CloudCycle 14: Secure and User-Friendly QR-Code Based Key Distribution


Kevin Körner, Thomas Walter
The Initial Problem
The Initial Problem

- How do I distribute key(s) to all my devices?

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Security</th>
<th>User-Friendly</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manually</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Cloud based</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>E-Mail based</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>OTR based</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Is there an ad hoc usable possibility?
Architecture and Workflow

- The distribution device might be a mobile device as well!
- Useful for ad hoc public key distribution! (see also Threema[TH14])
Main Use Cases
Main Use Cases
More Use Cases

• Distributing
  - symmetric keys between *devices*
  - symmetric keys between *persons*
  - keys from *non digital media* (e.g. printings) to *devices*
    - E.g. to provide project keys to all members
  - …

• The easier the distribution is…
  … keys can become complex at will
  … same keys can be used on any user device
  … asymmetric cryptosystems become more usable for non technological persons
Architectural Choices

• Receiver / Distribution Application
  - May be operated inside one application
  - May be split up in several applications

• Key store / Trust store (*stores*)
  - May be operated centralized
    ▪ Operation system integration
    ▪ Inside the receiver/distribution application
    ▪ Additional key-management-application
  - May be operated decentralized
  - May be operated with an „refreshing“-approach
Centralized Key Management

+ Usability
  + Keys can be managed inside one store
  + No "forgotten" keys in the updating process
+ Implementation
  + Centralized access-interface
  + Usage of existing standards / libraries
  + Combined application for receiving, distributing, and storing

- Security breaches
  - affect all keys
  - affect all data of all applications that interacted with the store
- Access standard needs to be forced and implemented
Decentralized Key Management

+ Organisation
  + Task based stores
  + Security breaches affect **only** keys inside the cracked store
  + Each application may implement its own store technology

- Usability
  - Key integration in multiple stores
  - Key updating in multiple stores
  - Technology update

- Implementation
  - Code reuse vs. updating used libraries
  - Implementing distribution / receiving functionality for each store
Updating Key Management

• Separate distribution / receiving functionality from storing
• Selection of influenced stores in the distribution/receiving process
• Requires an secure and standardized access interface

+ Task based stores
+ Security breaches affect **only** keys inside the cracked store
+ Each application may implement its own store technology
+ Keys can be managed inside **one** application
+ Combined application for receiving, distributing, and storing
What have we done yet?

- Android prototype for distribution, receiving and storing keys

- Android key access prototype application (text encryption)
What have we done yet?

• Java-based distribution application for *common* operating systems
Evaluation of QR-Codes as Distribution Encoding

- ZXing-Library supports common QR-Codes (177x177 Modules)
- Those can be used to encode maximum 2.953 Byte (see [ISO00])
  - RSA-Key with length of 4096Bit (~0.5 Kilobyte) ✓
  - Using PKCS#12 as file format (~4 Kilobyte) ✗

- Increasing key size may cause problems
- Old camera systems may cause problems
  - Samsung Galaxy 551, 3,2 megapixels camera → 1000 Byte
  - Huawei Ascend Y300, 5 megapixels camera → 1400 Byte
  - …

- Current solution: Split up the data on multiple adjustable QR-Codes
  - Not really user-friendly (up to 16 separate user interactions) :-/
What is coming next

• Reimplementation of the distribution using another technology
  - E.g. colored Barcode, iQR-Code, …
  - Main goal: One-Click distribution for an as high as possible percentage of mobile devices
• Implementing the prototype for iOS and Windows Phone

• Publishing the implemented applications using an open source licensing model
• Publishing the prototype on the supported app stores

• Evaluation of the systems and publishing of the results
Thank you!
Any Questions?

Contact:

Kevin Körner

Wilhelmstraße 106, 72074 Tübingen
Telefon: +49 7071 29-70384
kevin.koerner@uni-tuebingen.de
References


Project Web Page

• http://ceres.zdv.uni-tuebingen.de/forschung/ufo-kyt.html