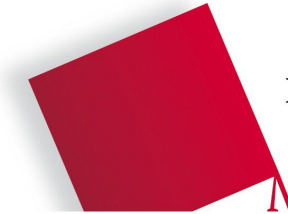
TÉCNICO LISBOA



HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN MÜNCHEN

Mobile Security

Implementation of a secure TODO-App on Android

João Pedro Moreira Sousa & Michael Fuchs

Munich, 12. June 2020

Content

1. Motivation
2. Preliminary considerations
   - 2.1. Mobility aspects
   - 2.2. Safety aspects
3. Scenario
   - 3.1. Architecture
   - 3.2. Threat modelling
   - 3.3. Implementation
4. Evaluation

Designed by stories / Freepik

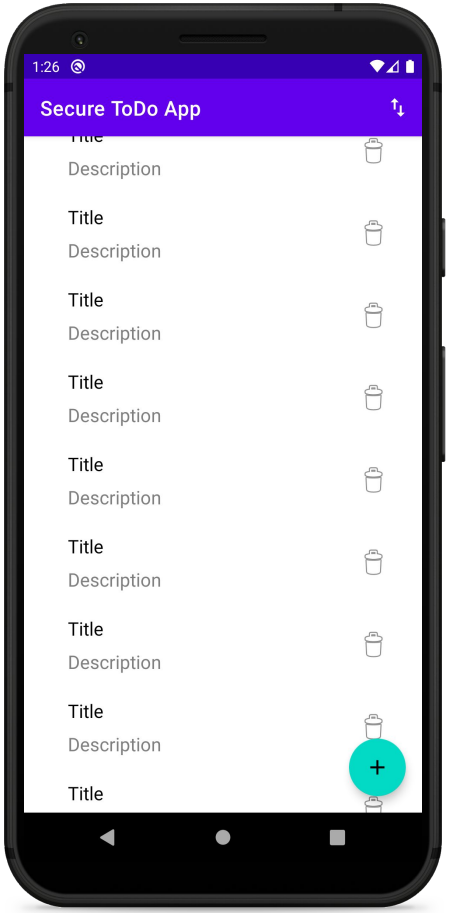
Motivation

Designed by pch.vector / Freepik

Internet is omnipresent in our everyday life

Motivation

- Starting with basic TODO-App
- security measures are iteratively implemented
- ability to backup data in an untrusted environment



Preliminary considerations

Mobility aspects

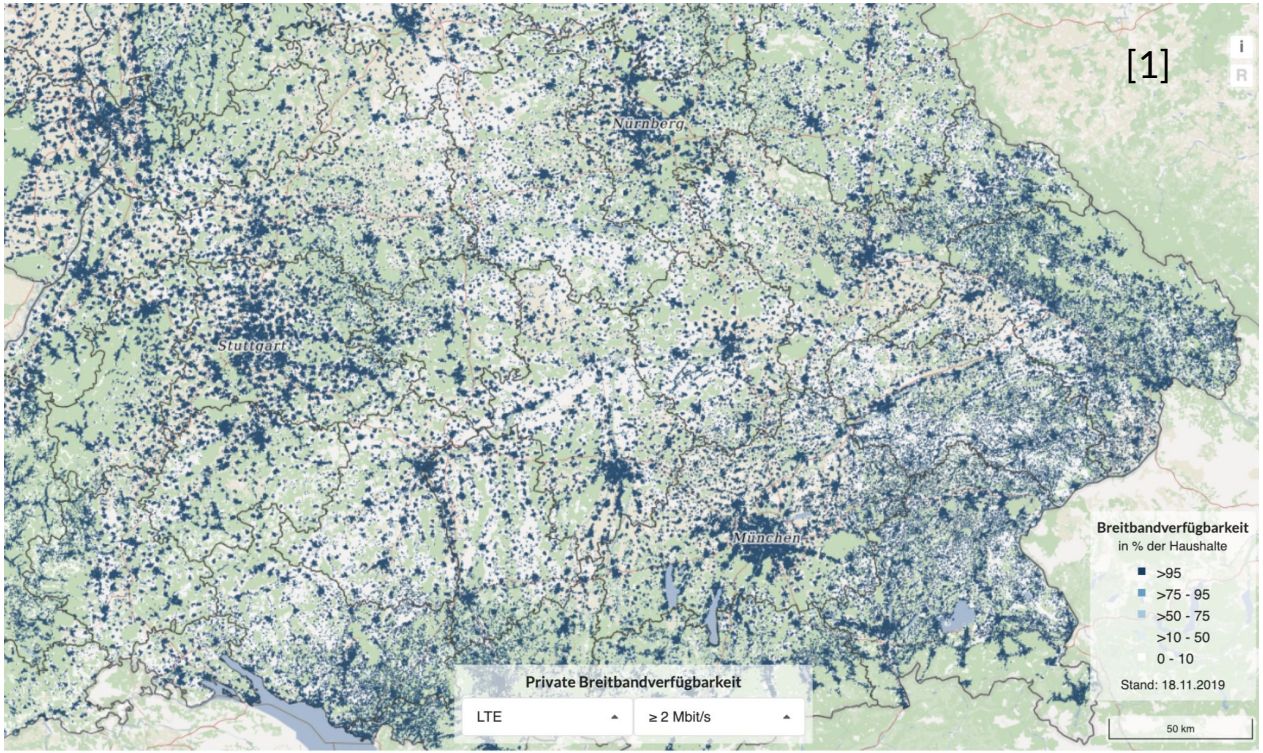
Designed by slidesgo / Freepik

Designed by stories / Freepik

Need for **communication** to reduce computing on smartphones

Preliminary considerations

Mobility aspects



Need for **data storage**: connection is not permanently guaranteed

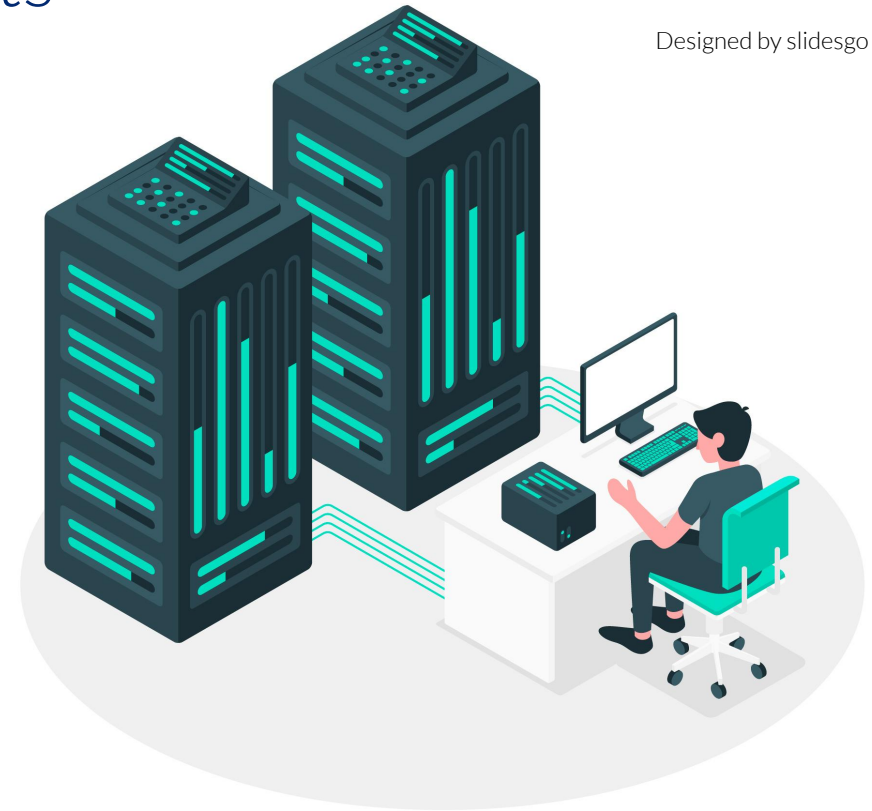
[1] Bundesministerium für Verkehr und digitale Infrastruktur. *BREITBANDATLAS*. 2020. URL: https://www.bmvi.de/DE/Themen/Digitales/Breitbandausbau/Breitbandatlas-Karte/start.html (visited on 05/31/2020).

TÉCNICO LISBOA

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN MÜNCHEN

Safety aspects

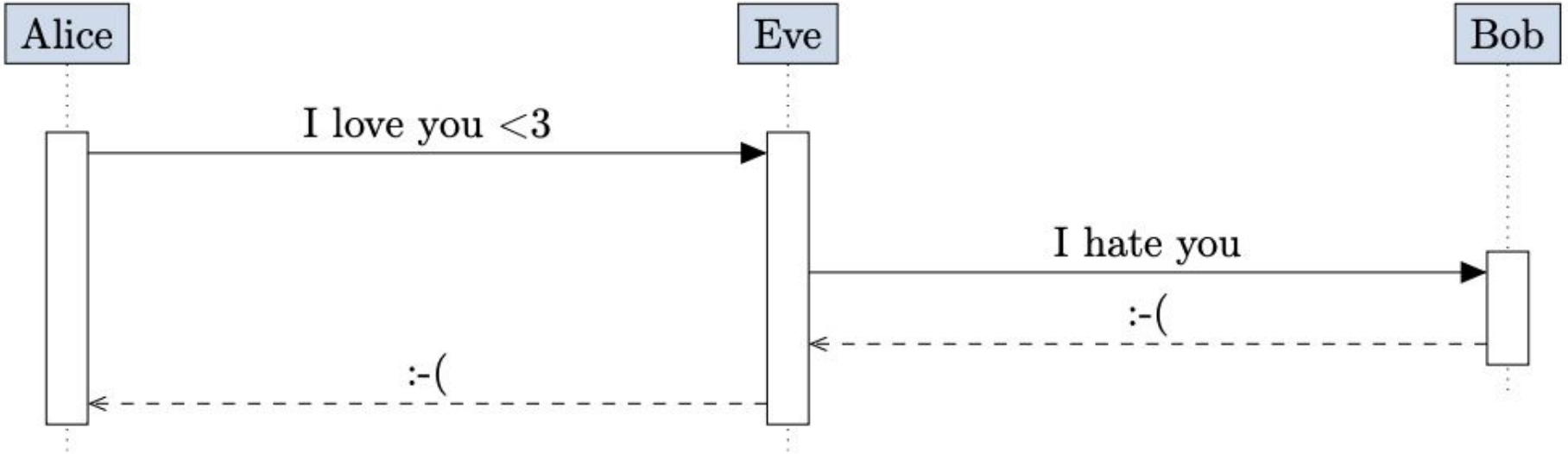
Designed by slidesgo / Freepik



mobile devices collect and handle extensive data from users

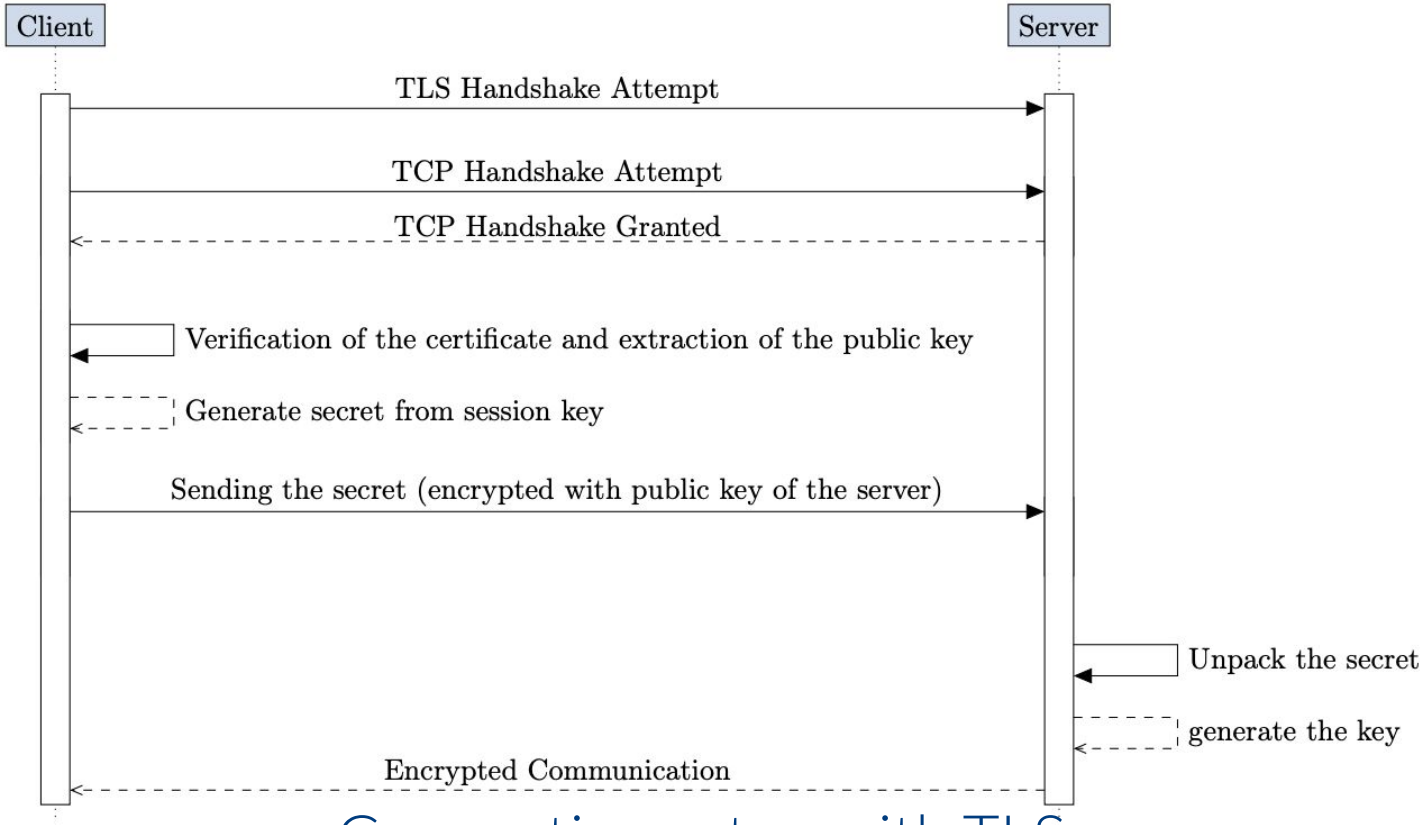
Safety aspects

Communication



Modification of messages by a Man-in-the-Middle attack

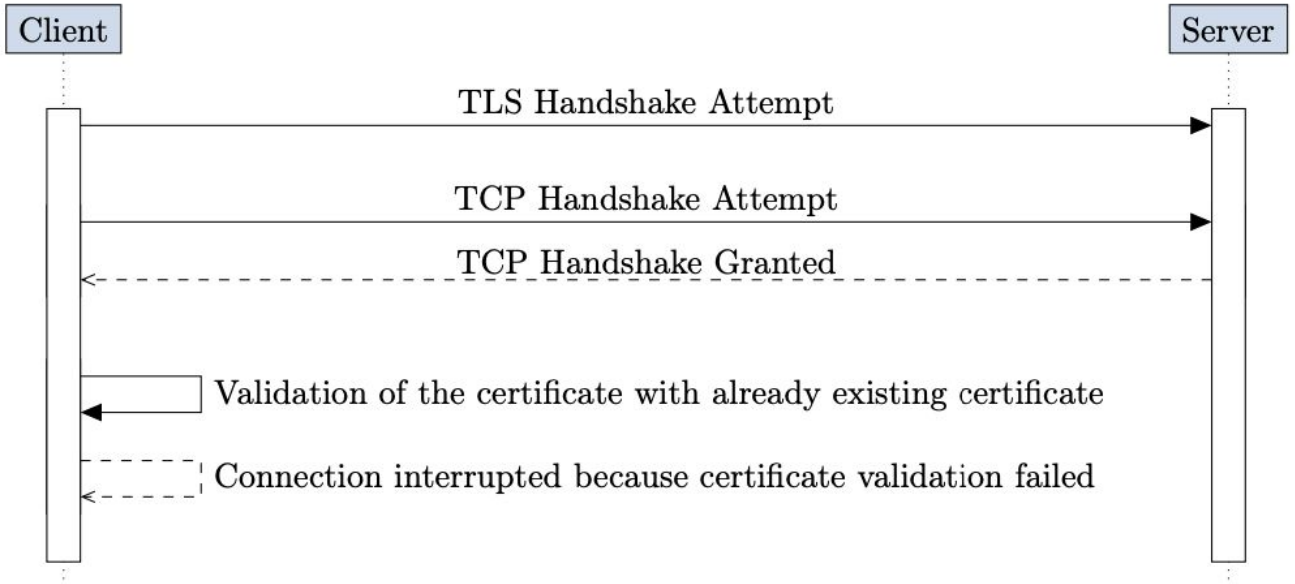
Safety aspects



Connection setup with TLS

Safety aspects

Communication



Procedure of a Man-in-the-Middle attack with **Certificate Pinning**

Safety aspects

Data Storage

effective encryption of data to prevent leak of it

TÉCNICO LISBOA

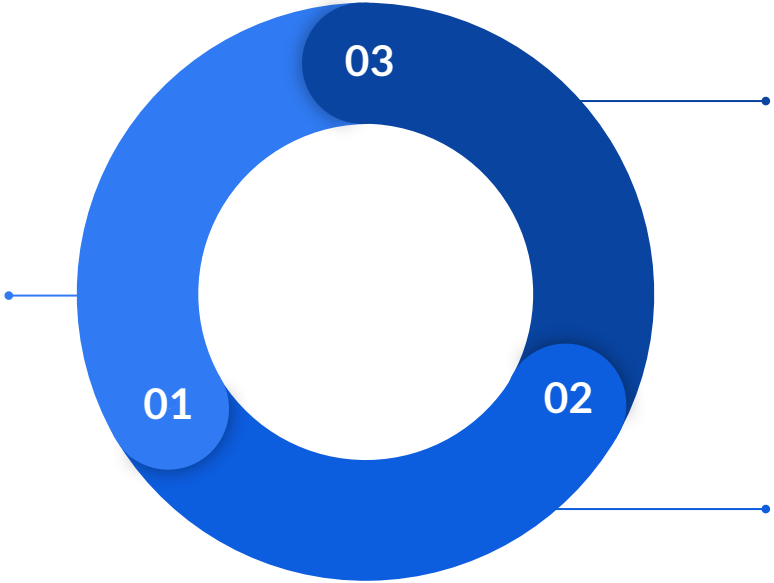
HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN MÜNCHEN

Safety aspects

Data Storage

Device is switched off

- data is decrypted only on startup
- no data in volatile memory
- attack → brute force



Device is on, and enabled

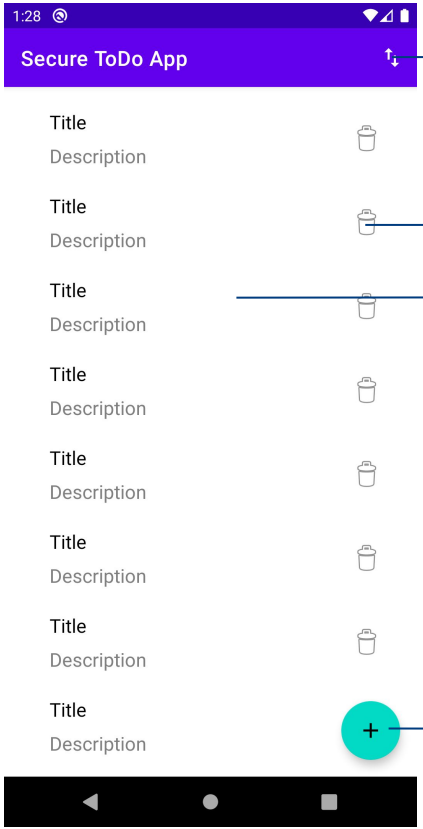
- data is already decrypted
- no other protection present

Device is on, but locked

- data is already decrypted
- lock screen is hurdle, but no sufficient protection

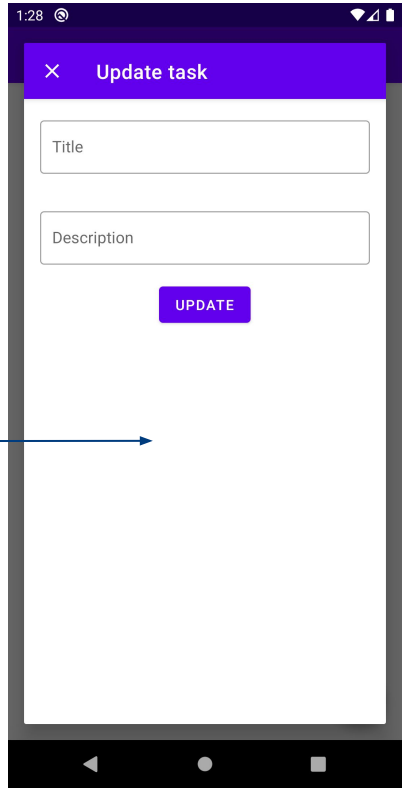
→ data shall be **stored encrypted** by the **application itself**

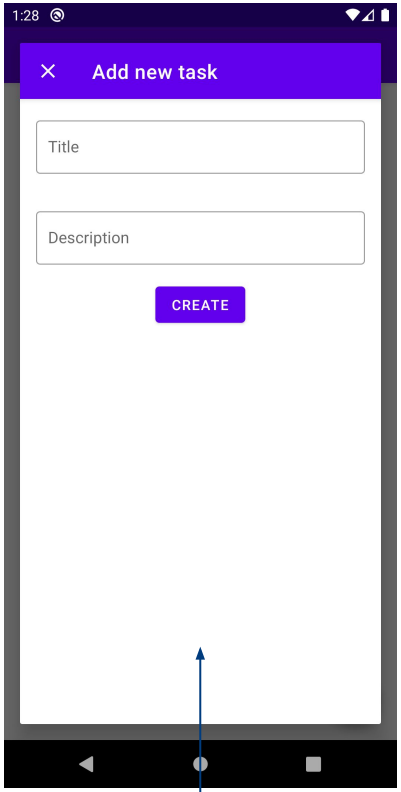
Scenario



Backup

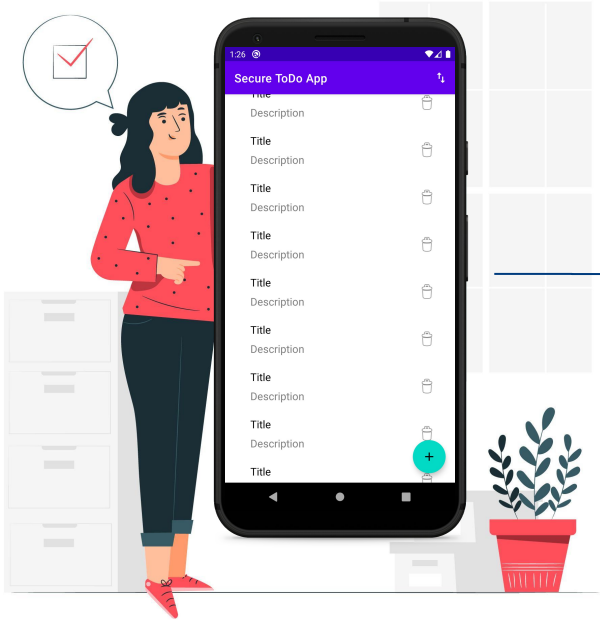
Delete



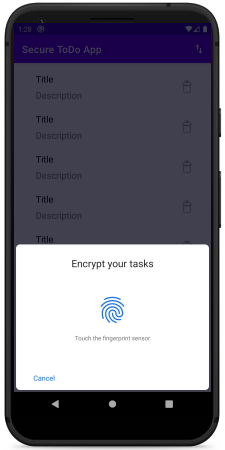


Activity tree of the app

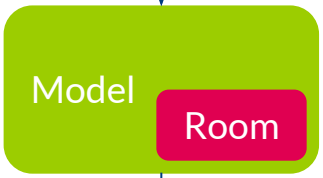
Scenario



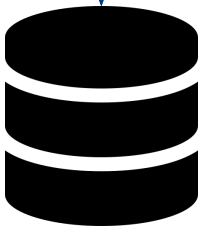
Designed by stories / Freepik



AES-GCM



Mock backend

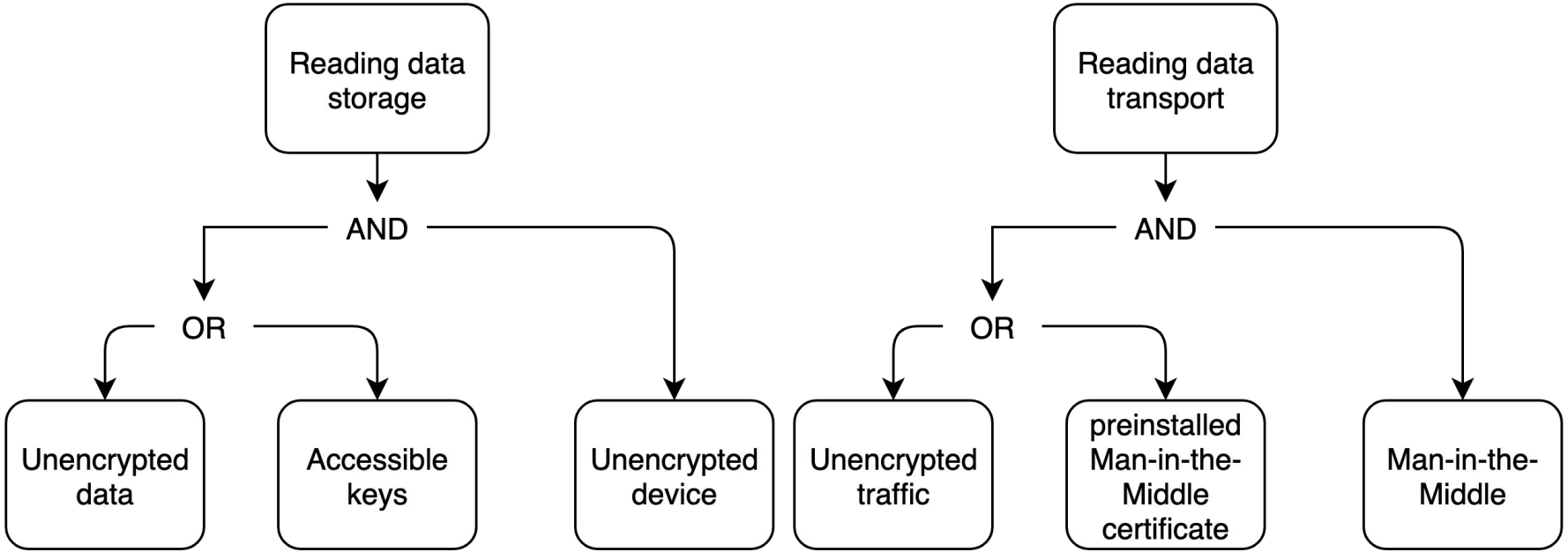


SQLite

Architecture of the app

Scenario

Threat modelling



Attack tree for communication and data storage

Scenario

Implementation

Scenario

Implementation: Data Storage

```
val cipher: Cipher = Cipher
    .getInstance(
        String.format(
            "%s/%s/%s"
            KeyProperties.KEY_ALGORITHM_AES,
            KeyProperties.BLOCK_MODE_GCM,
            KeyProperties.ENCRYPTION_PADDING_NONE
        )
    )
```

Creating the **Cipher**

Scenario

Implementation: Data Storage

```
val generator: KeyGenerator = KeyGenerator
    .getInstance(KeyProperties.KEY_ALGORITHM_AES, "AndroidKeyStore")
    .apply {
        init(parameterSpec)
    }

val key: SecretKey = generator.generateKey()
```

Provider: here **AndroidKeyStore**

Generating of the key generator and derivation of the key

Scenario

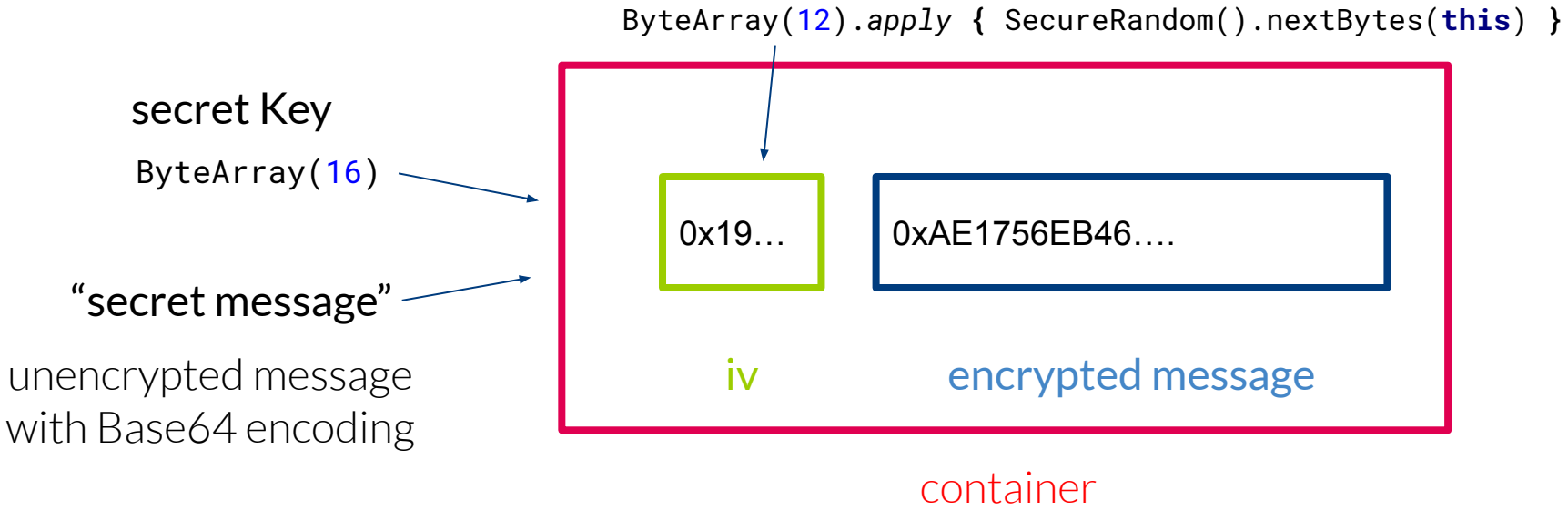
Implementation: Data Storage

```
val loadedSecretKey: SecretKey = KeyStore
    .getInstance("AndroidKeyStore")
    .apply {
        load(null)
    }
    .let {
        (it.getEntry("KeyName", null) as KeyStore.SecretKeyEntry)
        .secretKey
    }
```

Load data from **AndroidKeyStore**

Scenario

Implementation: Data Storage



Structure of the **data storage container**

Scenario

Implementation: Data Storage

```
class BiometricEncryption : (Context, String) -> String {
   /* ... */
override fun invoke(context: Context, clearText: String): String {
       BiometricPrompt
           .Builder(context)
           .setTitle("Usage of BiometricPrompt")
           .setNegativeButton("Cancel",AsyncTask.THREAD_POOL_EXECUTOR,
               DialogInterface.OnClickListener { dialog, _ ->
                   dialog.dismiss()
               }
            ).build()
...
```

Encrypt data with **BiometricPrompt**

Scenario

Implementation: Data Storage

```
...
.let { it.authenticate(BiometricPrompt.CryptoObject(cipher),
       CancellationSignal(),AsyncTask.SERIAL_EXECUTOR,
       object : BiometricPrompt.AuthenticationCallback() {
           override fun onAuthenticationSucceeded(result:
                                BiometricPrompt.AuthenticationResult?) {
               encryptedText = result?.cryptoObject?.cipher?.doFinal(
                                 clearText.toByteArray()).toString()
           }
       }
   )
   return encryptedText
}
```

Encrypt data with **BiometricPrompt**

Scenario

Implementation: Communication

```
<application  → AndroidManifest.xml
...
android:networkSecurityConfig="@xml/network_security_config"
...
</application>

<network-security-config>  → network_security_config.xml
    <domain-config cleartextTrafficPermitted="false">
        <domain includeSubdomains="true">example.com</domain>
    </domain-config>
</network-security-config>
```

Force **HTTPS** in Network Security Configuration

Scenario

Implementation: Certificate Pinning

```
<network-security-config>
    <domain-config cleartextTrafficPermitted="false">
        <domain includeSubdomains="true">example.com</domain>
        <trust-anchors>
            <certificates src="@raw/my_cert" />
        </trust-anchors>
    </domain-config>
</network-security-config>
```

→ network_security_config.xml

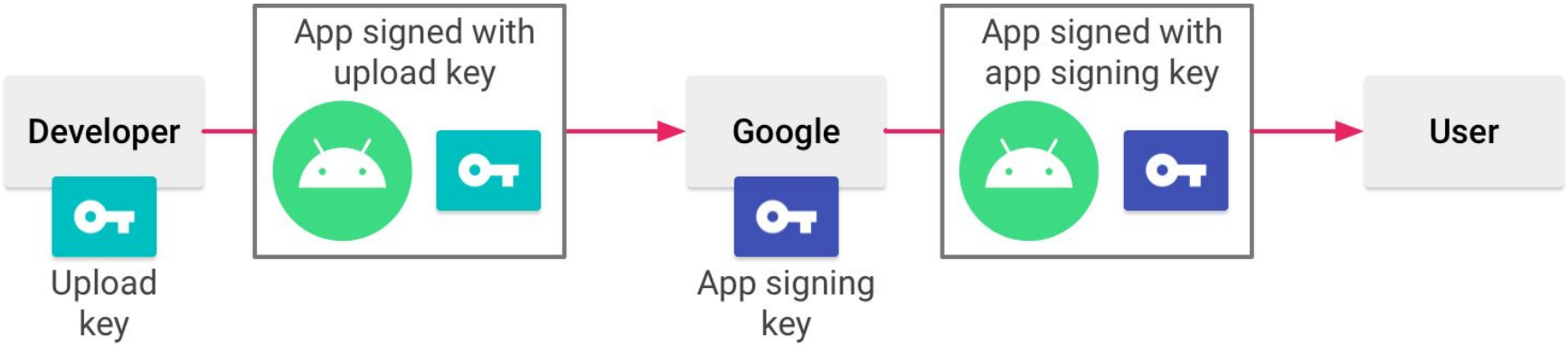
server certificate in pem-format

Enable explicit certificate pinning in Network Security Configuration

Scenario

Implementation: Signing

[2]

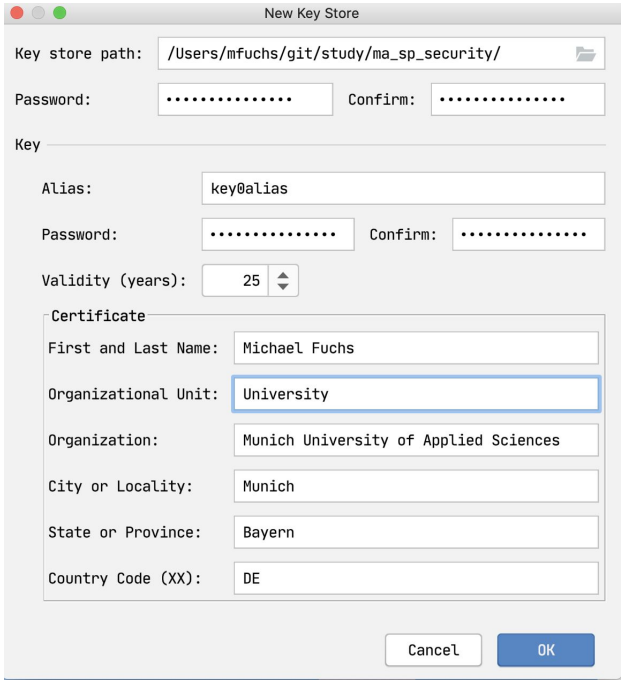


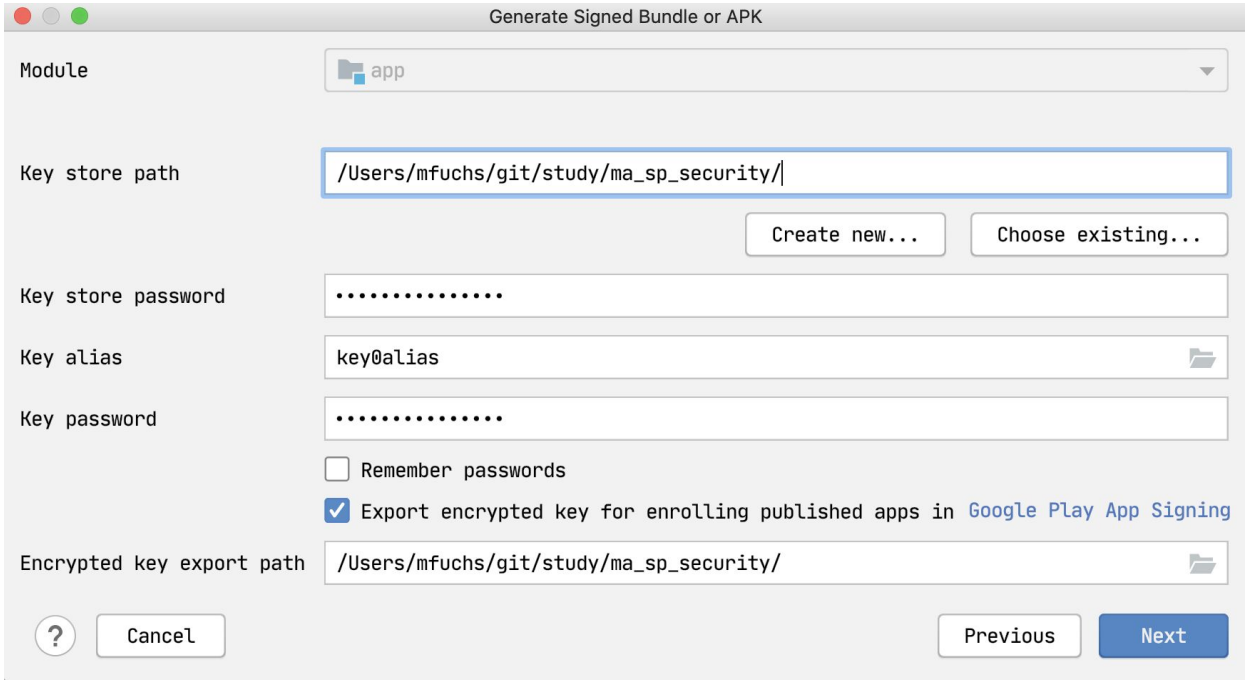
Scheme for signing Android applications

[2] Android Developers. *Sign your app*. 2020. URL: https://developer.android.com/ studio/publish/app-signing (visited on 05/31/2020).

Scenario

Implementation: Signing





Signing process in Android Studio

Scenario

Implementation: Keyboard

- auto-completion saves parts of input
  - used to improve the prediction
  - accessible throughout all applications

- auto-completion can be ensured by attribute

```
<EditText android:inputType="textVisiblePassword" />
```

  - but user is always free to install his own keyboard

Scenario

Implementation: Background

- application can be moved in background
  - screenshot is taken before moved into background
  - attacker → can extract desired information

- information must be cleaned up beforehand

```
window.setFlags(FLAG_SECURE, FLAG_SECURE)
```

Evaluation

- security mechanisms are very easy to implement
- biggest challenge → patchwork of version
  - lack of consistent security standards due to API-Level
  - external libraries are often reason for security holes
- guideline shows only basic scenarios
  - many other other scenarios need additional measures
  - e. g.: obfuscation on Android

Many thanks!

João Pedro Moreira Sousa
Master Control and Decision Systems
Técnico Lisboa
@ moreira.joao@hm.edu
joaosousa98

Michael Fuchs
B. Sc. Computer Science
Munich University of Applied Sciences
@ michael.fuchs@hm.edu
theexiile1305
TheEXiiLE

References

Josh Aas. *Why ninety-day lifetimes for certificates?* Nov. 2015. URL: https : / / letsencrypt.org/2015/11/09/why-90-days.html (visited on 06/01/2020).

Android Developers. *Android 9 features and APIs*. 2019. URL: https://developer. android.com/about/versions/pie/android-9.0 (visited on 05/31/2020).

Android Developers. *Android keystore system*. 2019. URL: https://developer. android.com/training/articles/keystore (visited on 05/31/2020).

Android Developers. *Application Signing*. 2020. URL: https://source.android. com/security/apksigning/ (visited on 05/31/2020).

Android Developers. *Guide to background processing*. 2020. URL: https://develop er.android.com/guide/background/ (visited on 05/31/2020).

Android Developers. *Handle input method visibility*. 2019. URL: https://developer. android.com/training/keyboard-input/visibility (visited on 05/31/2020).

Android Developers. *KeyStore*. 2019. URL: https://developer.android.com/referenc e/java/security/KeyStore (visited on 05/31/2020).

Android Developers. *Network security configuration*. 2019. URL: https://develope r.android.com/training/articles/security-config (visited on 05/31/2020).

Android Developers. *Security with HTTPS and SSL*. 2019. URL: https://develop er.android.com/training/articles/security-ssl (visited on 05/31/2020).

Android Developers. *Sign your app*. 2020. URL: https://developer.android.com/ studio/publish/app-signing (visited on 05/31/2020).

M J Dworkin. *Recommendation for block cipher modes of operation* : tech. rep. Gaithersburg, MD: National Institute of Standards and Technology, Nov. 2007. DOI: 10.6028/NIST.SP.800-38d.

R. Fielding and J. Reschke. *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*. Tech. rep. Fermont: Internet Engineering Task Force (IETF), June 2014. DOI: 10.17487/rfc7230.

Adam Langley. *ImperialViolet - Public key pinning*. May 2011. URL: https://www. imperialviolet.org/2011/05/04/pinning.html (visited on 05/31/2020).

Michael Muckin and Scott C. Fitch. *A Threat-Driven Approach to Cyber Security*. Tech. rep. Bethesda: Lockheed Martin Corporation, 2017. URL: http://ce.sharif. edu/courses/95-96/2/ce746-1/resources/root/Resources/Lockheed%20Mart in%20Threat-Driven%20Approach%20whitepaper.pdf.

E. Rescorla. *Diffie-Hellman Key Agreement Method*. Tech. rep. Fermont: Internet Engineering Task Force (IETF), June 1999. DOI: 10.17487/RFC2631.

E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*. Tech. rep. Fermont: Internet Engineering Task Force (IETF), Aug. 2015. DOI: 10 . 17487 / RFC8446.