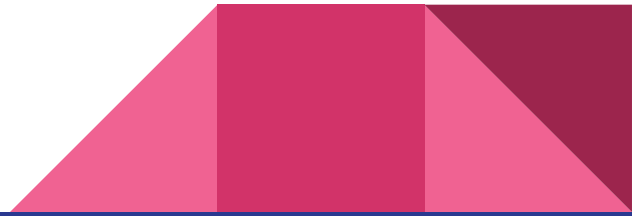
JavaCard Key Unwrapping

Paul, Tobias, Marvin

Ablauf

1. Problemstellung
2. Java Card (die ersten Schritte)
    - 2.1. Tools
    - 2.2. APDUs
    - 2.3. Hello World
    - 2.4. Hello World - Live Demo
3. Key Unwrapping
    - 3.1. Erklärung
    - 3.2. Implementierung
    - 3.3. Live Demo



Problemstellung

Problemstellung – Initialisierung

- Kommunikation Client <-> Server, z.B. mittels OpenVPN

- Server besitzt statische Schlüssel
- Clientschlüssel werden aus Serverschlüsseln vom Server generiert
- Für zukünftige sichere Übertragung des Clientschlüssels wird ein *Wrapping* dessen berechnet (u.a. HMAC und Verschlüsselung mit Serverschlüsseln)

Problemstellung – Verbindungsaufbau

- Kommunikation Client <-> Server, z.B. mittels OpenVPN

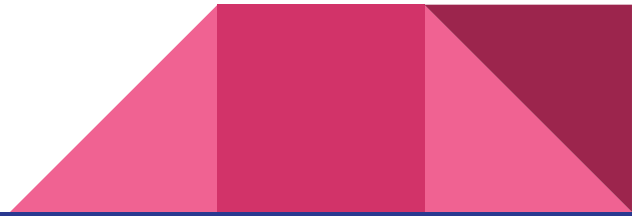
- Server besitzt statische Schlüssel
- Clientschlüssel werden aus Serverschlüsseln vom Server generiert
- Für zukünftige sichere Übertragung des Clientschlüssels wird ein *Wrapping* dessen berechnet (u.a. HMAC und Verschlüsselung mit Serverschlüsseln)

- Client überträgt seinen gewrapten Schlüssel an Server
- Server führt *Unwrapping* durch und erhält Clientschlüssel
- Verschlüsselte Kommunikation kann beginnen

Problemstellung – Absicherung

- Schutz der Serverschlüssel durch SmartCard

- Serverschlüssel in JavaCard einliefern
- *Unwrapping* von des ankommenden gewrapten Clientschlüssels auf JavaCard ausführen
- Clientschlüssel zurückliefern



Tools

JavaCard

- Version 2.2.2 und 3.0.4

Cardreader

pcscd

- JavaCard scannen und testen (`pcsc_scan`)

GlobalPlatformPro

- Applets managen und damit kommunizieren

Ant

- Applet Build Tool
- ant-javacard.jar

[Quelle: https://sarwiki.informatik.hu-berlin.de/images/0/07/NXP_J3H145_DI.jpg]

APDUs

APDUs

- “Application Protocol Data Unit”
- Kommando-/Datenblock des Kommunikationsprotokolls zwischen einem Chipkartenleser und einer Chipkarte
- Struktur der APDU definiert in ISO 7816
- Darin angegebene Befehle werden im Applet implementiert

APDUs – Command APDUs

- Struktur

CLA	INS	P1	P2	Lc	Data	Le

Abkürzung	Name	Bytelänge
CLA	Class	1
INS	Instruction	1
P1	Parameter 1	1
P2	Parameter 2	1
Lc	Length command	0, 1 (regulär), 3 (extended APDU)
Data	Data	Lc
Le	Length expected	0, 1 (regulär), 2, 3 (extended APDU)

APDUs – Command APDUs

4 Typen

- Case 1
- Case 2
- Case 3
- Case 4

CLA	INS	P1	P2			
CLA	INS	P1	P2	Le		
CLA	INS	P1	P2	Lc	Data	
CLA	INS	P1	P2	Lc	Data	Le

APDUs – Response APDUs

- Struktur

Data	SW1	SW2

- 2 Typen, abhängig von Le = oder ≠ 0
- SW1 und SW2 ergeben zusammen das Statuswort (Returncode)
- Bsp.: SW = 0x9000 bedeutet “Bearbeitung erfolgreich”

Abkürzung	Name	Bytelänge
Data	Data	Le
SW1	Statusbyte 1	1
SW2	Statusbyte 2	1

Hello World

Hello World

`install(byte bArray[], short bOffset, byte bLength)`

- Erstellen einer Instanz der Klasse mittels Konstruktor

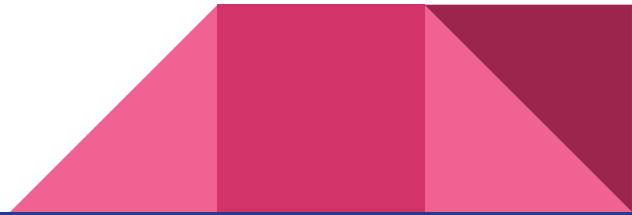
Konstruktor

- Klassenspezifische Initialisierungen + `register()` um Instanz zu registrieren

`process(APDU apdu)`

- Prüfen von CLA und INS und Wahl einer Funktion

INS-spezifische Funktionen



Hello World – Live Demo

Key Unwrapping

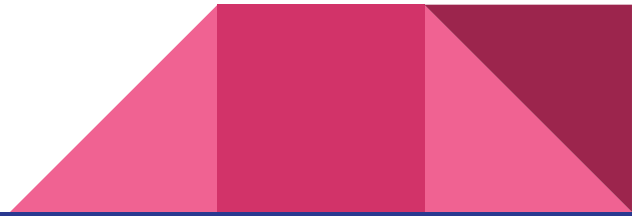
tls-crypt-v2

Unterschiede zu tls-crypt(-v1):

- Client-spezifische Schlüssel
- Beim Verbindungsaufbau wird zunächst ein „gewrappter” Key verschickt
- Metadaten (Identifizierung, Validierung)

Nachteile:

- Mehraufwand durch das Unwrapping
- Flexible Metadaten können zu Fehlern führen



Ablauf

Initialisierung:

1. Serverschlüssel *Ke* und *Ka* werden generiert
2. Für jeden Client:
   a. Clientschlüssel *Kc* wird generiert
   b. *WKc* wird erstellt, indem *Kc* durch *Ke* und *Ka* „gewrappt” wird
   c. Client speichert *Kc* und *WKc* ab

Operative Phase:

3. Client schickt beim Verbindungsaufbau *WKc* an den Server
4. Server „unwrappt” *WKc* mit *Ke* und *Ka*
5. Verschlüsselte Kommunikation durch *Kc*

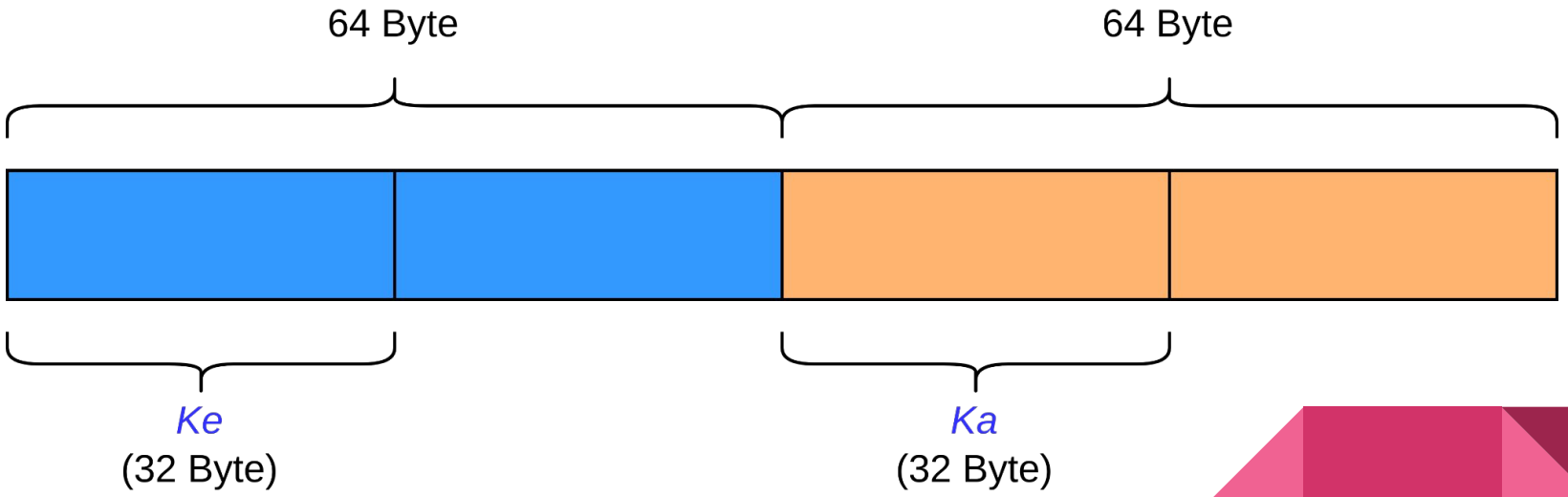
Initialisierung des Server-Keys

- openvpn --genkey tls-crypt-v2-server

```
 ~ openvpn --genkey tls-crypt-v2-server
-----BEGIN OpenVPN tls-crypt-v2 server key-----
GLOmDRWjuDAU5ihRysluzr3txTwbgQz5hFG57YT6nAQskHF0cnDMJS7NFZiT18w5
arCieMHyN+NwosNQF0XYXd+U5Aa08tffn6ZvrsHb3ih9WDvNeCyTtertWs1M4/57
U5IAWUK7u/vMlc92plDcUN8B7fV0zMGq69PWz9fWlSI=
-----END OpenVPN tls-crypt-v2 server key-----
```

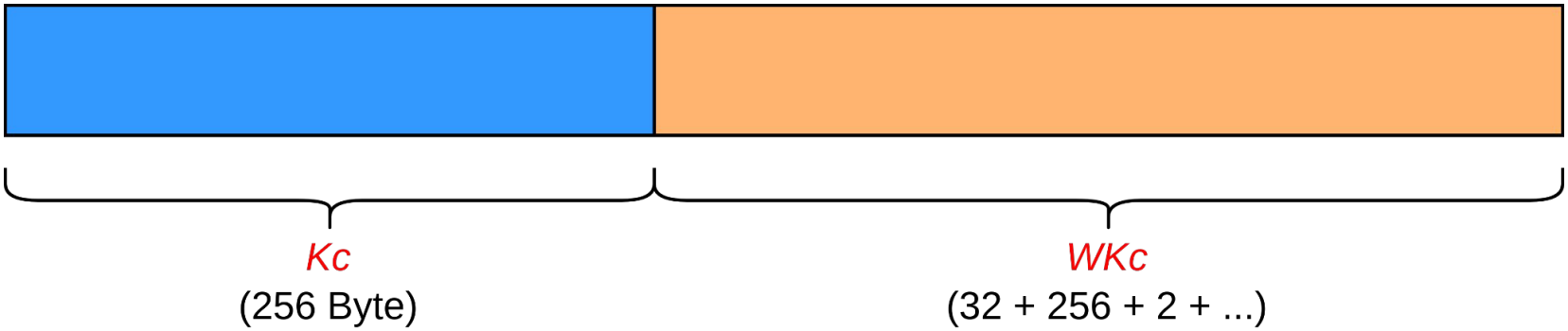
Initialisierung des Server-Keys

- openvpn --genkey tls-crypt-v2-server



Initialisierung eines Client-Keys

- openvpn --genkey tls-crypt-v2-client --tls-crypt-v2 <server.key>



Wrapping & Unwrapping

len = length(*WKc*)

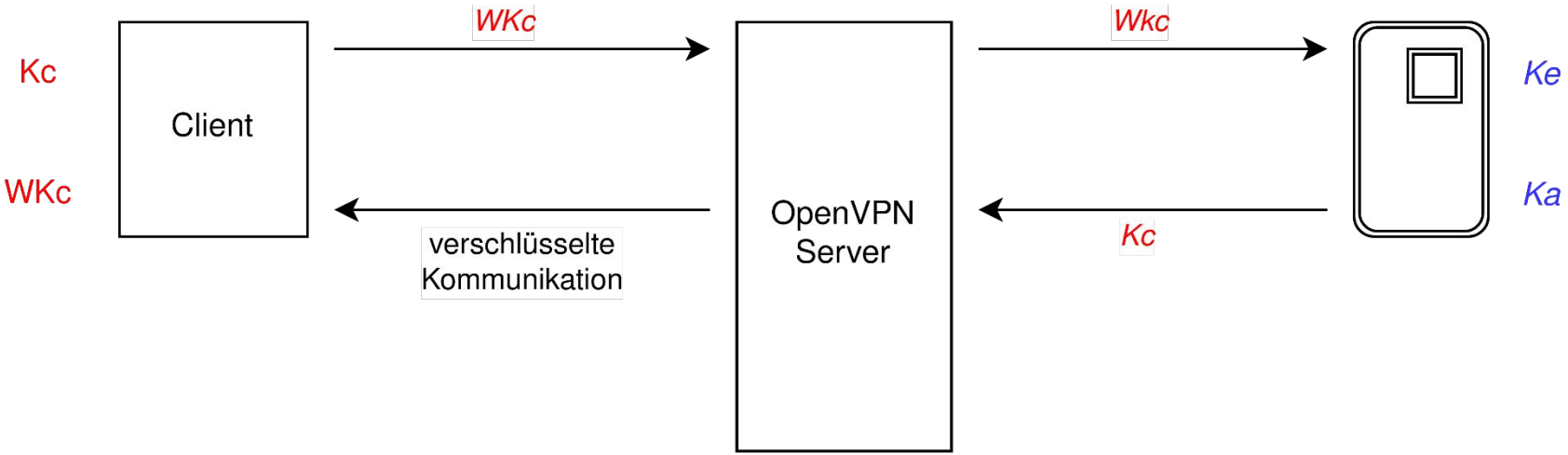
T = HMAC-SHA256(*Ka*, *len* || *Kc* || *metadata*)

IV = 128 *most significant bits* (MSB) von *T*

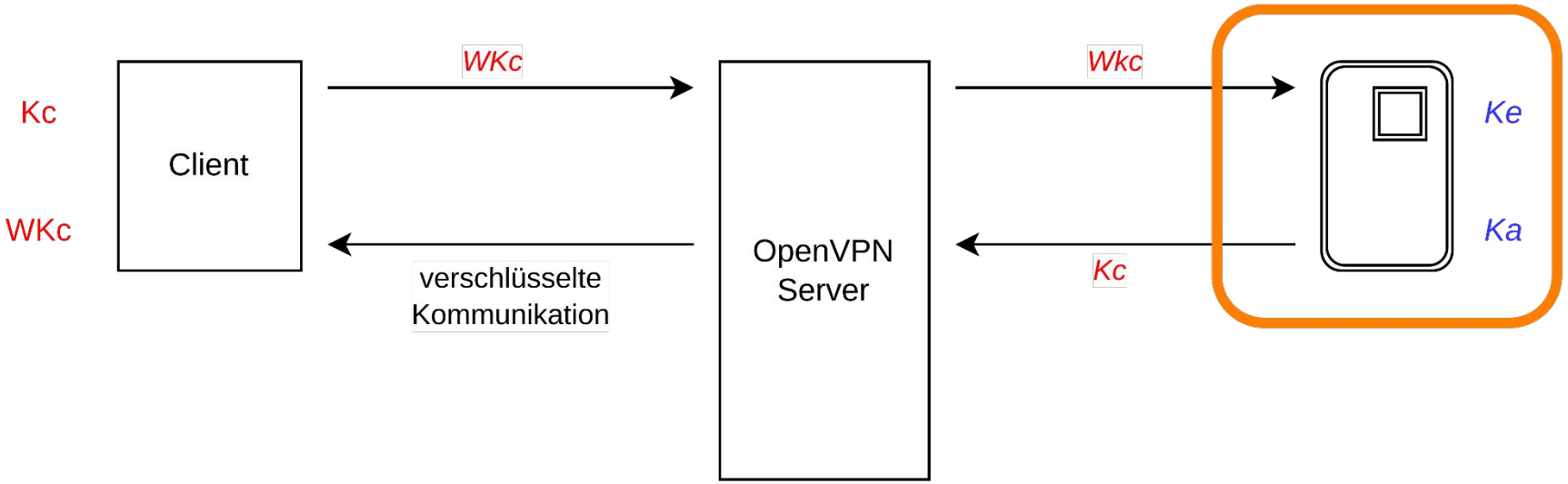
WKc = *T* || AES-256-CTR(*Ke*, *IV*, *Kc* || *metadata*) || *len*

- Die Länge *len* kann vorab bestimmt werden.
- Das Unwrapping soll auf der Java Card geschehen.

Unwrapping auf der Java Card



Unwrapping auf der Java Card



Implementierung

AES-256-CTR:

- Unterstützt ab Version 3.0.5 (Unsere Karte hat Version 3.0.4)

HMAC-SHA256:

- Nicht unterstützt, daher eigene Implementierung

Extended APDUs:

- Daten werden nacheinander in dafür vorgesehenen Buffer geschrieben

Implementierte Funktionen (Instructions)

INS_IMPORT_KEY:

B0 | 10 | 01 00 | 20 | *Ke*

B0 | 10 | 02 00 | 20 | *Ka*

INS_UNWRAP_KEY: ca. 150 Unwraps / Minute

B0 | 20 | 00 00 | 00 01 2B | *WKc* | 01 00 → len(*Kc*)

Fehlercodes (nach ISO7816)

0x9862 → Authentication Error, application specific (incorrect MAC):

- Berechnete Prüfsumme unterscheidet sich von empfangenen ***T***

0x6700 → Wrong length:

- Empfangener *WKc* ist zu kurz oder zu lang
- Die Länge der empfangenen Daten unterscheiden sich von len(*WKc*)
- Die zu importierenden Schlüssel haben die falsche Länge

0x6A86 → Incorrect P1 or P2 parameter

Key Unwrapping - Live Demo