Absicherung des Internets der Dinge

Artemij Olegovic Voskobojnikov
Freie Universität Berlin
Fakultät für Mathematik und Informatik
Berlin, Germany
Email: voskobojnikov.artemij@gmail.com

Benjamin Swiers
Freie Universität Berlin
Fakultät für Mathematik und Informatik
Berlin, Germany
Email: swiers.benjamin@googlemail.com

***Zusammenfassung*—Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.**

I. Einleitung

Bennys Penis schmeckt nach Lakritz

Juli, 2015

II. Anwendung des IoT in der Industrie

In Unternehmen wird meist zwischen zwei Informations- und Kommunikationstechnikbereichen unterschieden. Zum einen existiert in Unternehmen die *Produktions-IT*, welche jegliche Controller von Produktionsanlagen oder Logistikanbindungen umfasst. Zum anderen gibt es ebenfalls den Bereich der *Business-IT*. Darunter fallen jegliche Anwendungen, die von den Angestellten verwendet werden und in keinem direkten Zusammenhang mit der Produktion stehen. Ein Beispiel dafür wären *Content-Management-Systeme*.

III. Industrie 4.0

Die *Industrie 4.0* vernetzt beide Bereiche miteinander und als Resultat entstehen sogenannte *cyberphysikalische Systeme* (*CPS*).
Bei einem *CPS* handelt es sich meist um ein Gerät mit beschränkten Ressourcen. Dies bedeutet, dass diese Systeme in der Rechenleistung und im Energieverbrauch beschränkt sind. Zusätzlich gibt es Anforderungen an solche Systeme, die unbedingt erfüllt werden müssen. So müssen *CPS* ständig verfügbar und ausfallsicher sein, sodass es im schlimmsten Fall nicht zum Produktionsstillstand kommen kann.

A. Anforderungen an solche Systeme

Aufgrund der Vernetzung beider *IKT-Bereiche*, werden Anforderungen des jeweiligen Subsystems übernommen. Dies bedeutet, dass Sicherheitslösungen im Office-Bereich nicht ohne weiteres angewendet werden können, da in einem *CPS* auch Auswirkungen auf den anderen IKT-Bereich (in diesem Beispiel die *Produktions-IT*) abgeschätzt werden müssen. Aus dieser Problematik resultieren somit neue Probleme und Schwachstellen, die möglicherweise bei den anfänglich separierten Systemen nicht existierten.
Im Detail bedeutet das, dass sich insbesondere die Sicherheitsanforderungen unterscheiden und bekannte Sicherheitslösungen der *Business-IT* wie VPN oder SSL/TLS-Verschlüsselung nicht auf die *Produktions-IT* übertragen werden können.
Der Hauptgrund dafür ist die Tatsache, dass Komponenten der *Produktions-IT* zertifiziert sind und eine Verwendung von Verschlüsselungen einen Eingriff bedeuten würde, welcher im schlimmsten Fall zu Verlust der Zertifizierungen führen würde. Ebenfalls können Verschlüsselungen zu möglichen Latenzen

führen, die im Office-Bereich verkraftbar sind, in einer Produktion aber eine Nichtfunktionalität bedeuten würden.
Ein weiterer Aspekt, der in der *Business-IT* nicht bedacht werden musste, sind physikalische Angriffe, die in der Produktion denkbar sind. So können Mitarbeiter Komponenten verändern oder gänzlich entfernen [1].

In den folgenden Abschnitten werden mögliche Probleme der *Industrie 4.0* aufgezeigt und es werden Lösungsansätze demonstriert, die aktuell im Einsatz sind.

B. Cloud-Computing

1) Unterarten von Clouds: Cloud-Computing ist nicht gleich Cloud-Computing. Es gibt verschiedene Services, die auf verschiedene Anwendergruppen zugeschnitten sind, wie die nachfolgende Grafik demonstriert.

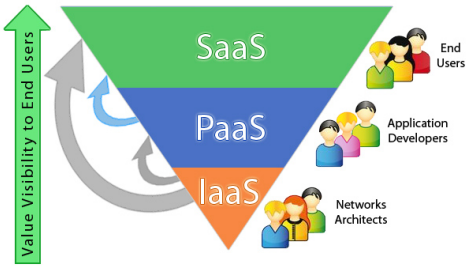

Abbildung 1: Unterarten von Clouds

Die umgekehrte Pyramide verdeutlicht dabei den unterschiedlichen Umfang des jeweiligen Dienstes. Dabei wird zwischen drei verschiedenen Services unterschieden. Bei *Software-as-a-Service* (SaaS) werden vollständige Systeme den Nutzern zur Verfügung gestellt. Der Zugriff auf diese erfolgt mittels Browser.
Product-as-a-Service (PaaS) ist in der Regel auf Anwendungsentwickler bzw. fortgeschrittene Nutzer zugeschnitten, welche möglicherweise eigene Applikationen auf den Systemen verwenden wollen. Die letzte Unterart *Infrastructure-as-a-Service* (IaaS) bietet dem Anwender lediglich ein Grundgerüst [2]. Es können personalisierte Dienste oder Betriebssysteme installiert werden. Große Unternehmen wählen häufig IaaS als Cloud-Lösung, weil diese eine komplette Plattform bieten und ein Drittanbieter für die Wartung dieser zuständig ist. Darüber hinaus ist der Drittanbieter im Falle eines Angriffs auch verantwortlich.

C. Industrie und die Cloud

Aufgrund der Verwendung von beschränkten Ressourcen wird oftmals auf *cloudbasierte Lösungen* industriellen im industriellen Bereich zurückgegriffen, wie die Abbildung 2 aufzeigt. Diese Verlagerung bedeutet, dass Rechenoperationen sowie die gesamte Datenspeicherung nicht auf dem limitierten Gerät selbst durchgeführt werden muss, es stattdessen vielmehr mit der Cloud in ständiger Kommunikation steht.
Trotz der verbreiteten Verwendung, sind Cloudlösungen nicht durchgehend akzeptiert. Dies ist der Statistik in Abbildung 3

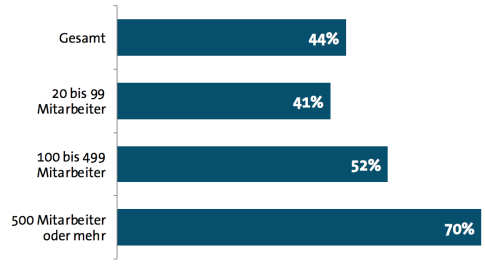

Abbildung 2: Clouds in der Industrie

zu entnehmen. Dabei handelt es sich um eine Statistik, die bei der *Pressekonferenz Cloud Monitor 2015* präsentiert wurde [3]. Die größten Bedenken beziehen sich auf die Sicherheit der Cloud, insbesondere auf Schutz der sensiblen (personenbezogenen) Daten. Aus diesem Grund wird im Folgenden auf die Absicherung dieser eingegangen.

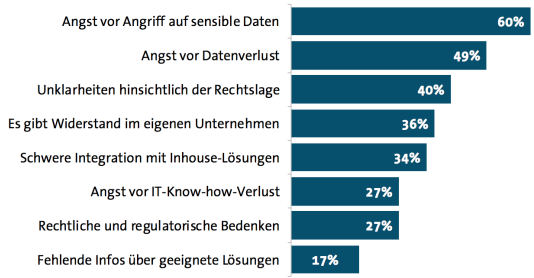

Abbildung 3: Sicherheitsbedenken der Nutzer

D. Trusted Cloud

Bei der *Trusted Cloud* handelt es sich um ein Programm des Bundesministeriums für Wirtschaft und Energie. Das Ziel dieses Projektes war es die Entwicklung einer sicheren Cloud-Infrastruktur, die böswilligen Administratoren den Zugriff auf sensible Daten verwehren soll. [4]
Das Grundkonzept sind speziell gesicherte Serverschränke, die bei unberechtigten Eingriffen sich herunterfahren. Zusätzlich werden jegliche Daten verschlüsselt auf den Festplatten gespeichert. Bei der Verarbeitung der Daten wird darüber hinaus zusätzlich nur flüchtiger Speicher verwendet, sodass nach Herunterfahren des Systems auf keinerlei Daten zugegriffen werden kann [1].

E. Zugriffskontrolle bei Cloud-Diensten

Aufgrund der Vielzahl von verschiedenen Nutzern bei einem Cloud-Dienst muss ebenfalls der Zugriff auf die Datensätze abgesichert werden. Dies könnte mittels einer *rollenbasierten Zugriffskontrolle* geschehen, doch besteht hierbei der Nachteil, dass ein möglicher Fehler in der Implementierung zur Aufhebung der Rollen führen würde. Dies ist nur möglich, weil solche Systeme im Regelfall vollständig in Software umgesetzt werden.

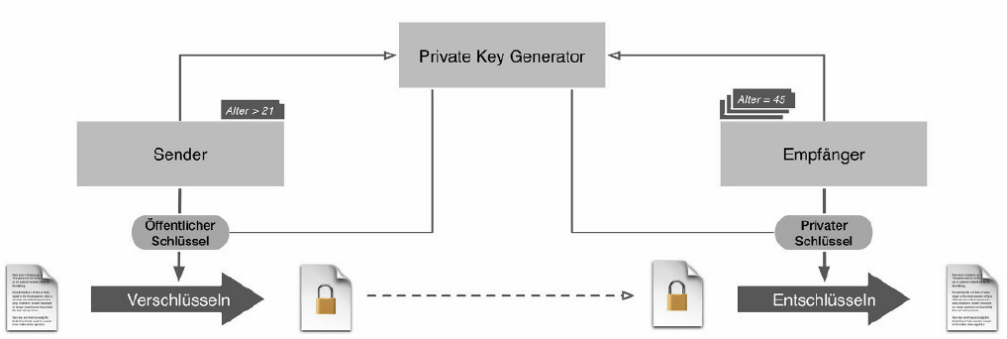

Abbildung 4: Attributbasiertes Verschlüsselungssystem

Eine bessere Möglichkeit bietet ein *attributbasiertes Verschlüsselungsystem* (s. Abbildung 4).

Zugrunde liegt hier ein *asymmetrisches Verschlüsselungsverfahren*. Die *Public Keys* sowie *Private Keys* werden dabei vom *Private Key Generator* ausgestellt.
In der Praxis würde also nur eine Person zugreifen können, die das nötige Attribut besitzt [1].
Bei mehreren Anwendern könnte man eine gruppenbasierte Zugriffskontrolle einführen. Dafür kann beispielsweise der Verzeichnisdienst *Active Directory* (AD) verwendet werden. Mit Hilfe von AD können nun Gruppen erstellt werden und digitale Zertifikate gruppenweit verteilt werden. Eine Zugriffskontrolle für mehrere Nutzer wäre damit gewährleistet.

F. Suchen auf verschlüsselten Daten

Aufgrund der Tatsache, dass die Daten verschlüsselt auf den Datenträgern vorliegen, müssen spezielle Algorithmen verwendet werden, die eine Suche auf verschlüsselten Datensätzen ermöglichen. Dies ist notwendig, da so die Daten zu jedem Zeitpunkt verschlüsselt auf dem Server vorliegen und nur clientseitig entschlüsselt werden. Böswillige Administratoren haben so im besten Fall keine Angriffsmöglichkeit (serverseitig).

G. Secure Indexes

In dieser Ausarbeitung beschränken wir uns auf *Symmetric Searchable Encryption* (SSE), also auf Verschlüsselungsverfahren, die auf einem symmetrischen Schlüssel/Schema basieren. Ein Index ist dabei eine Datenstruktur, die bei Eingabe eines Teilwortes die Pointer zu den Dokumenten wiedergibt, in denen das Teilwort enthalten ist [5]. Zusätzlich kann man einen Index als sicher ansehen, wenn die Suche nach einem Teilwort w nur mit Hilfe einer *Trapdoor* durchgeführt werden kann und ein Index alleine keinerlei Auskunft über den Inhalt gibt. Die Generierung einer sogenannten *Trapdoor* erfolgt dabei mittels des privaten Schlüssels des Nutzers. Im Folgenden wird eine beispielhafte *SSE* nach Goh [6] erläutert:
Zu Beginn muss der Begriff der *Bloom-Filter* eingeführt werden, da diese existenziell für die Erstellung der sicheren Indizes sind. Bei einem *Bloom-Filter* handelt es sich um eine Datenstruktur, welche eine Menge $S = \{s_1, ..., s_n\}$ mit n Elementen repräsentiert. Solch ein Filter besteht aus einem m-stelligen Bitarray, in welchem jede Stelle anfangs auf 0 gesetzt wird. Anschließend werden r unabhängige Hashfunktionen $h_1, ..., h_r$ gewählt, wobei $h_i : \{0,1\}^* \rightarrow [1, m]$ für $i \in [1, r]$ gilt. Zusätzlich werden für jedes Element $s \in S$ im m-stelligen Array die Positionen $h_1(s), ..., h_r(s)$ auf 1 gesetzt.
Die für die weitere Betrachtung relevante Operation ist nun das Prüfen, ob ein Element a in der Menge S ist. Hierfür werden die Positionen $h_1(a), ..., h_r(a)$ geprüft und falls all diese Positionen eine 1 sind, gehört a zur Menge S [7].
In dem folgenden Abschnitt wird die Konstruktion von *sicheren Indizes* nach [6] erklärt.

H. Konstruktion nach Goh [6]

- `Keygen(s)`: Ein Sicherheitsparameter s wird gegeben. Eine pseudozufällige Funktion $f : \{0,1\}^n \times \{0,1\}^s \rightarrow \{0,1\}^s$ wird gewählt sowie ein privater Schlüssel $K_{priv} = (k_1, ..., k_r) \leftarrow \{0,1\}^{sr}$
- `Trapdoor(`K_{priv}`, w)`: Eingabe für die *Trapdoor* sind der private Schlüssel K_{priv} und das Wort w, die Ausgabe für die beiden Parameter ist $T_w = (f(w, k_1), ..., f(w, k_r)) \in \{0,1\}^{sr}$
- `BuildIndex(`D`,` K_{priv}`)`: Eingabe für die Funktion `BuildIndex` ist das Dokument D mit einer einzigartigen ID $D_{id} \in \{0,1\}^n$ und einer Liste von Worten $(w_0, ..., w_t) \in \{0,1\}^{nt}$ sowie dem privaten Schlüssel $K_{priv} = (k_1, ..., k_n) \in \{0,1\}^{sr}$ Für jedes einzigartige Wort w_i, wobei $i \in [0, t]$ werden folgende drei Berechnungen angestellt
  1) Berechnung der *Trapdoor*: $(x_1 = f(w_i, k_i), ..., x_r = f(w_i, k_r)) \in \{0,1\}^{sr}$
  2) Berechnung eines eindeutigen Codewortes für jedes w_i: $(y_1 = f(D_{id}, x1), ..., y_r = f(D_{id}, xr)) \in \{0,1\}^{sr}$
  3) Abschließend wird das Codewort $y_1, ..., y_r$ in den *Bloom-Filter* des Dokumentes mit der D_{id} eingefügt

  Der Output ist der Index $I_{D_{id}} = (D_{id}, BF)$ für das Dokument mit der ID D_{id}
- `SearchIndex(`T_w`,` I_D`)`: Der Input ist die *Trapdoor* $T_w = (x_1, ..., x_r) \in \{0,1\}^{sr}$ für das jeweilige Wort w und ein Index I_D mit $I_{Did} = (D_{id}, BF)$ für das

jeweilige Dokument mit der ID D_{id}. Es müssen die folgenden Berechnungen durchgeführt werden:

1) Berechne das Codewort für das Wort w in D_{id} : $(y_1 = f(D_{id}, x_1), ..., y_r = f(D_{id}, xr)) \in \{0,1\}^{s,r}$
2) Wenn der *Bloom-Filter* nur Einsen an allen r Stellen mit $y_1, ..., y_r$ enthält
3) Falls dies der Fall ist, kam das Wort vor, gebe 1 aus. Ansonsten 0

Das obige Verfahren soll lediglich als Beispiel dienen. Es sind weitere Möglichkeiten denkbar. So können auch *asymmetrische Verfahren* angewendet werden ([8]).

I. Schutz von limitierten Geräten

Limitierte Geräte müssen ebenfalls geschützt werden. Dabei unterscheidet man zum einen zwischen der Kommunikation zwischen den jeweiligen Gerätschaften und dem eigentlichen Geräteschutz. Im Folgenden konzentrieren wir uns auf den internen Schutz limitierter Geräte.

J. Sichere Komponenten

Bei der Entwicklung limitierter Geräte werden häufig sichere Bauteile verwendet, welche beispielsweise dafür sorgen, dass die interne Schlüsselspeicherung sicher ist und die Schlüssel auch nicht ohne weiteres gelesen werden können. Die Abbildung 5 veranschaulicht einen beispielhaften Aufbau eines limitierten Gerätes.

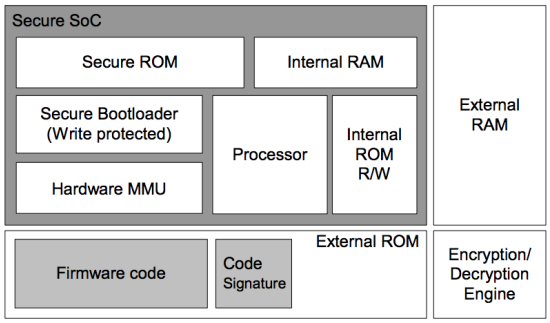

Abbildung 5: Diagramm eines System-on-a-Chip

1) Secure SoC: Das *Secure SoC* bietet Schutz für die geräteinternen Schlüssel. Diese werden verschlüsselt im *Secure Read-only-Memore* (Secure ROM) abgespeichert. Bei den Schlüsseln handelt es sich oftmals um Schlüssel, welche für digitale Signaturverfahren wie RSA, DSA oder auch ECDSA verwendet werden. Diese Schlüssel werden wie bereits gesagt verschlüsselt im ROM abgelegt. Dabei wird häufig auf Verschlüsselungsverfahren wie AES zurückgegriffen.[9]
Ein SoC bietet im besten Fall folgende Eigenschaften:

- Auf Secure ROM kann nicht physikalisch zugegriffen werden
- Die Buse innerhalb des Systems können nicht abgehört werden
- Das Ersetzen von Komponenten soll zu einer Nichtfunktionalität führen

Der letzte Punkt ist dabei besonders interessant, da das Gerät dadurch mögliche Angriffe verhindern kann. Der *Secure Bootloader* sorgt dabei, dass das Gerät nur mit einer korrekten Firmware hochfährt. Zusätzlich können *Physical Unclonable Functions* verwendet werden, deren Funktionsweise im Folgenden erläutert wird.

K. Physical Unclonable Functions

Keine zwei Stromkreise sind identisch. Diese Idee dient als Grundlage bei der Erstellung von *Physical Unclonable Functions* (PUFs). Durch Anwendung von PUFs sind Bauteile oder gesamte Geräte eindeutig identifizierbar.
In besten Fall sind PUFs eine Hardwareanalogon von Hashfunktionen. So gibt es im besten Fall sehr wenige Kollisionen und die Funktion selbst soll einfach durchführbar sein.
Formalisiert bedeutet das, dass es eine Challge c gibt und eine Funktion f, die diese Challenge als Eingabe nimmt. Die Ausgabe wäre dann $r = f(c)$. Dabei ist zu erwähnen, dass zwei Bedingungen gelten sollen:

- Für jede Challenge hat ein (integrierter) Stromkreis die gleiche Response
- Für jede Challenge haben unterschiedliche (integrierte) Schaltkreise unterschiedliche Responses

Die folgende Abbildung demonstriert eine beispielhafte PUF.

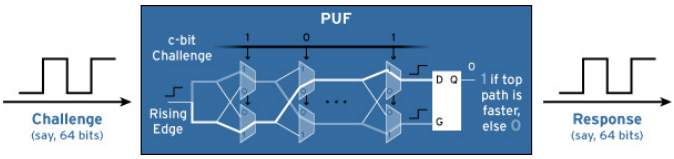

Abbildung 6: Multiplexer-PUF

Die PUF kann wie folgt formalisiert werden: $MUX - PUF : \{0,1\}^{64} \rightarrow \{0,1\}^{64}$.
Sie erwartet also eine 64bit-Folge als Eingabe und produziert eine dementsprechende 64-bit-Folge als Ausgabe.
Diese PUF besitzt zwei verschiedene Pfade und je nachdem auf welchem Pfad das Signal schneller war, wird eine 1 oder eine 0 ausgegeben. Die Anzahl der Challenge-Response-Paare ist bei dem obigen beispiel 2^{64}[10].

Dieses Verfahren ermöglicht eine Identifikation von Komponenten (Schaltkreisen). Als Folge daraus können limitierte Geräte in einem internen Speicher die Challenge-Response-Paare für die Komponenten speichern und sich selbstständig im Betrieb überprüfen. Falls nämlich für eine Komponente kein Challenge-Response-Paar, also keine eindeutige Identifikation vorliegt, kann sich das Gerät abschalten oder gar nicht erst hochfahren (*Secure Boot*)[11].
Darüber hinaus können PUFs auch zur sicheren Kommunikation zwischen Geräten verwendet werden. So können bestimmte Kommunikationspartner nur nach erfolgreicher Identifikation kommunizieren und werden andernfalls abgewiesen [1], [12].

IV. Fazit

In dieser Ausarbeitung wurden zwei Bereiche des *Internet of Things* untersucht. Zu Beginn wurde auf die Sicherheitsmechanismen im Heimbereich eingegangen, es wurden mögliche Schwachstellen und Verbesserungen genannt. Im zweiten Teil wurde deutlich, dass die Kombination zweier IKT-Bereiche neue Herausforderungen mit sich bringt, die separat bei beiden System zuvor in dieser Form nicht existierten.

Literatur

[1] C. Eckert, N. Fallenbeck, "Industrie 4.0 meets IT-Sicherheit: eine Herausforderung!" Heidelberg Berlin 2015.

[2] "Security Services für Industrie 4.0," June 2015. [Online]. Available: http://www.channelpartner.de/a/security-services-fuer-industrie-4-0,3044992

[3] Cloud-monitor. [Online]. Available: https://www.bitkom.org/Presse/Anh%C3%A4nge-an-PIs/2015/M%C3%A4rz/Cloud-Monitor.pdf

[4] Wind River Systems, "Security in the Internet of Things," 2015.

[5] R. Curtmola, J. Garay, S. Kamara, R. Ostrovsky, "Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions," *Proceedings of the 13th ACM conference on Computer and communications security, CCS '06, ACM*, 2006.

[6] E.-J. Goh, "Secure Indexes," *IACR Cryptology ePrint Archive 2003*, 2003.

[7] Broder A., Mitzenmacher M., "Network Applications of Bloom Filters: A Survey," 2002.

[8] Boneh D., Di Crescenzo G., Ostrovsky R., Persiano G., "Public Key Encryption with keyword Search," 2004.

[9] A. MS, "Security needs in embedded systems," *IACR Cryptology ePrint Archive*, 2008.

[10] "Physical Unclonable Functions." [Online]. Available: http://studiopresence.com/client/verayo/technology

[11] G. E. Suh, S. Devadas, "Physical Unclonable Functions for Device Authentication and Secret Key Generation," *Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE*, 2007.

[12] C. Eckert, "IT-Sicherheit und Industrie 4.0," *Fachzeitschrift für Innovation, Organisation und Management*, 2014.

[13] S. Devadas, "Physical unclonable functions and applications."