Keep it Like a Secret:
When Android Apps Contain Private Keys

Will Dormann
CERT/CC

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

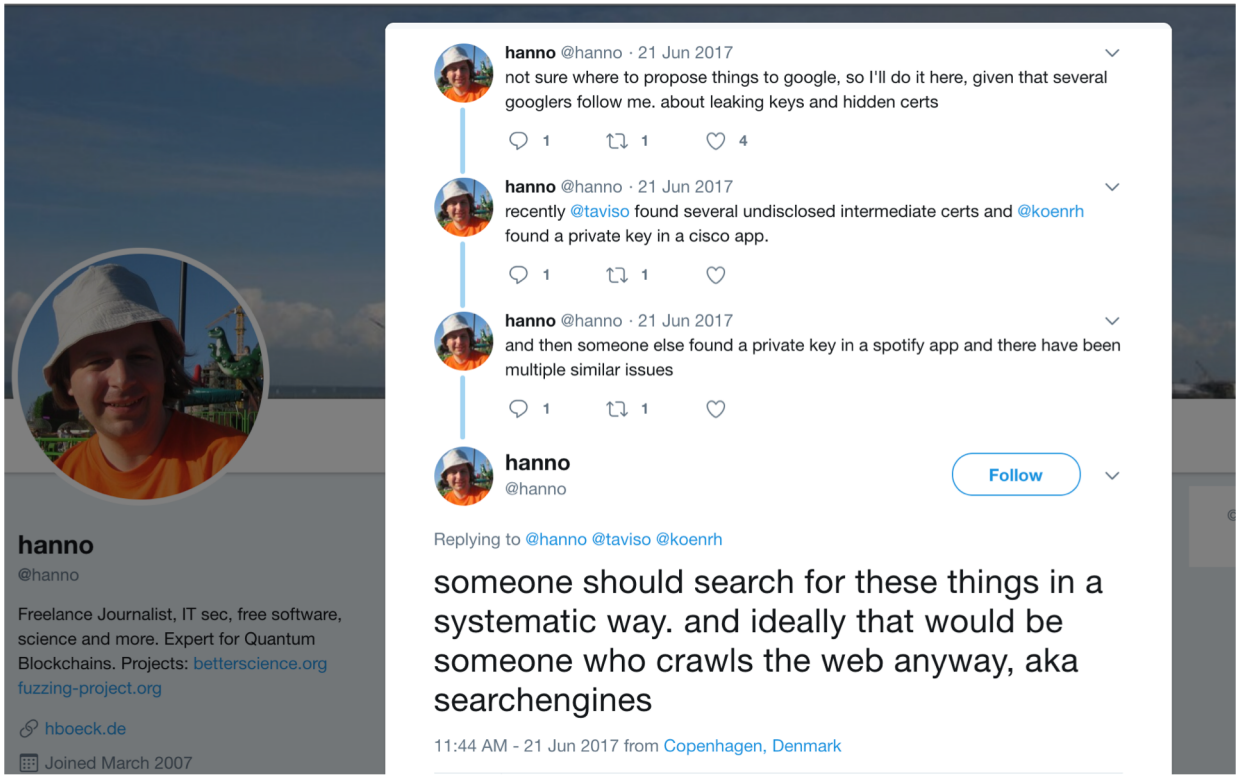
Legal

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

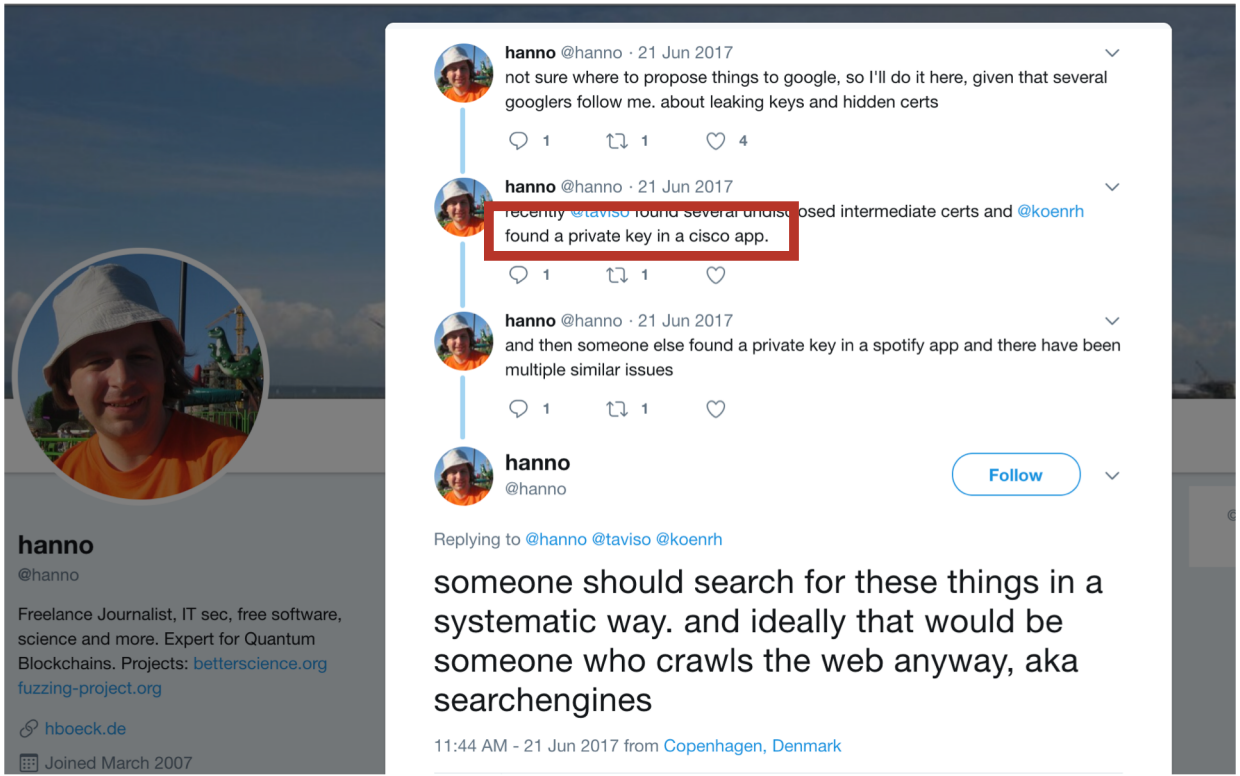
DM18-0416

Background

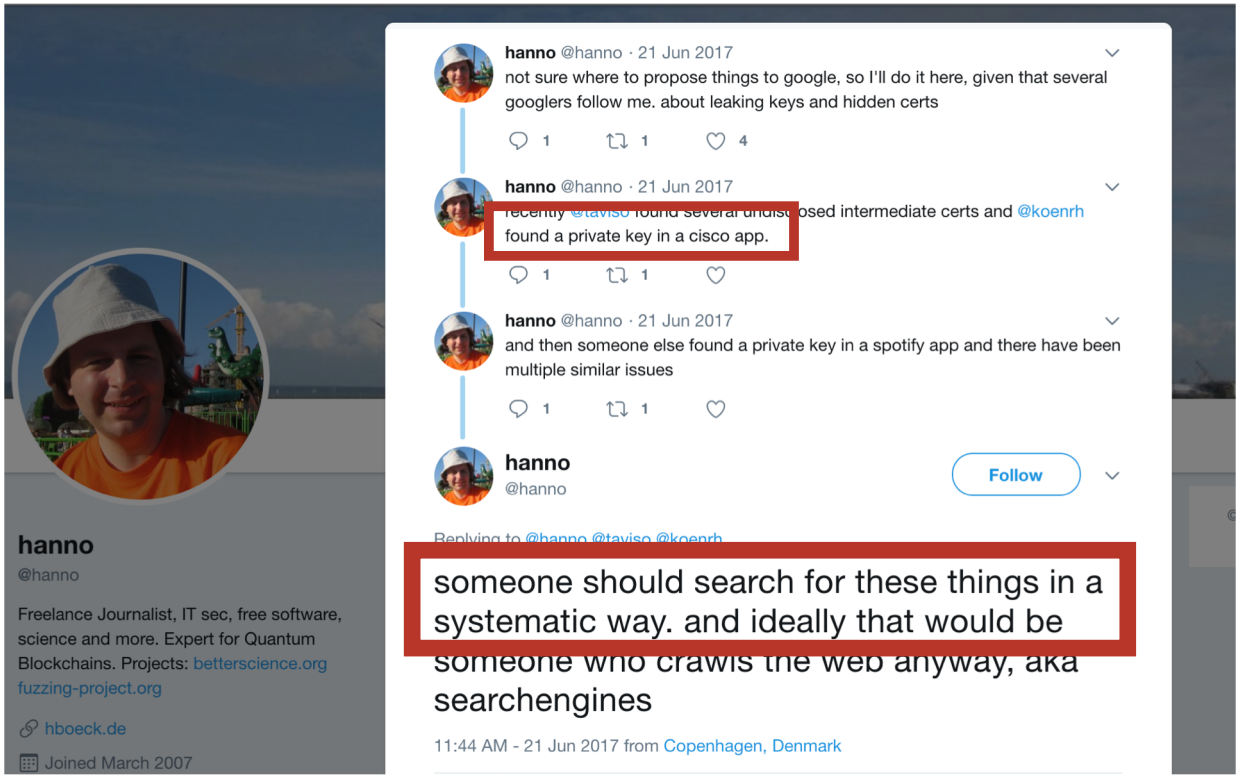
The Inspiration



The Inspiration



The Inspiration



Prior Android Work

#RSAC

The Numbers

	Total	Percent
Free Apps Tested	1,000,500	Most?
Vulnerable Apps Discovered	23,667	2.4%
Vulnerable App Authors Notified	23,301	98.5%
Email responses	1,593	6.8%
Email responses with fix details	25	0.1%

"There are now 1 million apps in the Google Play store."
July 24, 2013

http://mashable.com/2013/07/24/google-play-1-million/

CERT | Software Engineering Institute Carnegie Mellon

RSAConference2015

https://www.rsaconference.com/events/us15/agenda/sessions/1638/how-we-discovered-thousands-of-vulnerable-android

Prior Android Work

#RSAC

The Numbers

	Total	Percent
Free Apps Tested	1,000,500	Most?
Vulnerable Apps Discovered	23,667	2.4%
Vulnerable App Authors Notified	23,301	98.5%
Email responses	1,593	6.8%
Email responses with fix details	25	0.1%

"There are now 1 million apps in the Google Play store."
July 24, 2013

http://mashable.com/2013/07/24/google-play-1-million/

CERT | Software Engineering Institute Carnegie Mellon

RSAConference2015

https://www.rsaconference.com/events/us15/agenda/sessions/1638/how-we-discovered-thousands-of-vulnerable-android

Why Private Keys Matter

Why Look For Private Keys?

Private key files can be used for a number of purposes, including:

Why Look For Private Keys?

Private key files can be used for a number of purposes, including:

- Signing Android Applications

Why Look For Private Keys?

Private key files can be used for a number of purposes, including:

- Signing Android Applications
- Signing IOS Applications

Why Look For Private Keys?

Private key files can be used for a number of purposes, including:

- Signing Android Applications
- Signing IOS Applications
- Encrypting HTTPS traffic

Why Look For Private Keys?

Private key files can be used for a number of purposes, including:

- Signing Android Applications
- Signing IOS Applications
- Encrypting HTTPS traffic
- More!

Getting Android Apps

Downloading APK Files

```
while true; do
   cat /usr/share/hunspell/*.dic | sed 's/\/[^/]*$//' | shuf \
      | xargs -n1 python apkspider.py -k
   echo "Download stopped!?"
   sleep 60
done
```

Processing steps

1. Search for term with gplaycli

Processing steps

1. Search for term with gplaycli
2. Check for new app/version combinations

Processing steps

1. Search for term with gplaycli
2. Check for new app/version combinations
3. Download each new app version in parallel with gplaycli

Processing steps

1. Search for term with gplaycli
2. Check for new app/version combinations
3. Download each new app version in parallel with gplaycli
4. Extract APK details with androguard androapkinfo.py (files, certificates, etc.)

Processing steps

1. Search for term with gplaycli
2. Check for new app/version combinations
3. Download each new app version in parallel with gplaycli
4. Extract APK details with androguard androapkinfo.py (files, certificates, etc.)
5. Pick out key files due to MIME / file extension

Processing steps

1. Search for term with gplaycli
2. Check for new app/version combinations
3. Download each new app version in parallel with gplaycli
4. Extract APK details with androguard androapkinfo.py (files, certificates, etc.)
5. Pick out key files due to MIME / file extension
6. Delete old app versions

Processing steps

1. Search for term with gplaycli
2. Check for new app/version combinations
3. Download each new app version in parallel with gplaycli
4. Extract APK details with androguard androapkinfo.py (files, certificates, etc.)
5. Pick out key files due to MIME / file extension
6. Delete old app versions
7. Repeat

Processing steps

1. Search for term with gplaycli
2. Check for new app/version combinations
3. Download each new app version in parallel with gplaycli
4. Extract APK details with androguard androapkinfo.py (files, certificates, etc.)
5. Pick out key files due to MIME / file extension
6. Delete old app versions
7. Repeat

https://github.com/matlink/gplaycli
https://github.com/androguard/androguard

Learning from mistakes

Don’t end up with a directory with > 1M files in it!

Learning from mistakes

Don’t end up with a directory with > 1M files in it!

by_sha256/c9/ab/34/
c9ab3448251e8a866fb9415b0376022c30aec6e22a5179e64daf24913c32de5d/com.facebook.orca.apk

Learning from mistakes

Don’t end up with a directory with > 1M files in it!

by_sha256/c9/ab/34/
c9ab3448251e8a866fb9415b0376022c30aec6e22a5179e64daf24913c32de5d/com.facebook.orca.apk

Learning from mistakes

Don’t end up with a directory with > 1M files in it!

by_sha256/c9/ab/34/
c9ab3448251e8a866fb9415b0376022c30aec6e22a5179e64daf24913c32de5d/com.facebook.orca.apk

Learning from mistakes

Don’t end up with a directory with > 1M files in it!

by_sha256/c9/ab/34/
c9ab3448251e8a866fb9415b0376022c30aec6e22a5179e64daf24913c32de5d/com.facebook.orca.apk

Picking out Key Files

Problem: You have over **2 billion** files from Android apps. Which to look at?

Picking out Key Files

Problem: You have over **2 billion** files from Android apps. Which to look at?

Answer: Leverage your sqlite database.

```
select file from files where apkpath = ? and (
file like '%.key' or
file like '%.pem' or
file like '%.der' or
file like '%.p12' or
file like '%.bks' or
file like '%.pfx' or
type like '%key%')
```

Key File Formats

PKCS#1

```
-----BEGIN RSA PRIVATE KEY-----
MIIBOwIBAAJBANxtmQ1Kccdp7HBNt8zgTai48Vv617bj4SnhkcMN99sCQ2Naj/sp
... (snip) ...
NiCYNLiCawBbpZnYw/ztPVACK4EwOpUy+u19cMB0JA==
-----END RSA PRIVATE KEY-----
```

- RSA only
- Not password protected

https://github.com/kjur/jsrsasign/wiki/Tutorial-for-PKCS5-and-PKCS8-PEM-private-key-formats-differences

PKCS#5 - Protected

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,E83B4019057F55E9

iIPs59nQn4RSd7ppch9/vNE7PfRSHLoQFmaAjaF0DxjV9oucznUjJq2gphAB2E2H
... (snip) ...
y5IT1MZPgN3LNkVSsLPWKo08uFZQdfu0JTKcn7NPyRc=
-----END RSA PRIVATE KEY-----
```

- RSA only
- Encryption details included (mode, seed)

https://github.com/kjur/jsrsasign/wiki/Tutorial-for-PKCS5-and-PKCS8-PEM-private-key-formats-differences

PKCS#8 - Plain

```
-----BEGIN PRIVATE KEY-----
MIIBVAIBADANBgkqhkiG9w0BAQEFAASCAT4wggE6AgEAAkEA6GZN0rQFKRIVaPOz
... (snip) ...
LaLGdd9G63kLg85eldSy55uIAXsvqQIgfSYaliVtSbAgyx1Yfs3hJ+CTpNKzTNv/
Fx80EltYV6k=
-----END PRIVATE KEY-----
```

- Multiple algorithms
- Not password protected

https://github.com/kjur/jsrsasign/wiki/Tutorial-for-PKCS5-and-PKCS8-PEM-private-key-formats-differences

PKCS#8 - Protected

```
-----BEGIN ENCRYPTED PRIVATE KEY-----
MIIBpjBABgkqhkiG9w0BBQ0wMzAbBgkqhkiG9w0BBQwwDgQIU9Y9p2EfWucCAggA
... (snip) ...
IjsZNp6zmlqf/RXnETsJjGd0TXRWaEdu+XOOyVyPskX2177X9DUJoD31
-----END ENCRYPTED PRIVATE KEY-----
```

- Multiple algorithms
- Password protected

https://github.com/kjur/jsrsasign/wiki/Tutorial-for-PKCS5-and-PKCS8-PEM-private-key-formats-differences

DER Encoding

Key files can be PEM (base64) or DER (binary) encoding

```
openssl asn1parse -inform der -in KEYFILE
```

If "`prim: OBJECT`" is in the results, and openssl returns with exit code 0, you have a DER-encoded key.

PKCS#12

PKCS12 is a **container** for public keys, private keys, certificates, etc.

PKCS#12

PKCS12 is a **container** for public keys, private keys, certificates, etc.

- Password-protection is optional

PKCS#12

PKCS12 is a **container** for public keys, private keys, certificates, etc.

- Password-protection is optional
- A password causes container-level encryption

PKCS#12

PKCS12 is a **container** for public keys, private keys, certificates, etc.

- Password-protection is optional
- A password causes container-level encryption
- You won't know if a protected PKCS12 file contains a private key until **after** you crack it

Java KeyStore

JKS is a **container** for public keys, private keys, certificates, etc.

Java KeyStore

JKS is a **container** for public keys, private keys, certificates, etc.

- Password-protection is optional

Java KeyStore

JKS is a **container** for public keys, private keys, certificates, etc.

- Password-protection is optional
- **No** container-level encryption is used

Java KeyStore

JKS is a **container** for public keys, private keys, certificates, etc.

- Password-protection is optional
- **No** container-level encryption is used
- Private keys share the same password as the container by default

Bouncy Castle Keystore

BKS is a **container** for public keys, private keys, certificates, etc.

Bouncy Castle Keystore

BKS is a **container** for public keys, private keys, certificates, etc.

- Password-protection is optional

Bouncy Castle Keystore

BKS is a **container** for public keys, private keys, certificates, etc.

- Password-protection is optional
- **No** container-level encryption is used

Bouncy Castle Keystore

BKS is a **container** for public keys, private keys, certificates, etc.

- Password-protection is optional
- **No** container-level encryption is used
- Private keys share the same password as the container by default

Bouncy Castle Keystore

BKS is a **container** for public keys, private keys, certificates, etc.

- Password-protection is optional
- **No** container-level encryption is used
- Private keys share the same password as the container by default
- BKS-V1 keystores use only 16 bits for container-level integrity
  https://insights.sei.cmu.edu/cert/2018/03/the-curious-case-of-the-bouncy-castle-bks-passwords.html

Key File Deception

MIME Magic to the Rescue

Some apps try to hide their keys. But androguard lists files and their MIME type based on their magic bytes. Here are some Java KeyStore files that have been found:

MIME Magic to the Rescue

Some apps try to hide their keys. But androguard lists files and their MIME type based on their magic bytes. Here are some Java KeyStore files that have been found:

- `/assets/keystore/comet`

MIME Magic to the Rescue

Some apps try to hide their keys. But androguard lists files and their MIME type based on their magic bytes. Here are some Java KeyStore files that have been found:

- `/assets/keystore/comet`
- `/assets/key2.txt`

MIME Magic to the Rescue

Some apps try to hide their keys. But androguard lists files and their MIME type based on their magic bytes. Here are some Java KeyStore files that have been found:

- `/assets/keystore/comet`
- `/assets/key2.txt`
- `/assets/Resources/Keyfile pw sc7`

MIME Magic to the Rescue

Some apps try to hide their keys. But androguard lists files and their MIME type based on their magic bytes. Here are some Java KeyStore files that have been found:

- `/assets/keystore/comet`
- `/assets/key2.txt`
- `/assets/Resources/Keyfile pw sc7`
- `/assets/data/10305.bin`

MIME Magic to the Rescue

Some apps try to hide their keys. But androguard lists files and their MIME type based on their magic bytes. Here are some Java KeyStore files that have been found:

- `/assets/keystore/comet`
- `/assets/key2.txt`
- `/assets/Resources/Keyfile pw sc7`
- `/assets/data/10305.bin`
- `/assets/www/res/bin/md-trucks.apk`

MIME Magic to the Rescue

Some apps try to hide their keys. But androguard lists files and their MIME type based on their magic bytes. Here are some Java KeyStore files that have been found:

- `/assets/keystore/comet`
- `/assets/key2.txt`
- `/assets/Resources/Keyfile pw sc7`
- `/assets/data/10305.bin`
- `/assets/www/res/bin/md-trucks.apk`
- `/assets/fonts/data.ttf`

MIME Magic to the Rescue

Some apps try to hide their keys. But androguard lists files and their MIME type based on their magic bytes. Here are some Java KeyStore files that have been found:

- `/assets/keystore/comet`
- `/assets/key2.txt`
- `/assets/Resources/Keyfile pw sc7`
- `/assets/data/10305.bin`
- `/assets/www/res/bin/md-trucks.apk`
- `/assets/fonts/data.ttf`
- `/assets/ic_launcher-web.png`

Keep it Like a Secret: When Android Apps Contain Private Keys

Password Cracking

Why crack passwords?

Why crack passwords?

- You don’t know what’s in a PKCS#12 keystore until you’ve decrypted it.

Why crack passwords?

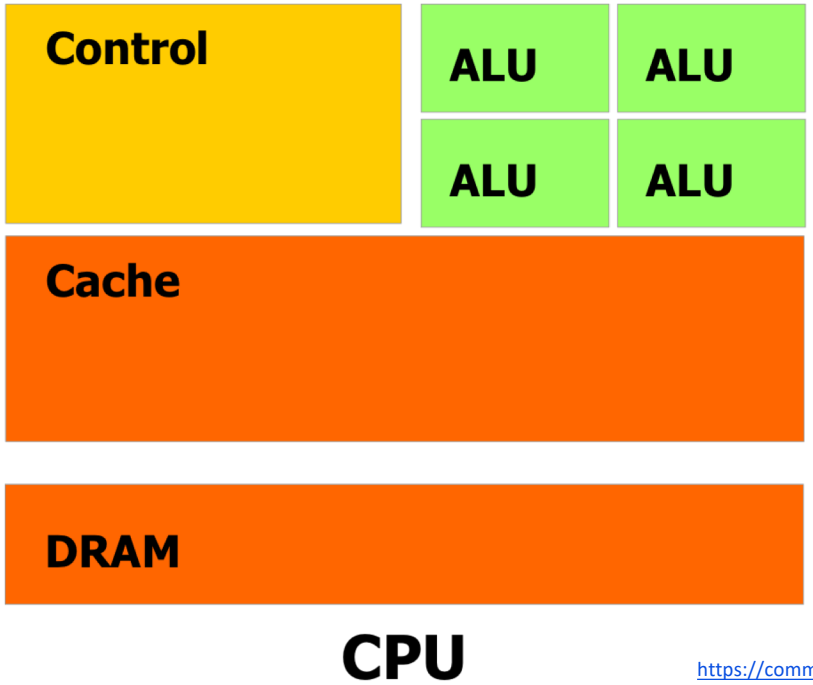
- You don’t know what’s in a PKCS#12 keystore until you’ve decrypted it.
- The impact of a key leak is related to its protection

GPU hardware

GPUs are massively parallel, which is a great match for password cracking.

GPU hardware

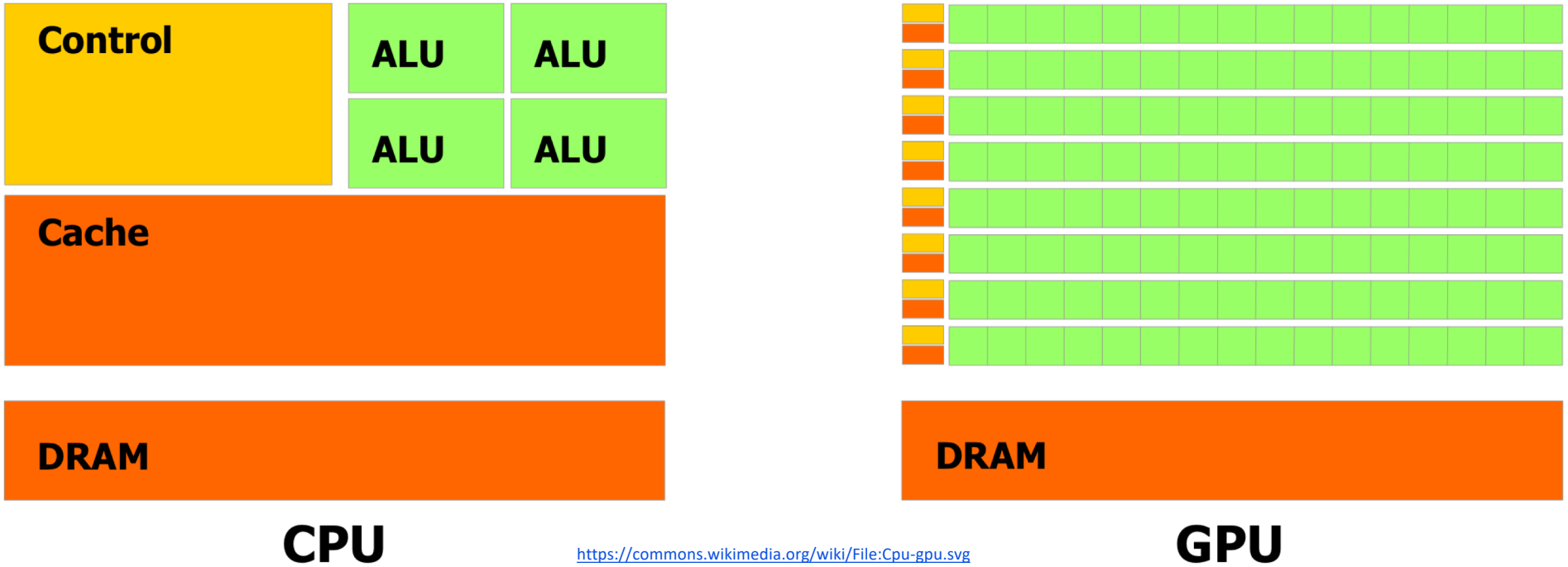
GPUs are massively parallel, which is a great match for password cracking.



https://commons.wikimedia.org/wiki/File:Cpu-gpu.svg

GPU hardware

GPUs are massively parallel, which is a great match for password cracking.



https://commons.wikimedia.org/wiki/File:Cpu-gpu.svg

Password-cracking software

John the Ripper

https://github.com/magnumripper/JohnTheRipper

Password-cracking software

John the Ripper

https://github.com/magnumripper/JohnTheRipper

Hashcat

https://hashcat.net/hashcat/

Keep it Like a Secret: When Android Apps Contain Private Keys

Password Strength

Password strength

One measure of password strength is its resistance to bruteforce attacks. This is a factor of:

Password strength

One measure of password strength is its resistance to bruteforce attacks. This is a factor of:

- Charset size

Password strength

One measure of password strength is its resistance to bruteforce attacks. This is a factor of:

- Charset size
- Length

Password strength example

Symbols used: digits 0-9 (Charset size:10)

Password strength example

Symbols used: digits 0-9 (Charset size:10)

Length: 1

Password strength example

Symbols used: digits 0-9 (Charset size:10)

Length: 1
Total passwords: 10 (10^1)

Password strength example

Symbols used: digits 0-9 (Charset size:10)

Length: 1
Total passwords: 10 (10^1)

Length: 2

Password strength example

Symbols used: digits 0-9 (Charset size:10)

Length: 1
Total passwords: 10 (10^1)

Length: 2
Total passwords: 100 (10^2)

Password strength example

Symbols used: digits 0-9 (Charset size:10)

Length: 1
Total passwords: 10 (10^1)

Length: 2
Total passwords: 100 (10^2)

Length: 3

Password strength example

Symbols used: digits 0-9 (Charset size:10)

Length: 1
Total passwords: 10 (10^1)

Length: 2
Total passwords: 100 (10^2)

Length: 3
Total passwords: 1000 (10^3)

Password strength example

Symbols used: digits 0-9 (Charset size:10)

Length: 1
Total passwords: 10 (10^1)

Length: 2
Total passwords: 100 (10^2)

Length: 3
Total passwords: 1000 (10^3)

Increasing password length grows password strength **exponentially**!

Password strength example

Symbols used: digits 0-9 (Charset size:10)

Length: 1
Total passwords: 10 (10^1)

Length: 2
Total passwords: 100 (10^2)

Length: 3
Total passwords: 1000 (10^3)

Increasing password length grows password strength **exponentially**!

Real-world password cracking times

Password	Charset size	Length	Crack Time
TMAE	26	4	1 second
66aK	62	4	2 seconds
tcdely	26	6	2 seconds
v4lz6o	36	6	27 seconds
>^	32	2	162 seconds
dcgkrnel	26	8	186 seconds
8C;e	84	4	407 seconds (7 mins)
QFsRe7	62	6	1022 seconds (17 mins)
147wyf7g	36	8	9569 seconds (3 hours)

Random passwords found via MD5 with John the Ripper incremental mode on 64 CPU cores

Real-world password cracking times

Password	Charset size	Length	Crack Time
147wyf7g	36	8	9569 seconds (3 hours)
:@(	32	3	16459 seconds (5 hours)
QLvD:	74	5	59861 seconds (17 hours)
KDJCTFWY	26	8	279959 seconds (3 days)
ahpcwckjq	26	9	472538 seconds (5 days)
{~>[	22	4	475956 seconds (5 days)
lavyovfass	26	10	619300 seconds (7 days)
ob13zgx33	36	9	796192 seconds (9 days)
fqmRDnGY	52	8	1874245 seconds (22 days)

Random passwords found via MD5 with John the Ripper incremental mode on 64 CPU cores

Example Passwords

Which passwords are good?

Rules of the game:

Given an example password, tell me if the password is **weak** or **strong.**

Weak or Strong?

Password:
`changeit`

Weak or Strong?

Password:
`changeit`

Answer:
WEAK

Weak or Strong?

Password:
`changeit`

Answer:
WEAK

Reason:
Present in “rockyou” password list

https://github.com/danielmiessler/SecLists/tree/master/Passwords

Weak or Strong?

Password:
`7y6t5r4e`

Weak or Strong?

Password:
`7y6t5r4e`

Answer:
WEAK

Weak or Strong?

Password:
`7y6t5r4e`

Answer:
WEAK

Reason:
Present in “rockyou” password list

https://github.com/danielmiessler/SecLists/tree/master/Passwords

Weak or Strong?

Password:
7y6t5r4e

Answer:
WEAK

Reason:
Present in “rockyou” password list

https://github.com/danielmiessler/SecLists/tree/master/Passwords

Weak or Strong?

Password:
`7y6t5r4e`

Answer:
WEAK

Reason:
Present in “rockyou” password list

https://github.com/danielmiessler/SecLists/tree/master/Passwords

Weak or Strong?

Password:
7y6t5r4e

Answer:
WEAK

Reason:
Present in “rockyou” password list

https://github.com/danielmiessler/SecLists/tree/master/Passwords

Weak or Strong?

Password:
7y6t5r4e

Answer:
WEAK

Reason:
Present in “rockyou” password list

https://github.com/danielmiessler/SecLists/tree/master/Passwords

Weak or Strong?

Password:
7y6t5r4e

Answer:
WEAK

Reason:
Present in “rockyou” password list

https://github.com/danielmiessler/SecLists/tree/master/Passwords

Weak or Strong?

Password:
`7y6t5r4e`

Answer:
WEAK

Reason:
Present in “rockyou” password list

https://github.com/danielmiessler/SecLists/tree/master/Passwords

Weak or Strong?

Password:
7y6t5r4e

Answer:
WEAK

Reason:
Present in “rockyou” password list

https://github.com/danielmiessler/SecLists/tree/master/Passwords

Weak or Strong?

Password:
7y6t5r4e

Answer:
WEAK

Reason:
Present in “rockyou” password list

https://github.com/danielmiessler/SecLists/tree/master/Passwords

Weak or Strong?

Password:
7y6t5r4e

Answer:
WEAK

Reason:
Present in “rockyou” password list

https://github.com/danielmiessler/SecLists/tree/master/Passwords

Weak or Strong?

Password:
Huawei@123

Weak or Strong?

Password:
Huawei@123

Answer:
WEAK

Weak or Strong?

Password:
Huawei@123

Answer:
WEAK

Reason:
Cracked by hashcat “naïve” strategy

https://github.com/brannondorsey/naive-hashcat

Weak or Strong?

Password:
Huawei@123

Answer:
WEAK

- Capitalize first letter

Reason:
Cracked by hashcat “naïve” strategy

https://github.com/brannondorsey/naive-hashcat

Weak or Strong?

Password:
Huawei@123

Answer:
WEAK

- Capitalize first letter
- Numbers and symbols at end

Reason:
Cracked by hashcat “naïve” strategy

https://github.com/brannondorsey/naive-hashcat

Weak or Strong?

Password:
Huawei@123

Answer:
WEAK

- Capitalize first letter
- Numbers and symbols at end
- Hashcat rules target human nature

Reason:
Cracked by hashcat “naïve” strategy

https://github.com/brannondorsey/naive-hashcat

Weak or Strong?

Password:
`delaunay55`

Weak or Strong?

Password:
`delaunay55`

Answer:
WEAK

Weak or Strong?

Password:
`delaunay55`

Answer:
WEAK

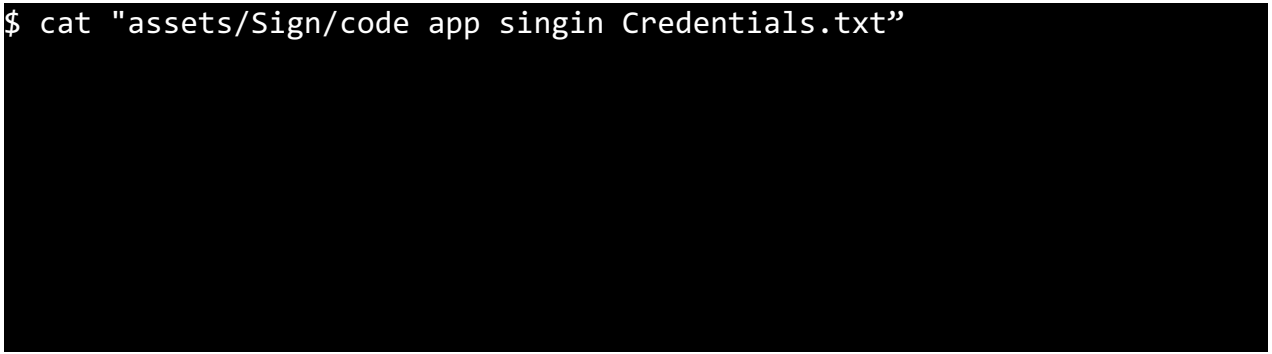
Reason:
Author left a note.

Weak or Strong?

Password:
`delaunay55`

Answer:
WEAK

Reason:
Author left a note.



Weak or Strong?

Password:
`delaunay55`

Answer:
WEAK

Reason:
Author left a note.

```
$ cat "assets/Sign/code app singin Credentials.txt"
   signingConfigs {
        config {
            keyAlias 'codehospitality'
            keyPassword 'delaunay55'
            storeFile file('C:/Users/Al/keystores/codehospitality.jks')
            storePassword 'delaunay55'
        }
    }
```

Weak or Strong?

Password:
`SamSam1cooke!`

Weak or Strong?

Password:
`SamSam1cooke!`

Answer:
WEAK

Weak or Strong?

Password:
`SamSam1cooke!`

Answer:
WEAK

Reason:
Author left a note.

Weak or Strong?

Password:
SamSam1cooke!

Answer:
WEAK

Reason:
Author left a note.

Weak or Strong?

Password:
SamSam1cooke!

Answer:
WEAK

Reason:
Author left a note.

```
$ cat /assets/www/config/README.txt
The apple developer credentials are:
[redacted]@[redacted]
[redacted]


The Google Play credentials are:
[redacted]@[redacted]
[redacted]

The password for the sean.p12 file:
SamSam1cooke!
```

Weak or Strong?

Password:
yXPZ1gzxFkV68NKquIpa93BLeHiC7Yjd

Weak or Strong?

Password:
yXPZ1gzxFkV68NKquIpa93BLeHiC7Yjd

Answer:
WEAK

Weak or Strong?

Password:
yXPZ1gzxFkV68NKquIpa93BLeHiC7Yjd

Answer:
WEAK

Reason:
It's in the application.

Weak or Strong?

Password:
yXPZ1gzxFkV68NKquIpa93BLeHiC7Yjd

Answer:
WEAK

Reason:
It's in the application.

Weak or Strong?

Password:
yXPZ1gzxFkV68NKquIpa93BLeHiC7Yjd

Answer:
WEAK

Reason:
It's in the application.

```
$ cat res/values/strings.xml
…
<string
name="lyfpay_certificate_password">yXPZ1gzxFkV68NKquIpa93BLeHiC7Yjd
</string>
…
```

Weak or Strong?

Password:
`305P6N2cEl4nkH!T$1q*`

Weak or Strong?

Password:
`305P6N2cEl4nkH!T$1q*`

Answer:
WEAK

Weak or Strong?

Password:
`305P6N2cEl4nkH!T$1q*`

Answer:
WEAK

Reason:
It’s in the application.

Weak or Strong?

Password:
`305P6N2cEl4nkH!T$1q*`

Answer:
WEAK

Reason:
It's in the application.

```
$ cat decompiled/com/[redacted]/security/SecurityUtil.java
```

Weak or Strong?

Password:
305P6N2cEl4nkH!T$1q*

Answer:
WEAK

Reason:
It's in the application.

```
$ cat decompiled/com/[redacted]/security/SecurityUtil.java
public class SecurityUtil {
    private static final char[] certPass;

    static {
        SecurityUtil.certPass = new char[]{'3', '0', '5', 'P', '6',
'N', '2', 'c', 'E', 'l', '4', 'n', 'k', 'H', '!', 'T', '$', '1', 'q',
'*'};
    }
```

Weak or Strong?

Password:
gaypark@lespark#gzxyindex.php

Weak or Strong?

Password:
`gaypark@lespark#gzxyindex.php`

Answer:
WEAK

Weak or Strong?

Password:
`gaypark@lespark#gzxyindex.php`

Answer:
WEAK

Reason:
It’s in the application.

Weak or Strong?

Password:
`gaypark@lespark#gzxyindex.php`

Answer:
WEAK

Reason:
It's in the application.

Decompiled `liblespark.so` in IDA:

Weak or Strong?

Password:
gaypark@lespark#gzxyindex.php

Answer:
WEAK

Reason:
It's in the application.

Decompiled `liblespark.so` in IDA:

```
EXPORT
Java_com_redwolfama_peonylespark_util_jni_PasswordUtil_getSSLPassword
Java_com_redwolfama_peonylespark_util_jni_PasswordUtil_getSSLPassword
PUSH            {R3,LR}
LDR             R3, [R0]
LDR             R1, =(aGayparkLespark - 0x11FE)
LDR.W           R3, [R3,#0x29C]
ADD             R1, PC  ; "gaypark@lespark#gzxyindex.php"
BLX             R3
POP             {R3,PC}
; End of function
Java_com_redwolfama_peonylespark_util_jni_PasswordUtil_getSSLPassword
```

Weak or Strong?

Password:
gaypark@lespark#gzxyindex.php

Answer:
WEAK

Reason:
It's in the application.

Decompiled `liblespark.so` in IDA:

```
EXPORT
Java_com_redwolfama_peonylespark_util_jni_PasswordUtil_getSSLPassword
Java_com_redwolfama_peonylespark_util_jni_PasswordUtil_getSSLPassword
PUSH            {R3,LR}
LDR             R3, [R0]
LDR             R1, =(aGayparkLespark - 0x11FE)
LDR.W           R3, [R3,#0x29C]
ADD             R1, PC  ; "gaypark@lespark#gzxyindex.php"
BLX             R3
POP             {R3,PC}
; End of function
Java_com_redwolfama_peonylespark_util_jni_PasswordUtil_getSSLPassword
```

Weak or Strong?

Password:
`T.G-8~KvkNwvB&7g`

Weak or Strong?

Password:
`T.G-8~KvkNwvB&7g`

Answer:
STRONG

Weak or Strong?

Password:
`T.G-8~KvkNwvB&7g`

Answer:
STRONG

Reasons:

- It is long enough
- It uses a large character set
- It has no patterns recognizable by a cracker

Keys used by apps

If a private key is contained within and used by an application...

Keys used by apps

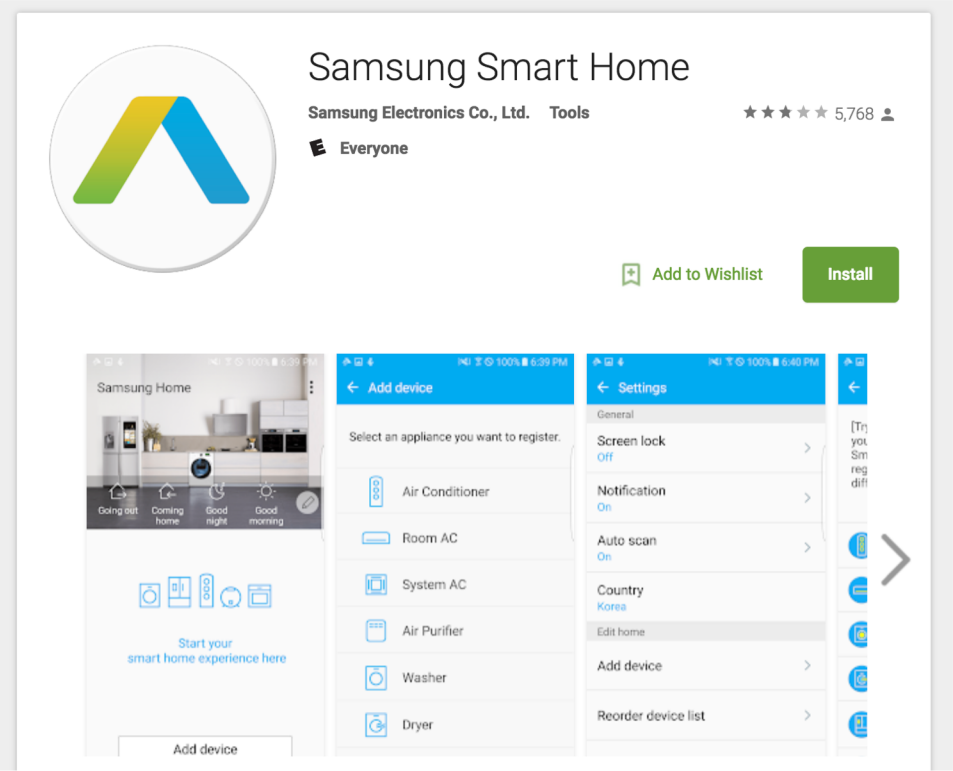
If a private key is contained within and used by an application...

it doesn’t matter how strong you’ve made it!

Keep it Like a Secret: When Android Apps Contain Private Keys

Case Study: Samsung SmartHome

Samsung SmartHome



Samsung SmartHome

ADDITIONAL INFORMATION

Updated
February 5, 2018

Installs
1,000,000 - 5,000,000

Current Version
3.1072.19.203

Requires Android
4.0 and up

Content Rating
Everyone
Learn more

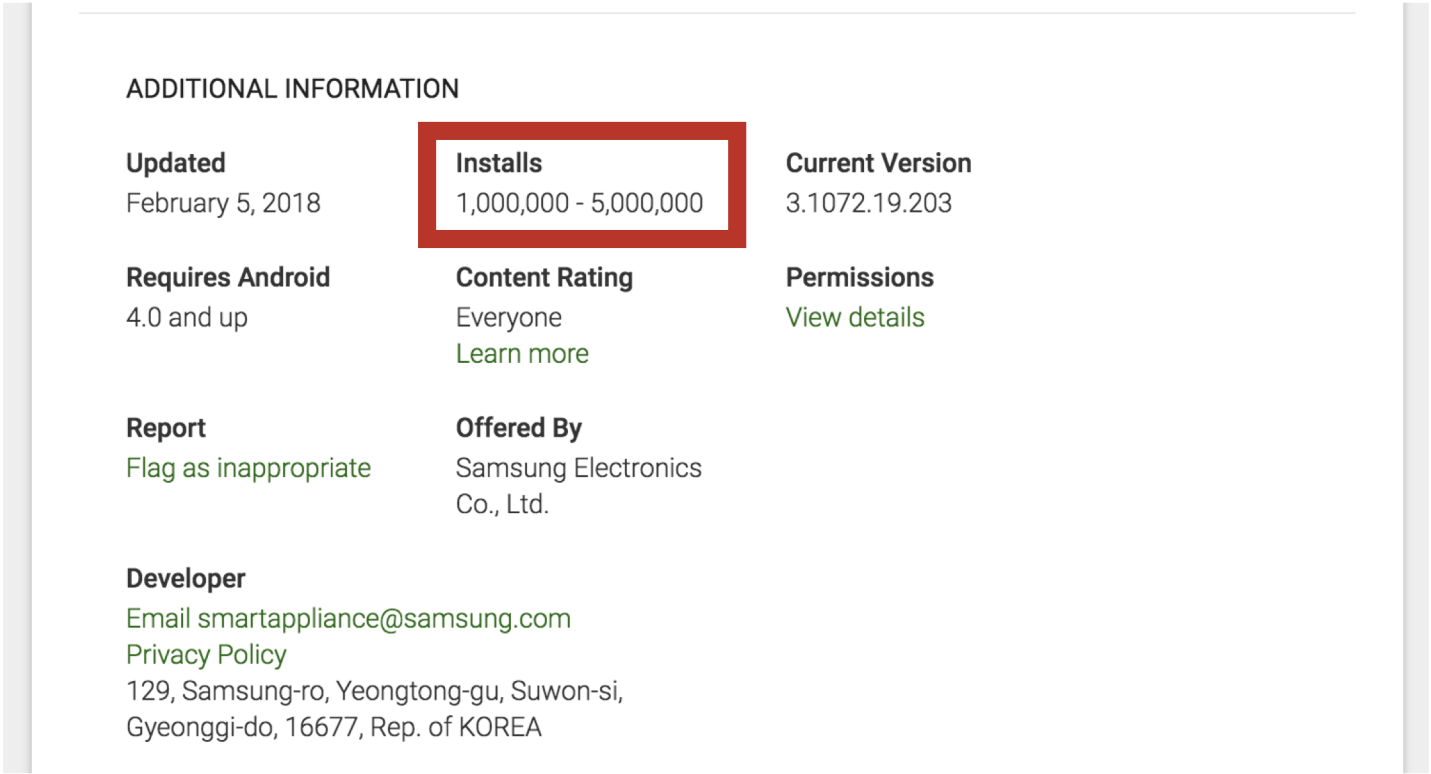
Permissions
View details

Report
Flag as inappropriate

Offered By
Samsung Electronics Co., Ltd.

Developer
Email smartappliance@samsung.com
Privacy Policy
129, Samsung-ro, Yeongtong-gu, Suwon-si, Gyeonggi-do, 16677, Rep. of KOREA

Samsung SmartHome



ADDITIONAL INFORMATION

Updated
February 5, 2018

Installs
1,000,000 - 5,000,000

Current Version
3.1072.19.203

Requires Android
4.0 and up

Content Rating
Everyone
Learn more

Permissions
View details

Report
Flag as inappropriate

Offered By
Samsung Electronics Co., Ltd.

Developer
Email smartappliance@samsung.com
Privacy Policy
129, Samsung-ro, Yeongtong-gu, Suwon-si, Gyeonggi-do, 16677, Rep. of KOREA

Finding a private key used by a popular app

```
sqlite> select file, keyused, dlcount from keys inner join apks on keys.appname = apks.appname
where file like '%COMMON14K_M_priv%' order by apks.dlcount desc;
```

Finding a private key used by a popular app

```
sqlite> select file, keyused, dlcount from keys inner join apks on keys.appname = apks.appname
where file like '%COMMON14K_M_priv%' order by apks.dlcount desc limit 10;
keys/com.samsung.smarthome/assets/certificates/COMMON14K_M_priv.pem|0|1000000
keys/com.samsung.smarthome.refrigerator/assets/certificates/COMMON14K_M_priv.pem|1|10000
keys/com.samsung.smarthome.dvm/assets/certificates/COMMON14K_M_priv.pem|0|10000
keys/com.samsung.dacorsmarthome/assets/certificates/COMMON14K_M_priv.pem|0|500
sqlite>
```

Finding a private key used by a popular app

```
sqlite> select file, keyused, dlcount from keys inner join apks on keys.appname = apks.appname
where file like '%COMMON14K_M_priv%' order by apks.dlcount desc limit 10;
keys/com.samsung.smarthome/assets/certificates/COMMON14K_M_priv.pem|0|1000000
keys/com.samsung.smarthome.refrigerator/assets/certificates/COMMON14K_M_priv.pem|1|10000
keys/com.samsung.smarthome.dvm/assets/certificates/COMMON14K_M_priv.pem|0|10000
keys/com.samsung.dacorsmarthome/assets/certificates/COMMON14K_M_priv.pem|0|500
sqlite>
```

Finding the reference to the private key

```
$ grep -rl COMMON14K_M_priv.pem .
```

Finding the reference to the private key

```
$ grep -rl COMMON14K_M_priv.pem .
decompiled/com/samsung/smarthome/refrigerator/SmartHomeLauncherActivity.java
smali/com/samsung/smarthome/refrigerator/SmartHomeLauncherActivity.smali
original/META-INF/MANIFEST.MF
original/META-INF/CERT.SF
strings.txt
$
```

Finding the reference to the private key

```
$ grep -rl COMMON14K_M_priv.pem .
decompiled/com/samsung/smarthome/refrigerator/SmartHomeLauncherActivity.java
smali/com/samsung/smarthome/refrigerator/SmartHomeLauncherActivity.smali
original/META-INF/MANIFEST.MF
original/META-INF/CERT.SF
strings.txt
$
```

Finding the reference to the private key

```
$ cat decompiled/com/samsung/smarthome/refrigerator/SmartHomeLauncherActivity.java
…
 String[] v0_1 = new String[4];
        v0_1[0] = "DHE-RSA-AES128-SHA";
        v0_1[1] = "DHE-RSA-AES256-SHA";
        v0_1[v4] = "AES128-SHA";
        v0_1[3] = "AES256-SHA";
        SSLswitch.setServerCiphers(v0_1);
        SSLswitch.setClientCiphers(v0_1);
        SSLswitch.setRunSSL(this.getApplicationContext(), true);
        SSLswitch.setShp_certificates("ca-certificates_20151109.crt");
        SSLswitch.setCertChain("CertChain.pem");
        SSLswitch.setMyCert("COMMON14K_M_CertChain.pem");
        SSLswitch.setPrivateKey("COMMON14K_M_priv.pem");
        SSLswitch.setPassword("KKJSB04XvDQMC0rm2LHTHhAyK3iBVfpHBuQCXwKEYN4=");
        SSLswitch.setDHParam("dh_param_1024.pem")
```

Finding the reference to the private key

```
$ cat decompiled/com/samsung/smarthome/refrigerator/SmartHomeLauncherActivity.java
…
 String[] v0_1 = new String[4];
        v0_1[0] = "DHE-RSA-AES128-SHA";
        v0_1[1] = "DHE-RSA-AES256-SHA";
        v0_1[v4] = "AES128-SHA";
        v0_1[3] = "AES256-SHA";
        SSLswitch.setServerCiphers(v0_1);
        SSLswitch.setClientCiphers(v0_1);
        SSLswitch.setRunSSL(this.getApplicationContext(), true);
        SSLswitch.setShp_certificates("ca-certificates_20151109.crt");
        SSLswitch.setCertChain("CertChain.pem");
        SSLswitch.setMyCert("COMMON14K_M_CertChain.pem");
        SSLswitch.setPrivateKey("COMMON14K_M_priv.pem");
        SSLswitch.setPassword("KKJSB04XvDQMC0rm2LHTHhAyK3iBVfpHBuQCXwKEYN4=");
        SSLswitch.setDHParam("dh_param_1024.pem")
```

Finding the reference to the private key

```
$ echo 'KKJSB04XvDQMC0rm2LHTHhAyK3iBVfpHBuQCXwKEYN4=' | base64 -D
```

Finding the reference to the private key

```
$ echo 'KKJSB04XvDQMC0rm2LHTHhAyK3iBVfpHBuQCXwKEYN4=' | base64 -D
(RN4

      J+2x U G _ `
```

Finding the reference to the private key

```
$ cat decompiled/com/samsung/smarthome/refrigerator/SmartHomeLauncherActivity.java
…
 String[] v0_1 = new String[4];
        v0_1[0] = "DHE-RSA-AES128-SHA";
        v0_1[1] = "DHE-RSA-AES256-SHA";
        v0_1[v4] = "AES128-SHA";
        v0_1[3] = "AES256-SHA";
        SSLswitch.setServerCiphers(v0_1);
        SSLswitch.setClientCiphers(v0_1);
        SSLswitch.setRunSSL(this.getApplicationContext(), true);
        SSLswitch.setShp_certificates("ca-certificates_20151109.crt");
        SSLswitch.setCertChain("CertChain.pem");
        SSLswitch.setMyCert("COMMON14K_M_CertChain.pem");
        SSLswitch.setPrivateKey("COMMON14K_M_priv.pem");
        SSLswitch.setPassword("KKJSB04XvDQMC0rm2LHTHhAyK3iBVfpHBuQCXwKEYN4=");
        SSLswitch.setDHParam("dh_param_1024.pem")
```

Decoding the key password

```
$ cat decompiled/com/sec/smarthome/framework/security/SSLswitch.java
```

Decoding the key password

```
$ cat decompiled/com/sec/smarthome/framework/security/SSLswitch.java
…

public static void setPassword(String arg1) {
    SSLswitch.password = PswGenerator.decryptPBE(arg1);
}
```

Decoding the key password

```
$ cat decompiled/com/sec/smarthome/framework/security/SSLswitch.java
…

public static void setPassword(String arg1) {
    SSLswitch.password = PswGenerator.decryptPBE(arg1);
}
```

Decoding the key password

```
$ cat decompiled/com/sec/smarthome/framework/security/SSLswitch.java
…

public static void setPassword(String arg1) {
    SSLswitch.password = PswGenerator.decryptPBE(arg1);
}
```

Decoding the key password

```
$ cat decompiled/com/sec/smarthome/framework/security/PswGenerator.java
// Decompiled by JEB v2.3.10.201802032133

package com.sec.smarthome.framework.security;

…

public static String decryptPBE(String arg9) {
        String v0_5;
        byte[] v1_1;
        String v4 = null;
        int v3 = 0;
        byte[] v6 = PswGenerator.createKey(PswGenerator.salt, PswGenerator.key_data);
…
```

Decoding the key password

```
$ cat decompiled/com/sec/smarthome/framework/security/PswGenerator.java
// Decompiled by JEB v2.3.10.201802032133

package com.sec.smarthome.framework.security;

…

public static String decryptPBE(String arg9) {
        String v0_5;
        byte[] v1_1;
        String v4 = null;
        int v3 = 0;
        byte[] v6 = PswGenerator.createKey(PswGenerator.salt, PswGenerator.key_data);
…
```

Decoding the key password

```
$ cat decompiled/com/sec/smarthome/framework/security/PswGenerator.java
// Decompiled by JEB v2.3.10.201802032133

package com.sec.smarthome.framework.security;

…

public static String decryptPBE(String arg9) {
        String v0_5;
        byte[] v1_1;
        String v4 = null;
        int v3 = 0;
        byte[] v6 = PswGenerator.createKey(PswGenerator.salt, PswGenerator.key_data);
…
```

VERY COMPLICATED

Decoding the key password

```
$ cat decompiled/com/sec/smarthome/framework/security/PswGenerator.java
// Decompiled by JEB v2.3.10.201802032133

package com.sec.smarthome.framework.security;

…

public static String decryptPBE(String arg9) {
        String v0_5;
        byte[] v1_1;
        String v4 = null;
        int v3 = 0;
        byte[] v6 = PswGenerator.createKey(PswGenerator.salt, PswGenerator.key_data);
…
```

VERY COMPLICATED

... but it doesn't matter!

Using the existing code

```
$ cat PswGenerator.Java.diff
--- PswGenerator.orig.java  2018-02-16 14:20:53.930636617 -0500
+++ PswGenerator.java   2018-02-20 13:17:30.717678481 -0500
@@ -1,8 +1,8 @@
 // Decompiled by JEB v2.3.10.201802032133

…



+
 public class PswGenerator {
+
+  public static void main(String[] args)
+  {
+    String pw;
+    System.out.println("Attempting to decode 'KKJSB04XvDQMC0rm2LHTHhAyK3iBVfpHBuQCXwKEYN4='...");
+    pw = PswGenerator.decryptPBE("KKJSB04XvDQMC0rm2LHTHhAyK3iBVfpHBuQCXwKEYN4=");
+    System.out.printf("\nDecoded password: %s\n", pw);
+   }
+
```

Using the existing code

```
$ cat PswGenerator.Java.diff
--- PswGenerator.orig.java  2018-02-16 14:20:53.930636617 -0500
+++ PswGenerator.java   2018-02-20 13:17:30.717678481 -0500
@@ -1,8 +1,8 @@
 // Decompiled by JEB v2.3.10.201802032133

…


+
 public class PswGenerator {
+
+  public static void main(String[] args)
+  {
+    String pw;
+    System.out.println("Attempting to decode 'KKJSB04XvDQMC0rm2LHTHhAyK3iBVfpHBuQCXwKEYN4='...");
+    pw = PswGenerator.decryptPBE("KKJSB04XvDQMC0rm2LHTHhAyK3iBVfpHBuQCXwKEYN4=");
+    System.out.printf("\nDecoded password: %s\n", pw);
+   }
+
```

Running their code to get the password

```
$ javac PswGenerator.java
```

Running their code to get the password

```
$ javac PswGenerator.java
$ java PswGenerator
```

Running their code to get the password

```
$ javac PswGenerator.java
$ java PswGenerator
Attempting to decode 'KKJSB04XvDQMC0rm2LHTHhAyK3iBVfpHBuQCXwKEYN4='...

Decoded password: 1%^t[redacted]
$
```

Running their code to get the password

```
$ javac PswGenerator.java
$ java PswGenerator
Attempting to decode 'KKJSB04XvDQMC0rm2LHTHhAyK3iBVfpHBuQCXwKEYN4='...

Decoded password: 1%^t[redacted]
$
```

Running their code to get the password

```
$ javac PswGenerator.java
$ java PswGenerator
Attempting to decode 'KKJSB04XvDQMC0rm2LHTHhAyK3iBVfpHBuQCXwKEYN4='...

Decoded password: 1%^t[redacted]
$
```

Password is 16 characters and complicated

Decrypting the key

```
$ openssl pkey -in assets/certificates/COMMON14K_M_priv.pem -passin pass:1%^t
```

Decrypting the key

```
$ openssl pkey -in assets/certificates/COMMON14K_M_priv.pem -passin pass:1%^t
-----BEGIN PRIVATE KEY-----
MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQCq48EeXQVO31cA
YIHHYrIB4dAepFvvmWAI1lN1v214Hhq1X+0hhXE8iNX7Kn1kGkBLfiu3nxTvb+fB
tL5ZifIJDbBY2ZoXwq4pfb6cwkAlLgZ1grZ4BY7LZp5eMtQwOdbPCL8ySjrmDca8
YW70qZAZlGWY+1NGV4ydGih3t6OwGSOopwt6dptX0lzV+euKsy5pAJNIJ7JgkR/d
uMcNNK4tYaPyAHVdmcxxJw+e33AdcWCZIGGFbfbAVuNl+05B93lxZHF3jM/uhL0l
/ei3uG8febIDb73mEWD8g3obVeaImU6+cm7KyFCP1Pwlw3xjAV90jqJkIrNXfk92
zqSll0XfAgMBAAECggEAQDqKAxeDPznCQWoEXSRDM2HkDyBHTa6f4bJse/0O7+6g
…
jmbxYeD9BQKBgQDP6gT3G3lju+rVTzvLoYl6DJ3urI2j8QlZ7y/koqiKnSnbu5a7
Y2hq0lVsHMdbjX+roVphrbAEUmoDpxO5KaJrfW2VKMcYSAvjXR/JAH1NPneSjDgp
srgog/yrJvMXAXIEIe245KcC4UKqbDGV8GP7/XZJYE+W33Yls8d81Y2vRQKBgEoc
s0I0WYMQtzK0qTZzPpv6XmfSnTmxi6Lyre/Xl3q1QELsWGXrt1S9Ge7t8Ie0tNPT
5d/50tSgWED4hLv59yV8gVNu0bLlfSQbbU0+PG4aeeO7mMJU+khjccvhB03Sbk8R
7yWlSImPF8a2mbA6OPsOyILycVKYYprI6MeOJP6pAoGAD38lqdZk5azOnuu0/mC9
KT2GmbdlUENz3FIfnjzdE1a766+dovDEnfZxXCAXi/Ycy87ABX/kwroD8zpec6tl
oxfi+WwlI3dAJgemYpe71po/HJaNBV3V2rLyZ5ur7vI8Ya/d36ldg4vea5dXzK5d
yzkb0/Odq8A00aztCY1eWgU=
-----END PRIVATE KEY-----
```

Getting the public key from the private key

```
$ openssl pkey -in assets/certificates/COMMON14K_M_priv.pem -passin pass:1%^t[redacted] -pubout
```

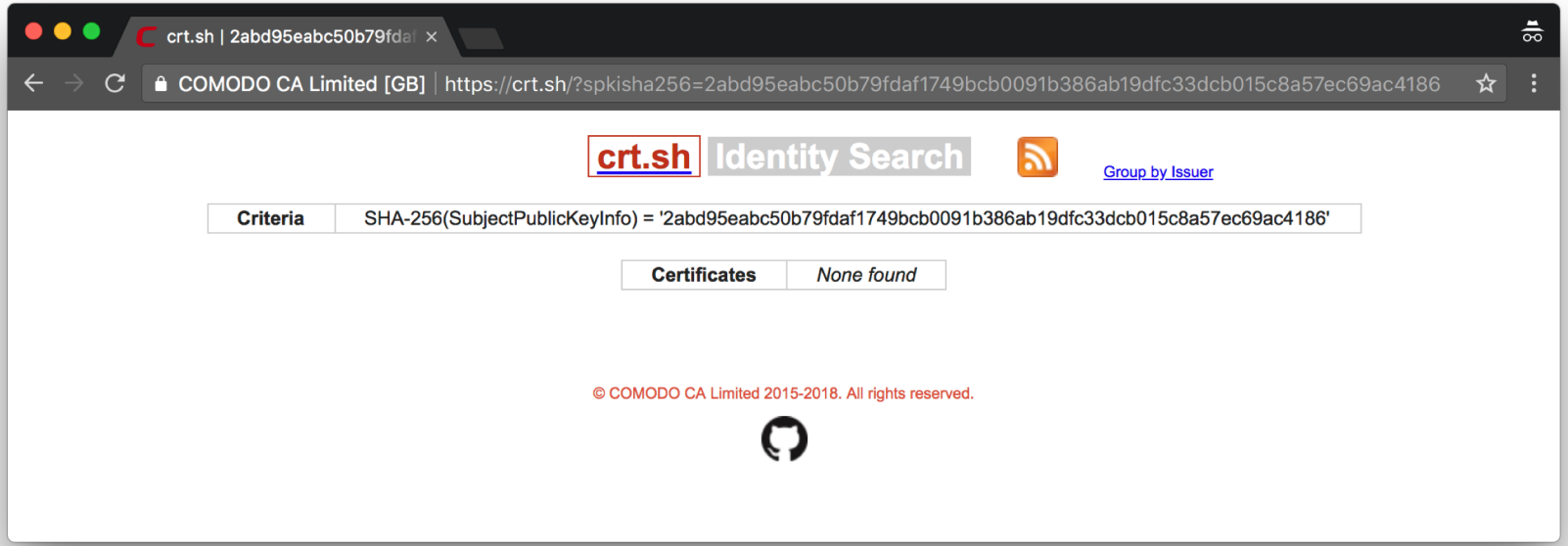
Getting the public key from the private key

```
$ openssl pkey -in assets/certificates/COMMON14K_M_priv.pem -passin pass:1%^t          -pubout
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAquPBHl0FTt9XAGCBx2Ky
AeHQHqRb75lgCNZTdb9teB4atV/tIYVxPIjV+yp9ZBpAS34rt58U72/nwbS+WYny
CQ2wWNmaF8KuKX2+nMJAJS4GdYK2eAWOy2aeXjLUMDnWzwi/Mko65g3GvGFu9KmQ
GZRlmPtTRleMnRood7ejsBkjqKcLenabV9Jc1fnrirMuaQCTSCeyYJEf3bjHDTSu
LWGj8gB1XZnMcScPnt9wHXFgmSBhhW32wFbjZftOQfd5cWRxd4zP7oS9Jf3ot7hv
H3myA2+95hFg/IN6G1XmiJlOvnJuyshQj9T8JcN8YwFfdI6iZCKzV35Pds6kpZdF
3wIDAQAB
-----END PUBLIC KEY-----
```

Getting the public key hash

```
$ openssl pkey -in assets/certificates/COMMON14K_M_priv.pem -passin pass:1%^t[redacted] -pubout
-outform der | sha256sum
2abd95eabc50b79fdaf1749bcb0091b386ab19dfc33dcb015c8a57ec69ac4186  -
```

Checking if public key hash is in crt.sh



Comparing the Private Key Modulus to the Public one

```
$ openssl pkey -in assets/certificates/COMMON14K_M_priv.pem -passin pass:1%^t[redacted] \
| openssl rsa -noout -modulus | openssl sha256
```

Comparing the Private Key Modulus to the Public one

```
$ openssl pkey -in assets/certificates/COMMON14K_M_priv.pem -passin pass:1%^t[redacted] \
| openssl rsa -noout -modulus | openssl sha256
(stdin)= 1c03a8197edd52063a9694ee76d789ca2a53fc6962638831e3af2f0ff281405a
```

Comparing the Private Key Modulus to the Public one

```
$ openssl pkey -in assets/certificates/COMMON14K_M_priv.pem -passin pass:1%^t[redacted] \
| openssl rsa -noout -modulus | openssl sha256
(stdin)= 1c03a8197edd52063a9694ee76d789ca2a53fc6962638831e3af2f0ff281405a
$ openssl x509 -noout -modulus -in assets/certificates/COMMON14K_M_CertChain.pem \
| openssl sha256
```

Comparing the Private Key Modulus to the Public one

```
$ openssl pkey -in assets/certificates/COMMON14K_M_priv.pem -passin pass:1%^t[redacted] \
| openssl rsa -noout -modulus | openssl sha256
(stdin)= 1c03a8197edd52063a9694ee76d789ca2a53fc6962638831e3af2f0ff281405a
$ openssl x509 -noout -modulus -in assets/certificates/COMMON14K_M_CertChain.pem \
| openssl sha256
(stdin)= 1c03a8197edd52063a9694ee76d789ca2a53fc6962638831e3af2f0ff281405a
```

Comparing the Private Key Modulus to the Public one

```
$ openssl pkey -in assets/certificates/COMMON14K_M_priv.pem -passin pass:1%^t[redacted] \
| openssl rsa -noout -modulus | openssl sha256
(stdin)= 1c03a8197edd52063a9694ee76d789ca2a53fc6962638831e3af2f0ff281405a
$ openssl x509 -noout -modulus -in assets/certificates/COMMON14K_M_CertChain.pem \
| openssl sha256
(stdin)= 1c03a8197edd52063a9694ee76d789ca2a53fc6962638831e3af2f0ff281405a
```

Comparing the Private Key Modulus to the Public one

```
$ openssl pkey -in assets/certificates/COMMON14K_M_priv.pem -passin pass:1%^t[redacted] \
| openssl rsa -noout -modulus | openssl sha256
(stdin)= 1c03a8197edd52063a9694ee76d789ca2a53fc6962638831e3af2f0ff281405a
$ openssl x509 -noout -modulus -in assets/certificates/COMMON14K_M_CertChain.pem \
| openssl sha256
(stdin)= 1c03a8197edd52063a9694ee76d789ca2a53fc6962638831e3af2f0ff281405a
```

Comparing the Private Key Modulus to the Public one

```
$ openssl pkey -in assets/certificates/COMMON14K_M_priv.pem -passin pass:1%^t[redacted] \
| openssl rsa -noout -modulus | openssl sha256
(stdin)= 1c03a8197edd52063a9694ee76d789ca2a53fc6962638831e3af2f0ff281405a
$ openssl x509 -noout -modulus -in assets/certificates/COMMON14K_M_CertChain.pem \
| openssl sha256
(stdin)= 1c03a8197edd52063a9694ee76d789ca2a53fc6962638831e3af2f0ff281405a
```

The modulus hashes match, so the certificate is for the private key!

Getting certificate details

```
$ openssl x509 -text -in assets/certificates/COMMON14K_M_CertChain.pem
```

Getting certificate details

```
$ openssl x509 -text -in assets/certificates/COMMON14K_M_CertChain.pem

Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 10 (0xa)
    Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=KR, O=Samsung Electronics, CN=RemoteAccessCA(CE)
        Validity
            Not Before: Jan  1 00:00:00 1960 GMT
            Not After : Jan  1 00:00:00 2060 GMT
        Subject: C=KR, O=Samsung Electronics, CN=COMMON14K_M/emailAddress=COMMON14K_M@samsung.com

…

            X509v3 Basic Constraints:
                CA:TRUE
            X509v3 Subject Alternative Name:
                DNS:samsung.com, DNS:localhost
```

Getting certificate details

```
$ openssl x509 -text -in assets/certificates/COMMON14K_M_CertChain.pem

Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 10 (0xa)
    Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=KR, O=Samsung Electronics, CN=RemoteAccessCA(CE)
        Validity
            Not Before: Jan  1 00:00:00 1960 GMT
            Not After : Jan  1 00:00:00 2060 GMT
        Subject: C=KR, O=Samsung Electronics, CN=COMMON14K_M/emailAddress=COMMON14K_M@samsung.com

…

            X509v3 Basic Constraints:
                CA:TRUE
            X509v3 Subject Alternative Name:
                DNS:samsung.com, DNS:localhost
```

Getting certificate details

```
$ openssl x509 -text -in assets/certificates/COMMON14K_M_CertChain.pem

Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 10 (0xa)
    Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=KR, O=Samsung Electronics, CN=RemoteAccessCA(CE)
        Validity
            Not Before: Jan  1 00:00:00 1960 GMT
            Not After : Jan  1 00:00:00 2060 GMT
        Subject: C=KR, O=Samsung Electronics, CN=COMMON14K_M/emailAddress=COMMON14K_M@samsung.com

…

            X509v3 Basic Constraints:
                CA:TRUE
            X509v3 Subject Alternative Name:
                DNS:samsung.com, DNS:localhost
```

Samsung private key conclusions

- A private key for a certificate authority (CA) is included in multiple Android apps

Samsung private key conclusions

- A private key for a certificate authority (CA) is included in multiple Android apps
- The certificate for said key can also be used to identify both “localhost” and “samsung.com” hosts

Samsung private key conclusions

- A private key for a certificate authority (CA) is included in multiple Android apps
- The certificate for said key can also be used to identify both “localhost” and “samsung.com” hosts

Don’t worry too much, as the certificate isn’t trusted by OS/browser versions I’ve seen.

Private Key Details From Samsung

- Older appliances and low-end Android devices do not have a trusted execution environment (TEE), such as TrustZone. This required that the key be included in the Android application.

Private Key Details From Samsung

- Older appliances and low-end Android devices do not have a trusted execution environment (TEE), such as TrustZone. This required that the key be included in the Android application.
- Samsung have applied protection methods including obfuscation/white-boxing/rooting detection to the application. Even with access the the private key, physical access to a device is required to complete pairing with the victim's device.

Private Key Details From Samsung

- Older appliances and low-end Android devices do not have a trusted execution environment (TEE), such as TrustZone. This required that the key be included in the Android application.
- Samsung have applied protection methods including obfuscation/white-boxing/rooting detection to the application. Even with access the the private key, physical access to a device is required to complete pairing with the victim's device.
- One of the old plug-in's for Smarthome app supporting older models lacked the protection mechanisms to protect sensitive information including the private key. So Samsung applied protections to protect the private key updated to app store.

Private Key Details From Samsung

- Older appliances and low-end Android devices do not have a trusted execution environment (TEE), such as TrustZone. This required that the key be included in the Android application.
- Samsung have applied protection methods including obfuscation/white-boxing/rooting detection to the application. Even with access the the private key, physical access to a device is required to complete pairing with the victim's device.
- One of the old plug-in's for Smarthome app supporting older models lacked the protection mechanisms to protect sensitive information including the private key. So Samsung applied protections to protect the private key updated to app store.
- Newer devices use SmartThings, which uses a trusted execution environment (TEE, Secure Element, or Artik) to protect keys securely.

Private Key Details From Samsung

- Older appliances and low-end Android devices do not have a trusted execution environment (TEE), such as TrustZone. This required that the key be included in the Android application.
- Samsung have applied protection methods including obfuscation/white-boxing/rooting detection to the application. Even with access the the private key, physical access to a device is required to complete pairing with the victim's device.
- One of the old plug-in's for Smarthome app supporting older models lacked the protection mechanisms to protect sensitive information including the private key. So Samsung applied protections to protect the private key updated to app store.
- Newer devices use SmartThings, which uses a trusted execution environment (TEE, Secure Element, or Artik) to protect keys securely.
- Samsung also implemented their IoT services in accordance with the common IoT standard, based on device unique certificates and Mutual SSL.

Things App Authors Say

Things App Authors Say

Things App Authors Say

```
I left this file (res/raw/clientv01.bks) in the actual app only to confuse a potential hacker who
would try to hack my app.
```

Things App Authors Say

```
Indeed, there is a static certificate saved in the app code. However, the reason for this is that
the app is a *honeypot* and the key is needed for emulating a "valid" HTTPS connection.
```

Things App Authors Say

```
Hola, muchas gracias por su correo.

Tengo alguna pregunta que espero pueda responderme:

1) Mi aplicación es muy sencilla, se basa en html + Jquery y está compilada con
Phonegapp. No esoty seguro de tener los conocimientos como para poder
solucionar el problema. ¿Cómo podría arreglar el problema que usted me está
reportando?
2) Si no consigo arreglar el el problema ¿Van a retirar la app de la PlayStore?

La app es gratuita y sin publicidad, es un recurso de lectura, un trabajo
cultural sin ánimo de lucro.
```

Things App Authors Say

```
The keys you refer to were not place by design by me -
they were added by default by the program used - Appinventor.
```

Things App Authors Say

```
The key included in "SSH Server" application is our intention.
This key is for consistence host key verification when user upgrades the app.
```

Things App Authors Say

Things App Authors Say

```
Good Morning,

Thank you very much for your interest in our [REDACTED]

Good luck with your outdated initiatives!
```

Things App Authors Say

```
Thank you so much and Good Job CERT Coordination Center.
```

Private Key Statistics

File Type Counts

File Type	Count
APK (Android applications)	1,701,930

File Type Counts

File Type	Count
APK (Android applications)	1,701,930
PKCS#1/5	549

File Type Counts

File Type	Count
APK (Android applications)	1,701,930
PKCS#1/5	549
PKCS#8	240

File Type Counts

File Type	Count
APK (Android applications)	1,701,930
PKCS#1/5	549
PKCS#8	240
PKCS#12	2,119

File Type Counts

File Type	Count
APK (Android applications)	1,701,930
PKCS#1/5	549
PKCS#8	240
PKCS#12	2,119
Java Keystore (JKS)	3,215

File Type Counts

File Type	Count
APK (Android applications)	1,701,930
PKCS#1/5	549
PKCS#8	240
PKCS#12	2,119
Java Keystore (JKS)	3,215
Bouncy Castle Keystore (BKS) V1	8,450

File Type Counts

File Type	Count
APK (Android applications)	1,701,930
PKCS#1/5	549
PKCS#8	240
PKCS#12	2,119
Java Keystore (JKS)	3,215
Bouncy Castle Keystore (BKS) V1	8,450
Bouncy Castle Keystore (BKS) V2	1,668

File Type Counts

File Type	Count
APK (Android applications)	1,701,930
PKCS#1/5	549
PKCS#8	240
PKCS#12	2,119
Java Keystore (JKS)	3,215
Bouncy Castle Keystore (BKS) V1	8,450
Bouncy Castle Keystore (BKS) V2	1,668
Openvpn (OVPN)	103

File Type Counts

File Type	Count
APK (Android applications)	1,701,930
PKCS#1/5	549
PKCS#8	240
PKCS#12	2,119
Java Keystore (JKS)	3,215
Bouncy Castle Keystore (BKS) V1	8,450
Bouncy Castle Keystore (BKS) V2	1,668
Openvpn (OVPN)	103 (64 unique)

File Type Counts

File Type	Count
APK (Android applications)	1,701,930
PKCS#1/5	549
PKCS#8	240
PKCS#12	2,119
Java Keystore (JKS)	3,215
Bouncy Castle Keystore (BKS) V1	8,450
Bouncy Castle Keystore (BKS) V2	1,668
Openvpn (OVPN)	103 (64 unique)
PGP Private	10

Key Property Counts

Key Property	Count
Private keys	6,180

Key Property Counts

Key Property	Count
Private keys	6,180
Unprotected private keys	650

Key Property Counts

Key Property	Count
Private keys	6,180
Unprotected private keys	650
Keys for certs seen by crt.sh	119

Key Property Counts

Key Property	Count
Private keys	6,180
Unprotected private keys	650
Keys for certs seen by crt.sh	119
Google Play signing private keys	1,948

Key Property Counts

Key Property	Count
Private keys	6,180
Unprotected private keys	650
Keys for certs seen by crt.sh	119
Google Play signing private keys	1,948
Apple Push Services private keys	87

Key Property Counts

Key Property	Count
Private keys	6,180
Unprotected private keys	650
Keys for certs seen by crt.sh	119
Google Play signing private keys	1,948
Apple Push Services private keys	87
Apple iPhone Developer private keys	21

Key Property Counts

Key Property	Count
Private keys	6,180
Unprotected private keys	650
Keys for certs seen by crt.sh	119
Google Play signing private keys	1,948
Apple Push Services private keys	87
Apple iPhone Developer private keys	21
Apple iPhone Enterprise private keys	68

PKCS#12 Password Cracking Statistics

Strategy	Count	Percent
Total	2119	100%

PKCS#12 Password Cracking Statistics

Strategy	Count	Percent
Total	2119	100%
rockyou.txt password list	870	41.4%

PKCS#12 Password Cracking Statistics

Strategy	Count	Percent
Total	2119	100%
rockyou.txt password list	870	41.4%
Strings from app code	729	34.4%

PKCS#12 Password Cracking Statistics

Strategy	Count	Percent
Total	2119	100%
rockyou.txt password list	870	41.4%
Strings from app code	729	34.4%
Manual analysis	18	0.8%

Java Keystore Password Cracking Statistics

Strategy	Count	Percent
Total	3215	100%

Java Keystore Password Cracking Statistics

Strategy	Count	Percent
Total	3215	100%
rockyou.txt password list	453	14.1%

Java Keystore Password Cracking Statistics

Strategy	Count	Percent
Total	3215	100%
rockyou.txt password list	453	14.1%
Strings from app code	35	1.1%

Java Keystore Password Cracking Statistics

Strategy	Count	Percent
Total	3215	100%
rockyou.txt password list	453	14.1%
Strings from app code	35	1.1%
hashcat-naive	1714	53.3%

Conclusions

Conclusions

If an application includes a private key, it:

Conclusions

If an application includes a private key, it:

- might be an accident.

Conclusions

If an application includes a private key, it:

- might be an accident.
- might not have the best design, if it is not an accident.

Conclusions

If an application includes a private key, it:

- might be an accident.
- might not have the best design, if it is not an accident.
- has an impact that depends on what the key is used for.

Recommendations for Google and Apple

If an application includes a private key:

Recommendations for Google and Apple

If an application includes a private key:

- don’t necessarily block the application from being uploaded, but...

Recommendations for Google and Apple

If an application includes a private key:

- don’t necessarily block the application from being uploaded, but...
- warn the user upon submission to the app store.

Recommendations for Google and Apple

If an application includes a private key:

- don’t necessarily block the application from being uploaded, but...
- warn the user upon submission to the app store.

Result: Fewer Mistakes

Contact Information

Presenter
Will Dormann
Senior Vulnerability Analyst, CERT/CC

Email: wd@cert.org
Twitter: @wdormann