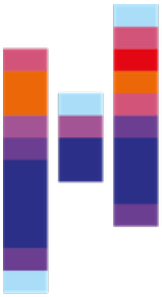

Hochschule
Flensburg
University of
Applied Sciences

**Wissenschaftliche Ausarbeitung**

# Sicherheitsrisiken und Schutzmechanismen in IoT-Anwendungen

22. Januar 2021

# Inhaltsverzeichnis

**1 Einleitung** **2**

**2 Anwendungsbereiche** **2**

**3 Angriffsmöglichkeiten** **4**

3.1 Herausforderungen bei mobilen Geräten . . . . . 4

3.2 Definition von Sicherheit . . . . . 5

3.3 Sicherheitsrisiken pro Domäne . . . . . 6

**4 Verteidigung gegen Angriffe** **6**

**5 Experiment** **10**

5.1 pi-aREST . . . . . 11

5.1.1 HTTP-Server . . . . . 11

5.1.2 MQTT-Client . . . . . 12

# 1 Einleitung

*Internet of Things (kz. IoT)* bezeichnet die Vernetzung von Geräten über das Internet. Diese werden oftmals, aufgrund der Verbindung zum Internet, auch als *„intelligente Geräte“ (engl. smart devices)* bezeichnet. Die Endgeräte können dabei sehr unterschiedlich sein, so reichen diese von von einfachen Messgeräten bis hin zu komplexen medizinischen Implantaten [5].

Aufgrund der Anbindung der Geräte an das Internet, ist die Kommunikationen mit diesen einfacher und Daten können leichter extrahiert und übermittelt werden. Diese Prozesse können zudem automatisiert werden, wodurch keine weitere menschliche Interaktion benötigt wird und Zeit bzw. Kosten eingespart werden können. Dadurch ist nicht verwunderlich, dass die Anzahl an IoT-Geräten in den letzten Jahren stark zugenommen hat. So sollen bis Ende 2020 über 25 Milliarden Geräte über das Internet vernetzt sein [5].

Die beeindruckende Anzahl an IoT-Geräten zeigt, dass die Technologie sich heutzutage etabliert hat, doch sollten hierbei nicht die Sicherheitsrisiken bei der Verwendung von IoT-Geräten außer Acht gelassen werden. Abhängig vom Anwendungsbereich der Geräte können diese für essentielle Funtkionalitäten eines System zuständig sein oder mit sehr sensiblen Daten arbeiten. Geräte, welche nicht ausreichend geschützt sind, können durch Angriffe bspw. heruntergefahren werden, was in Abhängigkeit zum Anwendungsbereich zu erheblichen Schäden führen kann [5].

Im Rahmen diese Ausarbeitung werden die Anwendungsbereiche, Sicherheitsrisiken und Schutzmechanismen von IoT-Geräten vorgestellt und anhand eines praktischen Beispiels demonstriert.

# 2 Anwendungsbereiche

Aufgrund der allgemeinen Digitalisierung vieler Abläufe und Branchen werden IoT-Geräte umfangreich eingesetzt, in dem folgenden Kapitel sollen einige Anwendungsbereiche der IoT vorgestellt werden.

**Umweltbeobachtung.** Bei der Umweltbeobachtung werden bestimmte Bereiche der Umwelt auf bestimmte ökologische Parameter überwacht. So werden bspw. die Luft-, Wasser- oder Bodenqualität von bestimmten Gebieten oder Standorten untersucht, um potentielle Umweltprobleme oder -gefahren frühzeitig feststellen zu können. Ein detailliertes Beispiel wäre die Erdbebenüberwachung, bei der IoT-Geräte mit seismischen Sensoren ausgestattet werden, um frühzeitig Erdbeben zu erkennen. Im Falle eines Erdbebens können die Geräte weitere System benachrichtigen, welche daraufhin Versorgungseinrichtungen oder Datenzentren abschalten, sowie Menschen in kritischen Wohngebieten benachrichtigen können [5].

**Gesundheitswesen.** Ärzte benötigen oftmals stetige Information zum körperlichen Zustand eines Patienten, vor allem um herauszufinden, ob eine bestimmte Behandlung zum gewünschten Erfolg führt. IoT-Geräte, im speziellen kleine medizinische Implantate, können Ärzte dabei unterstützen bestimmte Gesundheitswerte auszulesen bzw. selbständig auszuwerten. Ärzte können dadurch einfacher Untersuchungen und Diagnosen durchführen [5].

**Industrie.** In der Produktion von Gütern finden IoT-Geräte sehr unterschiedliche Verwendungszwecke. Mithilfe der IoT können bspw. gesamte Lieferketten optimiert werden, indem IoT-Geräte bestimmte Teilprozesse (Produktion, Transport, etc.) überwachen und in bestimmten Situation zielgerichtete Maßnahmen einleiten. Außerdem werden mithilfe von IoT-Geräten eine Vielzahl von Daten ausgehoben, welche in wachsenden Anwendungsbereichen wie Big Data Analysen oder Künstlicher Intelligenz benötigt werden [5].

**Wearables.** Bei Wearables handelt es sich um technische Geräte, welche direkt am Körper getragen werden und mit verschiedenen Sensoren ausgestattet sind. Die bekanntesten Beispiele für Wearabeles sind Smart-Watches und Fitnessarmbänder und werden für unterschiedliche Anwendungsbereiche wie bspw. Gesundheitswesen, Sport oder Entertainment eingesetzt [5].

**Spielzeuge.** IoT-Geräte in Form von Spielzeugen sind oftmals mit Lautsprecher, Mikrofonen und weiteren technischen Komponenten ausgestattet. Funktionalitäten, welche klassische Spielzeuge nicht haben, wäre bspw. eine integrierte Spracherkennung, bei dir die Aussagen der Kinder aufgenommen, über ein weiteres System ausgewertet werden und das Spielzeug dementsprechend reagiert. Aufgrund der Verbindung zum Internet stehen den Spielzeugen dadurch eine Vielzahl von neuen Möglichkeiten zur Verfügung [5].

# 3 Angriffsmöglichkeiten

Wie in fast jedem anderen System müssen vor allem IoT-Anwendungen besonders bezüglich ihrer Sicherheit betrachtet werden. Die Anwendungsentwicklung innerhalb des IoT ist geprägt von vielseitig komplexen Problemen beim Etablieren der IT-Sicherheit. Grundsätzlich können Sicherheitsrisiken in drei Bereichen auftreten, die hier kurz erläutert werden sollen [5].

## 3.1 Herausforderungen bei mobilen Geräten

**Heterogenität.** Dieser Bereich behandelt Probleme, die durch die Vielzahl an Anwendungsmöglichkeiten in dem IoT entstehen. Dazu zählen unter anderem die verbauten Hardware-Komponenten, Protokolle für die Kommunikation unter Geräten und Variationen zwischen Geräten und ihrer Rechenleistung [5].

**Ressourcenbeschränkung.** Nicht jedes Gerät beinhaltet einen leistungsstarken Rechner. Oft stößt man bei der Entwicklung von IoT-Geräten auf das Problem, dass nicht genug Speicher oder Rechenleistung vorhanden ist. Nicht nur bei der Implementation der eigentlichen Anwendung macht dies Probleme. Bereits etablierte Sicherheitsalgorithmen können aufgrund fehlender Leistung nicht verwendet werden, da sie sich als unpraktikabel in dem IoT herausstellen. Sicherheitsziele können also aufgrund nicht ausreichender Hardware nicht erreicht werden [5]. Ein Beispiel hierfür ist das Arbeiten

mit einer Cloud. Sicherheitsziele ist hier unter anderem die Privatsphäre des Benutzers. Die üblichen kryptographischen Algorithmen, z.B. die attributbasierte Verschlüsselungen und Pairings, stellen sich als sehr unpraktikabel heraus, da diese zu viel Rechenleistung für die Berechnungen von bilinearen Abbildungen benötigen. Eine vielversprechende Lösung ist das Auslagern rechenintensiver Schritte an einem leistungsstarken Server [1].

**Dynamische Netzwerktopologie.** Gerade bei mobilen Geräten innerhalb des IoT haben wir mit „losen“ Verbindungen zu tun. Zum Beispiel ist es möglich, dass ein Smartphone mit diversen WLAN-Hotspots interagiert und damit keine feste Position innerhalb eines Netzwerkes besitzt [5].

## 3.2 Definition von Sicherheit

Damit die Sicherheit trotz der oben erwähnten Herausforderungen gewährleistet werden kann, genügt es nicht, das allseits bekannte CIA-Dreieck zu betrachten. Laut [5] sind Vertraulichkeit, Integrität und Verfügbarkeit nicht die einzigen Voraussetzungen, um Sicherheit für IoT-Geräte zu definieren. Stattdessen müssen individuelle Anforderungen für jeden Anwendungsbereich definiert werden.

Komplexe Anlagen besitzen zum Beispiel verschiedenste Sensoren und Aktoren, um beispielsweise Verfahrenstechniken anzuwenden und diese zu überwachen. Diese Sensoren werden häufig beim Implementieren von IT-Sicherheit vernachlässigt, nachdem sie verbaut wurden. Mögliche Sicherheitsanforderungen für diesen Bereich sind Vertraulichkeit, Authentifizierung, Nachrichtenauthentizität, Integrität, Verfügbarkeit, Zuverlässigkeit, Frische der Daten und Fälschungsdetektierung [5].

Andere Bereiche des IoT benötigen jedoch andere Anforderungen an die IT-Sicherheit. Im Gesundheitswesen wird die Privatsphäre des Patienten eine größere Rolle spielen, als bei technischen Anlagen, sodass die Anforderungen abweichen. Diese sind hier unter anderem Verfügbarkeit, Zuverlässigkeit, Authentifizierung, Integrität, Vertraulichkeit, Privatsphäre, Nachrichtenauthen-

tizität, Fälschungsdetektierung, Verantwortlichkeit und Nichabstreitbarkeit [5].

### 3.3 Sicherheitsrisiken pro Domäne

Im vorherigen Abschnitt wurde auf verschiedene IoT-Domänen und ihren Sicherheitsanforderungen eingegangen. Kurz zusammengefasst ist jede Domäne unterschiedlich zu behandeln, wenn es um Sicherheit geht. Angreifer, die Schwachstellen in solchen Systemen finden, haben unter Umständen Zugriff auf eine große Menge von sensiblen Daten. Mögliche Angriffe können auf Grundlage der Sicherheitsanforderungen aus Abschnitt 3.2 abgeleitet werden [5] und werden in Tabelle 5.1.1 dargestellt.

## 4 Verteidigung gegen Angriffe

Um Angriffen, die in Tabelle 5.1.1 genannt wurden, entgegenzuwirken, sollen nun einige Maßnahmen für die Abwehr besprochen werden.

**Gegenmaßnahmen gegen unsichere Web-Interfaces und Netzwerkdienste.** Damit ein Fluten von Authentifizierungsanfragen verhindert werden kann, können Standardports geblockt werden. Für SSH ist dies beispielsweise der Port 22 und für Telnet der Port 23 bzw. 2323. Grundsätzlich sollten nur die Ports geöffnet werden, die wirklich benötigt werden. Auch das *Universial Plug and Play Protocol* (UPnP) sollte deaktiviert werden. Außerdem werden häufig Standardeinstellungen wie Benutzernamen und Passwörter unverändert übernommen. Dies ist offensichtlich ein großes Problem, denn ein Angreifer wird diese Information ohne großen Aufwand herausfinden können. Man sollte also darauf achten, individuelle und starke Passwörter zu verwenden. Auch die Software selbst sollte regelmäßig Updates eingespielt bekommen, denn häufig werden Schwachstellen erst nach einem Release festgestellt und Patches veröffentlicht [5].

| **Anwendungsdomäne** | **Mögliche Angriffe** |
|---|---|
| Umgebungsüberwachung | DoS/DDoS, MitM, Node capture, Sinkhole, Hello flood, Traffic analysis, Hacking, Side channel, Sybil, Selective forwarding, Back hole, Tampering, Wormhole, Masquerading |
| Gesundheitswesen | DoS/DDoS, MitM, Malicious code, Spoofing, Hacking, Tampering, Eavesdropping, Hijacking, Replay, Backdoof, Tag tracking, Tag cloning, Identity theft, Masquerading, Node capture, Side channel |
| Feuerwehr | DoS/DDoS, MitM, Sybil, Hello flood, Node capture, Sinkhole, Black hole, Selective forwarding, Hacking, Tampering, Malicious code, Hijacking, Side channel, Traffic analysis, GPS jamming |
| Herstellung | DoS/DDoS, MitM, Masquerading, Backdoor, Identity theft, Replay, Hijacking, Hacking, Eavesdropping, Selective forwarding, Back hole, Sinkhole, Node capture, Sybil, Spoofing, Traffic analysis, Side channel, Tampering, Wormhole, Malicious code, Wormhole, Economic espionage |
| Wearables | DoS/DDoS, Eavesdropping, MitM, Malicious code, Identity theft, Hacking, Backdoor, Hijacking, inappropriate network configuration |
| Spielzeuge | DoS/DDoS, Eavesdropping, MitM, Identity theft, Hijacking, Hacking, Backdoor, Masquerading, Spoofing, Malicious code, inappropriate network configuration |

Tabelle 1: Sicherheitsrisiken pro Domäne [5]

**Gegenmaßnahmen gegen Routing-Protokolle.** Da IoT-Geräte sehr eingeschränkter Natur sind, ist das Schützen von IoT-Netzwerken eine echte Herausforderung [4]. Sinkhole-Angriffe können laut [5] verhindert werden, indem robuste Authentifikation-Schemata, geografisches Routing und systematisches Rerouting verwendet werden. Geografisches Routing bedeutet, dass Datenpakete abhängig der Position von Knotenpunkten und der Zieladresse innerhalb des Netzwerks umgeleitet werden. Die Chancen, dass Forwarding- und Black-Hole-Angriffe durchführbar sind, können mithilfe von Source-Routing reduziert werden. Verwendet man Multipath-Routing, dann können Selective-Forwarding-Angriffe abgewehrt werden. Hierbei werden verschiedene Pfade innerhalb des Netzwerks verwendet. Hello-Flood-Angriffe können durch bidirektionale Authentifikation verhindert werden. Wormhole-Angriffe können ebenfalls durch geografisches Routing erschwert durchführbar gemacht werden. Aber auch durch physische Überwachung oder Source-Routing können Angriffe dieser Art verhindert werden. Gegenmaßnamen gegen Sybil-Angriffe ist die Evaluierung der Einzigartigkeit von Geräten im Netzwerk. Dies bedeutet lediglich, dass jedes Gerät eine eindeutige Identifikationsnummer besitzen muss. Auch mithilfe von sogenannten Random-Key-Pre-Distribution-Schemata [3] können Sybil-Angriffe verhindert werden.

**Gegenmaßnahmen gegen unerlaubten Informationstransport.** Bevor ein Smart-Device einen Datenaustausch beginnt, sollte sich dieses Gerät vorerst authentifizieren. Ebenfalls sollte eine beidseitige Authentifizierung implementiert werden, damit beide Parteien verifizieren können, ob es sich um den richtigen Endpunkt handelt und sie mit dem gewünschten Ziel kommunizieren. Die Idee hierbei ist, dass zum Beispiel digitale Signaturen (SHA, ECDSA) bzw. symmetrische äquivalente (HMAC) ausgetauscht werden. Dies dient vor allem für die Einhaltung der Integrität der Daten. Ebenfalls können Signaturen verwendet werden, um einen Secure-Boot zu implementieren. Dies hilft dabei, dass das IoT-Gerät nur Codes ausführt, die vertrauenswürdig, also signiert sind. Vor allem wird dadurch verhindert, dass zum Beispiel die Firmware des Geräts überspielt wird. Um die Angriffe abzuwehren, die die Vertraulichkeit betreffen, werden Informationen über einen sicheren, ver-

schlüsselten Kanal übertragen. Dies betrifft auch die Kommunikation mit externen Diensten, wie einer Cloud [5].

**Gegenmaßnahmen gegen physische Angriffe.** Um zum Beispiel Node-Capturing zu verhindern, müssen Geräte physisch versteckt werden, um den Zugang zu erschweren. Außerdem können die Geräte so gebaut werden, sodass diese nur sehr schwer auseinanderbaubar sind. USB-Ports sollten zudem auch geschützt werden, damit ein Angreifer keine Schadsoftware über diese einspielen kann. Schutzmaßnahmen gegen das Manipulieren von Daten können durch regelmäßige Änderung der Schlüssel eingeführt werden. Side-Channel-Angriffe können durch spezielle, resistente Chipsätze verhindert werden. Das Messen von elektromagnetische Strahlung muss beim Entwickeln von IoT-Geräten ebenfalls berücksichtigt werden. Das Verschleiern von Informationen auf diesem Wege muss also ebenfalls implementiert werden, damit Angreifer nicht mithilfe von elektrischen Signalen Zugriff bekommen [5].

**Gegenmaßnahmen gegen GPS-Jamming.** Um GPS-Jamming-Angriffe abzuwehren, können Kerbfilter eingesetzt werden [5, 2].

**Gegenmaßnahmen gegen Tag-Tracking und -Cloning.** Eine Schutzmaßnahme gegen das Tag-Tracking ist, dass die Tags gegenüber einem Angreifer zufällig aussehen und eine gleiche Verteilung besitzen, damit dieser keine Schlüsse aus dem Aufbau der Tags ziehen kann. Um Tag-Cloning zu verhindern, muss sichergestellt werden, dass die eigentlichen Informationen unter dem Tag nicht zugänglich sind, um einen neuen validen Tag zu erzeugen [5].

**Gegenmaßnahmen gegen falsche Netwerkkonfiguration.** Die Schutzmaßnahme ist hier sozialer Natur. Ein Schulen der Benutzer über die Wichtigkeit der IT-Sicherheit ist unabdingbar. Es müssen starke Regeln für Passwörter verwendet und sicherheitsrelevantes Logging aktiviert werden [5].

# 5 Experiment

In diesem Kapitel werden die Sicherheitsgefahren und -risiken von IoT-Geräten demonstriert. Dafür wird eine Raspberry Pi-Sicherheitskamera mithilfe der Open-Source-Bibliothek *pi-aREST* entwickelt. Die Sicherheitskamera sendet mithilfe von *pi-aREST* in einem festgelegten Zeitintervall ein Bild, des überwachten Bereichs, an ein externes System. Die Entscheidung *pi-aREST* als Bibliothek für die Entwicklung des IoT-Gerätes zu verwenden, hatte mehrere Gründe. Ein Grund war, dass die Bibliothek im speziellen für die Arbeit mit dem Raspberry Pi entwickelt wurde und eine direkte Implementierung für das Erstellen von Bildern mithilfe einer angeschlossenen Kamera enthält. Außerdem ist der Quellcode Open-Source, wodurch die Bibliothek einfacher untersucht und auf potentielle Sicherheitsrisiken überprüft werden kann. Der Entwickler der Bibliothek bietet zudem einen eigenen Übewachungs- und Kontroll-Service, für die mit der Bibliothek entwickelten IoT-Geräte, an. Wodurch die Untersuchung der möglichen Sicherheitsprobleme anhand von realisitischen Szenarien demonstriert werden kann.

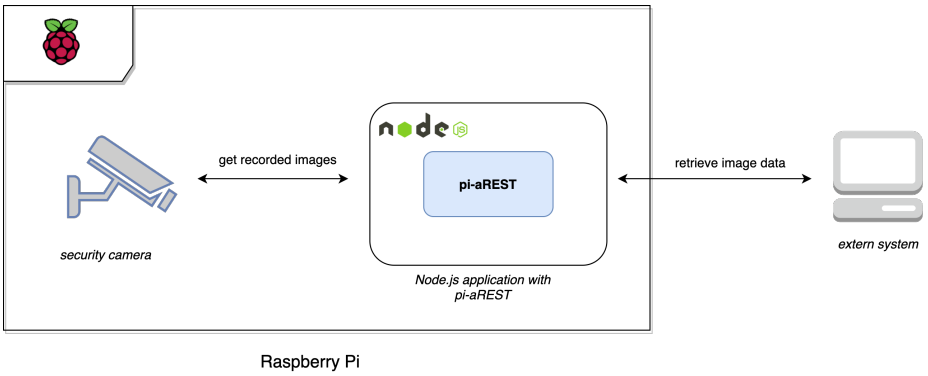


Abbildung 1: Sicherheitskamera-Architektur

## 5.1 pi-aREST

*pi-aREST* ist eine Bibliothek für die JavaScript-Laufzeitumgebung *Node.js*, welche mithilfe des Package-Managers *npm* in eine Node-Anwendung impor-

tiert werden kann. Durch das Einbinden von *pi-aREST* in eine eigene Node-Anwendung werden Schnittstellen zur Kommunikation mit anderen Systemem und Geräten geöffnet. Der Datenaustausch erfolgt dabei mithilfe der Protokolle *HTTP* und *MQTT* und kann sowohl in lokalen Netzwerken, als auch im Internet stattfinden.

### 5.1.1 HTTP-Server

Nachdem Ausführen der Node-Anwendung mit der importierten Bibliothek, startet *pi-aREST* einen *express* HTTP-Server auf dem IoT-Gerät zum bereitstellen einer REST-API. Die API besitzt verschiedene Endpunkte, welche entweder innerhalb des Netzwerkes oder aus dem Internet aufgerufen werden können.

| **Endpunkt** | **Aufgabe** |
|---|---|
| **/** | Liest die allgemeinen Raspberry Pi-Informationen (ID, Name) und weitere Metadaten, die in einem JavaScript Objekt festgehalten sind, aus und gibt diese zurück |
| **/:variable** | Anhand des übergebenen Parameters *variable* wird in einem Objekt, dass Metadaten zum Raspberry Pi enthält, überprüft, ob dieses ein Attribut mit diesem Namen enthält, wenn dies der Fall ist, wird es an den Anfragenden zurückgegebenen. Unter JavaScript kann das Attribut auch eine Funktion sein, falls dies der Fall ist, wird diese ausgeführt und das Ergebnis zurückgegebenen. |
| **/camera/snapshot** | Erstellt ein Bild durch die am Raspberry PI angeschlossene Kamera und hinterlegt diese im Dateisystem |
| **/:command/:pin** | Liest den Status des über den Parameter *pin* angegebenen digitalen Pins aus und sendet diesen Status zurück |
| **/digital/:pin/:state** | Ermöglicht das Setzten des Statuses eines Pins, indem über den Parameter *pin* der zu aktualisierende Pin ausgewählt wird und der Parameter *state* den neuen Status festlegt. |

Tabelle 2: Endpunkte des *pi-aREST* HTTP-Servers

### 5.1.2 MQTT-Client

Message Queuing Telemetry Transport (MQTT) ist ein Open-Source Kommunikationsprotokoll auf der Anwendungsebene. Das Protokoll zeichnet sich durch ressourcensparende Datenpakete mit einem geringen Overhead aus, wodurch es bspw. oftmals in der Kommunikation zwischen IoT-Geräten (M2M) eingesetzt wird. Das Protokoll basiert auf der Publish/Subscribe-Architektur, so nehmen Clients die Rollen eines *Publisher (Sender)* oder *Subscriber (Empfänger)* ein und kommunizieren über den sogenannten *Broker* miteinander.

**Publisher.** Dieser Client sendet Nachrichten an einen *Broker*, wobei die übersendeten Nachrichten eine spezielle *Topic (Bezeichner)* enthalten. Die übergebene *Topic* ist wichtig für das Weiterleiten der Nachrichten an andere Clients.

**Subscriber.** Im Gegenteil zum *Publisher* sendet der *Subscriber* keine Nachrichten an *Broker*, sondern abonniert *(subscriped* sich beim *Broker* für eine bestimtes *Topic*. Wenn ein *Publisher* eine Nachricht mit dem abonnierten *Topic* an den *Broker* sendet, erhalten alle *Subscriber* diese Nachricht.

**Broker.** Die Funktionalität des *Brokers* wurde bereits teilweise bei den anderen Protokoll-Teilnehmern beschrieben. Seine Hauptaufgabe ist das Steuern des Datenverkehrs zwischen den verschiedenen Clients. So nimmt dieser die Nachrichten der *Publisher* an und sendet diese nur an die, für das *Topic* abonnierte, *Subscriber*.

Beim Ausführen der Node-Anwendung mit *pi-aREST* verbindet sich das IoT-Gerät als Publisher-Client mit einem MQTT-Broker über den Port 1883. Die Broker-Adresse wird von der Node-Anwendung vorgegeben. *pi-aREST* sendet daraufhin die Verbindungsanfrage an den Broker. Das IoT-Gerät sendet nach dem erfolgreichen Verbindungsaufbau in einem regelmäßigen Zeitintervall die Daten, in im Falle des Experimentes die Überwachungsbilder, mit

einem speziellen Topic an den Broker.

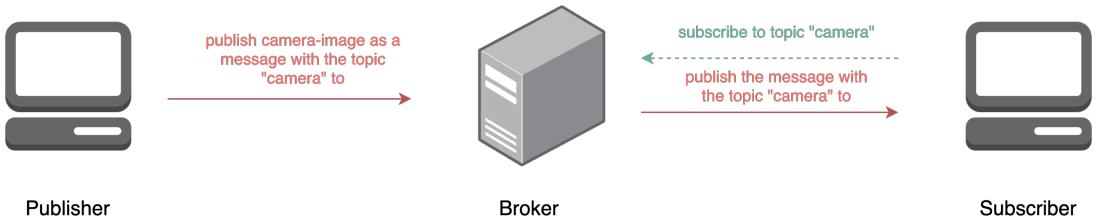


Abbildung 2: MQTT-Architektur für die Node-Anwendung

# Literatur


[1] S. Belguith, N. Kaaniche, M. Laurent, A. Jemai, and R. Attia. PHOABE: Securely outsourcing multi-authority attribute based encryption with policy hidden for cloud assisted IoT. *Computer Networks*, (133):141–156, 2018.

[2] D. Borio, C. O'Driscoll, and J. Fortuny. Gnss jammers: Effects and countermeasures. In *2012 6th ESA Workshop on Satellite Navigation Technologies (Navitec 2012) European Workshop on GNSS Signals and Signal Processing*, pages 1–7, 2012.

[3] Haowen Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *2003 Symposium on Security and Privacy, 2003.*, pages 197–213, 2003.

[4] B. D. Patel and A. D. Patel. A trust based solution for detection of network layer attacks in sensor networks. In *2016 International Conference on Micro-Electronics and Telecommunication Engineering (ICMETE)*, pages 121–126, 2016.

[5] M. G. Samaila, J. a. B. F. Sequeiros, M. M. Freire, and P. R. M. Inácio. Security threats and possible countermeasures in iot applications covering different industry domains. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, ARES 2018, New York, NY, USA, 2018. Association for Computing Machinery.