IOT: CONNECTED VEHICLE

ARCHITECTURE, CHALLENGE, FUTURE AND OPPORTUNITIES

JIAEN.GUO@BOSCH.COM;
OLIVER.SCHEELE@BOSCH.COM;
About the speakers
1. IoT Architecture & Infrastructure
   1. General infrastructure & architecture
   2. System architecture

2. Challenge
   1. Security
   2. Reliability & performance
   3. Operation
   4. Customer and country oriented data privacy

3. Future & Chance
   1. Branch
   2. Example & demo
Startup inside Bosch for mobile solutions & connected car solutions

Author: jiaen.guo@bosch.com

Leader backend, SW architect.
Security design/Dev
Cloud based DevOps

Group leader & project manager
Specialist of firmware/hardware
Overall systems engineering

Co-Author: oliver.scheele@bosch.com
1.1 General infrastructure & architecture

Connected Devices
- Wifi module
- Bluetooth module
- V2X (V2V, V2I)
- GSM/LTE module
- GNSS

Gateway
- Wifi Router
- Mobile Tower
- LAN
- APN (Private/Public)

Cloud/private Server
- Data storage
- Data handling
- Data exchange

Mobile devices/Frontend
- Bluetooth
- MNO/Wifi
- Display & (direct) Control

Monitoring/Frontend
- Internal Access to cloud
- Monitoring & Control
1.2 System architecture

Connected Devices (Data generator)
- Impl. of conn (Wifi/BT/MNO)
- Impl. of microcontroller
- Impl. of memory usage
- Impl. of data handling (sensor data, GNSS etc.)
- W/O operation system, e.g. OSGI or direct Firmware programming

Cloud platform or private Server (Data handling)
- Firewall/DNS/Loadbalancer
- Management platform Incl. User Mgt, Application Mgt and Marketplace
- Infrastructure services in Cloud marketplace, e.g. Oracle, MongoDB, MQ, Email, SMS,
- Data storage & exchange in real time
- Tools for application Mgt (deployment and monitoring)

Gateway (Proxy)
- Data Exchange between device and the backend

Mobile devices (Data generator + Access point for Mobile User)
- Data exchange with Device(s) in real time (BT)
- Generation of additional data, e.g. GPS, user data etc.
- Data exchange with backend

Monitoring center (Access point for supervisor)
- Data exchange with backend
- Monitoring of device data and application data.
2.1 Security Risks

- Man in the middle
- Man in the middle
- Man in the middle
- DDoS
- DDoS
- DDoS
- Data storage
- Data handling
2.1 Security Requirements

Aspects:
- End customer
- Business operator
- Software (incl. Firmware) supplier

General Data security
- Confidentiality (user authentication and authorization)
- Data integrity (data correctness)
- System availability

Connected Devices
(Data generator)
- Avoid the data manipulation
- Protect of the micro-controller against the hacking of the firmware

Gateway
(Data delegation)
- Keep the communication channel safe
- Nobody can manipulate the data package on the way

Backend in the Cloud
(Data storage/handling)
- Receive only data from registered devices
- Make sure the received data is not manipulated
- No data lost even the server is overloaded
- Authorized data access

Monitoring center
(Supervisor)
- Authorized data access
- Isolated from Internet by fireware
- Only the meta data is visible

Frontend/Mobile devices
(Data storage/handling)
- Authorized access of personal data

General key management
- Keep the key safe (physical isolation, nobody can touch it in the runtime)
- Kept in encrypted form
- Dynamic Data/Communication encryption (Symmetric/Asymmetric)
- A key loss may only have local impact, but no global system impact.
2.1 Security strategy and general solutions

- VPN (security of communication channel)
- Whitelist of IP/Url
- Application of signature for secure data exchange
- OAuth2 (SSO)
- Domain/Firewall based micro service development

![Diagram of deployment area]

Deployment area
- Reachable via internet
- Reachable only via intranet
- Reachable only inside same depl. area

IP-Range
VPN/Whitelist
VPN/Signature

Microservice 1
Microservice 2

API
Web
API

OAuth

API

DB
KeyStore
IdMgt
MQ
Mail
SMS
2.2 Reliability & performance requirements

- No data lost
- No unexpected data transformation

- Reliable transfer of big data / continuous data streaming

- Acceptable data latency during the transfer of big data in the whole system

Time interval (some seconds!)
2.2 Reliability & performance strategy and general solution

A. Cluster Design/Redundant system/MQ

real users or devices

Req/Resp

WebServer/Firewall

API (Checksum, signature check)

AppServer

Caching Syn

Caching Syn

Data Storage

real users or devices
### 2.2 Reliability & performance strategy and general solution

#### B. Fault-tolerant programming + data error tracking

Example: Parse jsonData using jsonjackson-databind-xxx.jar

```java
public String getGPS(String dataFromIoTDevice) {
    ObjectMapper mapper = new ObjectMapper();
    try {
        JsonNode result = mapper.readTree(dataFromIoTDevice);
        return result.get("gps").toString();
    } catch (JsonProcessingException e) {
        throw new YourException(your own errorMsg, e);
    }
    return null;
}
```

```
String dataFromIoTDevice: {"Deviceld": "1234567", "utc": 1493212463,
"gps": {"lat": 1.2000, "lng": 2.0000, "alt": 0.0, "sat": 1}, "data_version": "1.2.3.4"}
```

```
String dataFromIoTDevice: {"Deviceld": "1234567", "utc": 1493212463,
"gps": {"lat": 1.2000, "lng": 2.0000, "alt": 2.0, "sat": 1}, "data_version": "1.3M001"}
```

```
Example: Parse jsonData using jsonjackson-databind-xxx.jar

String dataFromIoTDevice:

```
{"
DeviceId": "1234567",
"utc": 1493212463,
"gps": {
"lat": 1.2000, "lng": 2.0000, "alt": 0.0, "sat": 1,
"data_version": "1.2.3.4"
}
```

```
...catch (JsonProcessingException e) {
    try{
        mapper.configure(com.fasterxml.jackson.core.JsonParser.Feature.LIMIT_STRING_READINGS, true);
        JsonNode result = mapper.readTree(dataFromIoTDevice);
        for (JsonNode jsonNode : allJsonChildNodes) {
            if (!StringUtils.isAsciiPrintable(jsonNode.toString())) {
                DBLog.error(jsonNode.key, jsonNode.toString());
            }
        }
        return result.get("gps").toString();
    } catch (JsonProcessingException e) {
        DBLog.error(dataFromIoTDevice)
        throw new YourException(your own errorMsg, e);
    }
}
```

Allow all strange characters if the json structure is valid!

Don't forget to report and follow this behaviour!
### 2.2 Reliability & performance strategy and general solution

**C. Efficient encryption cryptography: Symmetric (ECC) vs Asymmetric (RSA)**

<table>
<thead>
<tr>
<th>Basic Principles</th>
<th>ECC (Elliptic Curve Cryptogr.)</th>
<th>RSA (Rivest-Shamir-Adleman)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate keys through properties of</td>
<td>Factor the product of two large primes</td>
<td></td>
</tr>
<tr>
<td>elliptic curve equation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Key/signature length for the same</td>
<td>Key: 160 bits</td>
<td>Key 1,024 bits</td>
</tr>
<tr>
<td>sec level of 80 bits (2)^80</td>
<td>Signature: 320 bits</td>
<td>Signature: 320 bits</td>
</tr>
<tr>
<td>Signature benchmark (openssl 1.0.2</td>
<td>ECDSA using key of 256 bits: 9516/s</td>
<td>RSA using 2048 bits</td>
</tr>
<tr>
<td>beta on x86_64): sign/s</td>
<td></td>
<td>key: 1001/s</td>
</tr>
<tr>
<td>En-/De-cryption</td>
<td>Device individual en-/de-cryption for req and resp.</td>
<td>Encryption of Device Resp only based on</td>
</tr>
<tr>
<td></td>
<td></td>
<td>server pk.</td>
</tr>
<tr>
<td>Break of encryption</td>
<td>Need device key and server key</td>
<td>Only need one key</td>
</tr>
</tbody>
</table>

- **More efficient!**
- **More powerful!**
- **More secure!**
2.3 Operation

Mode: DevOps

Targets:
- Keep the development reliable & safe
- Quick response (not only reaction!) for questions from customers

DevOps

Push change

Repository

Trigger build

jenkins

Build + Unit test

Build/test error

Test failed

Auto deployment

Integration test

Test done

Deployment

report in real time
2.4 Customer and country oriented data privacy

- Anonymous vs identified personal data
- Which data belong to whom?
- How long can customer/device data be kept?
- Where should which data be kept?
- Special data policy on the country level?

![Diagram showing customer permission, personal data of User 1, and solutions for User 1 and User 2 with text annotations for IdentityMgt (Ldap) and AccessMgt (OAuth) as a solution.]

Solution: Certificate based Data exchange
3 Future & Chance

Use Case examples of connected vehicle

- **Live Screens**
  - Crash Detection
  - Health Check

- **Self track (DTC)**
  - Pos, fuel level, Mileage, temperature
  - Mileage, temperature

- **As reference for quality mgt of OEM**
  - During the production/design

- **Big Data**
  - IOS/Android: Search the next workshop

- **Workshop**
  - Connected with emergency call center

- **Ambulance**

- **Leasing/Taxi company**

- **Reduce the risks of insurance company**

- **Car tracking**
  - Storage/Display of the data history
  - Trip Log

- **SOS via 3G/4G**

- **Drive Score**
3 Project Example: Vehicle Link 3100

- Robert Bosch project with focus on retrofit connectivity
- Data based on OBD II and Bosch sensors (acceleration, gyro, GNSS, thermometer etc.)
- Communication based on Bluetooth and Cellular network (GSM/UMTS)
- NACL as powerful encryption and authentication library
- Cloud based backend development (Bosch IoT cloud, JEE, Microservices)