

Vorlesung Internet of Everything Wintersemester 2017/18

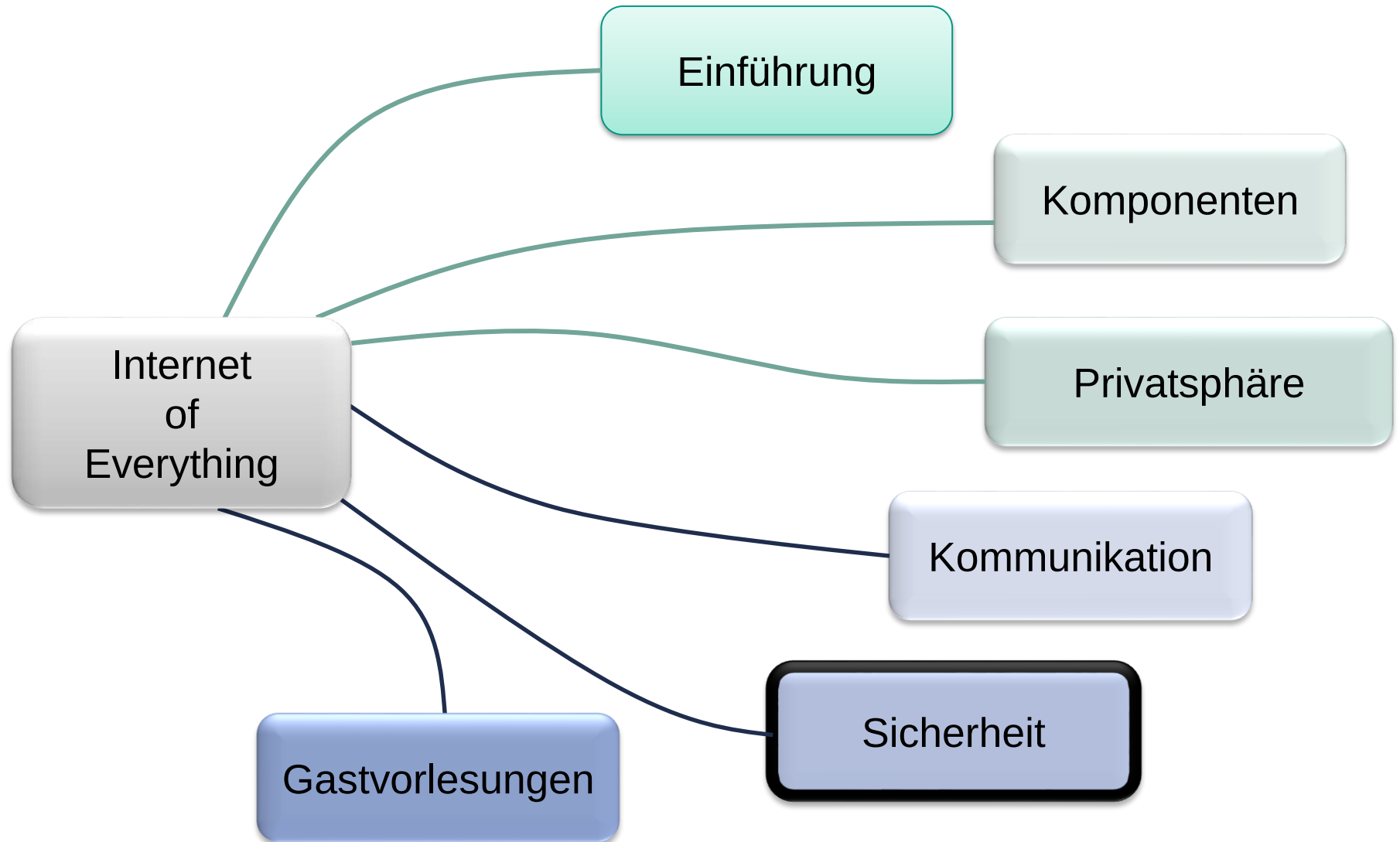
5. Sicherheit

Institut für Telematik, Prof. Zitterbart



© Peter Baumung

Inhalte der Vorlesung



Allgemeine Schutzziele (IT-Sicherheit)

- Schutzziel
 - Anforderungen an eine **Komponente** oder ein **System**, die erfüllt werden müssen, um schützenswerte **Güter** vor **Bedrohungen** zu schützen
- Häufige Kategorisierung in

C

I

A

 - **Confidentiality (Vertraulichkeit)**
 - Ein System bewahrt Vertraulichkeit, wenn es keine unautorisierte Informationsgewinnung ermöglicht
 - **Integrity (Integrität)**
 - Ein System bewahrt *starke* Integrität, wenn es nicht möglich ist, Daten unautorisiert zu manipulieren
 - Ein System bewahrt *schwache* Integrität, wenn unautorisierte Manipulationen an Daten nicht unbemerkt möglich sind
 - **Availability (Verfügbarkeit)**
 - Ein System bewahrt Verfügbarkeit, wenn es keine unautorisierte Einschränkung der Funktionalität des Systems ermöglicht
- Weitere Schutzziele
 - Authentizität
 - ...

Kryptografische Bausteine

■ Verschlüsselung

■ Symmetrische Verschlüsselung

- Kommunikationspartner verfügen über gemeinsamen Schlüssel zum Ver- und Entschlüsseln
- Hohe Effizienz, da meist einfache Operationen (Bit-Shift, XOR)
- Beispiel AES (Advanced Encryption Standard)

■ Asymmetrische Verschlüsselung

- Kommunikationspartner verfügen über öffentliche und private Schlüssel
- Basis: Faktorisierung von Zahlen gilt als schwer
- Beispiel: RSA (Rivest, Shamir, Adleman)

■ Integritätssicherung

■ Kryptografische Hashfunktion

- Verwendet symmetrische Kryptografie

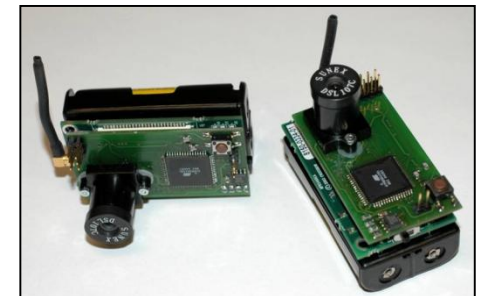
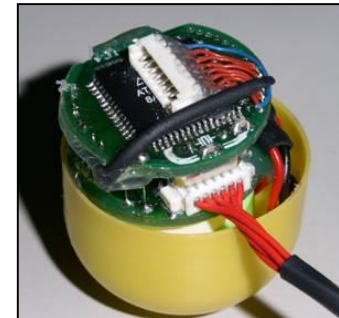
■ Digitale Signatur

- Verwendet asymmetrische Kryptografie

■ Schlüsselaustausch

Zur Erinnerung: Ressourcenmangel

- Potentiell leistungsschwache Sensoren (Kategorie C0 oder C1)
 - „schwacher“ Microcontroller
 - ca. 7 MIPS
 - $\approx 1/5$ eines Gameboys
 - 4 KByte RAM
 - $\approx 1/8000$ eines PDAs
- „Sparsame“ Kommunikation
 - Bluetooth/ZigBee mit 250 kbit/s
 - $\approx 1/400$ eines LAN
- Wenig Energie
 - Batteriebetrieben, 180 mAh Kapazität
 - $\approx 1/10$ eines Handys



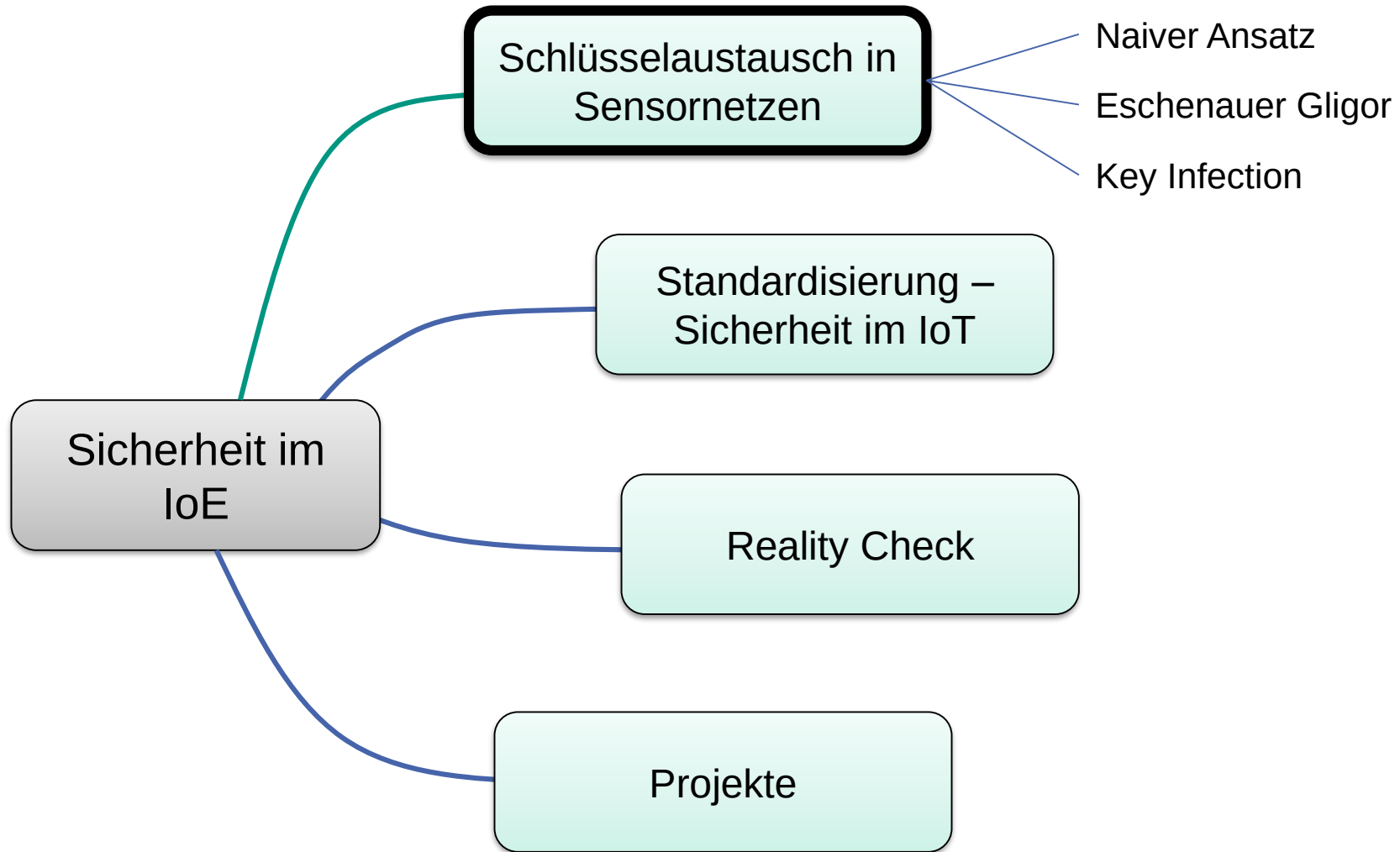
Einsatz von klassischer Kryptografie im IoE

- Kryptographische Operationen sind teuer
 - Verschlüsseln, Entschlüsseln, Hashing etc.
 - Vergleich (MICAz-Plattform) → Energieverbrauch
 - Addition: 625pAs
 - Symmetrisches Verschlüsseln (RC5/64Bit): 1,3μAs (x2.000)
 - RC5 (Rivest Cipher 5) gilt als sehr effiziente symmetrische Chiffre
 - Asymmetrische ECC-Punktmultiplikation: 4mAs-28mAs (x7Mio-x45Mio)
 - Elementare Rechenoperation für Kryptographie auf elliptischen Kurven
 - Ein einzelner symmetrischer Schlüssel benötigt mindestens 16 Byte Hauptspeicher
 - Skaliert nicht für große Netze mit tausenden Systemen
- Implementierung der kryptografischen Algorithmen benötigen Speicher
 - Typische Implementierungen benötigen etwa 2 kB RAM
 - Etwa die Hälfte des RAM eines MICAz Sensorknotens

Einsatz von klassischer Kryptografie im IoE

- Asymmetrische Kryptographie ist zu teuer
 - Schlüssel benötigen noch mehr Speicher, z.B. Faktor 20
 - Ein RSA-Schlüssel ist mindestens 1024 Bit lang
 - Paketgröße bei IEEE 802.15.4 ~ 100 Byte
 - Wie Schlüsselmateriale übertragen?
 - Typische Ver-/Entschlüsselungsdauern dabei im Sekundenbereich
 - RSA-1024 auf MICAz: ~10 Sekunden
 - Ungeeignet für Szenarien, in denen in kurzen Perioden Messungen vorgenommen werden
 - Asymmetrische Verschlüsselung kostet mehr Energie
 - Im Vergleich zu AES bis zu Faktor 100 und mehr
- Konsequenz
 - Vermeide asymmetrische Kryptographie im IoE. Verwende möglichst Lösungen mit symmetrischer Kryptographie.

Schwerpunkte des Kapitels im Überblick



Herausforderung Schlüsselaustausch

- Typische Eigenschaften der Komponenten/Architektur ein Problem
 - Ressourcenknappheit
 - Selbst-Organisation
 - Netztopologie
- Klassische Schlüsselaustauschprotokolle benutzen häufig „zentrale Komponenten“
 - z.B. Certificate Authorities, in Public-Key-Infrastrukturen
 - Zentrale Komponenten u.U. nicht immer erreichbar
- Einsatz von Diffie-Hellman aufgrund der Ressourcenbeschränkungen schwierig
 - Berechnungsdauer > 2 Sekunden
- Fazit: Austausch von Schlüsseln im IoE erschwert

Verfahren zum Schlüsselaustausch

- Single Mission Key
 - Einfach und problematisch

- Zufallsverteilte Schlüssellisten
 - Zwei Systeme besitzen mit gewisser Wahrscheinlichkeit einen gemeinsamen Schlüssel

- Key Infection
 - Angreifer darf während Schlüsselaustausch nicht anwesend sein

Single Mission Key

- Alle Systeme bekommen vor ihrer Ausbringung den selben Schlüssel K
 - Dieser wird verwendet um Kommunikation abzusichern
 - Verschlüsselung, Integritätssicherung ...

- Problem
 - Sobald ein einziges System korrumpiert wird, ist die gesamte Kommunikation im Netz unsicher

- Paarweise Schlüssel mit allen Systemen
 - Jeder bekommt prophylaktisch jeweils einen Schlüssel zur Kommunikation mit allen $n - 1$ anderen Systemen
 - Probleme
 - Speicherverbrauch bei größerem n , z.B. $n = 10.000$ Systemen und 128 Bit Schlüsseln mit ca. 160 *KByte* sehr hoch
 - Wie mit neuen Systemen die dem Netz beitreten umgehen ?
 - Wie können die Schlüssel auf alle Knoten verteilt werden ?

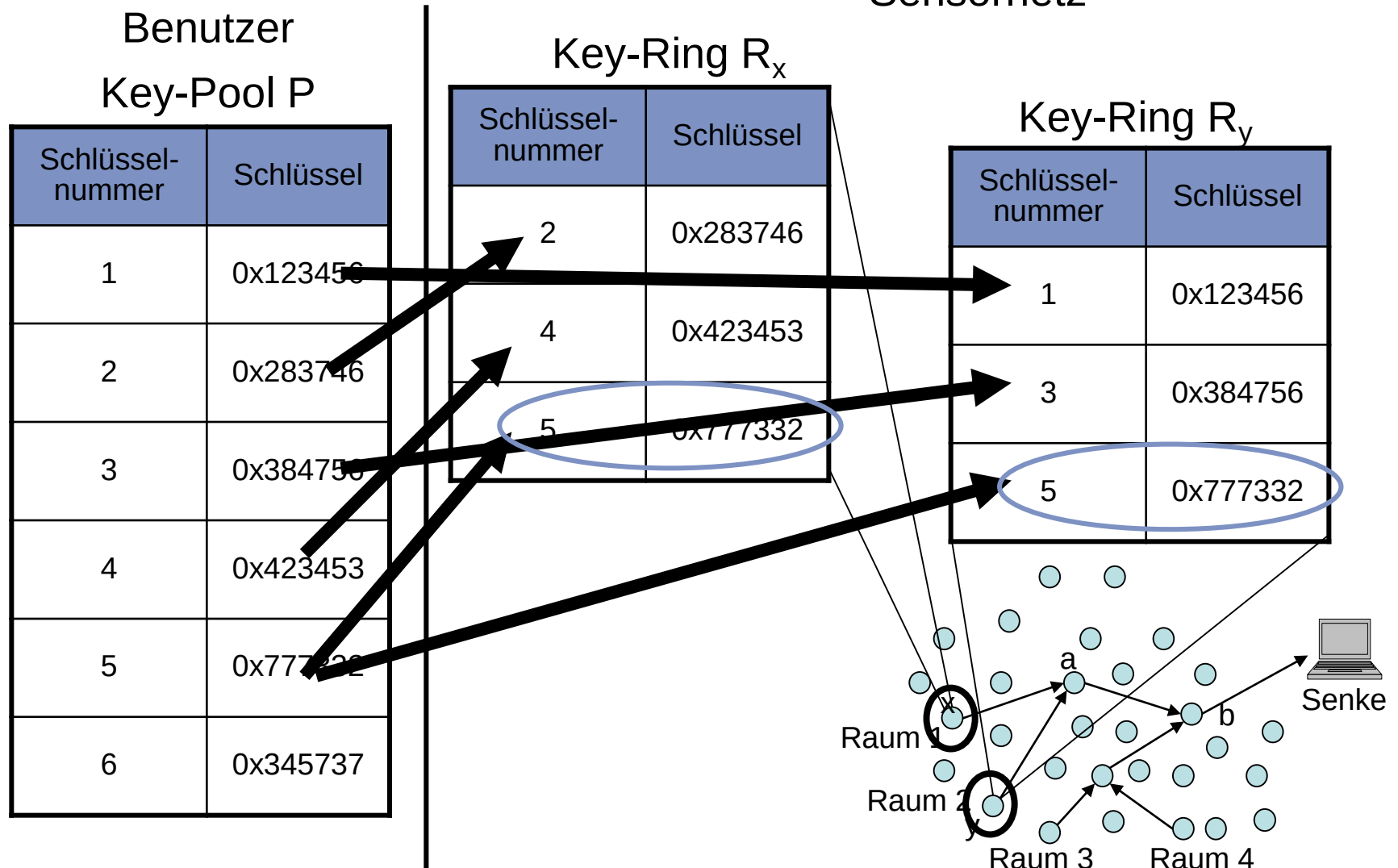
Zufallsverteilte Schlüssellisten



- Ansatz von Eschenauer und Gligor (EGLI)
 - Benutzer erstellt vorab offline eine große Liste von Schlüsseln, den *Key-Pool P*
 - Jedes System, das dem Netz beitreten soll, bekommt eine zufällige Teilmenge dieser Schlüssel, seinen *Key-Ring R*

- Nun gilt
 - Mit einer bestimmten *Wahrscheinlichkeit* besitzen zwei beliebige Systeme einen gemeinsamen Schlüssel auf ihren beiden Key-Ringen
 - Falls zwei Systeme keinen gemeinsamen Schlüssel auf ihren Key-Ringen besitzen, lässt sich ein sogenannter *Schlüsselpfad* zueinander konstruieren
 - Schlüsselpfad: Pfad über mehrere Systeme hinweg, die jeweils paarweise einen gemeinsamen Schlüssel auf ihren Key-Ringen besitzen

Beispiel



Finden von gemeinsamen Schlüsseln

- Wie erkennen zwei Systeme, dass sie über einen gemeinsamen Schlüssel verfügen?
 - Versenden von **Schlüssellisten**
 - System x sendet Liste seiner Schlüsselnummern „im Klartext“ an y: (2, 4, 5)
 - System y antwortet mit einer gemeinsamen Schlüsselnummer im Klartext: (5)

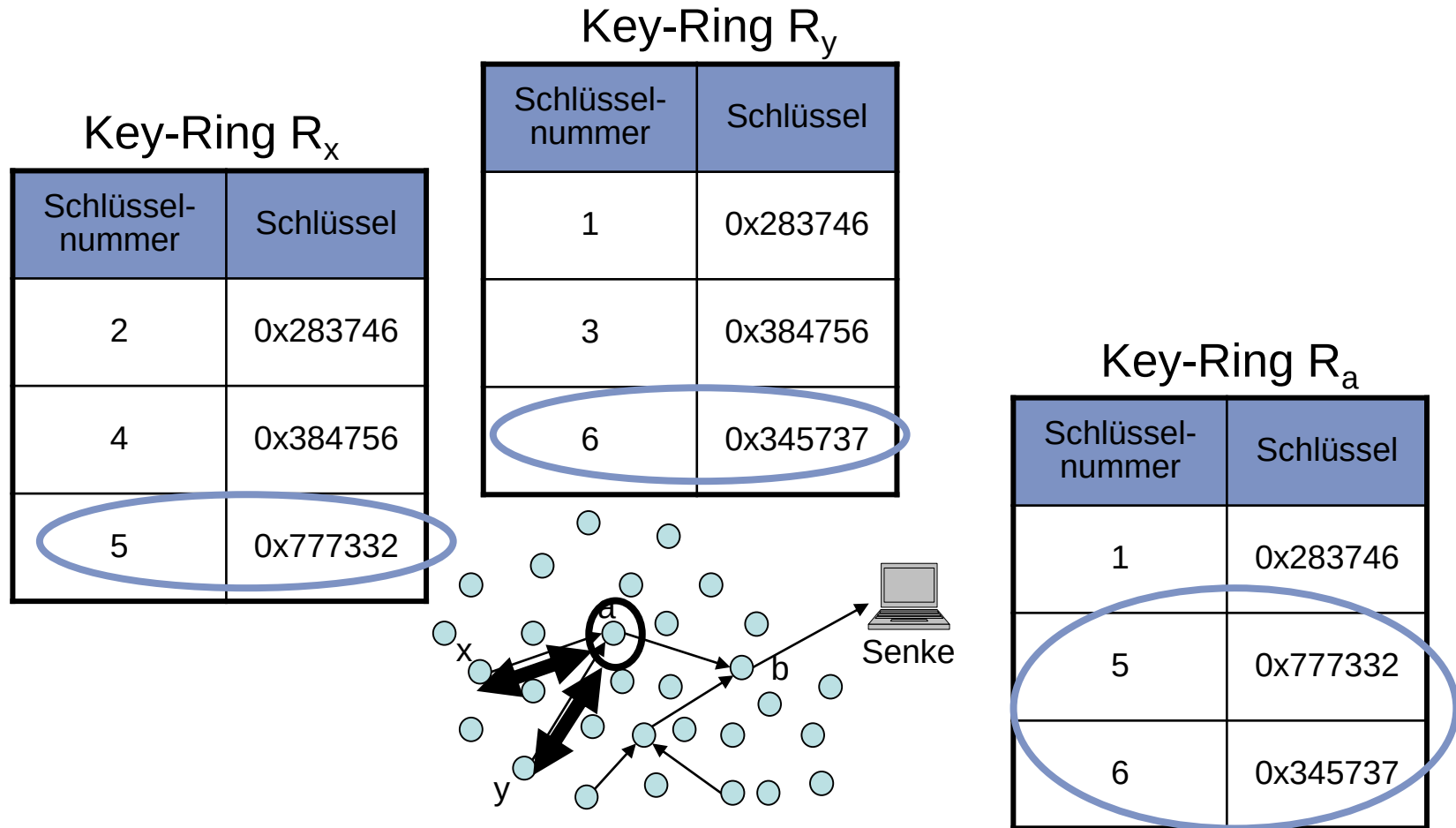
unsicher

- Versenden von **Klartext-** und **Chiffrat-Paaren**
 - System x sendet Zufallszahl Z und Chifftrate seiner Schlüssel an y:
 $(Z, ENC_2(Z), ENC_4(Z), ENC_5(Z))$
 - System y antwortet z.B. mit $(ENC_5(Z + 1))$, was x wiederum überprüfen kann

„teuer“!

Konstruktion Schlüsselpfade

- Problem: Systeme haben u.U. keinen gemeinsamen Schlüssel



Konstruktion Schlüsselpfade

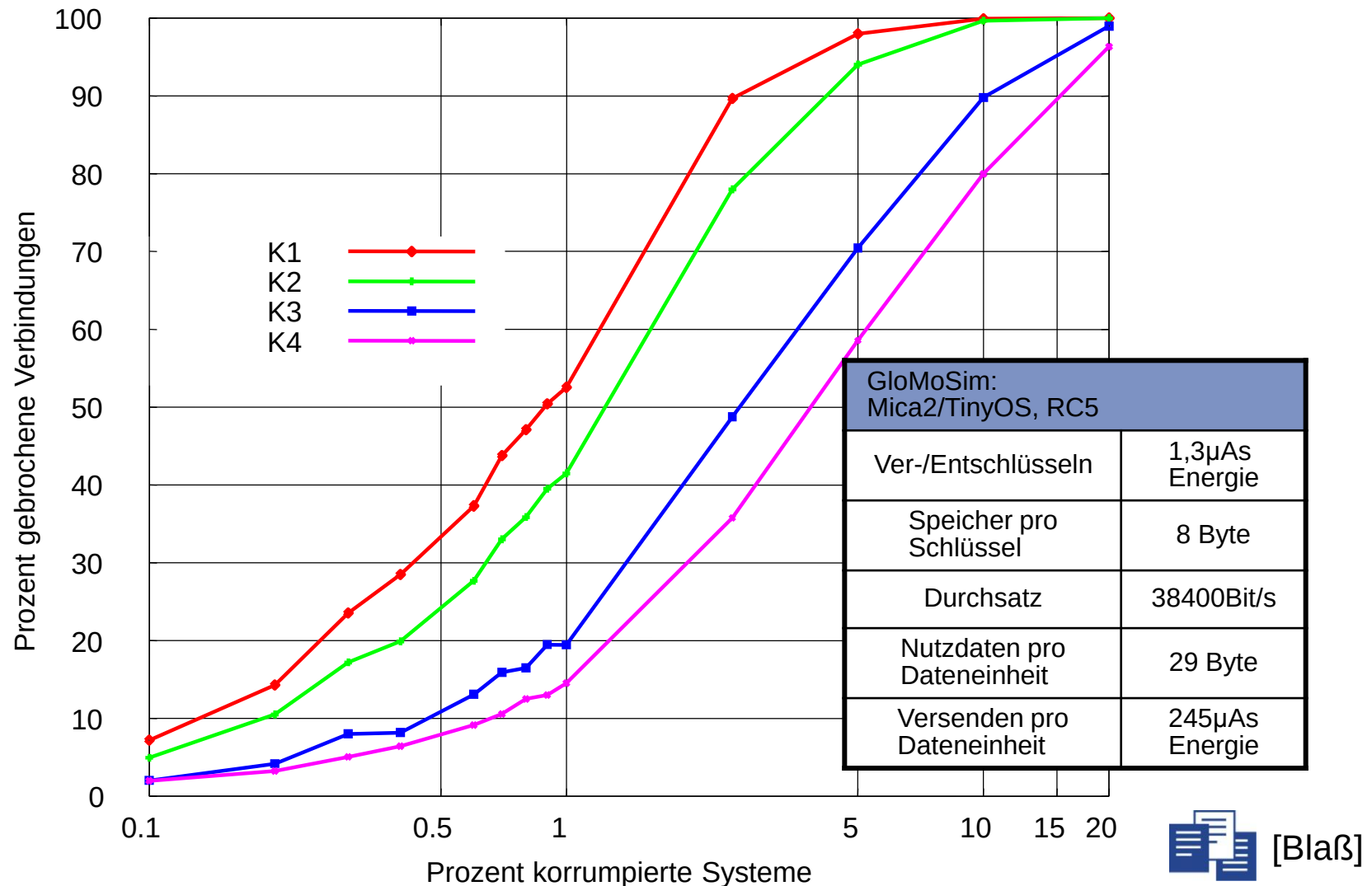
- Systeme x und y tauschen über z einen neuen gemeinsamen Schlüssel aus, z.B. (5)
- Schlüsselpfade können über viele Zwischenstationen hinweg existieren
 - Es ergibt sich ein Schlüsselgraph
 - Knoten im Graph sind die Systeme
 - Zwischen zwei Knoten im Graph existiert eine Kante, wenn die beiden Systeme einen gemeinsamen Schlüssel auf ihren Key-Ringen besitzen
 - Ziel: Schlüsselgraph muss zusammenhängend sein
 - Sonst gibt es Systeme, die niemals mit anderen „sicher“ kommunizieren können
 - Mit bestimmter Wahl der Größe von P und R kann man zeigen, dass der Schlüsselgraph mit $p > 99\%$ zusammenhängend ist
 - Vorschlag: Konfigurationen $(K_1, \dots, K_4)$ für verschiedene P und R

unsicher

	$ R $	$ P $
K_1	50	1000
K_2	75	2000
K_3	50	3700
K_4	75	8300


[EG02]

Simulationsergebnisse, $n = 1000$ Systeme



Sicherheitsbetrachtungen

- Angreifermodell
 - Angreifer kann gewissen Prozentsatz an Knoten korrumpieren und Schlüsselliste auslesen

- Bereits eine kleine Menge (1-5 %) korrumpierter Systeme erlaubt das Mithören oder Manipulieren vieler (>50 %) Verbindungen
 - Häufig werden Schlüssel zur Absicherung mehrerer „Verbindungen“ verwendet
 - „Verbindung“ hier: Kommunikation zwischen zwei Systemen, die mit einem Schlüssel geschützt ist

- Korruptierte Systeme auf dem Schlüsselpfad kennen ausgetauschten Schlüssel
 - Können Kommunikation abhören
 - Können Kommunikation manipulieren

Fazit: Zufallsverteilte Schlüssellisten

- Austausch neuer Schlüssel über Key-Ringe: **unsicher**
- Konstruktion von Schlüsselpfaden: **unsicher**
- Mehrere Verbindungen mit dem gleichen Schlüssel: **unsicher**
- [nicht gezeigt: Bei Ausfall eines Systems müssen verbleibende Systeme Schlüssel „streichen“]

- Aber
 - System kommt ohne zusätzliche Infrastruktur aus
 - Schlüsselfindung zwischen zwei beliebigen Systemen mit einer hohen Wahrscheinlichkeit
 - Trade-off zwischen Speicherverbrauch und Sicherheit der Schlüssel

Key Infection: Smart Trust for Smart Dust

■ Einsatzszenario

■ *Smart Dust Enviroment*

- Viele, sehr günstige und zufällig ausgebrachte Sensorknoten
 - Z.B. Abwurf aus dem Flugzeug

■ Eingeschränktes Angreifermodell

- Angreifer in Schlüsselaustauschphase nur bedingt präsent
 - Lokal eingeschränkt, kann also nicht kompletten Netzverkehr abhören
 - Keine aktiven Angriffe während der Schlüsselaustauschphase

■ Keine Infrastruktur notwendig

- Insbesondere kein Schlüsselaustauschserver

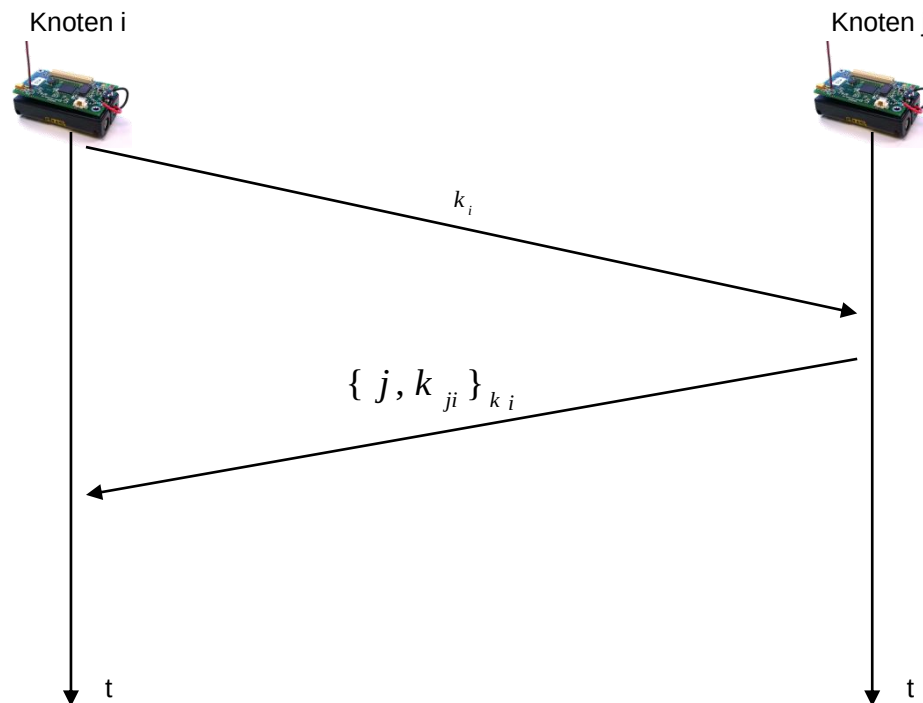
Key Infection

■ Ablauf des initialen Schlüsselaustauschs

- Wird für und von allen Knoten in gegenseitiger Kommunikationsreichweite durchgeführt

Ablauf:

- Knoten i wählt einen zufälligen Schlüssel k_i
- Dieser wird per Broadcast versendet
- Knoten j wählt gemeinsamen Sitzungsschlüssel k_{ji} und sendet diesen zusammen mit seiner ID verschlüsselt an Knoten i



Sicherheitsbetrachtung Key Infection

■ Simulation von verschiedenen Sensornetzkonfigurationen

- Unterschiedliche Dichte der Knoten
 - Sensornetz mit 10000 Knoten
 - Knoten haben 2,3,4 oder 5 Nachbarn
- Unterschiedlicher Anteil an korruptierten Knoten
 - 1, 2 oder 3 % böartiger Knoten

■ Ergebnisse

- Durchgeführt wurden jeweils 100 Simulationsdurchläufe
- Gemessen wurde hierbei der durchschnittliche Anteil an korruptierten Schlüsseln

Anzahl Nachbarn	Anteil an korruptierten Knoten		
	1 %	2 %	3 %
2	1,13 %	2,34 %	3,48 %
3	1,75 %	3,51 %	5,06 %
4	2,38 %	4,61 %	6,75 %
5	2,92 %	5,76 %	8,40 %



Key Infection

- Das Key Infection Protokoll besteht aus mehreren Teilen
 - Key Infection
 - Initialer Schlüsselaustausch im Klartext zwischen benachbarten Knoten
 - Multihop Key Exchange
 - Schlüsselaustausch zwischen nicht benachbarten Knoten
 - Secrecy Amplification
 - Verstärkung der Sicherheit der Schlüssel durch Nutzung von Mehrwege Austausch
- Multihop Key Exchange und Secrecy Amplification sind optional

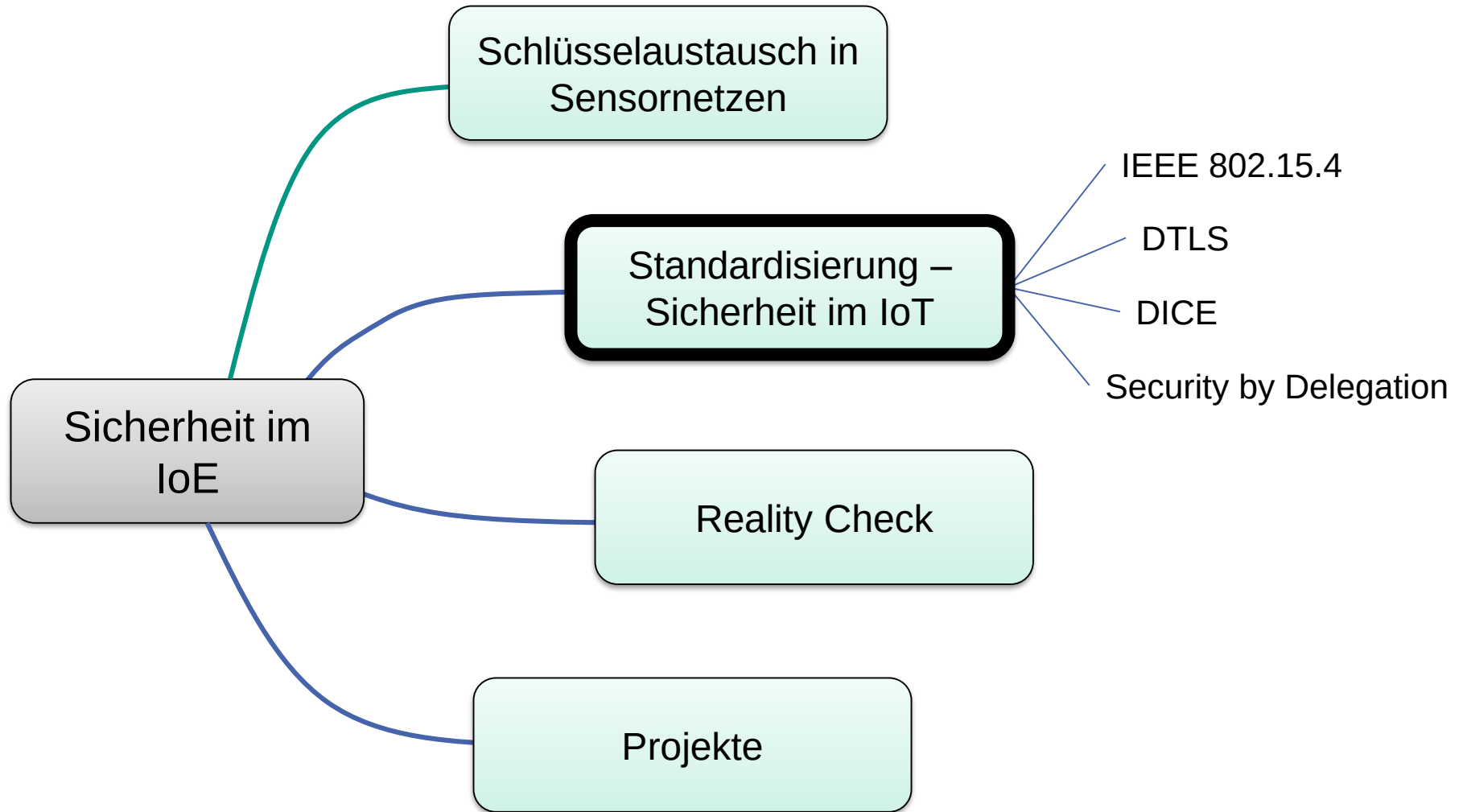
Zusammenfassung Key Infection

- Einfaches Schlüsselaustauschprotokoll
- Geringer Speicheraufwand für Schlüsselmateriale
 - Es werden nur tatsächlich genutzte Schlüssel gespeichert
- Theoretisch geringer Kommunikationsaufwand
 - Pro Schlüsselaustausch 2 Nachrichten
 - In der Praxis
 - Alle Knoten tauschen mehr oder weniger gleichzeitig Schlüssel aus
 - Kollisionen sehr häufig, Sendewiederholungen notwendig,
 - Erhöht Kommunikationsaufwand drastisch
- Weitere Kritikpunkte
 - Eingeschränktes Angreifermodell Voraussetzung
 - Weitere Voraussetzung: Symmetrische Links

Schlüsselaustauschverfahren

- Aktueller Stand der Forschung
 - Problem des Schlüsselaustausches stand lange im Fokus der Forschung
 - Neuere Ansätze beachten zunehmend auch asymmetrische Verfahren
 - Auch Unterstützung von hardware-basierten Verfahren im Fokus
 - IETF: Einige Verfahren sind auf dem Weg zur Standardisierung
 - Sowohl neue Ansätze als auch Adaptionen bereits existierender Protokolle
 - Verfolgt werden „klassischen“ Ansätzen mit bestehender Public-Key-Infrastruktur oder vordefinierten Vertrauensankern

Schwerpunkte des Kapitels im Überblick

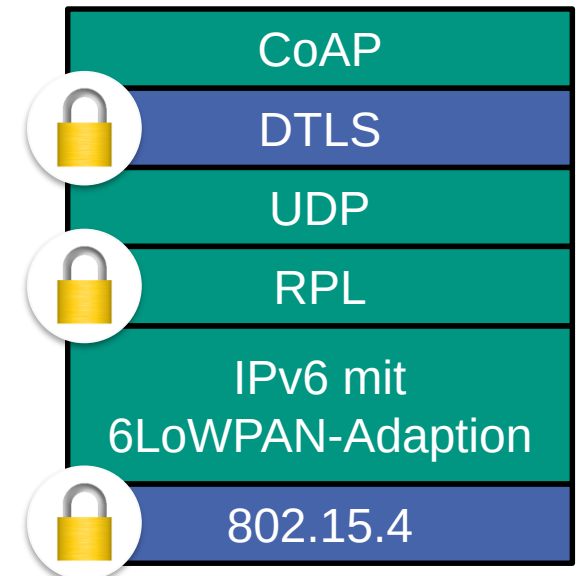


Standardisierung von Sicherheitsmechanismen für das IoT

- Sicherheit in IEEE 802.15.4
 - Link-Layer-Schutzmechanismen sind vorgesehen
- Stand der IETF Standardisierung bisher
 - 6LoWPAN und RPL standardisiert
 - Keine Sicherheitsmechanismen für übertragene Daten enthalten!
 - Aber: RPL kann eigene Dateneinheiten schützen
- (D)TLS ist umfangreicher Standard
 - Vereinfachung für Sicherheit und Effizienz
- CoAP als Anwendungsprotokoll standardisiert
 - Sichere Variante CoAPs verwendet DTLS

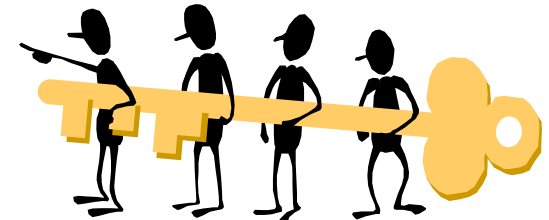


[s. Kapitel 4.2]



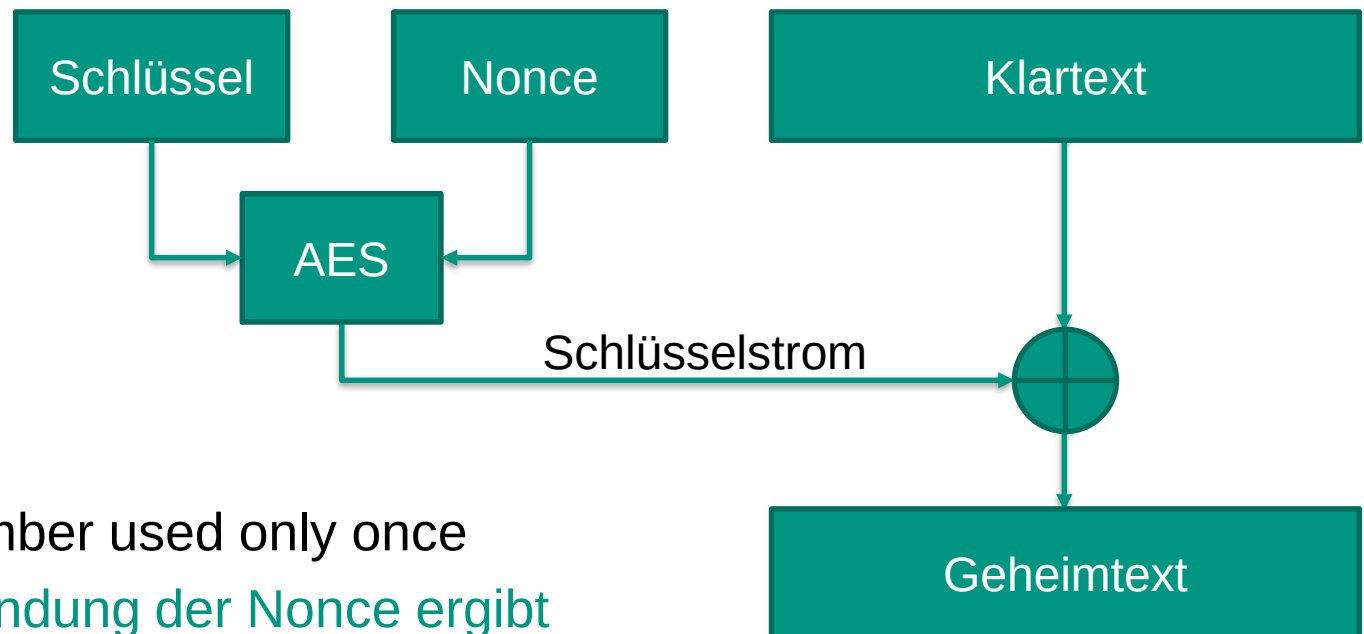
Sicherheit in IEEE 802.15.4-2015

- Optionaler Schutz von **Integrität** und **Vertraulichkeit**
 - ... und gegen **Wiedereinspielen** (Replay-Angriffe)
 - Anwendbar auf alle Dateneinheiten außer Schicht-2-Quittungen
- Verwendet **symmetrische Kryptoverfahren**
 - AES-CBC-MAC (Integritätssicherung)
 - AES-CCM (Verschlüsselung)
- Nicht spezifiziert: **Schlüsselaustausch!**
- **Schlüsselverwaltung**
 - Bis 255 Schlüssel können parallel genutzt werden
 - Auswahl anhand von Adressen oder expliziter Indizierung
 - Verwendung für **individuelle Verbindungen** und **Gruppenkommunikation**



Stromchiffren und Schlüsselverwendung

- IEEE 802.15.4 verwendet **AES-CCM**
 - Grundprinzip: Geheimtext = Schlüsselstrom **XOR** Klartext
 - Schlüsseldatenstrom wird aus Schlüssel und Nonce abgeleitet



- Nonce $\hat{=}$ Number used only once
- Wiederverwendung der Nonce ergibt gleichen Schlüsselstrom
 - Ermöglicht verschiedene Angriffsformen

Probleme bei der Schlüsselverwaltung und -verwendung

- Wie verhindern, dass Nonce und Schlüssel gemeinsam wiederholt genutzt werden?
- Aufbau der Nonce bei IEEE 802.15.4:

Absenderadresse (8 Byte)	Frame-Zähler (4 Byte)	Sicherheitsstufe (1 Byte)
konstant (je Knoten)	variabel	tendenziell konstant

- Zähler wird bei jeder Verwendung inkrementiert
 - Bis 802.15.4-2011: Ein Zähler je Zielsystem
 - Gleiche Schlüssel können für mehrere Adressaten genutzt werden
→ Wiederholte Nutzung eines Schlüssels mit gleicher Nonce möglich!
 - Seit 802.15.4-2015: Ein Zähler je Schlüssel
 - Zähler begrenzt Lebensdauer des Schlüssels → Erfordert Migrationsstrategie
- Verhalten bei Stromausfall?
 - Schlüssel und Zähler dürfen nicht unterschiedlich flüchtig gespeichert sein

TLS/DTLS im Internet of Everything

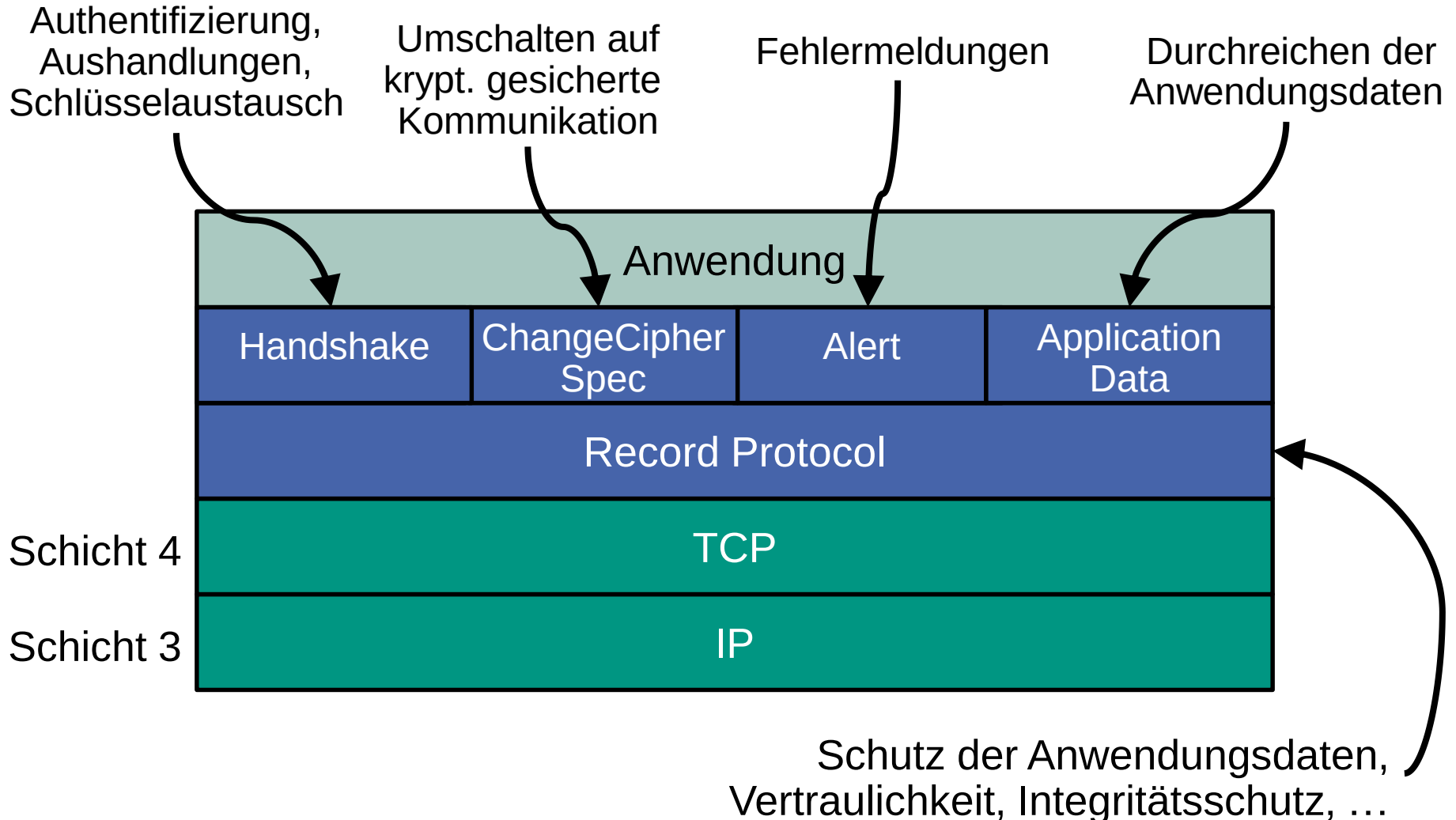
- Idee: **Nutzung bewährter Schutzmechanismen**
 - Versuche „das Rad neu zu Erfinden“ gehen oft schief!

- Internet-orientierter Ansatz
 - **Ende-zu-Ende Absicherung**
 - Konfiguration und Kontrolle auf Anwendungsebene

- Problem: **TLS ist für das IoE nur eingeschränkt geeignet**
 - Komplexer und umfangreicher Standard → Speicherbedarf!

- Daher: Weiterentwicklung von Standards
 - **DTLS** basiert auf TLS, ist aber besser für drahtlose IoE-Netze geeignet
 - **DICE** spezifiziert ein TLS/DTLS-Profil mit angepasstem Umfang

Grundlegende TLS-Protokollstruktur



Grundlegende Komponenten von TLS

■ Record Protocol erfüllt zwei Schutzziele

- Vertraulichkeit
- Integritätssicherung



■ Kryptografische Verarbeitung der zu übertragenden Daten

- Symmetrische Kryptografie zum Schutz der Vertraulichkeit
 - Verschiedene Algorithmen möglich, z.B. AES oder RC4
- Integritätssicherung mittels (H)MAC
- Schlüsselmateriale muss durch anderes Protokoll ausgehandelt werden
 - TLS Handshake Protocol zur Aushandlung des Schlüsselmateriale

■ Record Protocol nutzbar für Protokolle der höheren Schichten

- z.B. TLS Handshake Protocol, TLS Alert Protocol, ...



Unterschiede TLS und DTLS

- Einige Unterschiede in der Funktionsweise von TLS und DTLS
 - Jedoch gleiche Schutzziele erreichbar
- Grundproblem: TLS setzt zuverlässiges Transportprotokoll voraus
 - Paketverluste werden z.B. von TCP behandelt
 - IoE/IoT Anwendungen nutzen häufig UDP
 - Paketverluste werden nicht behandelt



- Änderungen in DTLS
 - Mechanismen zur Behandlung von Paketverlusten in DTLS integriert
 - Explizite Sequenznummern, Sendewiederholungen, Timer, ...
 - Erweiterung des Handshake-Protokolls zur Fragmentierung, Erhaltung der Reihenfolgetreue und dem Erkennen von Paketverlusten
 - Keine (direkte) Verwendung von Stromchiffren mehr möglich
 - Reihenfolgetreue notwendig, mit UDP nicht immer gegeben

DICE – DTLS in Constrained Environments

- Ziel: **Effiziente** Nutzung von DTLS in IoT/IoE-Anwendungen
 - Funktionalität in DTLS sehr Umfangreich
 - Viele Funktionen in IoT-Szenarien nicht notwendig
 - Unterstützung sehr vieler kryptografischer Algorithmen
 - Rechenzeit und Energieverbrauch auf IoT Plattformen z.T. zu hoch
- DICE definiert Konfigurationen, die *effizient und sicher* sind

- DICE bietet



[RFC 7925]

- **Authentifikation** zweier Kommunikationsendpunkte
- **Integrität** und **Authentizität** der Daten
- **Vertraulichkeit**
- Implementierbarkeit **auf ressourcenbeschränkten IoT/IoE-Systemen**

DICE DTLS-Profil

- Einige ausgewählte Funktionen und Konfigurationen in DICE
 - **Einschränkung der zu unterstützenden kryptografischen Algorithmen**
 - Einige wenige sichere, aber effiziente Cipher-Suites
 - **Einschränkung der zu unterstützenden Authentifizierungsmechanismen**
 - Pre-shared Keys, Raw Public Keys, Zertifikate
 - **Einschränkung der Protokollerweiterungen und Optionen**
 - TLS und DTLS bieten sehr viele Optionen und Konfigurationen
 - Sitzungswiederaufnahme, Keep-Alive Nachrichten, Schutz vor DoS-Angriffen, unterschiedliche Komprimierungsfunktionen, ..
 - Führt zu komplexen und speicheraufwändigen Implementierungen
 - Reduzierte bzw. angepasste Konfigurationen für IoT Geräte und Anwendungen
 - Einschränkung der Funktionalität, dafür höhere Effizienz

DICE Cipher-Suites

- Sehr viele **Cipher-Suites** in TLS/DTLS
 - **Beschreiben Kombination aus kryptografischen Mechanismen** für
 - Schlüsselaustausch
 - Authentifikation
 - Schutz von Integrität/Authentizität
 - Verschlüsselungsalgorithmen
 - >300 unterschiedliche Kombinationen
 - **Viele nicht praktikabel einsetzbar** auf ressourcenbeschränkten Systemen
 - Rechenzeit häufig mehrere Sekunden, z.B. RSA, DSA, DH, ...
 - ... **oder inzwischen veraltet und unsicher!**
- Kryptografische Algorithmen für DICE Profile
 - Symmetrische Verschlüsselung mit **AES**
 - Hashfunktion **SHA-256**
 - Asymmetrische Kryptografie auf Basis von **Elliptischen Kurven**
 - Effizienter implementierbar, geringere Schlüssellänge erforderlich

DICE Authentifizierungsmechanismen

- Einschränkung der zu unterstützenden
 - Schlüsselaustauschmechanismen
 - Authentifizierungsmechanismen
- Vorverteilte (pre-shared) symmetrische Schlüssel
 - Geringer Rechenaufwand, geringer Kommunikationsaufwand
 - Vorverteilung in kryptografischer Hardware (TPM) oder in Firmware
- Raw Public Keys
 - Jedes System besitzt asymmetrisches Schlüsselpaar
 - Nicht Teil eines Zertifikates, Authentifikation nicht Teil des Schlüsselaustauschs
 - Vorverteilung in kryptografischer Hardware (TPM) oder in Firmware
- Zertifikate
 - Nutzung von Diffie-Hellman auf Basis von elliptischen Kurven
 - Minimale X.509 Zertifikate auf Basis von ECDSA
 - Zertifikatvalidierung bzw. Prüfung auf Widerruf kann ausgelagert werden

Zusammenfassung DICE

- Erster Versuch, Sicherheitsmechanismen für IoT-Protokolle zu standardisieren
 - (Bisher) keine neuen Protokolle
 - Reduzierter Umfang an Funktionen und Optionen
 - **Optimierte Konfigurationen**
 - Speichereffizient
 - Effiziente Berechnung auf ressourcenbeschränkten Geräten
- **Beschränkung auf zwei Cipher-Suites** für DICE
 - Symmetrisch: **TLS_PSK_WITH_AES_128_CCM_8**
 - Asymmetrisch: **TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8**
- Zukünftige Entwicklung?
 - Trend weg von TLS, hin zu **Schutzmechanismen auf Anwendungsebene**
 - Object Security of CoAP (OSCOAP)
 -  [<https://tools.ietf.org/html/draft-ietf-core-object-security-01>]

Security by Delegation

- Existierende Sicherheitslösungen sind zu komplex für das IoE
 - Lösungsansatz: DTLS + DICE

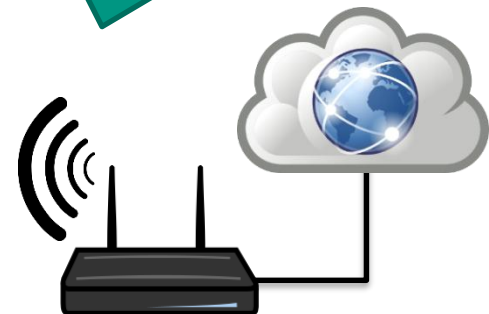
- Aber auch mit DICE
 - Asymmetrische Kryptographie ist teuer
 - Authentifizierung und Autorisierung der Nutzer ist ressourcenaufwändig
 - Rechenzeit/Energie (PKI, Zertifikatsvalidierung)
 - Speicher (Programmcode, Zugriffskontrolllisten)

- Ansatz Security by Delegation
 - Sicherheitsfunktionalität wird an vertrauenswürdigen Stellvertreter delegiert

– Speicher
– Rechenleistung
– Energie



+ Speicher
+ Rechenleistung
+ Energie

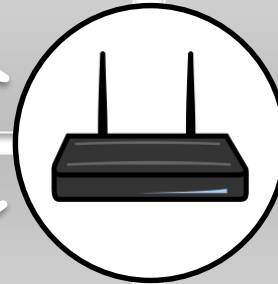
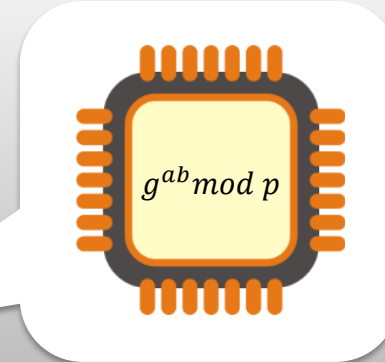


Delegationsformen

Authentifizierung und Autorisierung



Komplexe Berechnungen
(asymmetrische Kryptographie)



Authentifizierung und
Schlüsselaustausch



Aufbau sicherer Verbindungen



Vor- und Nachteile von Delegation

- Stellvertreter übernimmt Aufgaben mit hohem Ressourcenbedarf
 - Ermöglicht **Einsatz von Verschlüsselung mit einfacher Hardware**
 - Speicherbedarf
 - Rechenaufwand
 - **Zeitersparnis** beim Verbindungsaufbau
 - **Zentralisierte Verwaltung** von Zugriffsrechten
- Aber: **Zusätzlicher Kommunikationsaufwand**
 - Weniger Berechnung vs. mehr Datenaustausch – Energiebilanz?
- **Ressourcenbeschränkte Systeme müssen Stellvertreter vertrauen**
 - Führt sicherheitsrelevante Aufgaben in ihrem Auftrag durch
 - **Vermittelt Vertrauensbeziehungen** zwischen Endsystemen
 - **Stellvertreter sind kritische Angriffsziele**



Fallbeispiel: Delegated CoAP Authentication and Authorization Framework (DCAF)

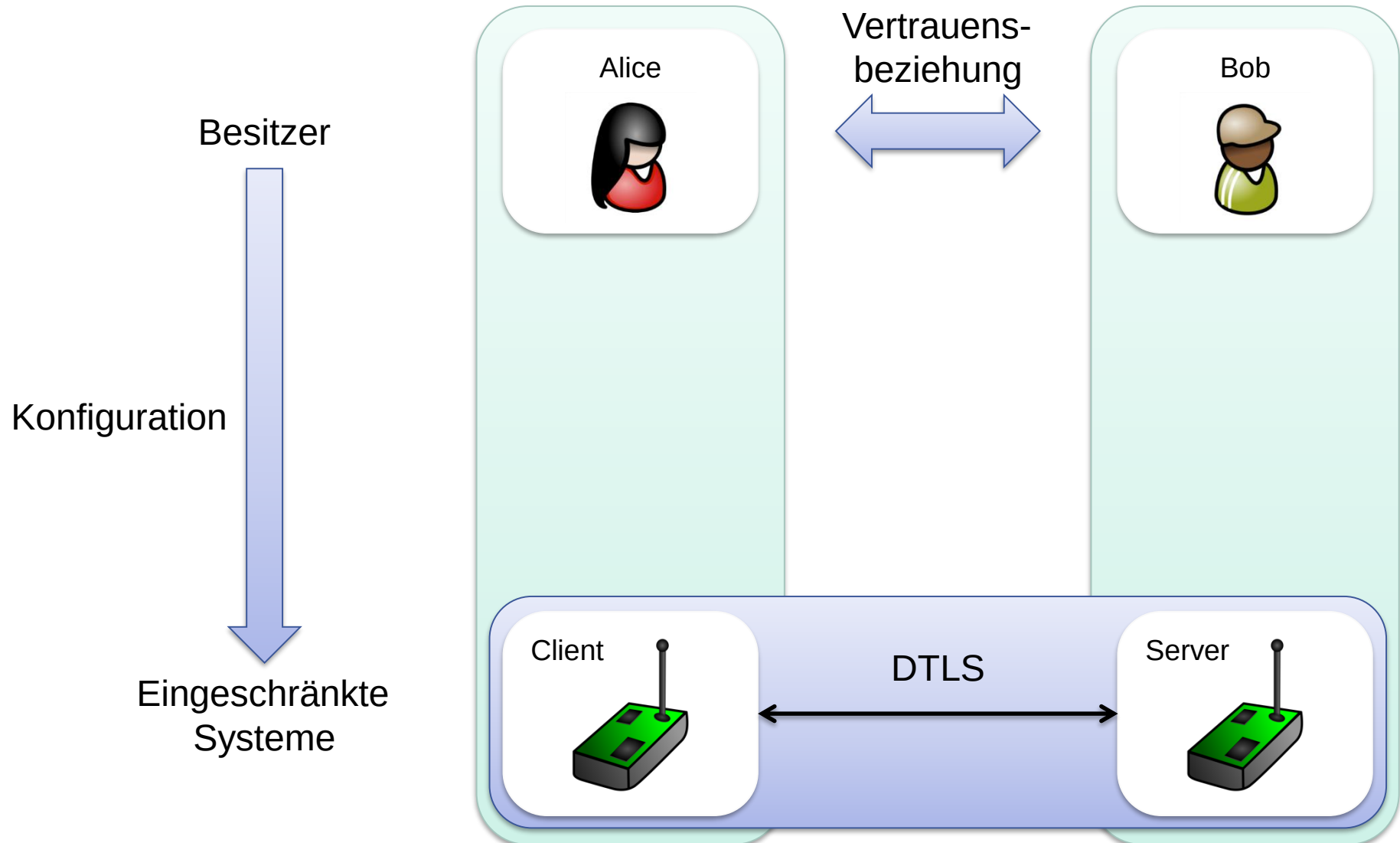
- In der IETF-Arbeitsgruppe Authentication and Authorization for Constrained Environments (ACE) diskutierter Protokollentwurf



[<https://tools.ietf.org/html/draft-gerdes-ace-dcaf-authorize-04>]

- Spezifiziert einen **Delegationsmechanismus für CoAP**
 - Eingebettet in CoAP, **keine eigene Protokollschicht**
 - Stellvertreter sind normale CoAP-Server
 - Kommunikation mit und zwischen Stellvertretern über CoAP POST
- Eigenschaften
 - Nur symmetrische Verschlüsselung auf ressourcenbeschränkten Geräten
 - Sowohl Client als auch Server können ressourcenbeschränkt sein
 - Individuelle Autorisierung für jede URI und CoAP-Methode
 - Beispiel: { PUT, GET } coaps://frontdoor.example.net/status/locked

Direkte Kommunikation ohne Delegation



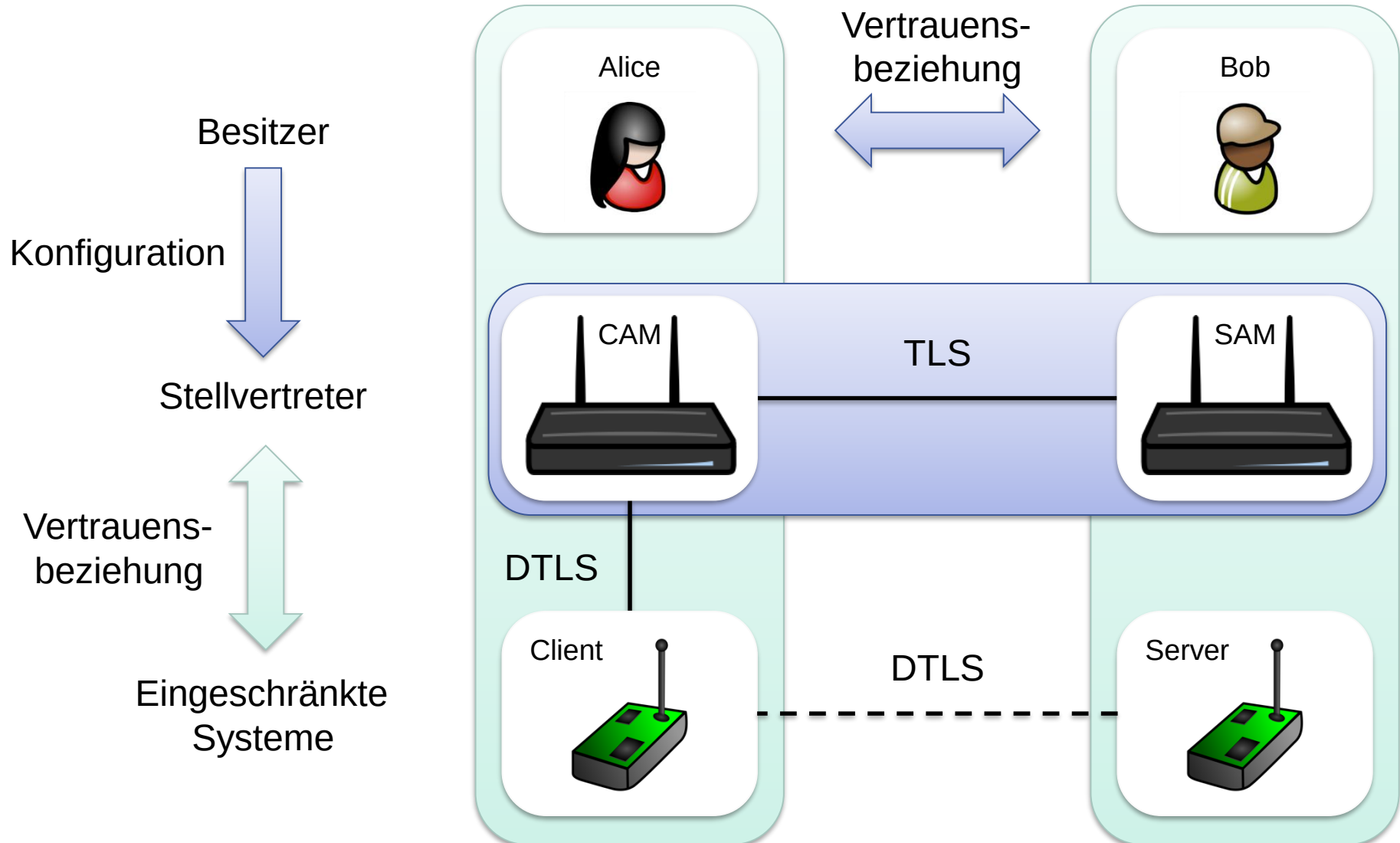
DCAF – Rollen

- Bereits bekannt
 - Besitzer
 - Eingeschränkte Systeme
 - Client
 - Server

- Vertrauensbeziehungen zwischen Besitzern werden auf die Konfiguration entsprechender Zugriffsberechtigungen abgebildet
 - Authentifizierung mittels Zertifikaten oder gemeinsamem Geheimnis

- Zusätzlich bei DCAF: Stellvertreter
 - Client Authentication Manager (CAM)
 - Übernimmt die Rolle des Clients im Autorisierungsprozess
 - Server Authentication Manager (SAM)
 - Vertritt den Server im Autorisierungsprozess

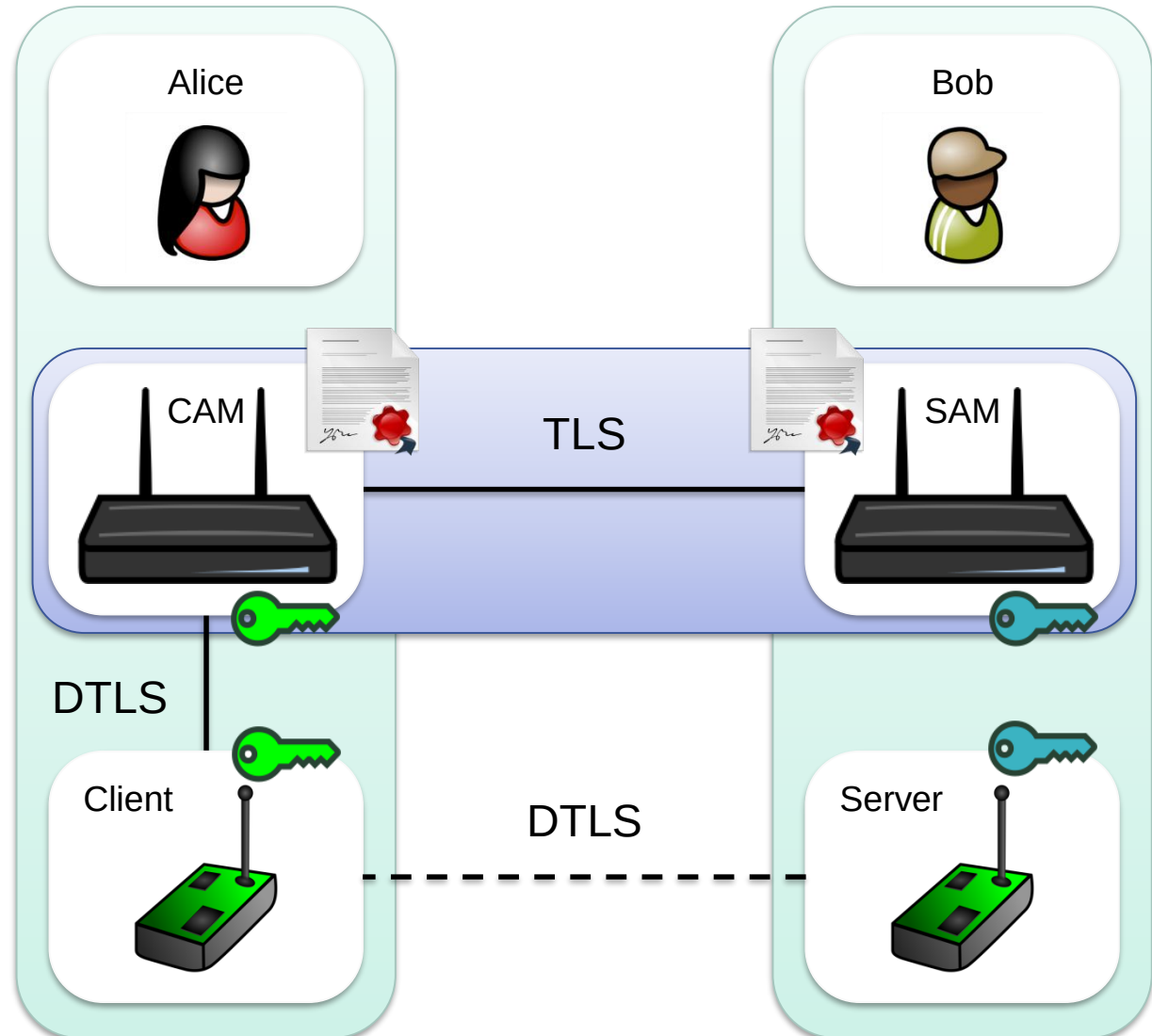
Delegation mit DCAF – Architektur



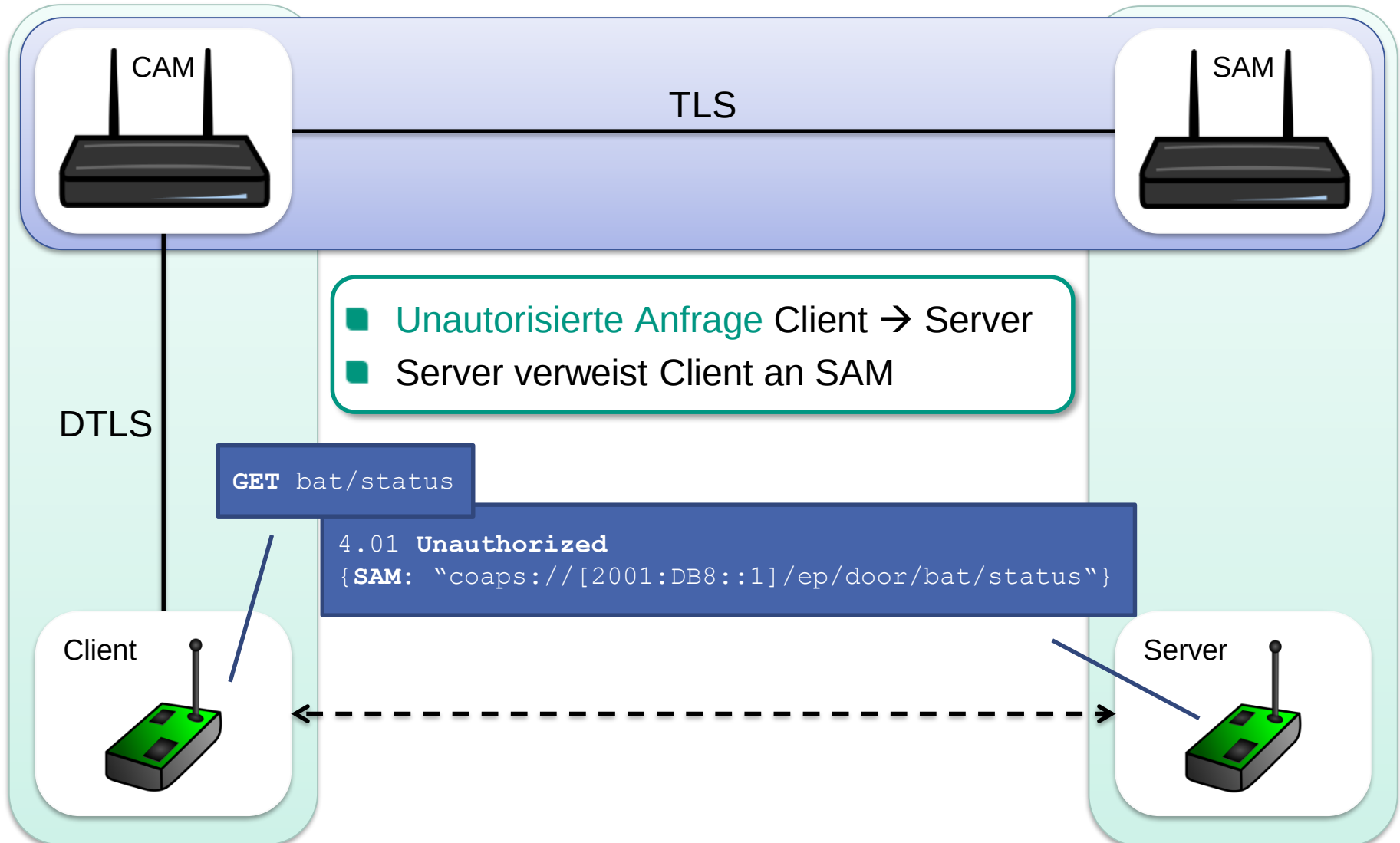
Delegation mit DCAF – Architektur

- Sym. Schlüssel für Client ↔ CAM und Server ↔ SAM
 - Authentifizieren Client/Server gegenüber dem Stellvertreter

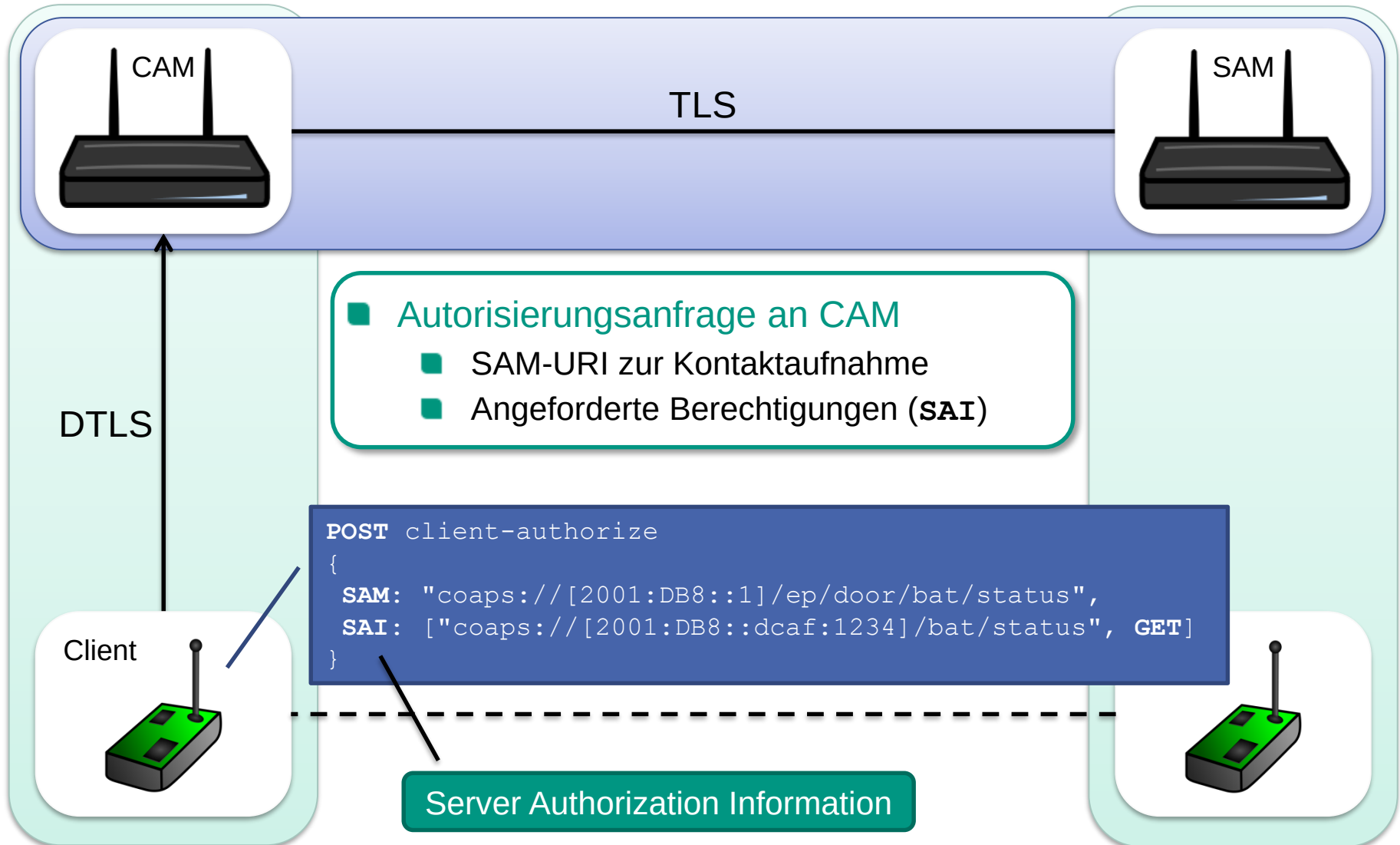
- Wechselseitige Authentifizierung von CAM/SAM
 - Zertifikate oder vergleichbare Mechanismen



Ablauf



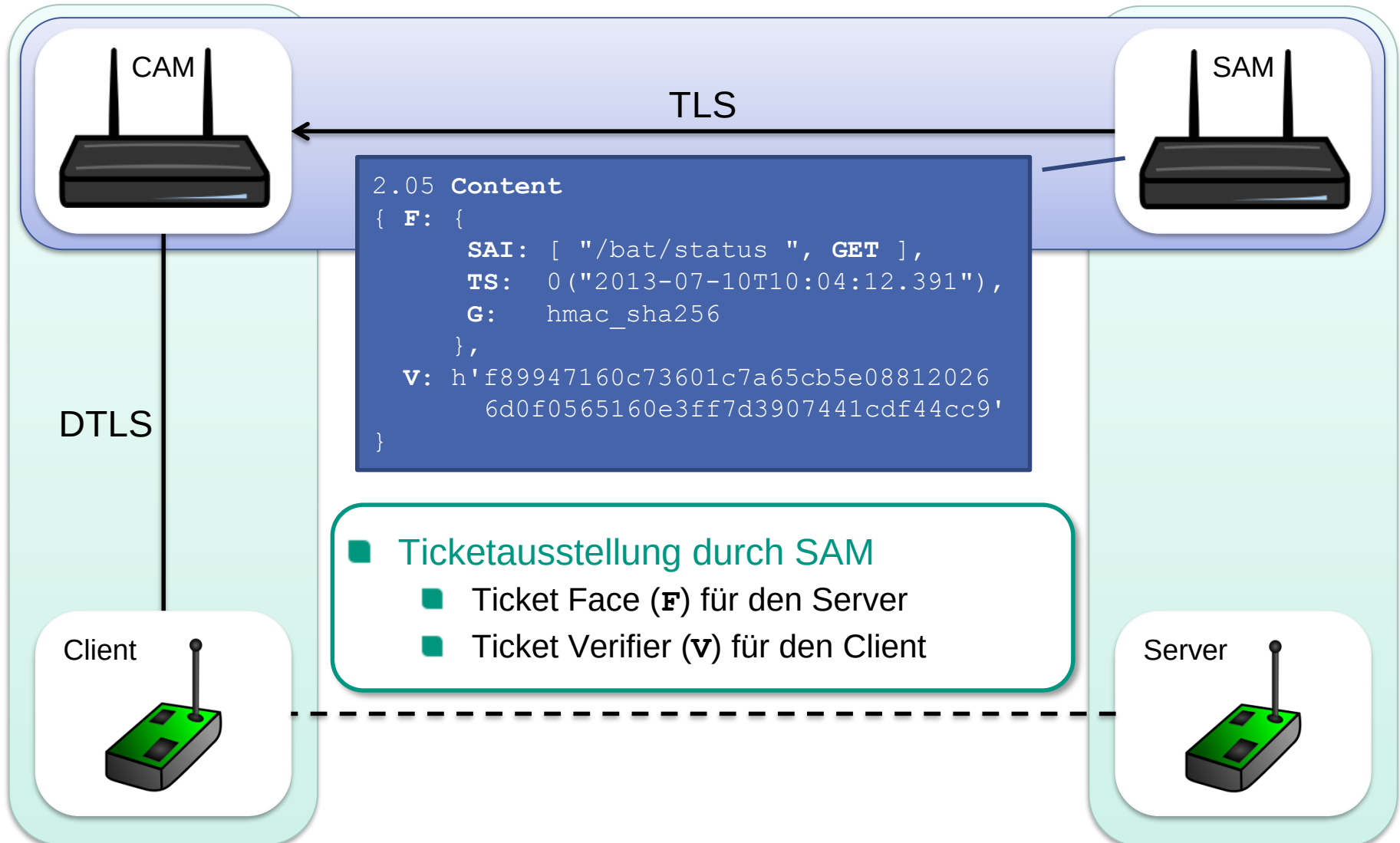
Ablauf



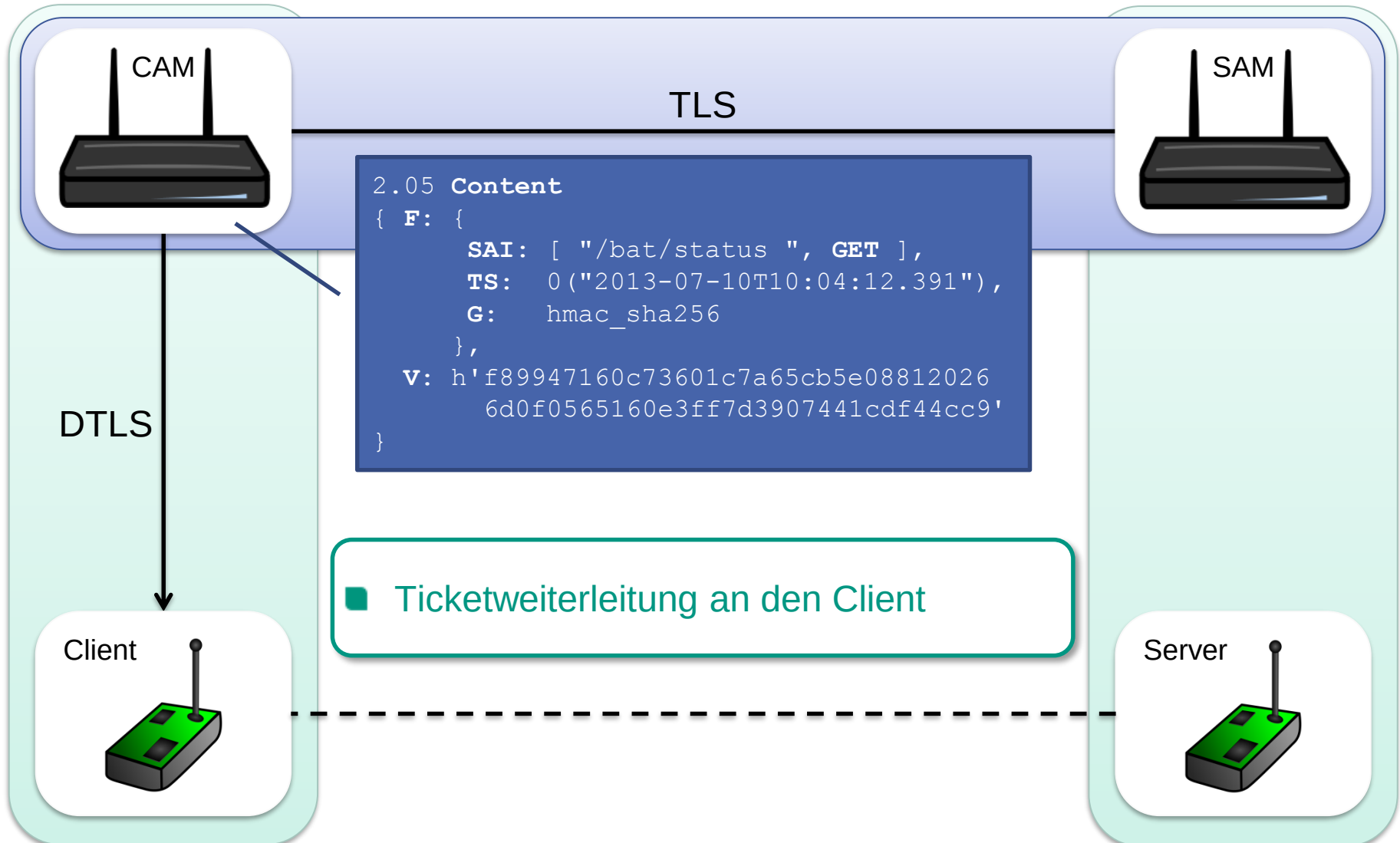
Ablauf



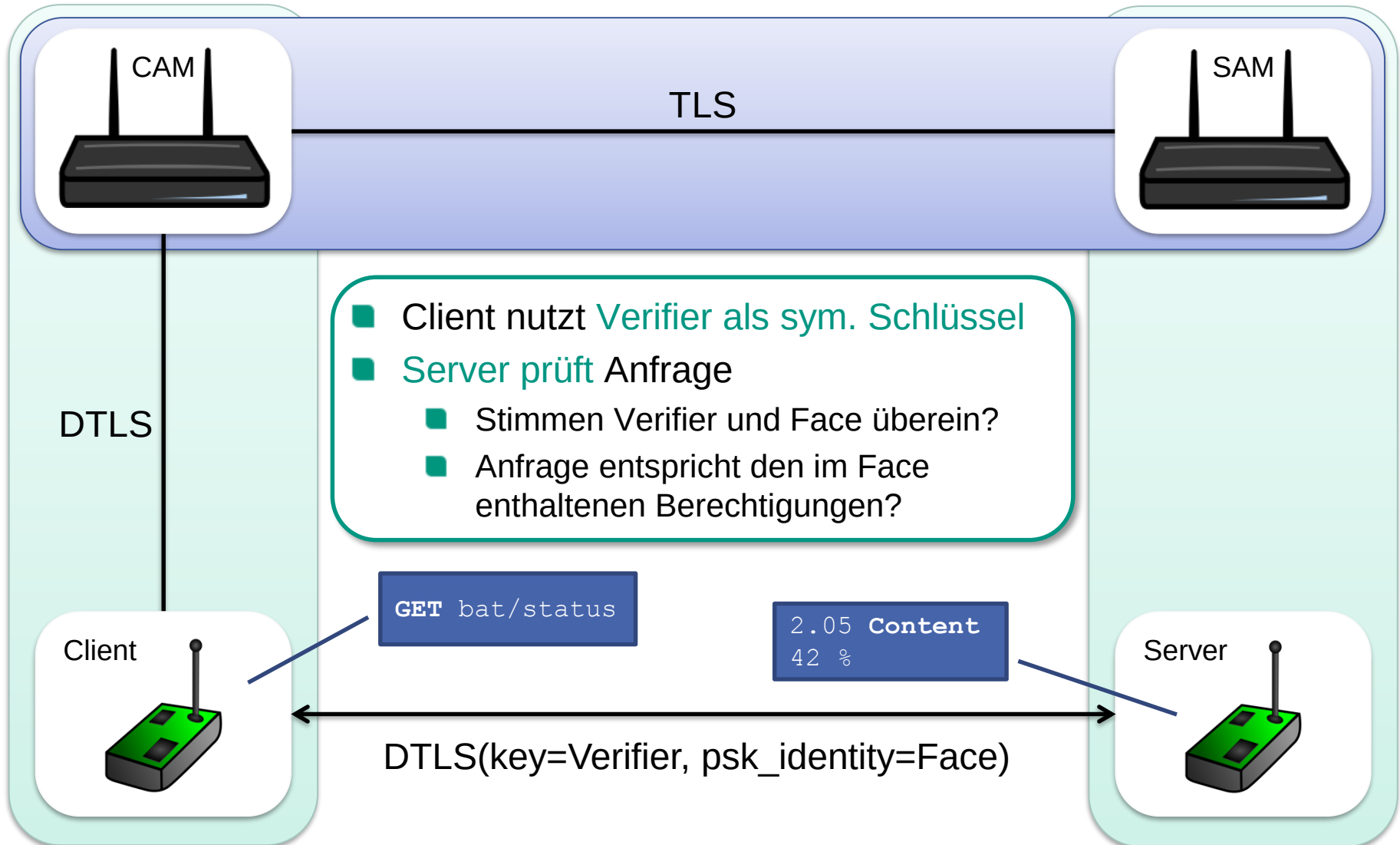
Ablauf



Ablauf



Ablauf



DCAF Ticket Face und Verifier

■ Ticket Face

- Gewährte Zugriffsberechtigungen (**SAI**)
- Zur Validierung erforderliche Informationen
 - Nonce (**N**) oder Zeitstempel (**TS**)
 - ... schützen gegen Replay-Angriffe (Angreifer „recycled“ Ticket)
- Eine **Methode zur Schlüsselableitung** (**G**) oder ...
- Einen **symmetrischen Schlüssel** für die Verbindung Client – Server (**v**)
 - ... erfordert dass das Ticket Face verschlüsselt wird (Grund: Nächste Folie)

■ Ticket Verifier (für den Client)

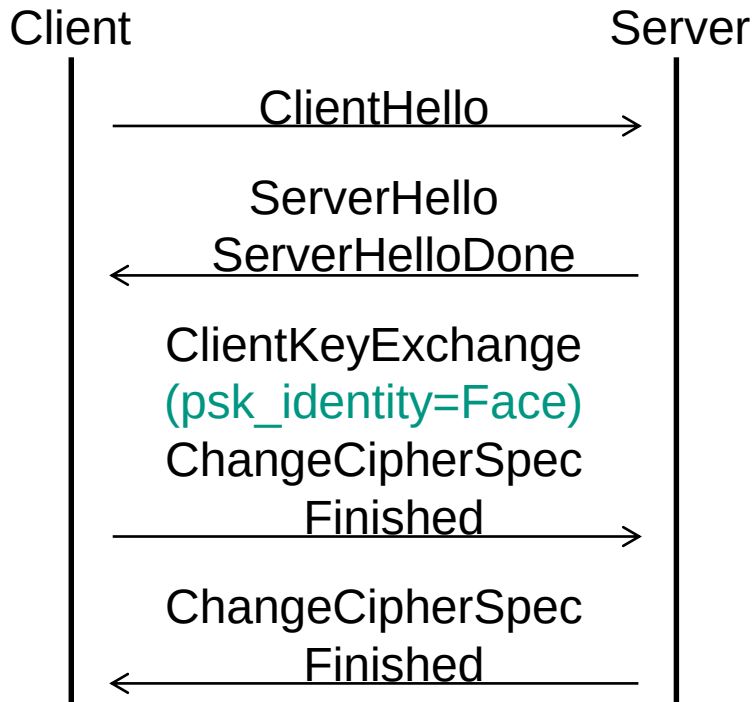
- **Symmetrischer Schlüssel**
- Eindeutig dem Ticket Face zugeordnet

```
{ F: {  
    SAI: [ "/bat/status ", GET ],  
    TS:  0("2013-07-10T10:04:12.391"),  
    G:   hmac_sha256  
  },  
  v: h'f89947160c73601c7a65cb5e08812026  
      6d0f0565160e3ff7d3907441cdf44cc9'  
}
```

Ticket bei Schlüsselableitung

Schlüsselübertragung und -ableitung

- Der **Server leitet den Verbindungsschlüssel aus dem Ticket Face ab**
 - Ticket Face muss an den Server übertragen werden
- DCAF nutzt dazu den „**PSK Identity Hint**“ im TLS-Handshake
 - Übertragung erfolgt im Klartext, da noch keine sichere Verbindung existiert
 - Schlüsselmaterial muss zusätzlich verschlüsselt sein



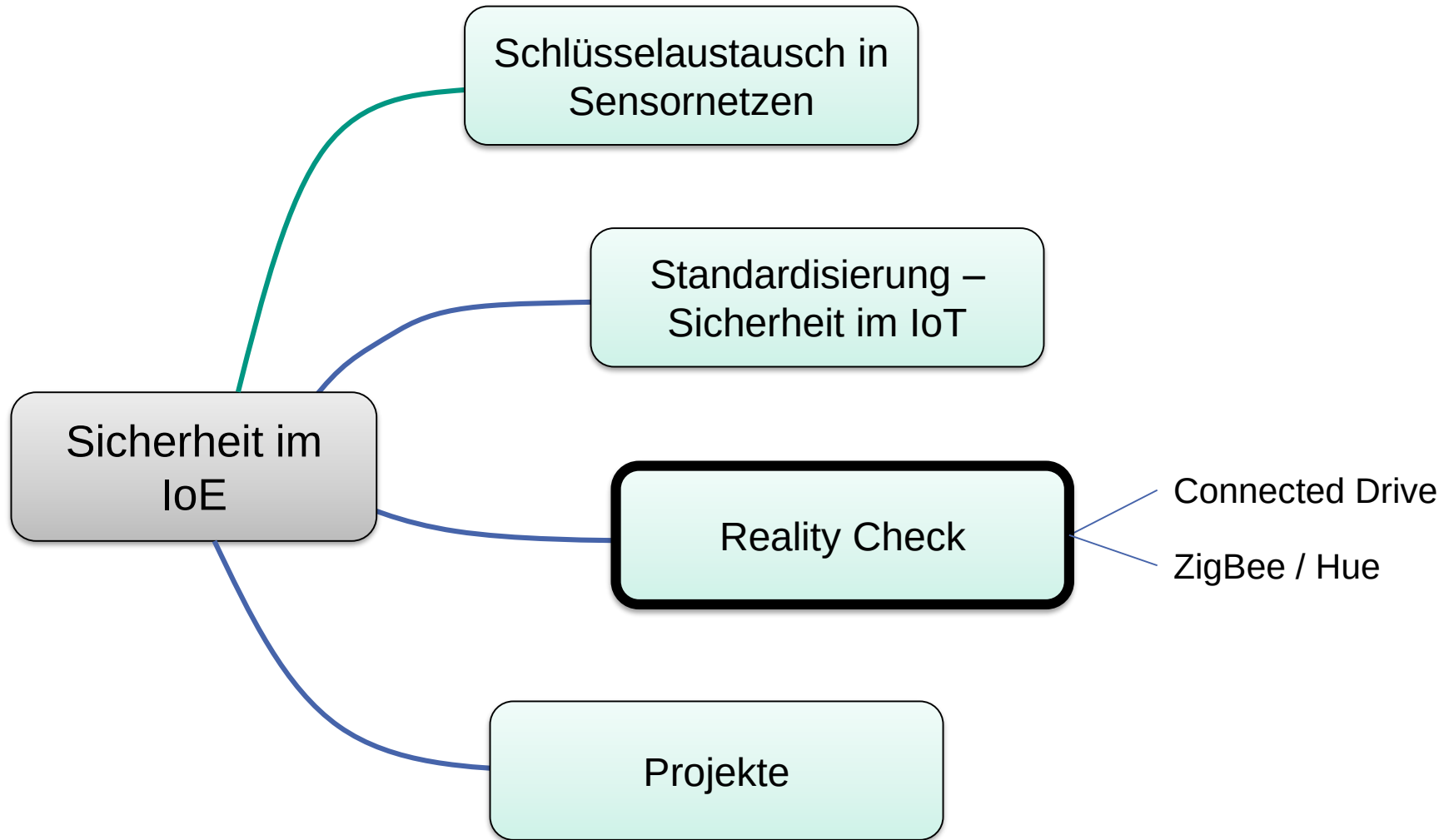
■ Schlüsselableitung

- Ticket Face unverschlüsselt:
 - $\text{Key} = \text{HMAC}_{\text{ServerKey}}(\text{Face})$
- Ticket Face verschlüsselt:
 - $\text{Key} = \text{Decrypt}_{\text{ServerKey}}(\text{Face}).v$
- **Verbindungsaufbau schlägt fehl wenn das Ticket Face manipuliert wurde**
 - Server errechnet falschen Schlüssel

Zusammenfassung Security by Delegation

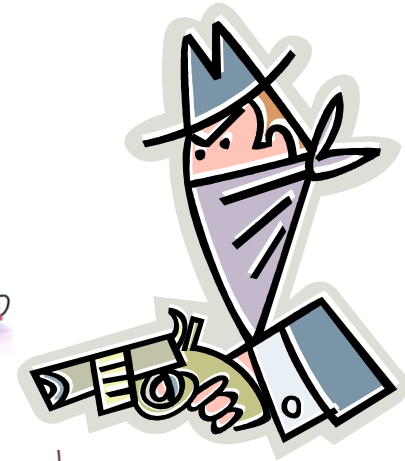
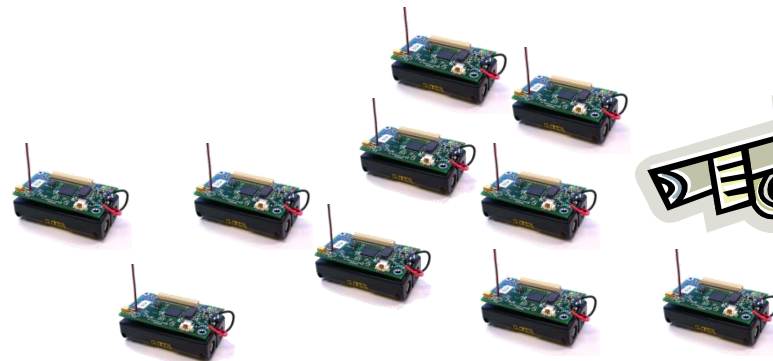
- Delegation von Sicherheitsfunktionalität
 - Authentifizierung und Autorisierung
 - Komplexe Berechnungen (asymmetrische Kryptographie)
 - Schlüsselaustausch
 - Aufbau sicherer Verbindungen
- An vertrauenswürdigen Stellvertreter
 - Sicherheitskritisch!
- Ziel: Entlastung ressourcenbeschränkter Systeme
 - Geringerer Speicherbedarf
 - Weniger Rechenzeit
- Kosten/Nutzen-Abwägung: Delegation erfordert Kommunikation

Schwerpunkte des Kapitels im Überblick



Zur Erinnerung: Angreifermodell im IoE

- Geräte häufig an öffentlichen oder leicht zugänglichen Orten
 - Angreifer kann physisch auf Geräte zugreifen
 - Kann Geräte evtl. einfach „klauen“ oder physisch zerstören
 - Manches wird auch einfacher
 - Speicher auslesen
 - Komplette re-programmieren
 - Korumpieren
- Funkmedium bietet Angreifern viele Möglichkeiten
 - ... und der Angreifer hat mehr Ressourcen



Angriffe auf/über den Funkkanal

- Passiv
 - Abhören (Eavesdropping)
- Aktiv
 - Einbringen eigener Dateneinheiten (Injection)
 - Wiederholen fremder Dateneinheiten (Replay-Angriff)
 - Weiterleiten von Funksignalen (Wormhole-Angriff)
 - Einbringen eigener Geräte (Intrusion)
 - Vortäuschen von Geräten (Sybil-Angriff)
- Denial of Service
 - Störung von Übertragungen (Jamming, Flooding)
 - Fragmentierung und Puffer-Überlastung
- Routing
 - (Selektives) Unterdrücken/Weiterleiten von Dateneinheiten (Sinkhole)
 - Manipulation von Metriken und Topologien



Angriffe auf die Hardware

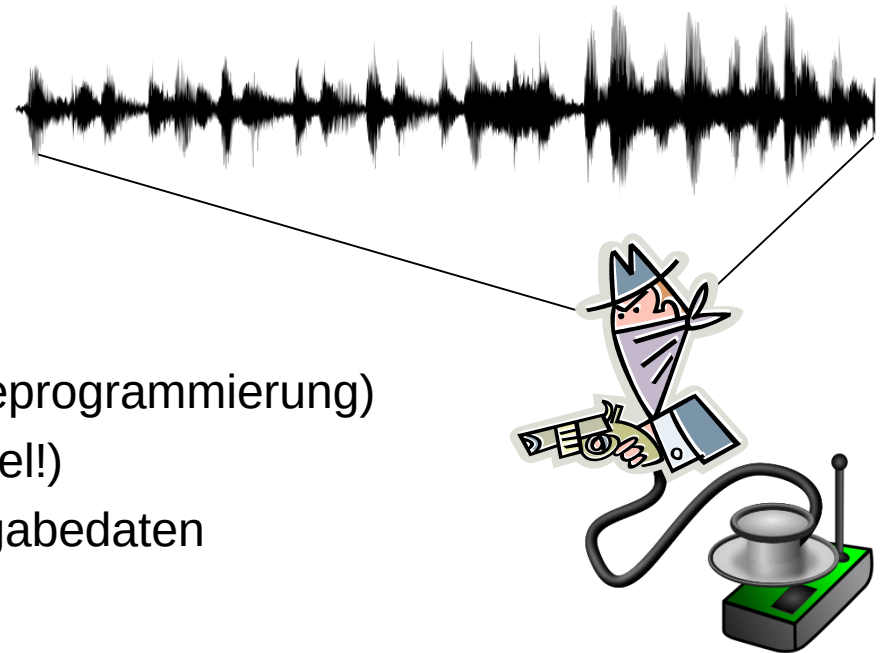
■ Passiv/Seitenkanalangriffe

→ Extraktion von Schlüsselmateriale

- Elektromagnetische Abstrahlungen (TEMPEST)
- Akustische Signale
- Leistungsaufnahme/Energieverbrauch
- Laufzeitverhalten

■ Aktiv

- Klonen von Hardware
- Manipulation der Anwendung (Reprogrammierung)
- Auslesen des Speichers (Schüssel!)
- Vorspiegeln falscher Sensor/Eingabedaten



FALLBEISPIEL

BMW CONNECTED DRIVE

Exkurs: BMW ConnectedDrive

- Cloud-basierter Dienst zur Steuerung des Autos aus der Ferne

Ein Smartphone als Fernbedienung für Ihren BMW

Mit den Remote Services verwandeln Sie Ihr Smartphone in eine intelligente und komfortable Fernbedienung für Ihr Fahrzeug – über die My BMW Remote App oder das BMW Callcenter. Ab sofort genügt ein Griff in die Jackentasche und Sie haben über Ihr iPhone oder Android-Gerät Zugriff auf Ihren BMW. Sie sind nicht mehr sicher, ob Sie Ihren BMW abgeschlossen haben? Oder haben Ihren Schlüssel nicht griffbereit? Dann ver- und entriegeln Sie Ihr Fahrzeug mit den Remote Services von BMW ConnectedDrive. Und sollten Sie Ihr Fahrzeug einmal nicht auf Anhieb finden, können Sie die Lichthupe und Hupe über das Smartphone betätigen und Ihr Fahrzeug so schnell ausfindig machen.



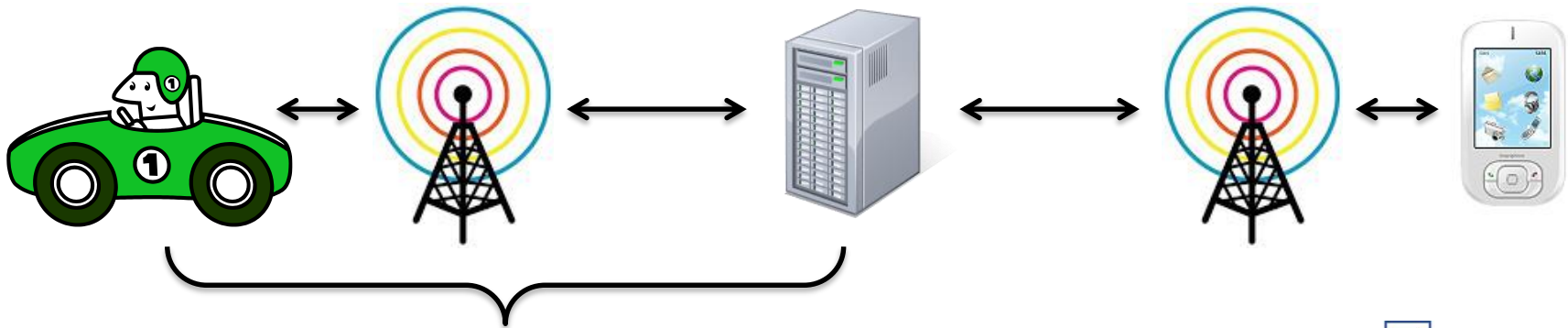
[BMWCD]

- Was kann da schon schief gehen?



Architektur

■ Cloud-Service

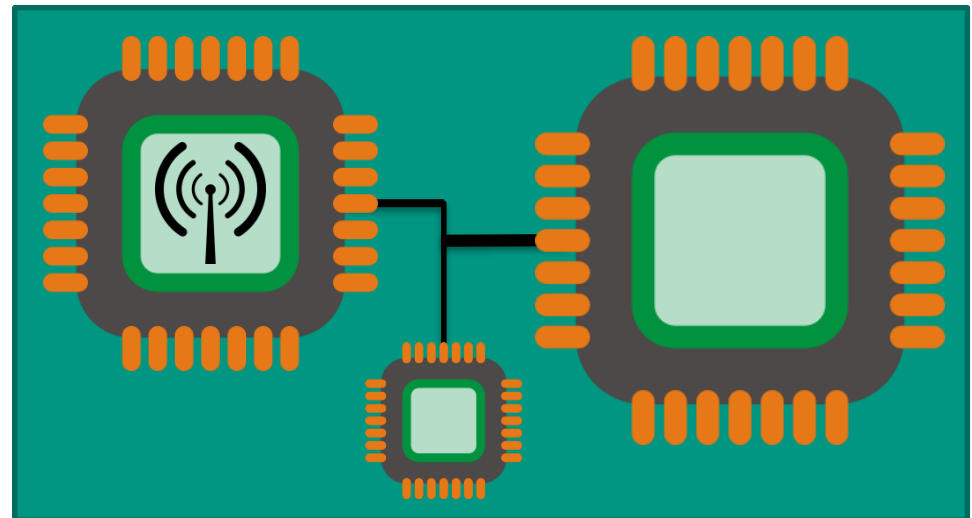


SMS + HTTP über GPRS/EDGE


[Heis15]

■ Steuergerät

- Hauptprozessor
Renesas SH-4A
- Mikrocontroller (Koprozessor)
Renesas V850ES
- Mobilfunk-Modem
Cinterion



Schutzmechanismen

- Wie funktioniert?
 - Sicherheitsmechanismen sind im Modem implementiert
 - Mikrocontroller realisiert Basisfunktionalität und überwacht das Modem
- Verschlüsselung
 - AES (sicher) und DES (unsicher)
- Signaturverfahren
 - DES-CBC-MAC, HMAC-SHA1 und HMAC-SHA256

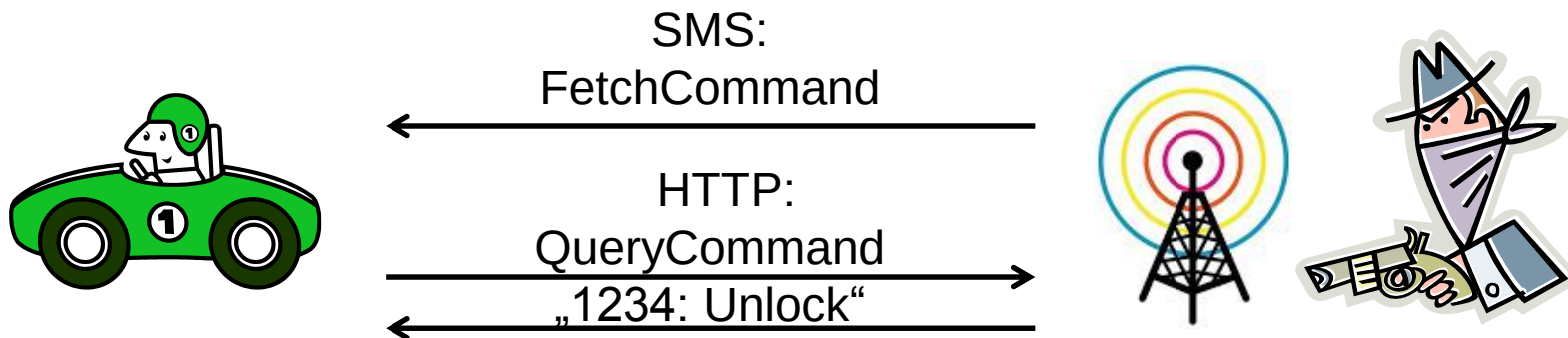
Bis jetzt nichts falsch gemacht. Bis jetzt ...

- Schlüsselmateriale
 - Im Modem fest einprogrammiert
 - Für alle Fahrzeuge gleich
 - Relativ einfach zu extrahieren



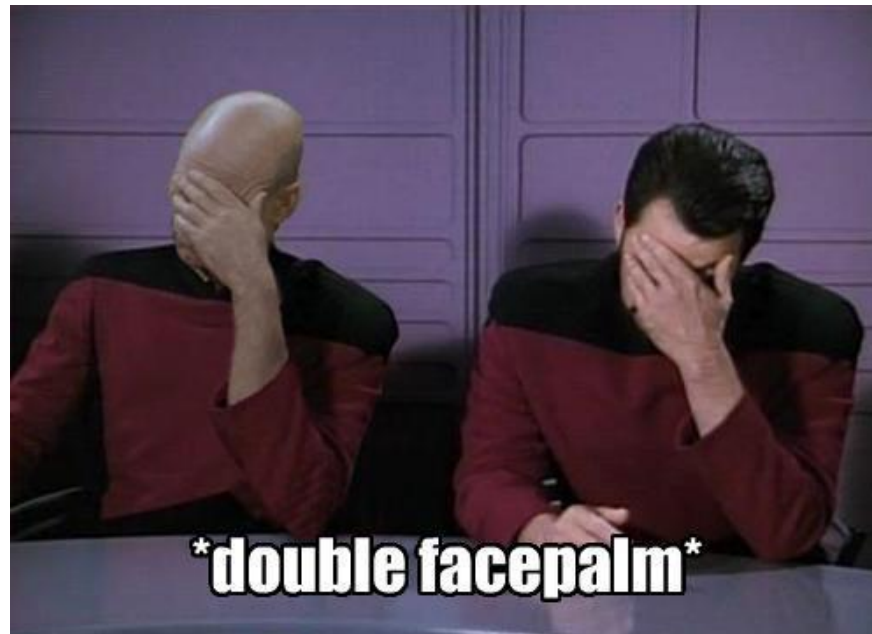
Angriff!

- Ausgangspunkt: Eine Mobilfunk-Basisstation (**IMSI-Catcher**)
 - Das Auto bucht sich in das gefälschte Netz ein
- Angreifer weckt das Steuergerät mit einer SMS
 - „Gesichert“ mit einem (nicht mehr) geheimen Schlüssel
- Auto fordert Befehle aus der Cloud an (über HTTP)
 - HTTP, nicht HTTPS! **Keine Transportverschlüsselung!**
- Angreifer antwortet mit gewünschtem Steuerbefehl
 - Nutzdaten sind mit bekannten Schlüsseln verschlüsselt und signiert



Angriff!

- Noch ist nichts passiert!
 - Auto akzeptiert Befehle nur wenn die Fahrgestellnummer übereinstimmt
 - Sonst sendet es seine Fahrgestellnummer als SMS an den Angreifer ...
- Letzte Hoffnung: ConnectedDrive kann deaktiviert werden ...
 - ... ist über den gleichen Angriff wieder aktivierbar (Konfigurationsupdate)



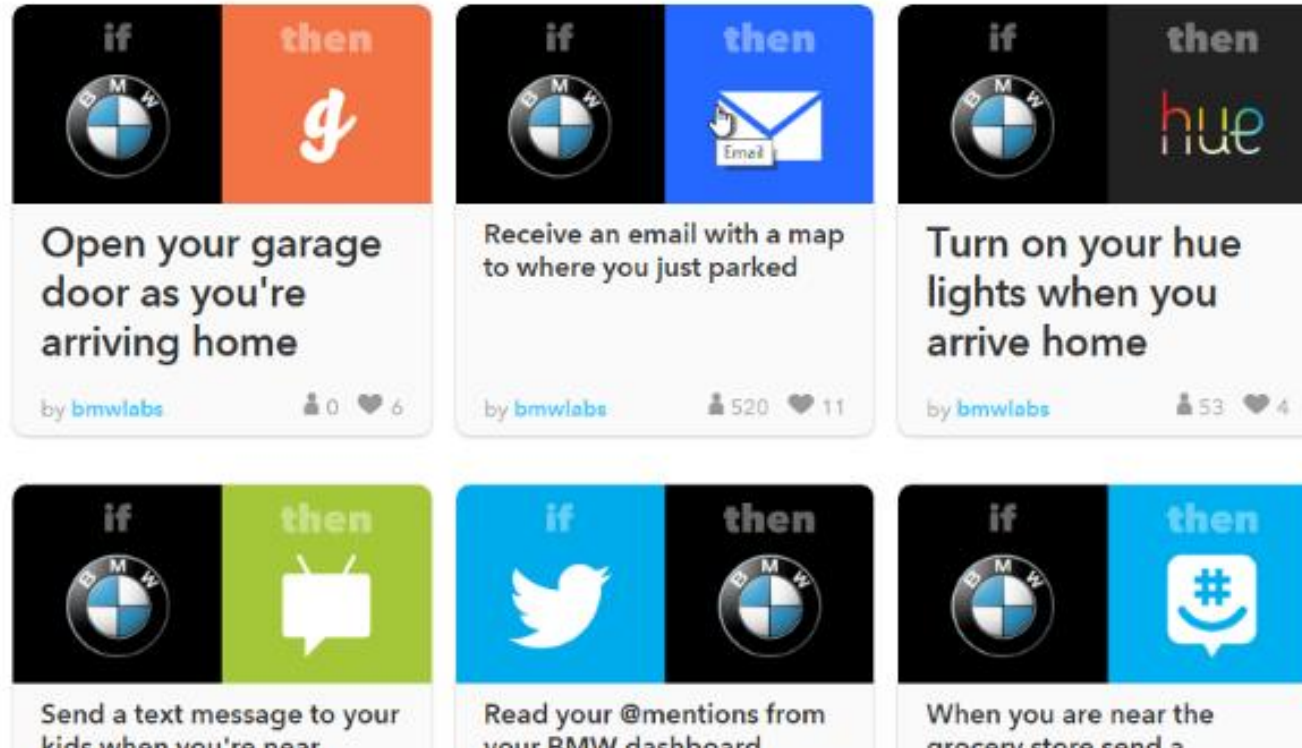
Was ist schief gelaufen?

- Ein **gemeinsamer Schlüsselsatz** für alle Fahrzeuge des Herstellers
 - Netzwerkschlüssel (Single Mission Key)
 - **Single Point of Failure**
- **Keine Authentifizierung** der ConnectedDrive Cloud
 - Mitgliedschaft im Netzwerk genügt
 - Angreifbar selbst wenn der Schlüsselsatz besser geschützt wäre
 - Angriffsskizze: Steuergerät als Krypto-Koprozessor nutzen (Blackbox).
Generierte Nachrichten werden von jedem anderen Steuergerät akzeptiert ...
- Unnötige Preisgabe von zur Validierung nutzbaren Informationen
 - Wobei die Fahrgestellnummer auch andernorts zu finden ist ...

- Nach „klassischen“ Maßstäben wäre das System sicher!
 - ... Dolev-Yao-Angreifer haben keinen Zugriff auf die Hardware ...
 - Im Internet of Everything ist diese Annahme nicht haltbar!
 - Einzelne Schlüssel sind sehr schwer geheim zu halten

Mein dein Auto, mein Haus?

- BWM integriert das Auto in das Internet of Things
 - Über den Automatisierungsdienst (lies: Cloud-Dienst ...) If This Then That
 - Das Auto steuert andere Geräte und Dienste ...
 - ... klingt nach einem Zukunftsmodell. Was soll da schon schiefgehen ...



Und wie geht es weiter?

c't 21/2016, S. 72ff: Risiko Zulieferindustrie

Spätestens wenn die Cloud direkt Einfluss auf das Gaspedal nimmt, spielt die Absicherung der Netzwerkarchitektur eine besondere Rolle. [...] Doch da beim Massengeschäft Auto jeder gesparte Cent zählt, ist es wahrscheinlicher, dass Synergieeffekte zwischen beiden Ebenen gesucht und gefunden werden. **So formulierte ein Zulieferer [...] die Überlegung, dass man die ständig wachsende Prozessorpower im Infotainment-Bereich auch für andere Steuerungskomponenten nutzbar machen könnte, um Kosten zu sparen. Eine physische Trennung [...] wäre damit vom Tisch.** Sie fände bestenfalls noch virtualisiert in Software statt.

- **Gemeinsame Hardwarenutzung** durch Steuersysteme und Infotainment
- Multimedia- und Infotainment-Systeme sind **potentielle Schwachstellen**
 - Schnittstelle zur Außenwelt, verarbeiten nicht vertrauenswürdige Daten
- **Sichere Trennung beider Welten?**
 - Virtualisierung ist problematisch (Komplexität → Fehleranfälligkeit)

FALLBEISPIEL PHILIPS HUE

Exkurs: Philips Hue

- Populäre Plattform für „intelligente“ Beleuchtungssysteme
 - Basierend auf ZigBee Light Link (ZLL)
- Leuchtmittel sind direkt über ZigBee steuerbar
- Zentrale Bridge für Anbindung zum Internet
 - Auch zur Steuerung über WLAN
 - *Und für Firmware-Updates*
- Integration in das Smart Home
 - Amazon Alexa
 - Apple HomeKit



Schutzmechanismen bei Philips Hue

- Im normalen Betrieb: **ZigBee-Verschlüsselung**
 - Ein **Netzwerkschlüssel** für alle Geräte im PAN
 - Sicher solange der Angreifer nicht an den Schlüssel gelangt
- **Gerätemanagement: ZLL TouchLink-Protokoll**
 - Verwaltung von Leuchtmitteln, unter anderem:
 - Zurücksetzen auf Werkszustand
 - Integration in neues Netzwerk (Schlüsselaustausch)
 - **Sicherung durch Beschränkung auf nahes Umfeld (< 1 Meter)**
- Philips Hue **Firmware**
 - Gesichert durch **symmetrische Verschlüsselung**
 - **Sperre (Fuse-Bit) verhindert Auslesen des Mikrocontrollers**
 - Programmcode
 - **Schlüsselmaterial**

Kleine Probleme ...

- ZLL TouchLink Protokoll
 - Reichweitenbeschränkung soll Zugriff aus der Ferne verhindern
 - Kann durch Fehler in der Protokollimplementierung umgangen werden
 - Erlaubt Zurücksetzen aller Geräte in Reichweite auf Werkszustand
- Verlust der Einstellungen erscheint harmlos ...
 - Aber: ZLL-Geräte sind kompatibel zu normalen ZigBee-Netzen
 - Nach Zurücksetzen: Netzwerksuche (Beacon Request)
 - ... und selbstständiger Beitritt zu neuem PAN
 - PAN-Koordinator signalisiert Bereitschaft zur Aufnahme im Beacon
- ... ermöglicht aber die Übernahme des Leuchtmittels



... und größere Fehler

■ Verschlüsselte Hue-Firmware

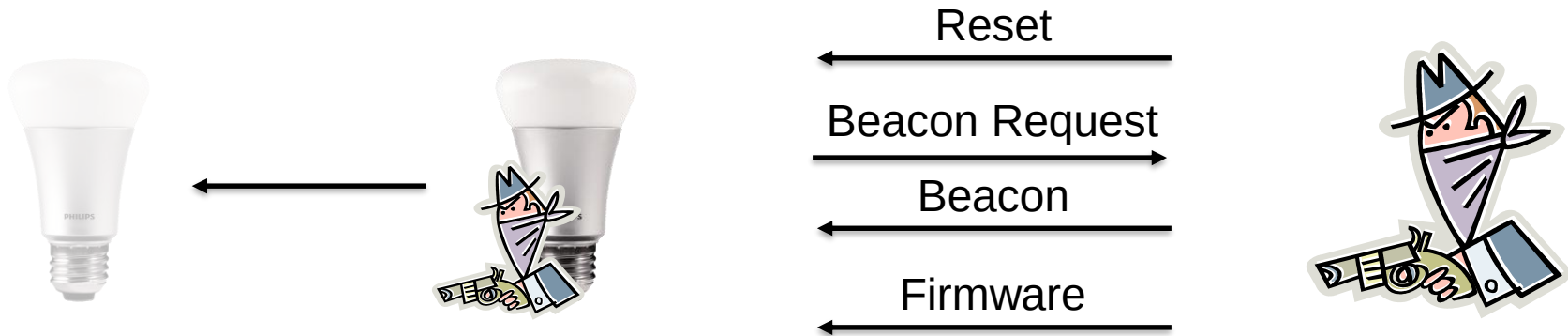
- Authentifizierung und Integritätsprüfung durch **symmetrischen Schlüssel**
- **Alle Firmware-Abbilder sind mit gleichem Schlüssel geschützt**
 - Extraktion über Seitenkanalangriff auf Hardware-AES-Implementierung



Angriff der IoT-Viren!

■ Ablauf

- Angreifer setzt Leuchtmittel über ZLL TouchLink zurück
- Netzwerksuche: Angreifer „lädt“ sein Opfer in eigenes PAN ein
 - Volle Kontrolle über das Leuchtmittel
- Angreifer startet Firmware-Update
 - Eigene Virus-Firmware ist mit dem Philips-Schlüssel signiert
 - Opfer akzeptiert Firmware und wird infiziert
 - Weiterverbreitung des Virus!



[RCSW16]

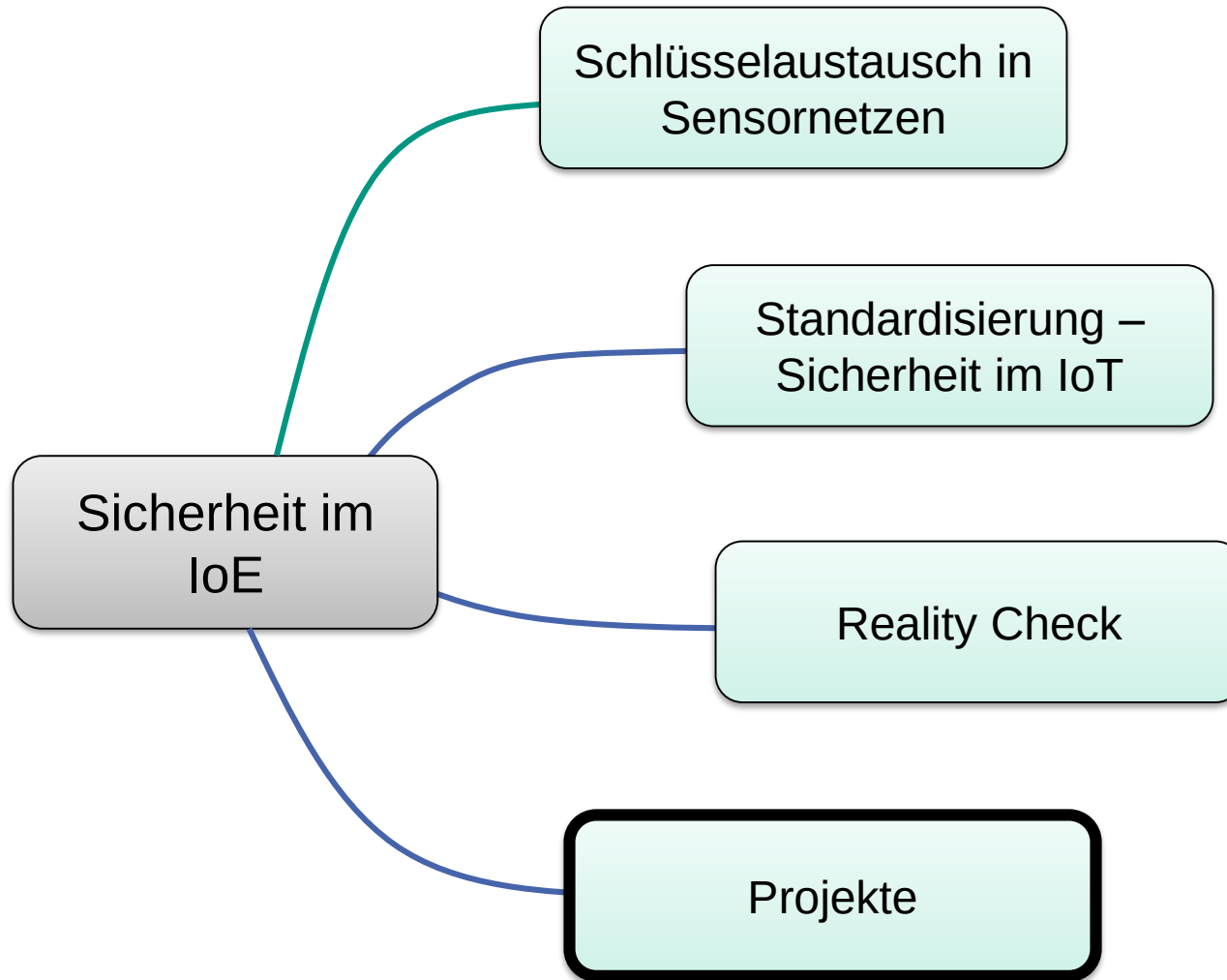


<https://www.youtube.com/watch?v=Ed1OjAuRARU>

Fazit: Philips Hue

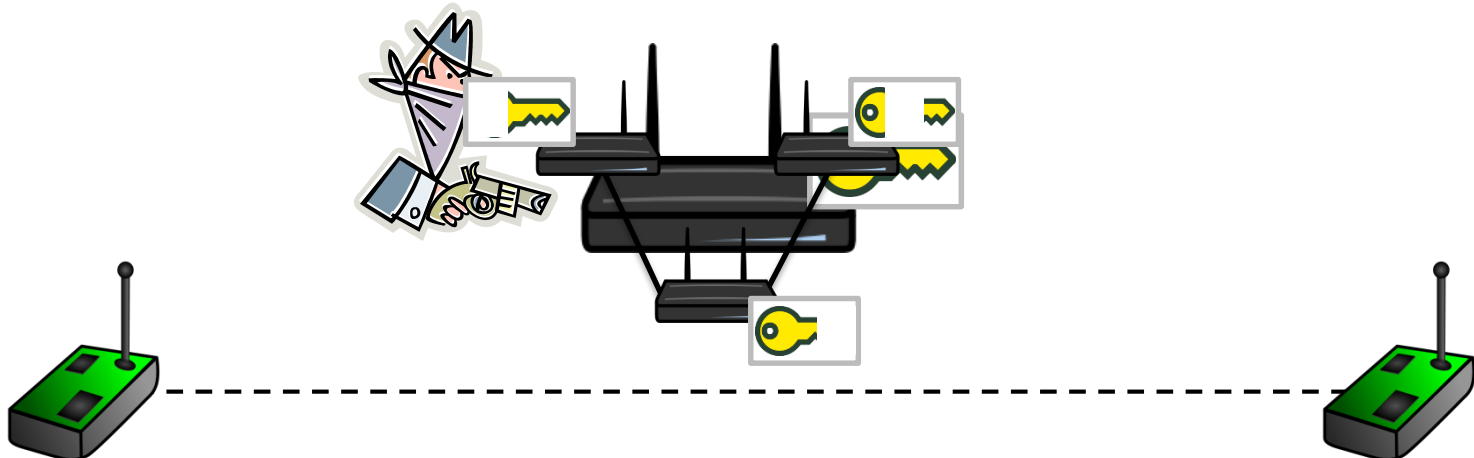
- Gemeinsamer symmetrischer Schlüssel für viele Systeme ist riskant
 - Extraktion durch Seitenkanäle und direkte Angriffe auf Hardware
- Sichere Gruppenkommunikation im IoE ist generelles Problem
 - Symmetrisch
 - Viele Schlüssel → Speicherbedarf, Verwaltungsaufwand
 - Single Mission Key → Es genügt ein Mitglied der Gruppe zu „brechen“
 - Asymmetrisch
 - Ressourcenbedarf (Speicher, Energie, Rechenzeit)
- Thema wird aktuell in der IETF Arbeitsgruppe ACE diskutiert
 - Austausch von Gruppenschlüsseln schwierig
 - Authentifikation von Absender mit symmetrischer Kryptografie schwierig
 - Interessenkonflikt: Sicherheit gegen Hardwarekosten

Schwerpunkte des Kapitels im Überblick



Security by Delegation

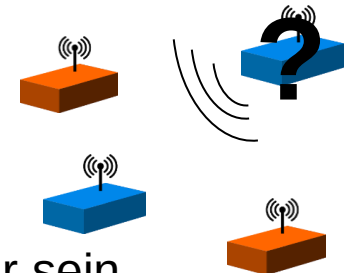
- Wie werden im Internet of Everything sichere Verbindungen aufgebaut?
 - Symmetrische Verschlüsselung benötigt (viele) pre-shared keys → Unflexibel
 - Asymmetrische Verschlüsselung ist (zu) teuer für IoE-Geräte
- Aufwendige Berechnungen auf vertrauenswürdigen Stellvertreter
- Problem: Stellvertreter ist single point of failure
 - Kennt alle Schlüssel im Netzwerk
 - Fehlfunktionen legen alle Geräte im Netzwerk still
- Ansatz: Stellvertreter auf mehrere Geräte aufteilen



Identitäten mit Blockchains

- Geräte brauchen eindeutige Identität
 - Z.B. für authentifizierte Kommunikation mit anderen Geräten
 - Zusätzlich zur Identität noch Attribute speichern, z.B. Gerätetyp
 - Außerhalb des IoE auch auf Personen anwendbar

- Zentralisiert kritisch wegen Privatsphäre und Einrichtungsaufwand
 - Ein dezentraler Ansatz ist vorzuziehen



- Widersprüchliche Ziele:
 - Vertrauenswürdige Identitäten sollen dezentral erzeugbar sein
 - Möglicherweise auch mehrere Identitäten pro Gerät
 - Attribute einer Identität sollen mit anderen Geräten geteilt werden können
 - Unbekannte Geräte sollen die Attribute nicht lesen dürfen

Ensure: New power grid structures for the energy transition

- Power grid faces new challenges
 - Classical grid: top-down distribution of electricity
 - Energy transition introduces many small decentralized producers and fluctuating resources
 - **Cyber attacks** are an increasing threat to the power grid
- Network is a central part of the power grid control system
- How can SDN technology be used to increase the power grid stability and security?
 - **SDN powered redundant data streams**
 - **Context dependent authorization of data streams**



Literatur



- [Blaß] Eric Blaß; **Sicherer, aggregierender Datentransport in drahtlosen Sensornetzen**. Dissertation Universität Karlsruhe, 2007
- [BMWCD] <http://www.bmw.de/de/topics/faszination-bmw/connecteddrive/services-apps/connecteddrive-services.html>, Abgerufen am 02.02.2016
- [EG02] L. Eschenauer, V. Gligor; **A Key Management Scheme for Distributed Sensor Networks**. Proceedings of ACM Computer and Communications Security, S. 41–47, Washington, USA, 2002, ISBN 1-58113-612-9
- [Gura04] N. Gura, et al.; **Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs**. Cryptographic Hardware and Embedded Systems (CHES'04). S. 119-132. Springer Berlin Heidelberg, 2004.
- [Heis15] <http://www.heise.de/newsticker/meldung/ConnectedDrive-Der-BMW-Hack-im-Detail-2540786.html>
- [Heis16] <http://www.heise.de/newsticker/meldung/BMW-steuert-via-IFTTT-das-Smart-Home-3085257.html>
- [Perrig] Adrian Perrig et al.; **Key Infection: Smart Trust for Smart Dust**. Proceedings of the IEEE International Conference on Network Protocols (ICNP), 2004.
- [RCSW16] E. Ronen, C. O'Flynn, A. Shamir A.-O. Weingarten: **IoT Goes Nuclear: Creating a ZigBee Chain Reaction**. 2016