiding Android Applications

in ...

What is this all about?

Read the title! ;)

Hiding Android Applications

in ... images

Who are we?

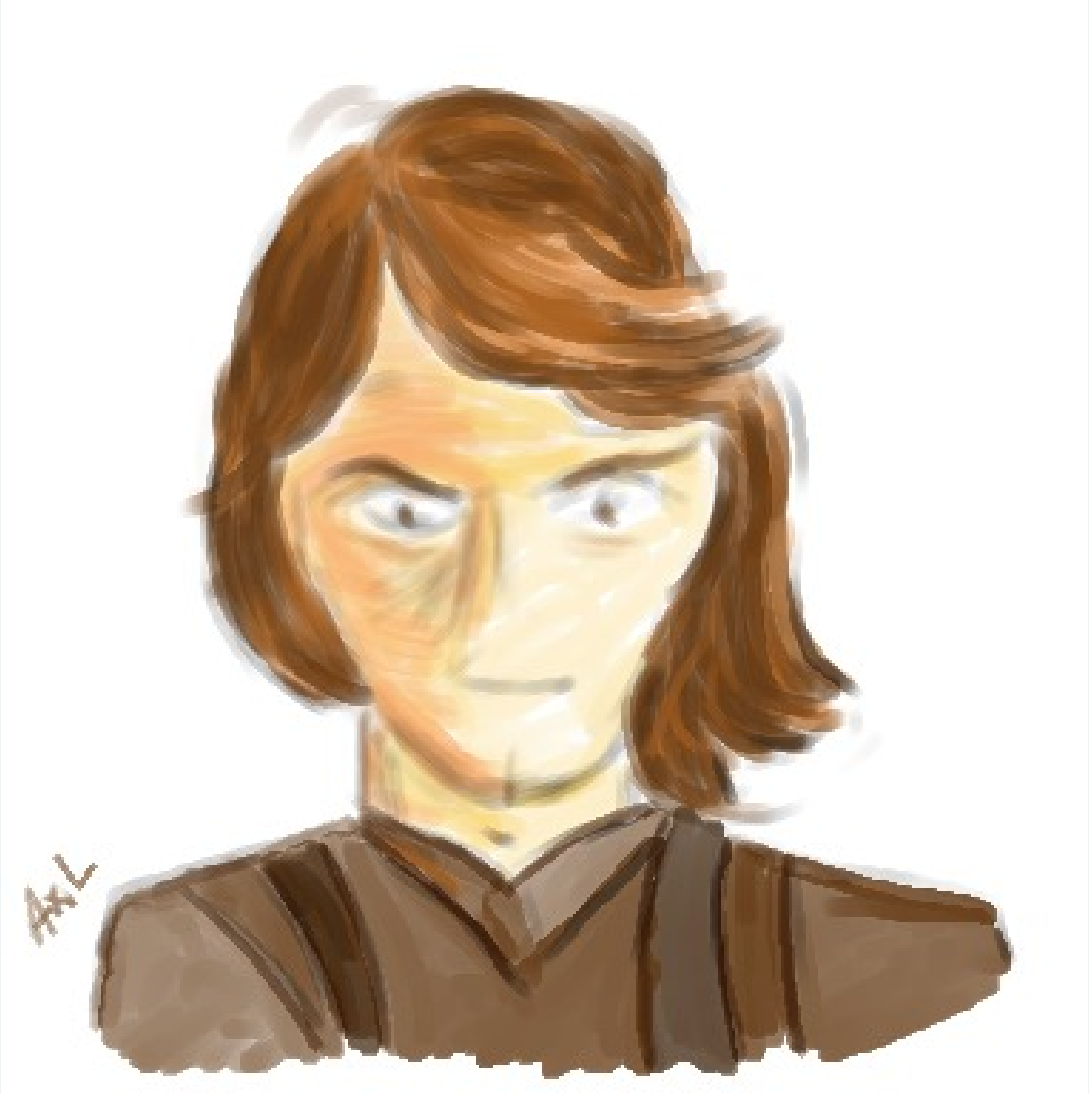
Axelle

```
axelle = {
  ‘‘realname’’ : ‘‘Axelle Apvrille’’,
  ‘‘job’’ : ‘‘Mobile/IoT Malware Analyst and Research’’,
  ‘‘company’’ : ‘‘Fortinet, FortiGuard Labs’’ }
```

Ange

```
ange = {
  ‘‘realname’’ : ‘‘Ange Albertini’’,
  ‘‘hobby’’ : ‘‘Corkami’’ }
```

What is this?



Nice? Thanks that's GIMP art from me ;)

It's an image!

file says...

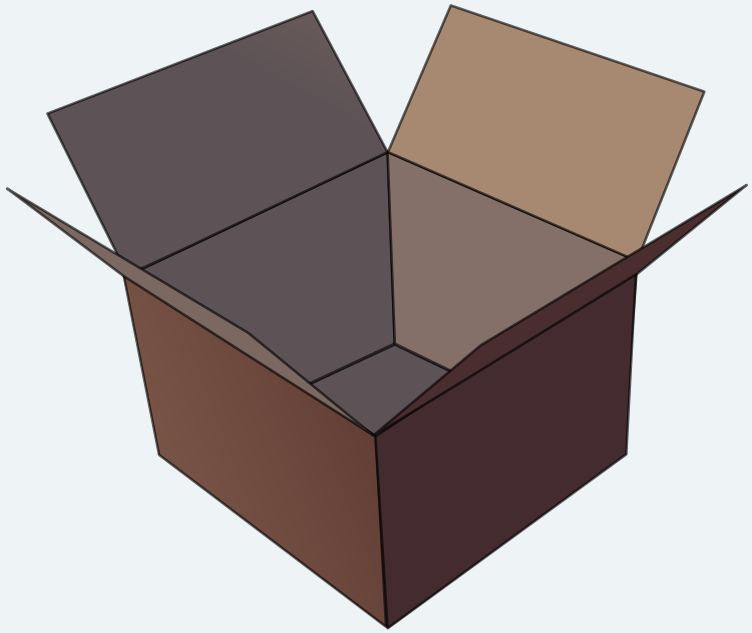
```
anakin.png: PNG image data, 636298042 x 1384184774, 19-bit
```

PNG file format

```
89 50 4e 47 0d 0a 1a 0a  00 01 b4 40 61 61 61 61  |.PNG.....@a
25 ed 23 3a 52 80 fb c6  13 cc 54 4d 74 f5 78 87  |%.#:R.....TMt
ba 7d b5 f6 93 63 43 f0  e0 b9 99 9b 37 06 cc 8f  |.}...cC.....7
32 59 5b 55 da 14 e2 87  68 f7 89 e5 88 14 fe 76  |2Y[U....h....
3e 0b cd 65 ec c4 7a 71  4d 95 c0 4e de 48 30 91  |>..e..zqM..N.
...
```

It is more than that!

AES Decrypt



Valid PNG

Valid Android Package (APK)

Embed this "PNG" in an Android app?

Imagine...

...if that PNG/APK is malicious!

- (Nearly) invisible to reverse engineering!
- The Android app is **encrypted**

Arg! What will I see?

- A **fat** image
- The wrapping application
  - Code that decrypts an asset
  - Code that loads/installs an application

But that depends how well the wrapping app is written

It can be *obfuscated*...

Demo

Party time!
Demo!
Wake up!

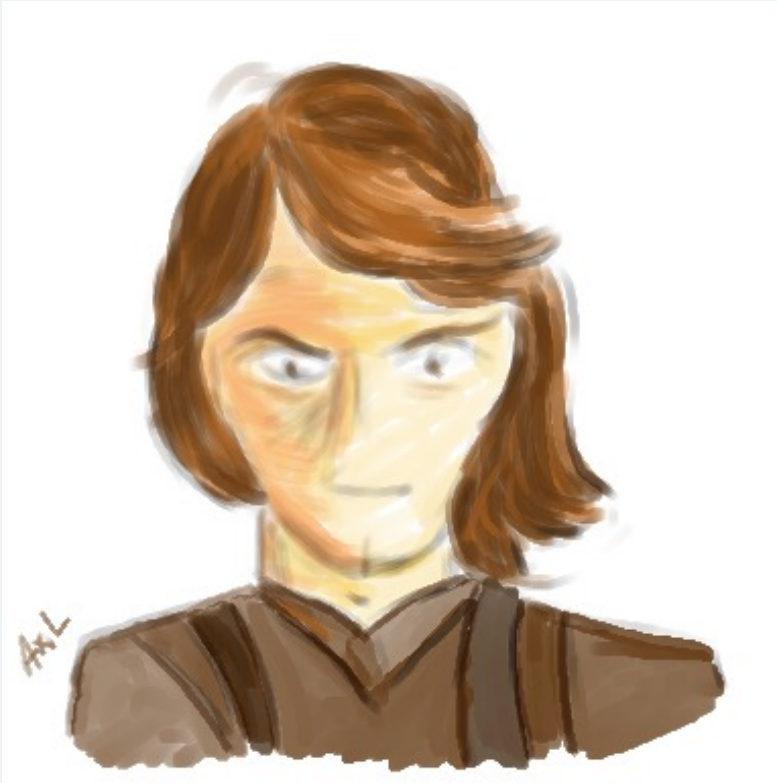
In case the demo crashes - lol

The APK looks genuine

```
Archive:  PocActivity-debug.apk
  Length      Date    Time    Name
---------  ---------- -----   ----
   508720  2014-09-11 13:41   assets/anakin.png
     1272  2014-09-11 14:03   res/layout/main.xml
     1988  2014-09-11 14:03   AndroidManifest.xml
     1444  2014-09-11 14:03   resources.arsc
     7515  2014-09-11 14:03   res/drawable-hdpi/logo.png
     2455  2014-09-11 14:03   res/drawable-ldpi/logo.png
     4471  2014-09-11 14:03   res/drawable-mdpi/logo.png
     8856  2014-09-11 14:03   classes.dex
      634  2014-09-11 14:03   META-INF/MANIFEST.MF
      687  2014-09-11 14:03   META-INF/CERT.SF
      776  2014-09-11 14:03   META-INF/CERT.RSA
---------                     -------
   538818                     11 files
```

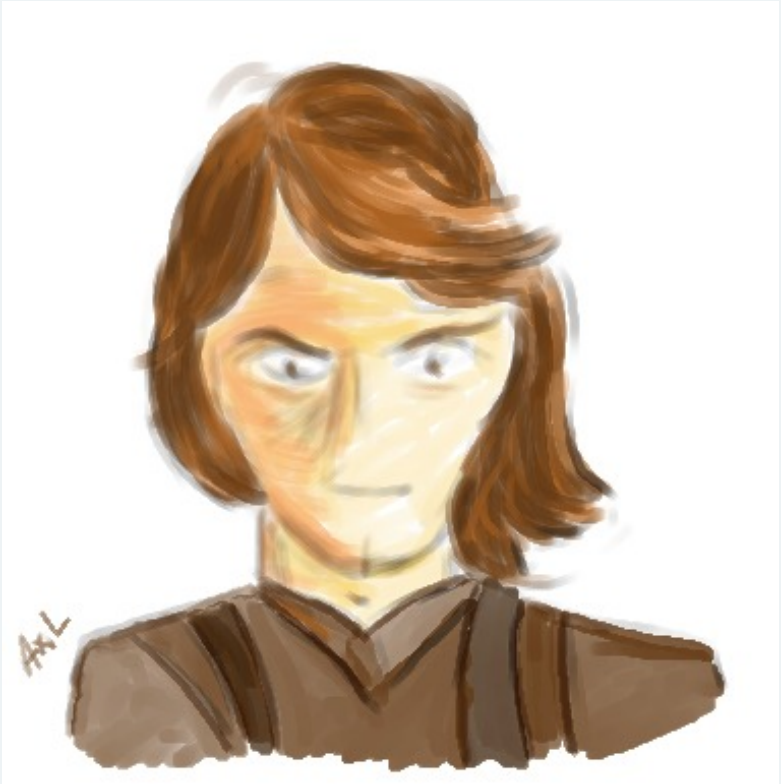
In case the demo crashes - lol

The image looks genuine: assets/anakin.png



In case the demo crashes - lol

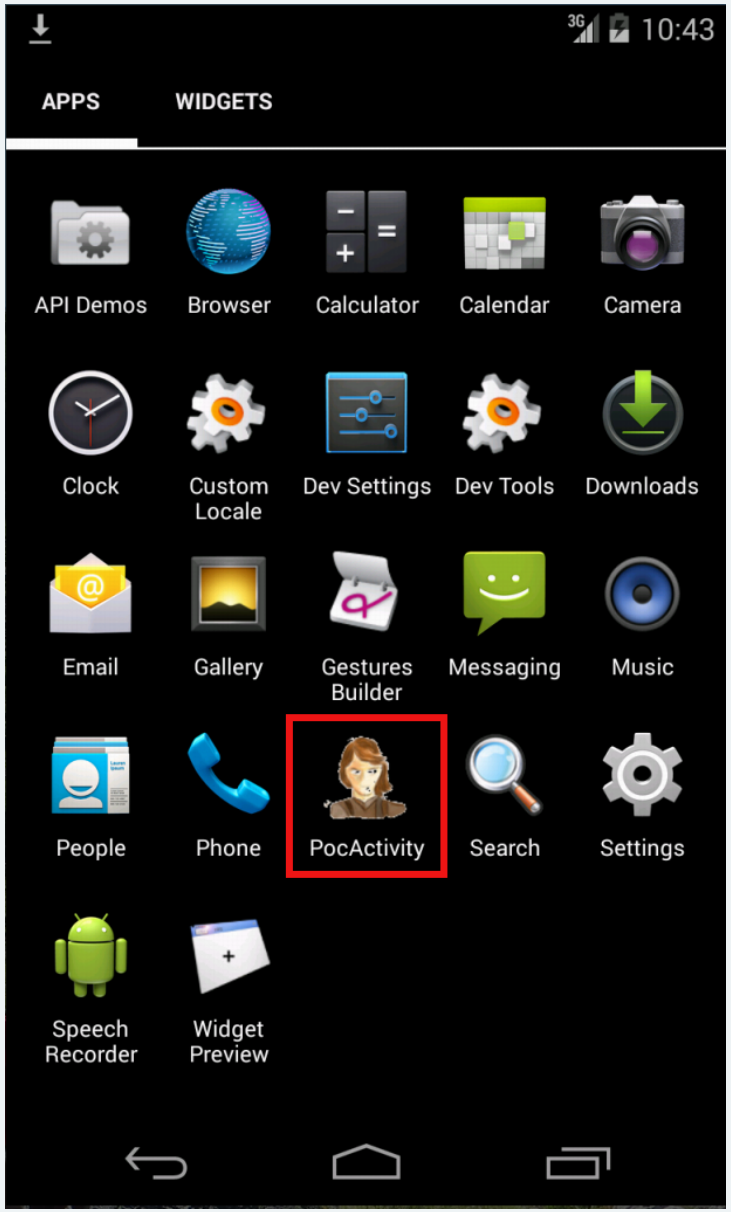
The image looks genuine: assets/anakin.png



Perhaps a bit 'fat'

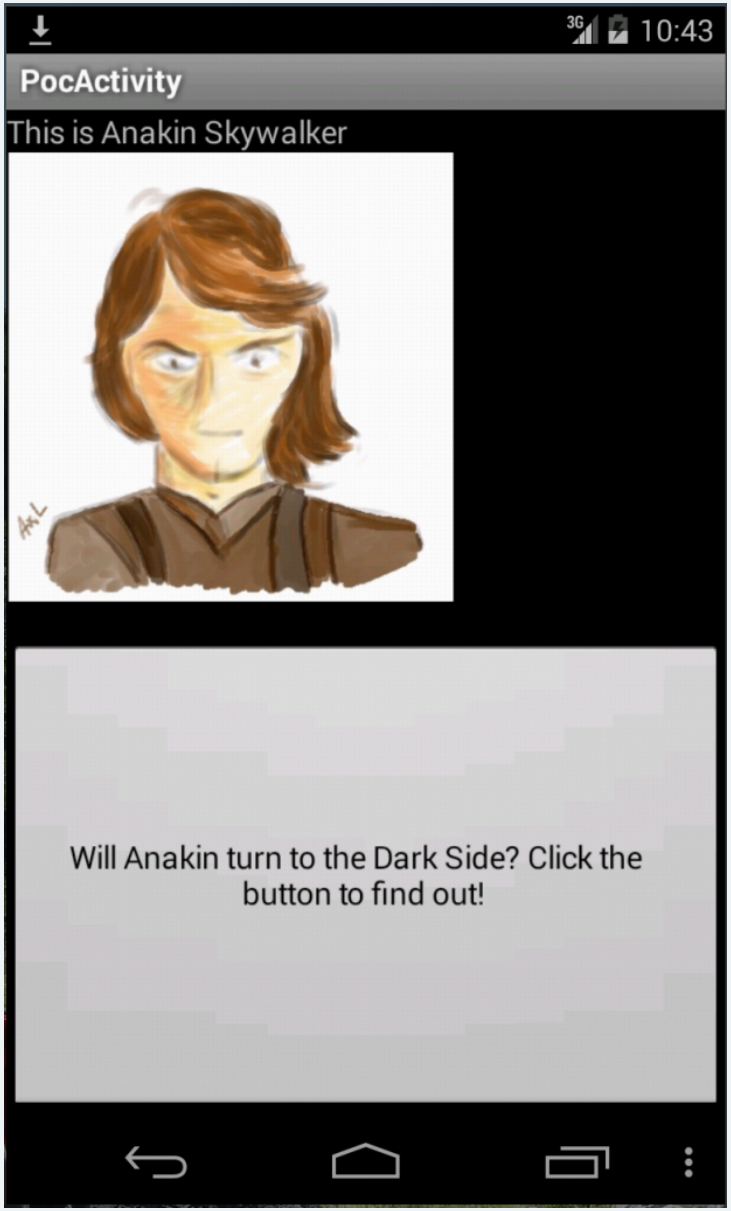
508720 bytes ($\approx$ 500K) for 382x385 pixels

In case the demo crashes - lol

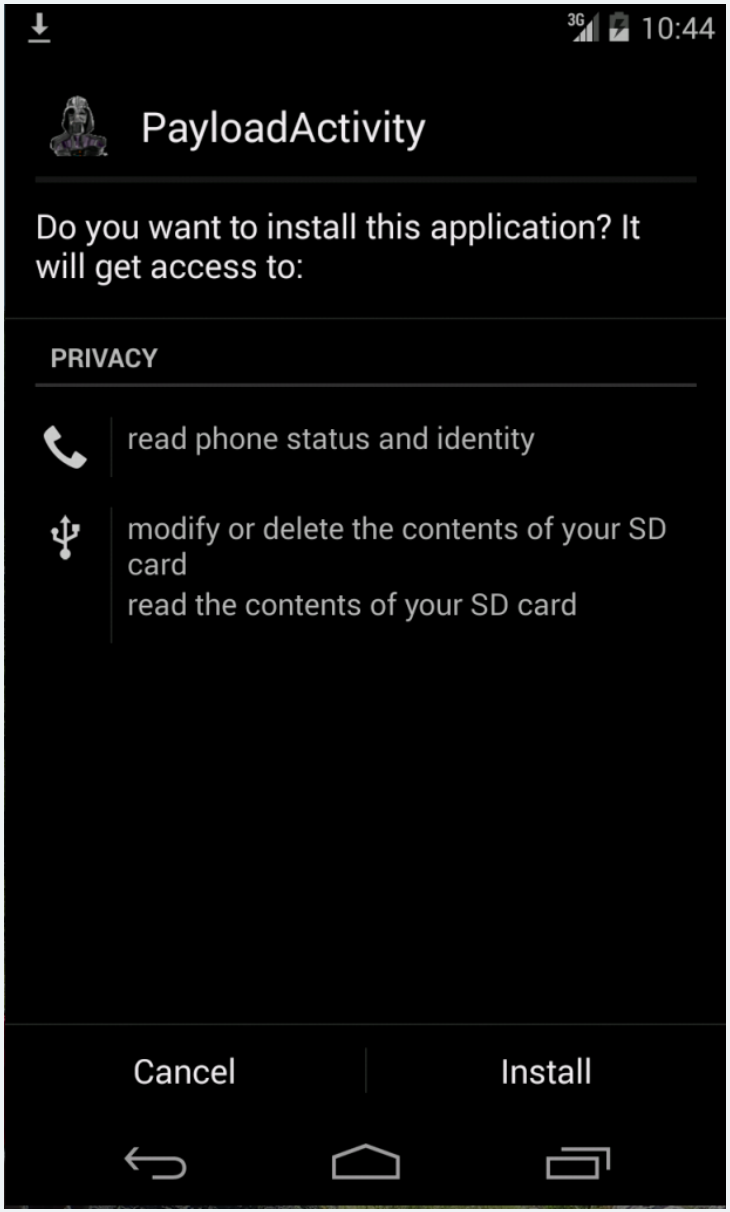



```
adb install
WrappingApk.apk
```

In case the demo crashes - lol

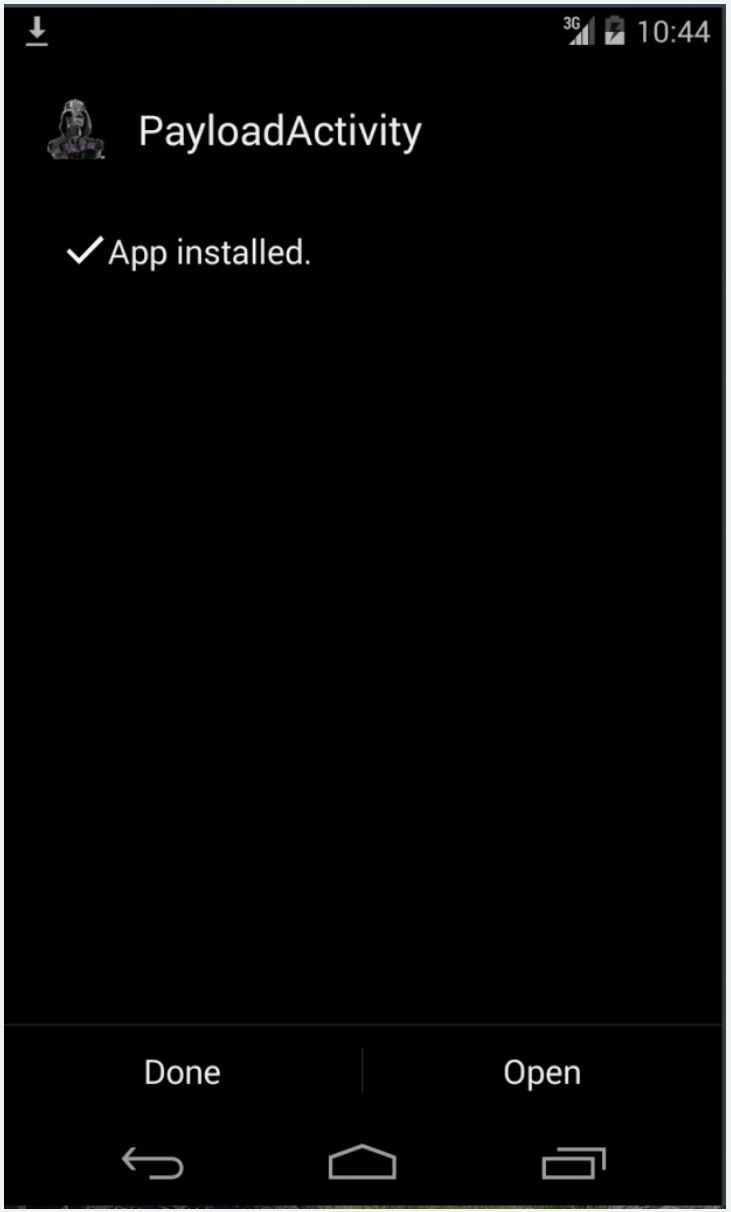


In case the demo crashes - lol



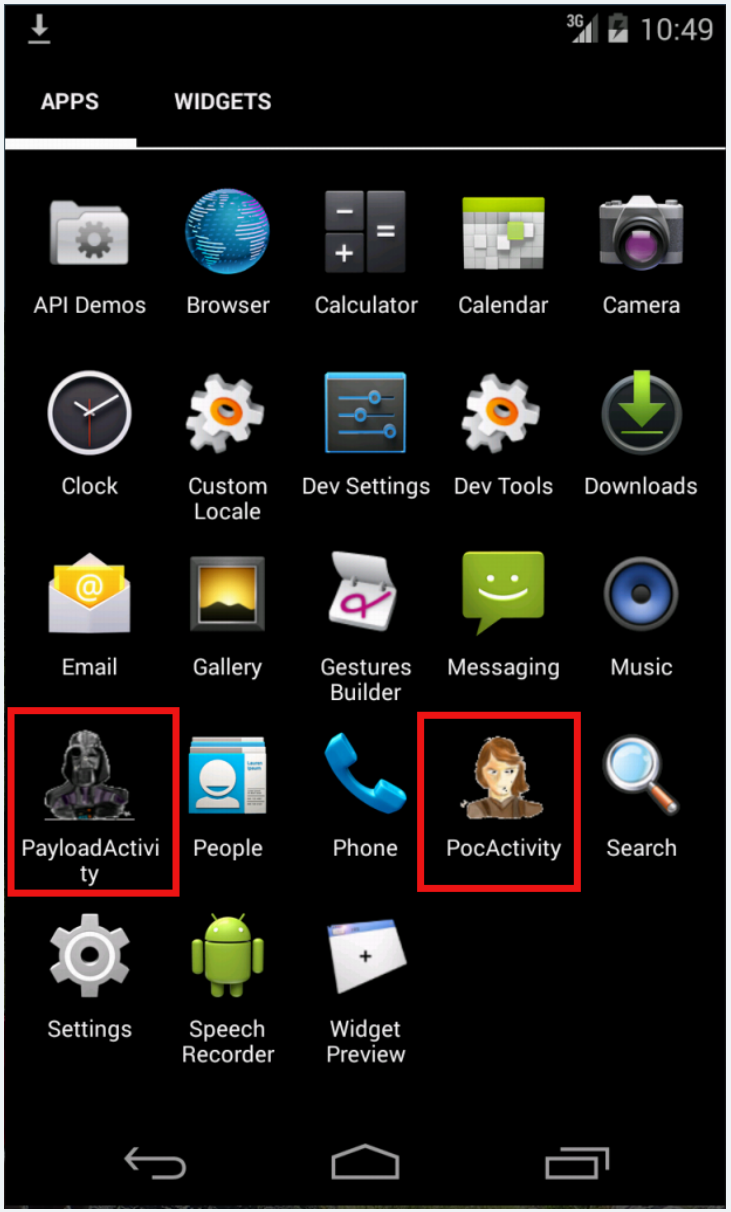
We could use `DexClassLoader` to hide this

In case the demo crashes - lol



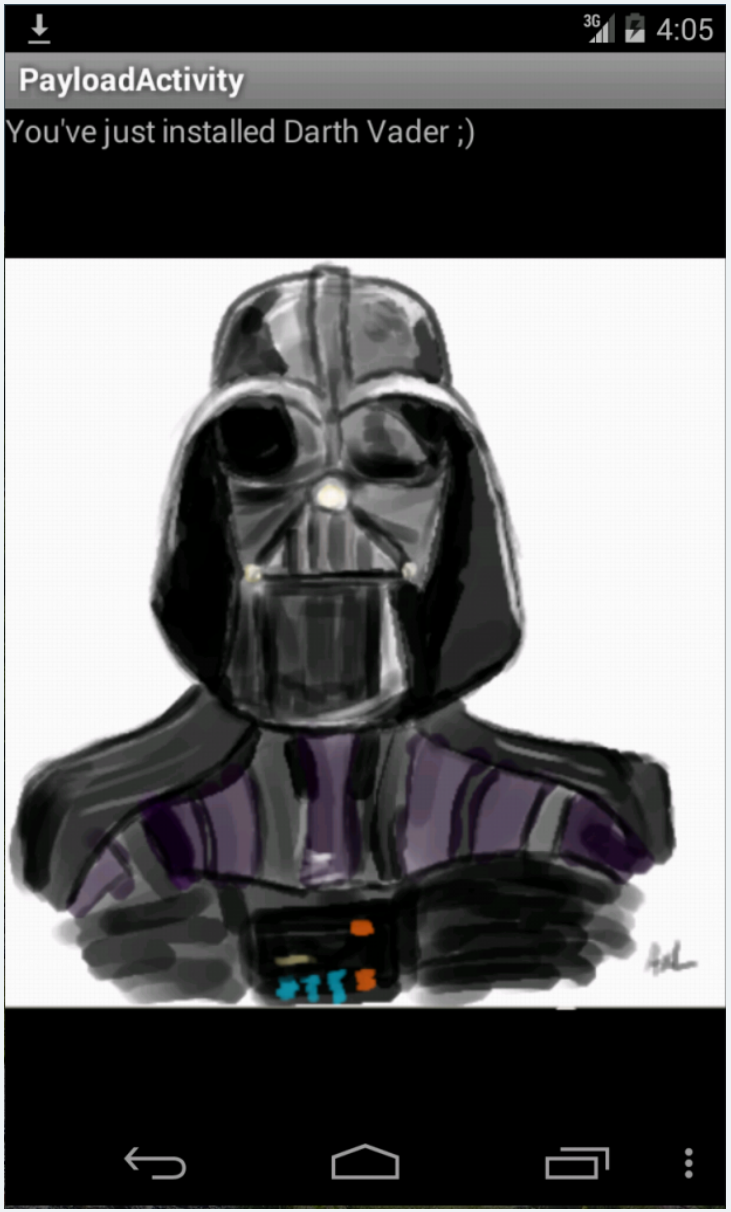
We could use `DexClassLoader` to hide this

In case the demo crashes - lol



We could use `DexClassLoader` to hide this

In case the demo crashes - lol



Payload gets executed

How do we do that?

1. We write a payload APK

How do we do that?

1. We write a payload APK
2. We encrypt it using AngeCryption: it looks like a valid PNG

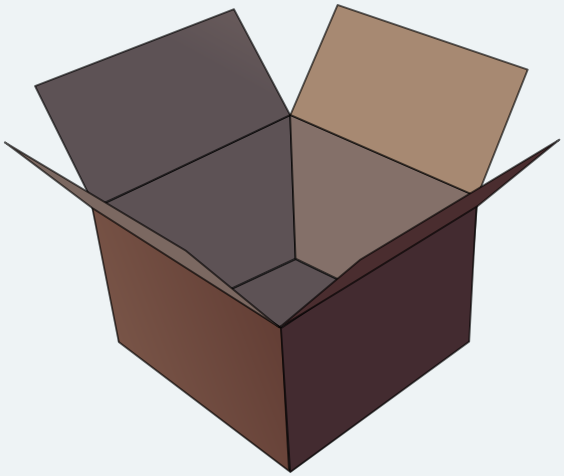
How do we do that?

1. We write a payload APK
2. We encrypt it using AngeCryption: it looks like a valid PNG
3. We hack it (a little)

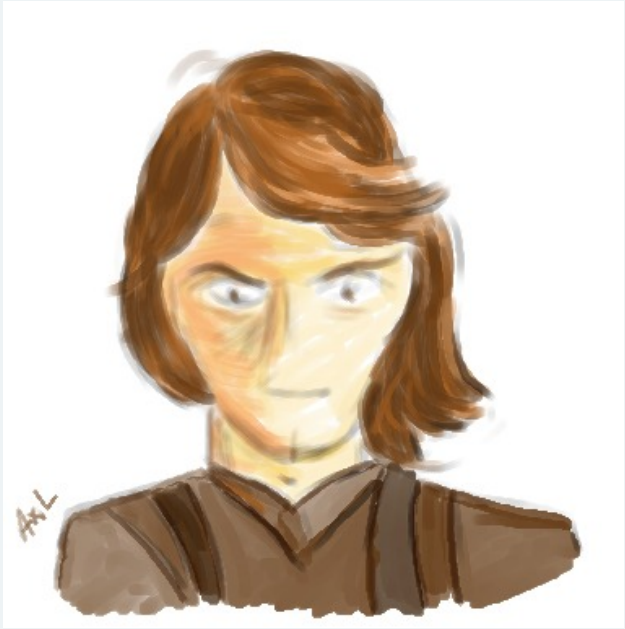
How do we do that?

1. We write a payload APK
2. We encrypt it using AngeCryption: it looks like a valid PNG
3. We hack it (a little)
4. We implement another APK containing the PNG

Power: controlling encryption!



encrypt



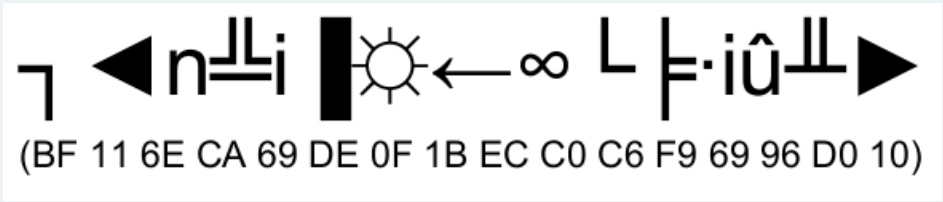
Android Package (APK)
Plaintext

Genuine PNG
Ciphertext

Is this possible?

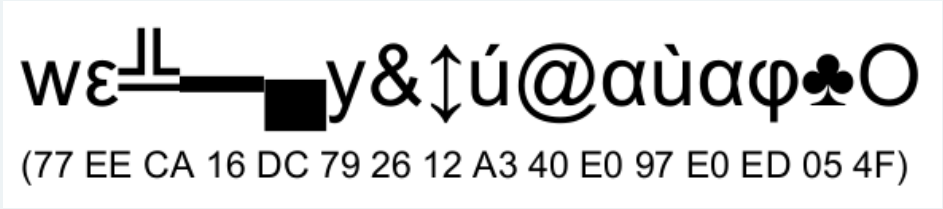
AES encryption in practice

key:'MySecretKey12345'
block:'a block of text.'



key:'MySecretKey12346'
block:'a block of text.'

key:'MySecretKey12345'
block:'a block of text!'



Can we control the output?

With a **tiny change** in the key in the key or the block, the output block is **completely different**

Can we control the output?

With a **tiny change** in the key in the key or the block, the output block is **completely different**

We can't control the output
The output block is (more or less) 'unpredictable'

Can we control the output?

With a **tiny change** in the key in the key or the block, the output block is **completely different**

We can't control the output
The output block is (more or less) 'unpredictable'

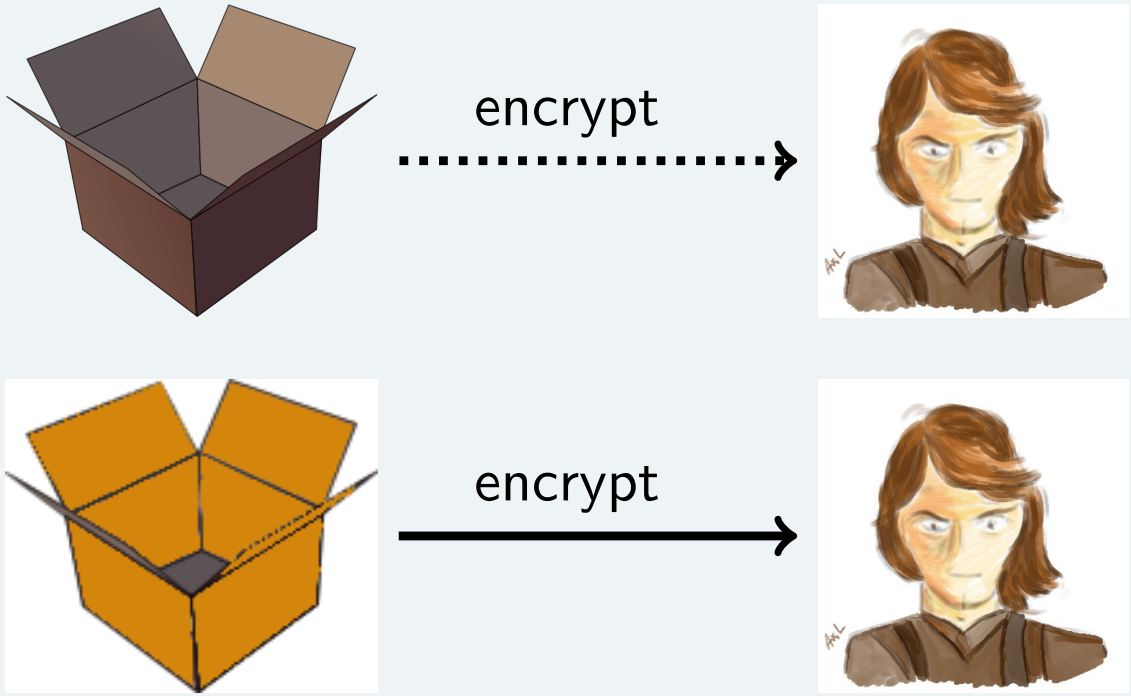
Yes, we can!
But there's a **trick - AngeCryption**

Controlling AES with AngeCryption

It will look the same ... but be slightly different

The APK **will look the same** to Android

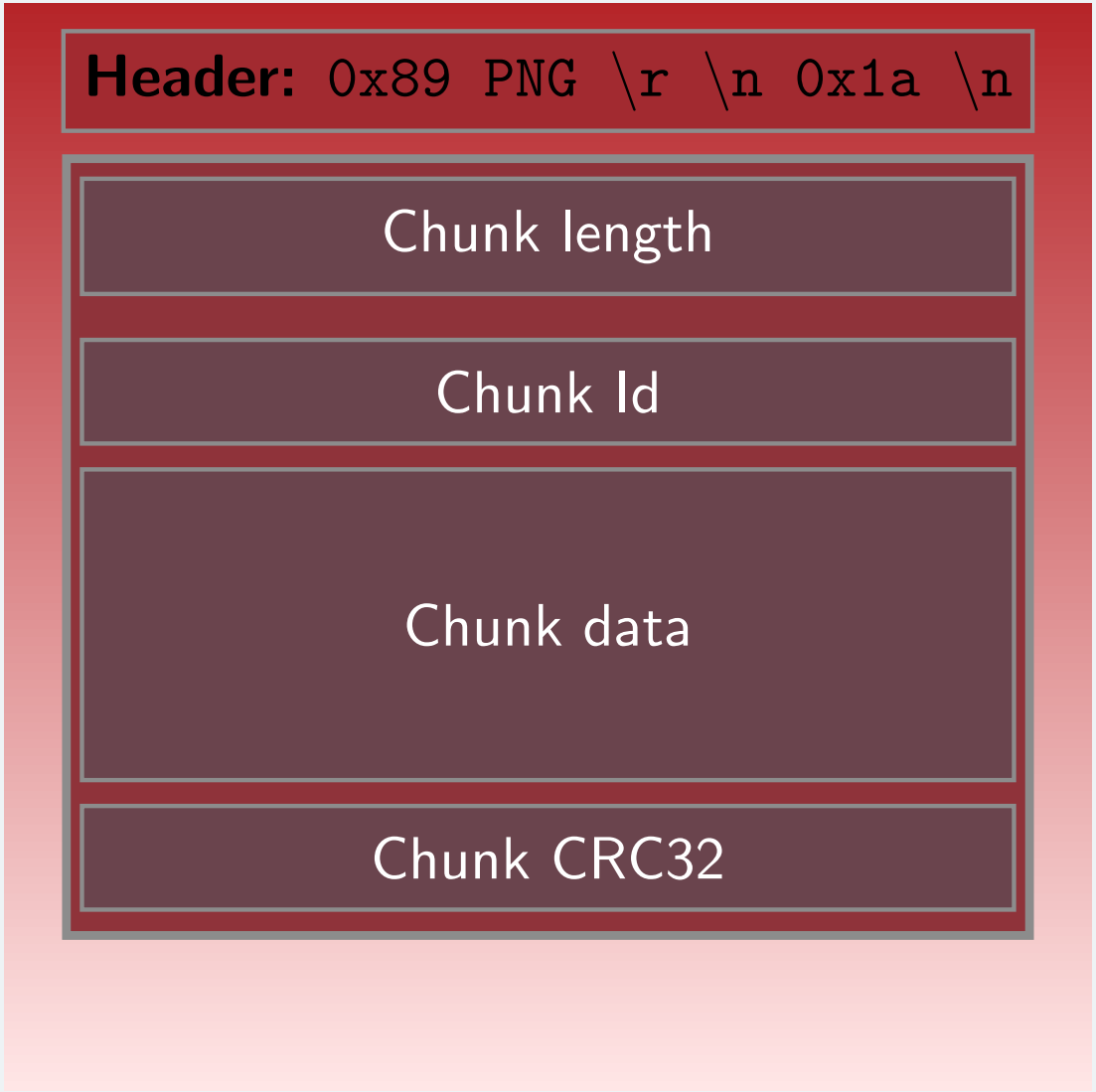
The PNG **will look the same** to our eyes



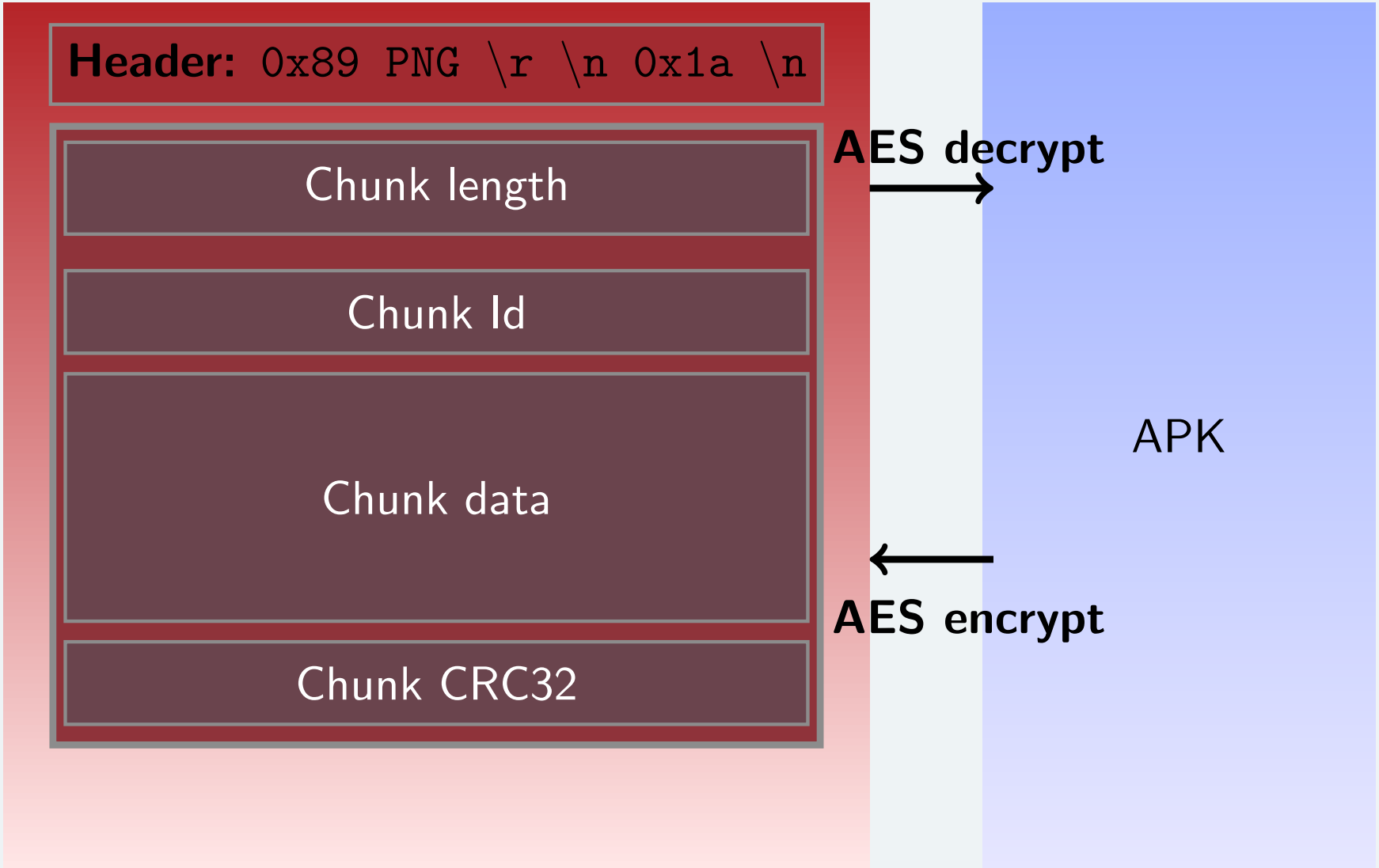
Android does not see the diff Your eye does not see the diff

Manipulate Plaintext so that it encrypts to this PNG

PNG file format 101



PNG file format 101



Crypto background

- AES is a **block cipher**
- It can only process a **block of 16 bytes**

What if my plaintext is longer?!

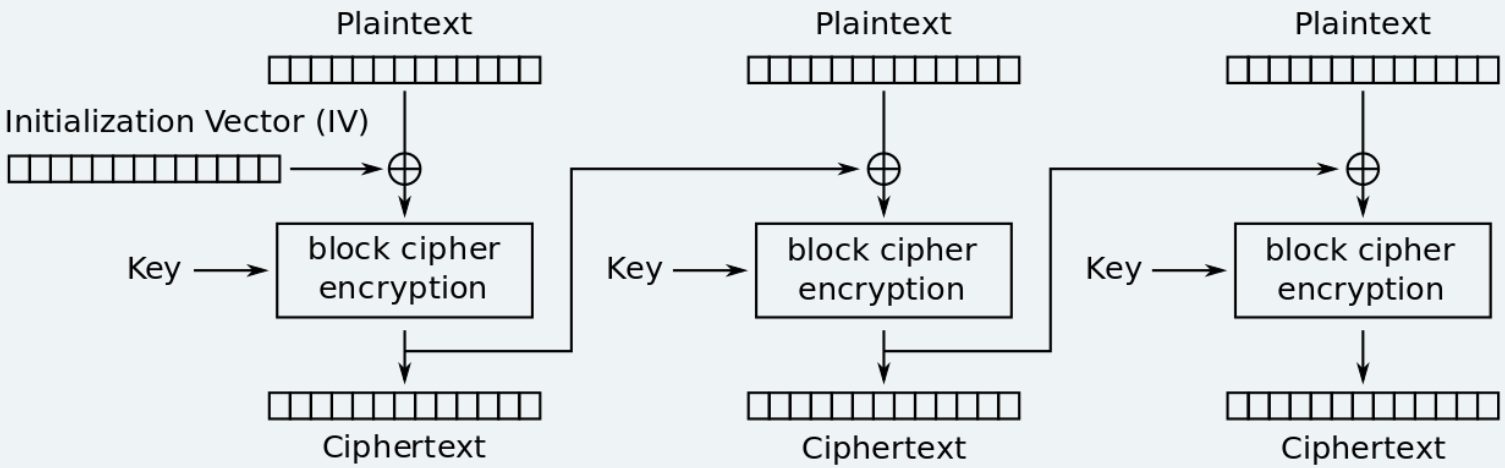
Chaining - 101

- We use **chaining**
- We apply **AES** on **block**
- ... well, that's for ECB (Electronic Code Book). Not very good.

Other chainings

- CBC, CFB, OFB... (see FIPS 81)
- We'll use **CBC** : **C**ipher **B**lock **C**haining

Cipher Block Chaining (CBC) - 101



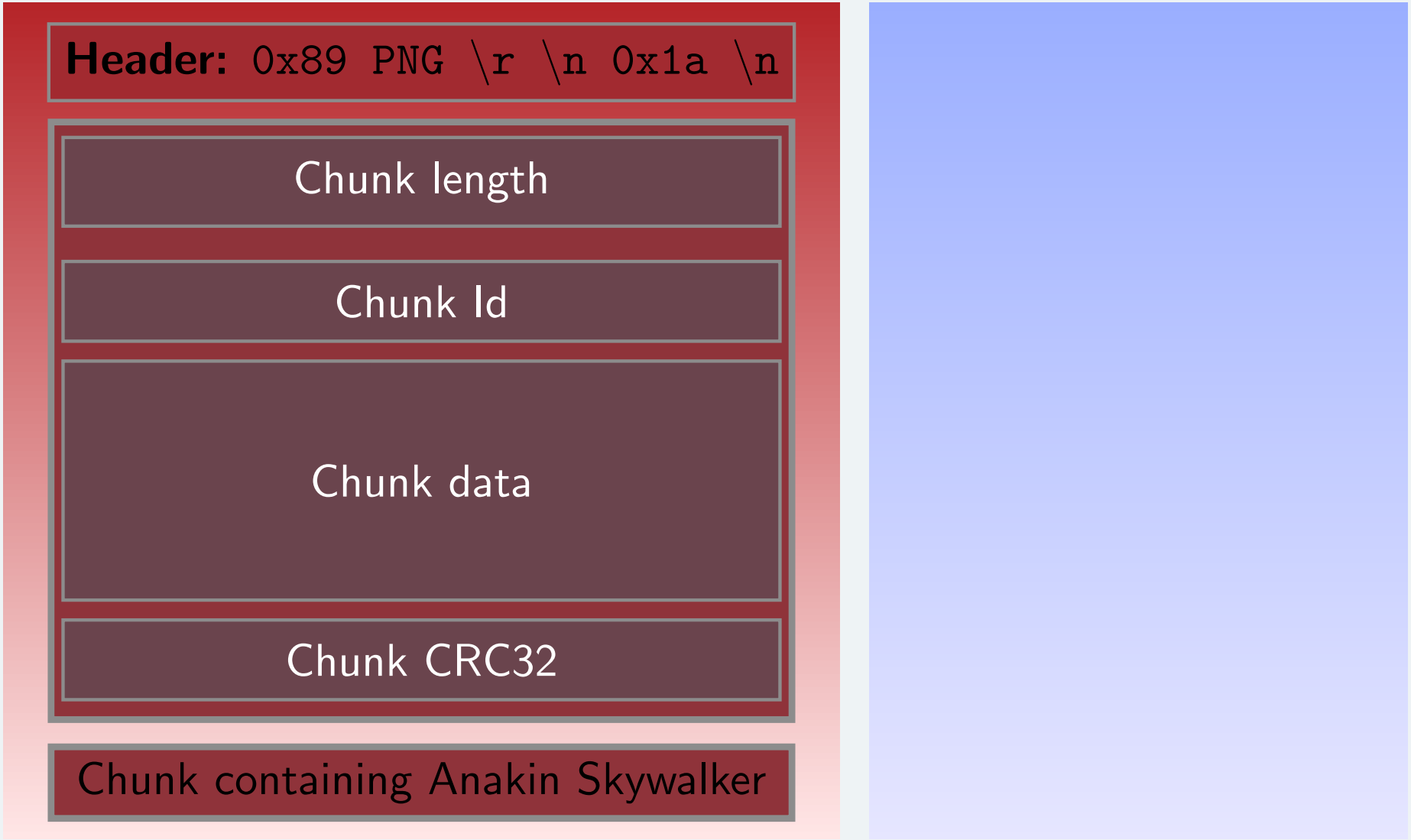
Cipher Block Chaining (CBC) mode encryption

IV is **Initialization Vector**

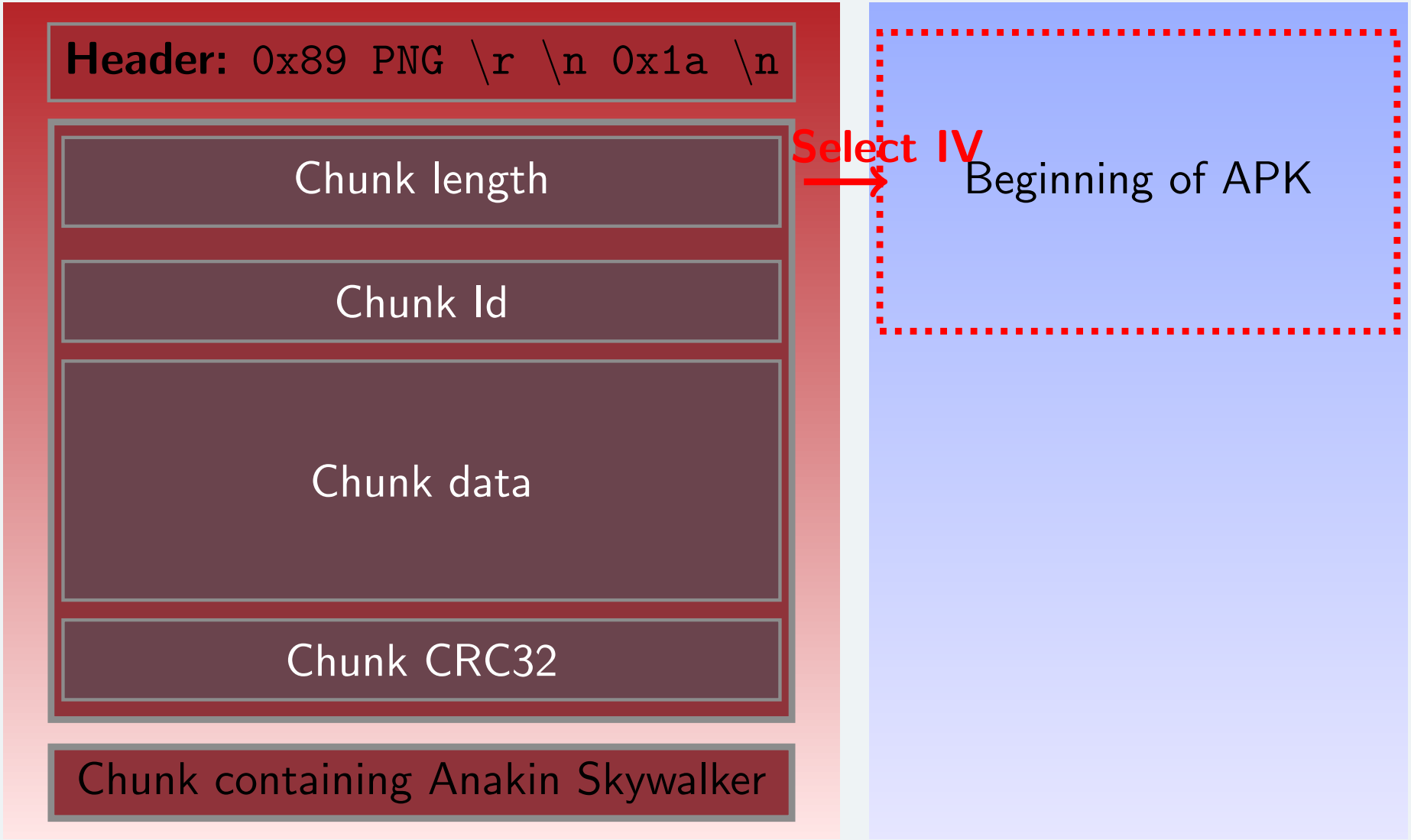
First block

- We have our **plaintext** P_0 and **ciphertext** C_0
- $C_0 = AES_K(P_0 \oplus IV)$
- We can choose the **key K** and **IV**!!!

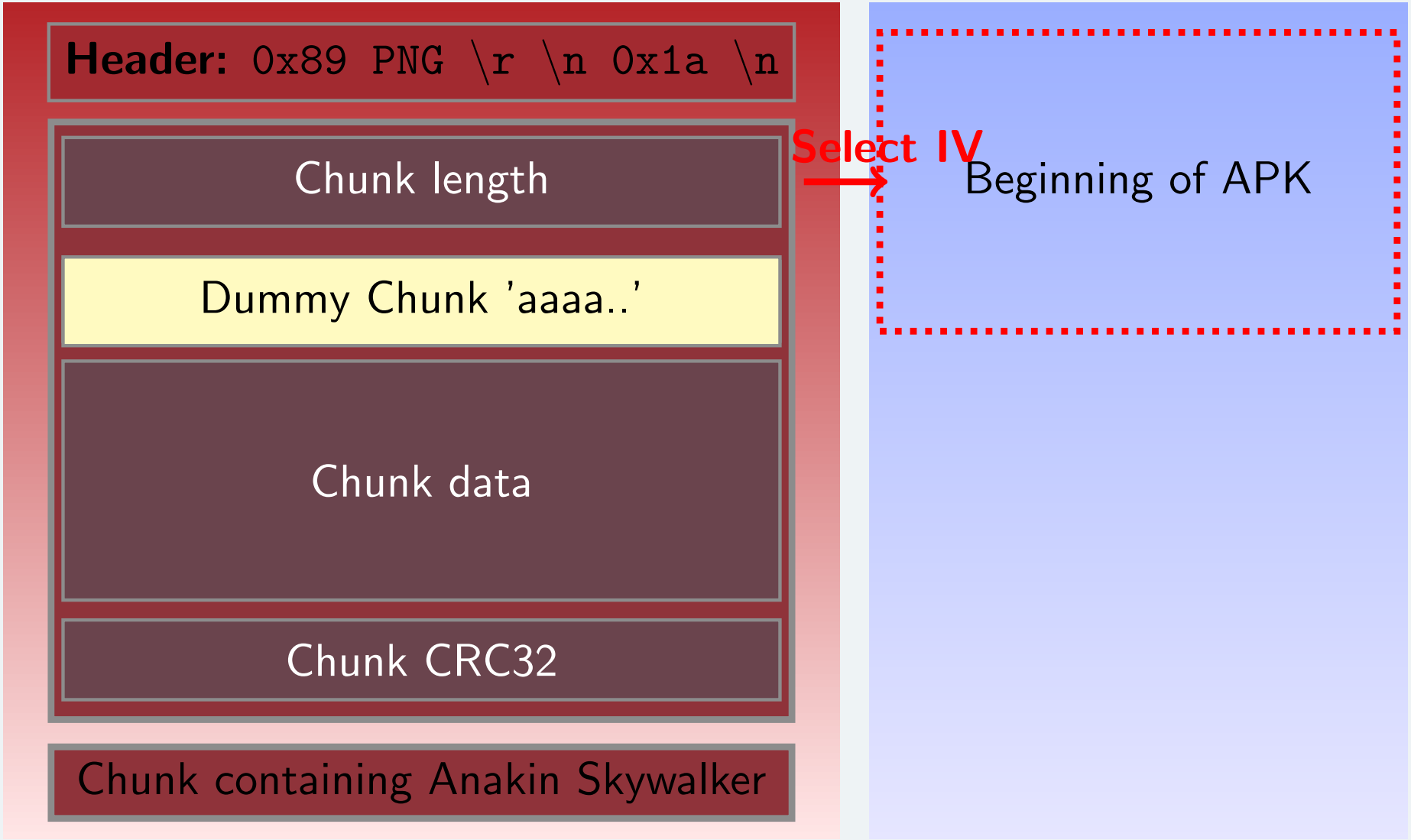
AngeCryption Explained



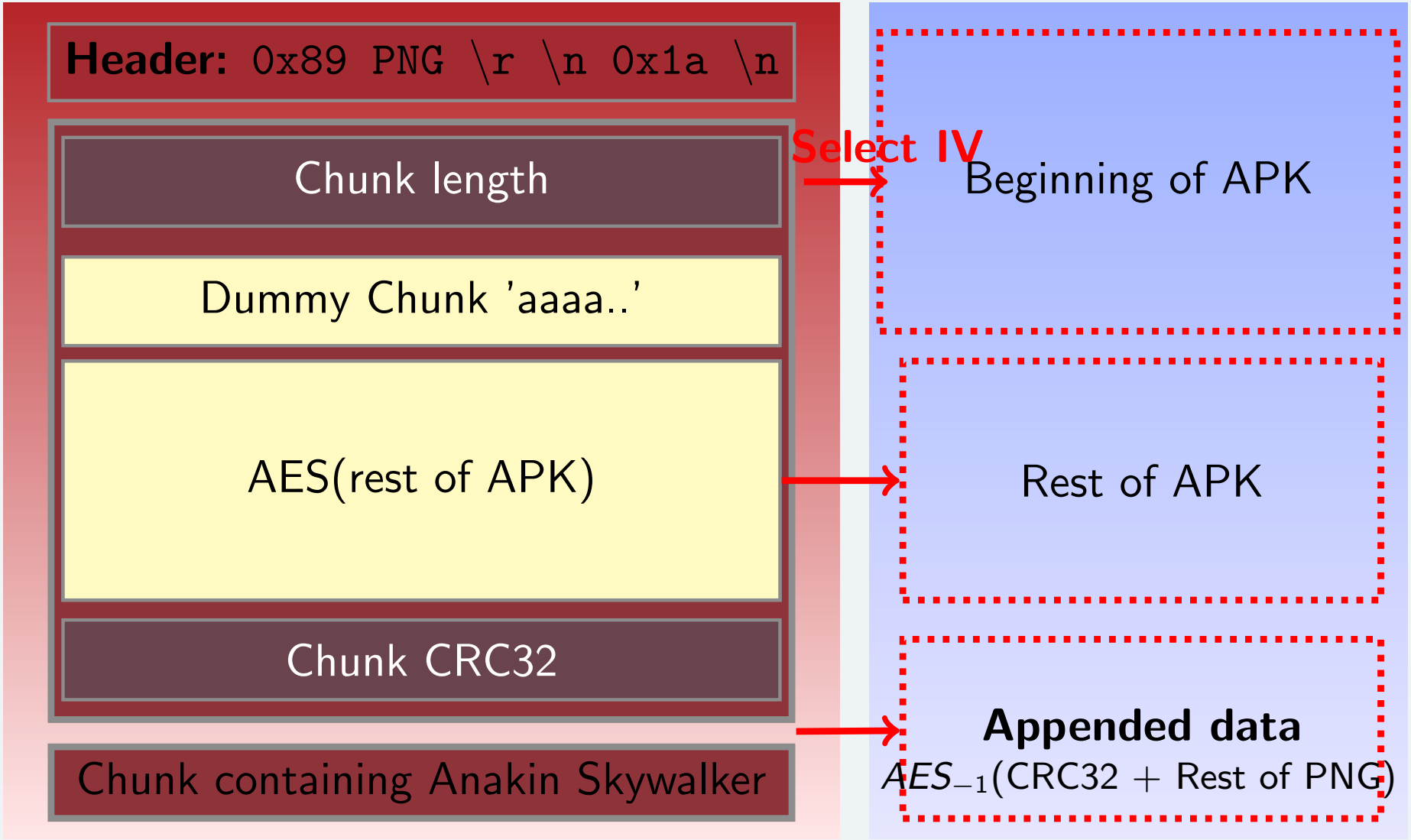
AngeCryption Explained



AngeCryption Explained



AngeCryption Explained



Another hack

The 'similar' APK is 'perfect'... except Android won't load it!
(unzip does not like it either)

```
EOCD not found, not Zip
file 'payload-similar.apk' is not a valid zip file
```

Why?

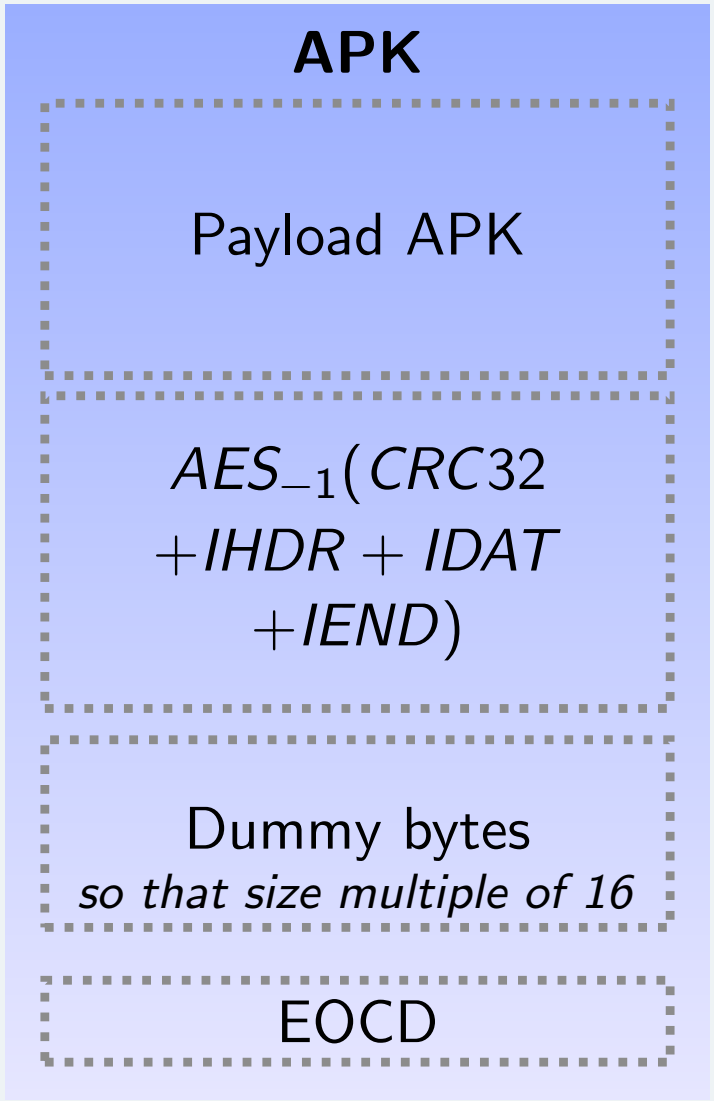
- EOCD is **End of Central Directory**: zip marker
- No EOCD at the end
- Too much appended data after EOCD

Solution

Let's add another EOCD at the end!

Will the APK still correspond to Anakin Skywalker?

YES



Will the APK still correspond to Anakin Skywalker?

YES

PNG

APK

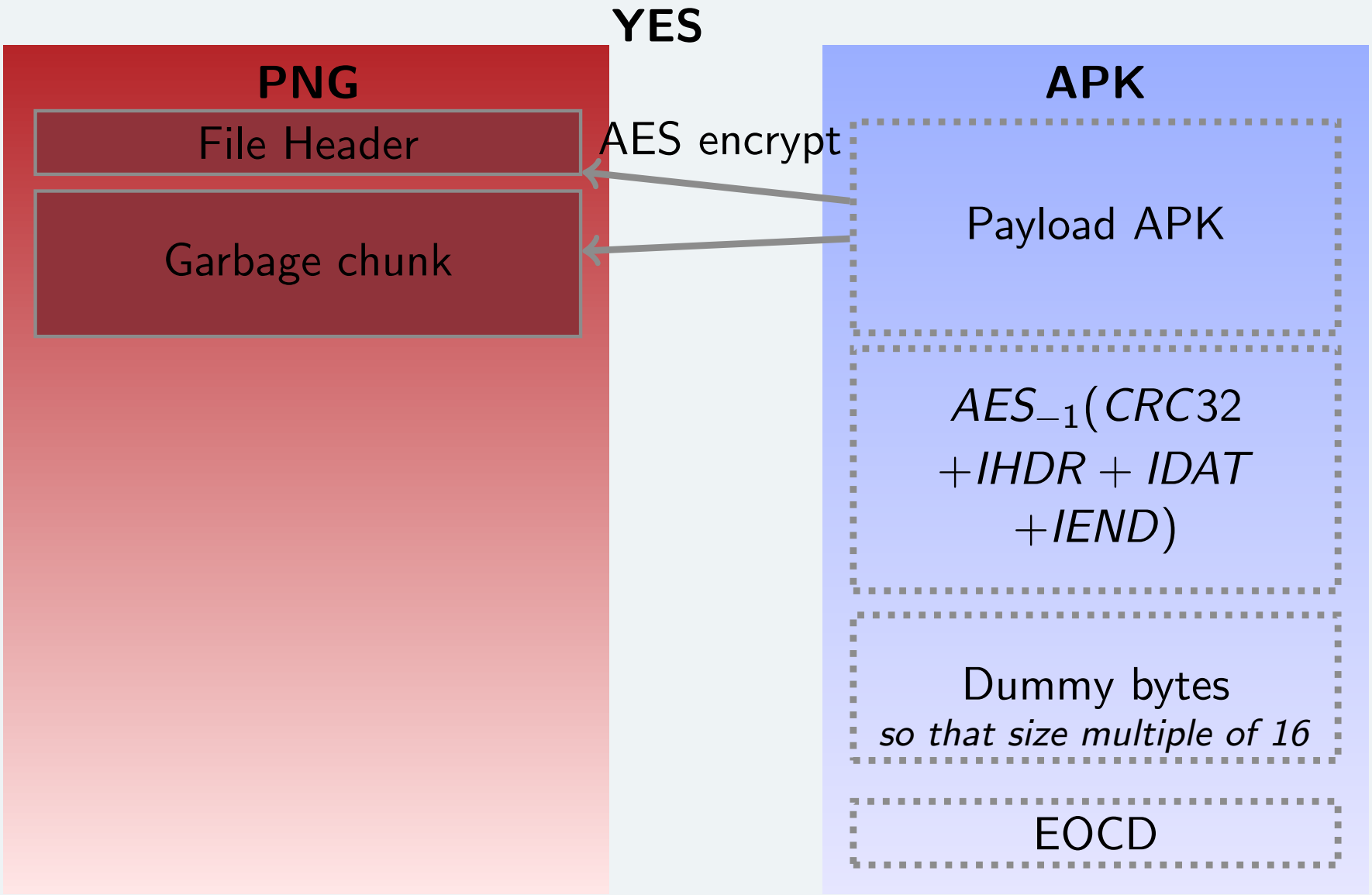
Payload APK

$AES_{-1}(CRC32 + IHDR + IDAT + IEND)$

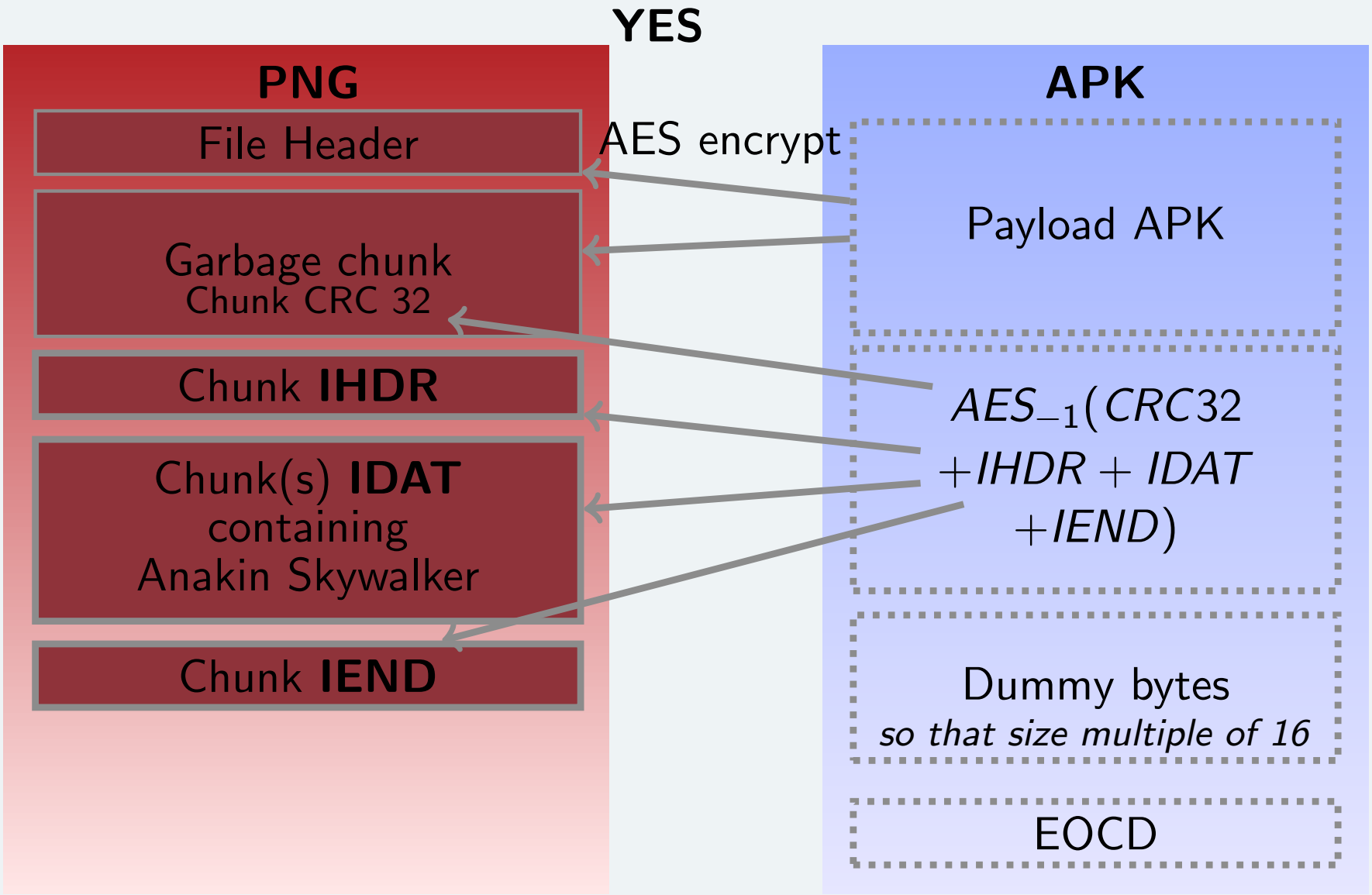
Dummy bytes
so that size multiple of 16

EOCD

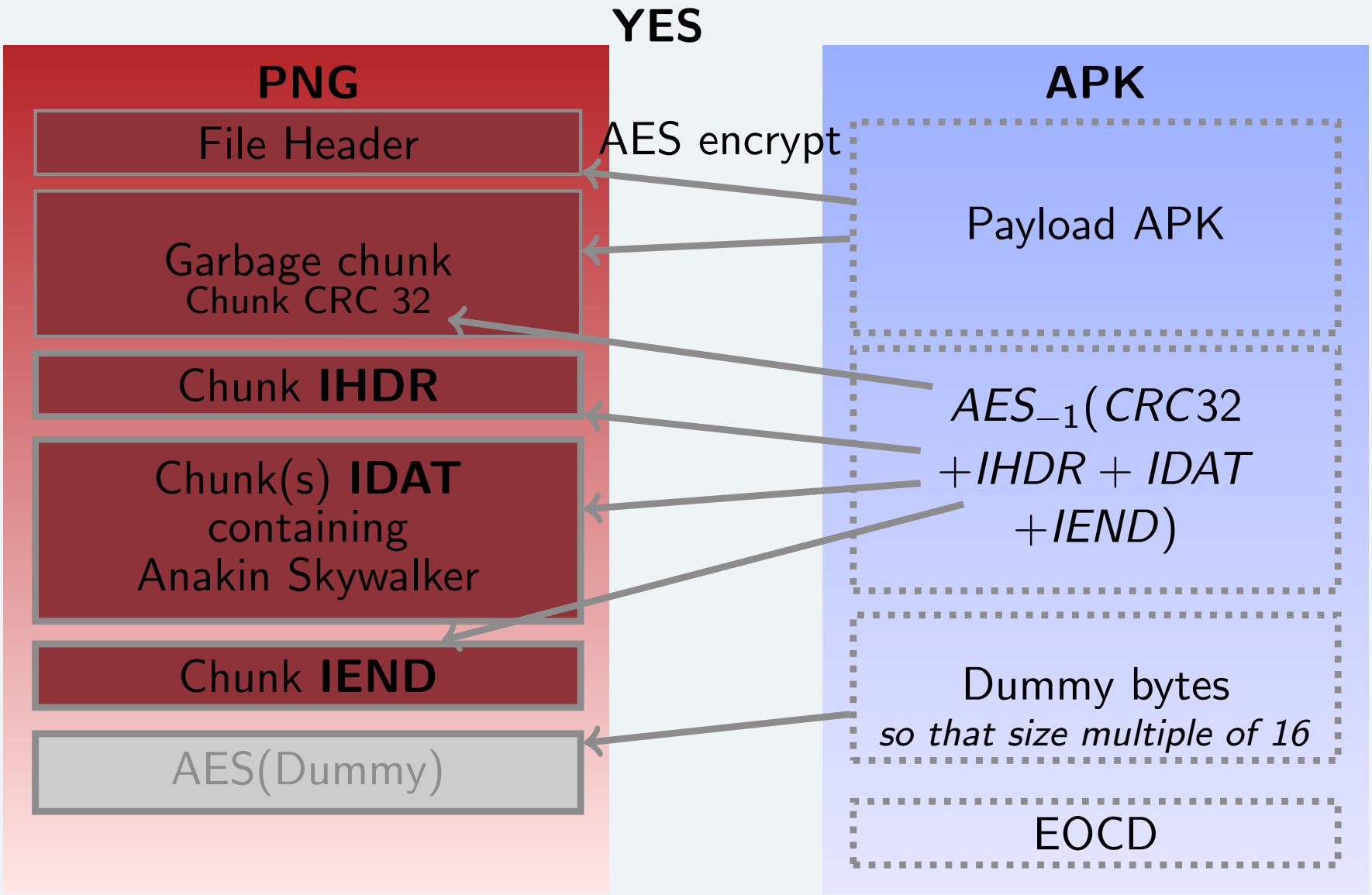
Will the APK still correspond to Anakin Skywalker?



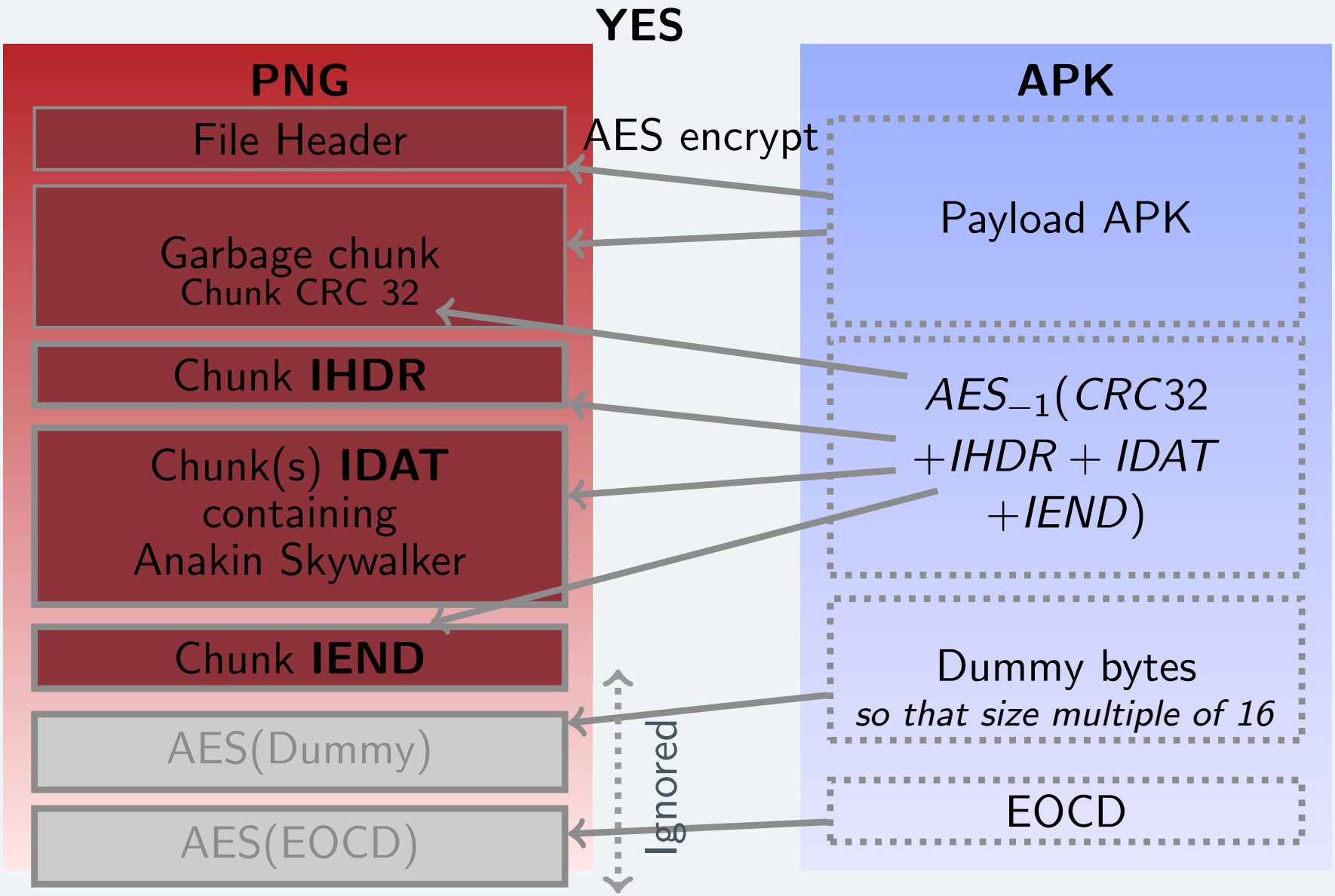
Will the APK still correspond to Anakin Skywalker?



Will the APK still correspond to Anakin Skywalker?



Will the APK still correspond to Anakin Skywalker?



Thank You !

Status

Works on Android 4.4.2

June 2014: Android Security Team notified $\approx$ fixed

Contact info

Me: **@cryptax** or `aapvrille` at `fortinet` dot `com`

Ange: **@angealbertini** or `ange` at `corkami` dot `com`

References

AngeCryption:
http://corkami.googlecode.com/svn/trunk/src/angecryption/

Code: https://github.com/cryptax/angeapk - *soon after conf'*

Corkami: https://code.google.com/p/corkami/

Fortinet's blog: http://blog.fortinet.com

Thanks to : @veorq, Android Security Team