Trusted Execution Environments für Mikrocontroller im IoT

Lars Pfau

Hochschule für Angewandte Wissenschaften Hamburg
lars.pfau@haw-hamburg.de

Zusammenfassung. Diese Ausarbeitung beschreibt Trusted Execution Environments und ihren Nutzen im Internet der Dinge (IoT). Dabei wird eine Analyse der aktuellen Bedrohungen im IoT durchgeführt. Darauf aufbauend werden Trusted Execution Environments als Lösung der zuvor beschriebenen Bedrohungen vorgestellt. Weiterhin werden die Herausforderungen beschrieben, die dem Einsatz von Trusted Execution Environments im IoT-Kontext entgegenstehen. Abschließend werden Implementierungen von Trusted Execution Environments für die RISC-V Befehlssatzarchitektur vorgestellt.

Schlüsselwörter: Trusted Execution Environments · Internet of Things · IoT Security

1 Einleitung

1.1 Internet of Things

Das Internet of Things (IoT) (dt. Internet der Dinge) beschreibt einen Technologietrend, bei dem Objekte aus der physischen Welt mit Sensoren, Aktoren und Netzwerkverbindungen ausgestattet und mit dem Internet verbunden werden [13]. Anwendungsbereiche, in denen Objekte (Things) vernetzt werden, sind Haushaltsgeräte, Gebäudeautomatisierung, industrielle Produktion, Medizin und viele weitere. Die Vernetzung ermöglicht Anwendungen von einfachen Wireless Sensor Networks bis hin zu komplexen Cyber-physischen Systemen.

Der weltweite Markt für IoT-Geräte wächst. Die Anzahl der IoT-Geräte wird von IoT Analytics auf 27 Milliarden im Jahr 2025 geschätzt [14]. McKinsey prognostiziert für das Jahr 2025 eine Wertschöpfung durch das Internet der Dinge zwischen 2.800 und 3.600 Mrd. USD [9].

1.2 IoT-Nodes

Als Hardware kommen im IoT häufig eingebettete Systeme mit Low-end Mikrocontrollern und drahtloser Netzwerkanbindung zum Einsatz. Der Energieverbrauch und die Stückkosten dieser IoT-Nodes sind wichtige Faktoren bei der Auswahl der Hardware [28].

Nach RFC7228 [8] enthalten solche Constrained IoT-Nodes nur wenige Kilobyte Arbeitsspeicher und einige Kilobyte Flash-Speicher. Entsprechend eingeschränkt ist der Software-Stack, der auf den Plattformen läuft. Auf besonders eingeschränkten Platformen kann nur Bare-metal Firmware eingesetzt werden. Auf leistungsfähigeren IoT-Nodes kommen auch Echtzeitbetriebsysteme (RTOS), wie zum Beispiel RIOT [5], zum Einsatz [28].

1.3 Gliederung der Ausarbeitung

Das Ziel dieser Ausarbeitung ist es zu zeigen, wie die Sicherheit von IoT-Nodes durch den Einsatz von Trusted Execution Environments verbessert werden kann. Dazu wird in Abschnitt 2 beschrieben, welche Besonderheiten die Sicherheit von IoT-Nodes beinhaltet und welche Bedrohungen durch Trusted Execution Environments adressiert werden. Abschnitt 3 erläutert, was Trusted Execution Environments sind, wie sie funktionieren und Abschnitt 4 beschreibt ihre Grenzen. Abschließend werden in Abschnitt 5 Implementierungen von Trusted Execution Environments für die RISC-V Befehlssatzarchitektur vorgestellt.

2 IoT Security

Tabelle 1. Registrierte Malware-Angriffe auf das IoT von 2018 bis 2022 [37]

Jahr	**Angriffe [#]**
2018	32.700.982
2019	34.296.891
2020	56.949.058
2021	60.139.968
2022	112.294.990

Die Sicherheit von IoT-Geräten ist 2016 in den Fokus der Sicherheitsforschung gerückt, nachdem das Mirai-Botnet [2] einen signifikanten Teil des Internets beeinträchtigt hatte. Durch die Übernahme einer hohen Anzahl von IoT-Nodes waren Angreifer in der Lage, einen erfolgreichen DDOS-Angriff auf den DNS-Dienstleister Dyn durchzuführen, der kritische Infrastruktur für die Namensauflösung im Internet bereitstellt. Dieser Vorfall hat gezeigt, dass neben Angriffen auf das IoT auch Angriffe mithilfe des IoT eine signifikante Bedrohung darstellen.

Seitdem hat sich die Bedrohungslage im Internet of Things weiter verschärft. Tabelle 1 ist zu entnehmen, dass die Anzahl der Angriffe auf das Internet der Dinge zunimmt. Demnach verzeichnete SonicWall im Jahr 2022 112 Millionen Angriffe mit IoT-Malware, 87 % mehr als im Vorjahr [37].

2.1 Besonderheiten von IoT-Nodes

Eine Besonderheit von IoT-Nodes ist ihre Ressourcenbeschränktheit. Diese macht es schwer starke Ende-zu-Ende-Verschlüsselung auf den IoT-Nodes zu implementieren. Außerdem macht sie die IoT-Nodes anfällig für Denial-of-Service-Angriffe [11].

Darüber hinaus sind IoT-Nodes, wie alle Softwaresysteme, anfällig für Schwachstellen in Software und Protokollen. Im Unterschied zu klassischen eingebetteten Systemen sind sie jedoch mit dem Internet verbunden, was erschwerend die Ausnutzung von Sicherheitslücken aus der Ferne ermöglicht [20].

Eine weitere Besonderheit stellt die physische Sicherheit der IoT-Nodes dar. IoT-Nodes sind unbeaufsichtigt und oft räumlich stark verteilt. Dadurch wird es Angreifern leicht gemacht, Zugang zu den Nodes zu erhalten [1]. Anders als bei einem Rechenzentrum kann der Zugang zu einem IoT-Node nicht immer kontrolliert werden.

2.2 Bedrohungen im IoT

Die Bedrohungen im Zusammenhang mit dem Internet der Dinge sind vielfältig und wurden in verschiedenen Studien aus unterschiedlichen Blickwinkeln untersucht [15,25,30,36]. Es ist wichtig, dass die Bedrohungen nicht nur aus der Sicht eines einzelnen IoT-Nodes, sondern im Kontext der gesamten IoT-Anwendung analysiert werden und auch die Bedrohungen für das Internet berücksichtigt werden. Die folgenden Bedrohungen und die sich daraus ergebenden Anforderungen sollen an dieser Stelle hervorgehoben werden:

Node Capturing Als Node Capturing wird ein Angriff bezeichnet, bei dem ein Angreifer die Kontrolle über einen IoT-Node übernimmt oder dessen Identität stiehlt [25]. Für den Angreifer kann dies der Ausgangspunkt sein, um weiter in das System vorzudringen und die gesamte IoT Anwendung zu kompromittieren [15].

Daraus ergibt sich die Anforderung, die Integrität und Authentizität der IoT-Nodes gegenüber der IoT-Anwendung sicherzustellen. Die Authentifizierung bildet gleichzeitig die Grundlage für eine Zugriffskontrolle, bei der ein IoT-Node nur Zugriff auf die Funktionen erhält, für die er autorisiert wurde [6]. Eine weitere Anforderung ist die Absicherung des Schlüsselmaterials. Insbesondere der Plattformschlüssel muss für einen Angreifer unzugänglich sein, selbst wenn dieser physischen Zugriff auf den IoT-Node hat [26].

Malware Physischer Zugriff sowie Schwachstellen in der Firmware eines IoT-Nodes können zur Änderung von Konfigurations- und Anwendungsdaten bis hin zur Ausführung beliebigen Codes durch den Angreifer führen. Malware kann sich in der Firmware eines Nodes einnisten und sich selbstständig auf weitere Nodes ausbreiten [30]. Eine große Anzahl identischer IoT-Nodes verstärkt diese Bedrohung [33].

Die Malware muss nicht zwangsläufig die Funktion der IoT-Nodes beeinträchtigen. Ein weiterer Anwendungsfall für IoT-Malware sind IoT-Botnetze, wobei IoT-Nodes von Angreifern genutzt werden, um DDoS-Angriffe gegen Ziele im Internet durchzuführen [2,36]. Aus diesen Bedrohungen resultiert die Notwendigkeit, die Integrität und Authentizität von Code und Daten der IoT-Nodes sicherzustellen.

Information Disclosure Malware, Schwachstellen in der Firmware und der physische Zugang können es Angreifern ermöglichen, sensible Informationen aus IoT-Nodes auszulesen [4]. Auch die Firmware eines IoT-Nodes kann als sensible Information betrachtet werden, da durch Reverse Engineering der Firmware Sicherheitslücken aufgedeckt werden können, die einen Angriff auf weitere Nodes ermöglichen [32]. Darüber hinaus können sich die Daten, die von allgegenwärtigen IoT-Nodes gesammelt werden, auch auf die Privatsphäre von Einzelpersonen beziehen. Insbesondere personenbezogene Daten unterliegen dem Schutz durch Verordnungen und Gesetze [11]. Neben der Integrität und Authentizität muss daher auch die Vertraulichkeit von Code und Daten sichergestellt werden.

Safety Eine weitere Form der Bedrohung ist gegeben, wenn Geräte, die mit dem Internet verbunden sind, kritische Funktionen übernehmen. Ein Angreifer könnte ein solches System in einen unsicheren Zustand versetzen oder die Verfügbarkeit des Geräts einschränken. Auf diese Weise könnte ein Schaden in der physischen Welt entstehen. Gerade bei medizinischen Geräten ist die Aufrechterhaltung kritischer Funktionen von entscheidender Bedeutung, da sonst im schlimmsten Fall Menschen zu Schaden kommen können [18]. In diesem Zusammenhang ist auch der Schutz vor Entladungsangriffen für die Batterie von Bedeutung [41].

In solchen Anwendungsbereichen besteht die Anforderung, kritische Dienste und Sicherheitsfunktionen auch dann aufrecht zu erhalten, wenn ein IoT-Node angegriffen wird oder bereits kompromittiert ist.

3 Trusted Execution Environments

Eine Technologie, um vielen dieser Bedrohungen zu begegnen sind Trusted Execution Environments. Ein Trusted Execution Environment (TEE) ist eine manipulationssichere Ausführungsumgebung, die die Integrität, Authentizität und Vertraulichkeit von Code und Daten innerhalb der Umgebung sicherstellt [35]. Die Sicherheit dieser Ausführungsumgebung wird auch dann gewährleistet, wenn die Anwendung in der normalen Ausführungsumgebung kompromittiert ist.

Die Idee von Trusted Execution Environments ist es, die Ausführungsumgebung durch einen Isolationsmechanismus in der Hardware in zwei logische Umgebungen aufzuteilen: Ein Secure Execution Environment und ein General-Purpose Execution Environment. Der Isolationsmechanismus hat dabei die Aufgabe, den Zugriff des General-Purpose Execution Environments auf geschützte Ressourcen einzuschränken [34]. Durch Mechanismen wie Secure Boot und Remote Attestation kann Vertrauen in das Secure Execution Environment aufgebaut werden.

3.1 Isolationsmechanismus

Es können verschiedene Isolationsmechanismen unterschieden werden, je nachdem auf welcher Privilegierungsebene (Ring) die Isolation zwischen den Ausführungsumgebungen erfolgt. Es gibt keine einheitliche Bezeichnung für die Isolationsmechanismen. Tabelle 2 zeigt verschiedene Bezeichnungen und Beschreibungen, die in der Literatur zu finden sind. Im Folgenden werden die Isolationsmechanismen daher nach dem Ring bezeichnet, an dem die Isolation stattfindet.

Tabelle 2. Beschreibungen von Isolationmechanismen für TEEs

Ring	**Ning et al.** [27]	**Pinto et al.** [31]	**Sabt et al.** [34]
3	Memory encryption	User space enclaves	—
0	—	OS partition	—
-1	—	Trusted hypervisor	Bare-metal hypervisor
-2	Memory restriction	Hardware technology	Special processor extensions
-3	Co-processors	Independend co-processor	—

Ring -3 Isolation Bei der Isolation an Ring -3 befindet sich das Secure Execution Environment auf einem Coprozessor. Der Coprozessor ist physikalisch vom General-Purpose Execution Environment getrennt und verfügt über einen eigenen Adressraum [34]. Der Coprozessor kann sich auf einem externen Chip befinden oder in den Hauptprozessor integriert sein [27]. Entscheidend ist, dass nur der Coprozessor Zugriff auf die geschützten Ressourcen hat.

Damit das General-Purpose Execution Environment auf die geschützten Ressourcen zugreifen kann, muss es über eine Hardware-Schnittstelle mit dem Coprozessor kommunizieren [34]. Coprozessoren spielen in IoT-Nodes aufgrund der Kosten für die zusätzlich benötigte Hardware nur eine untergeordnete Rolle.

Ring -1 Isolation Bei der Isolation an Ring -1 werden das General-Purpose Execution Environment und das Secure Execution Environment von einem Typ-1-Hypervisor in getrennten virtuellen Maschinen ausgeführt [34]. Dadurch kann der Zugriff des General-Purpose Execution Environment auf den physikalischen Adressraum eingeschränkt werden. Insbesondere wird der Zugriff auf den Speicher des Secure Execution Environments verhindert. Diese Form der Isolation kann auf allen Plattformen eingesetzt werden, die CPU-basierte Virtualisierung unterstützen.

Ring -2 Isolation Bei der Isolation an Ring -2 wird die Trennung des Secure Execution Environment von dem General-Purpose Execution Environment direkt in der CPU-Hardware implementiert [34]. Der Prozessor befindet sich zu jedem Zeitpunkt entweder im Secure-Zustand oder im Normal-Zustand. Im

Normal-Zustand ist der Zugriff der Hardware auf geschützte Ressourcen eingeschränkt [31].

3.2 Secure Monitor

Eine wichtige Komponente in Systemen, bei denen die Ausführungsumgebungen auf dem gleichen CPU-Kern ausgeführt werden, ist der Secure Monitor. Der Secure Monitor ist die Softwarekomponente, die im niedrigsten Privilegierunglevel der CPU ausgeführt wird. Die Aufgabe des Secure Monitor ist es, die Isolation zwischen dem General-Purpose Execution Environment und den Ressourcen des Secure Execution Environments sicherzustellen. Er ist somit für die korrekte Konfiguration des Isolationsmechanismus verantwortlich und führt die Kontextswitches zwischen General-Purpose und Secure Execution Environment durch. Darüber hinaus stellt der Secure Monitor IPC-Mechanismen zwischen dem General-Purpose- und dem Secure Execution Environment zur Verfügung [35].

3.3 Secure Boot

Der Isolationsmechanismus und der Secure Monitor bilden die Grundlage für Trusted Execution Environments. Die Trennung zwischen einem Secure Execution Environment und einem General-Purpose Execution Environment ist jedoch nicht ausreichend um die Integrität und Authentizität des Codes innerhalb des Secure Execution Environments zu gewährleisten. Zhao et al. [42] weisen in ihrem Paper beispielsweise auf die Gefahr hin, dass eine Anwendungen im General-Purpose Execution Environment den Code des Secure Execution Environments überschreiben kann, wenn diese Zugriff auf den Flash-Speicher hat.

Es ist daher zunächst notwendig, Vertrauen in die Integrität des Secure Execution Environments aufzubauen. Dies wird durch einen Prozess namens Secure Boot erreicht. Die Idee von Secure Boot ist es, ausgehend von einem vertrauenswürdigen Bootloader, eine Chain-of-Trust vom ersten CPU-Befehl bis in die Anwendung hinein aufzubauen [21]. Secure Boot garantiert somit die Authentizität der Secure Software.

Die Root-of-Trust für Secure Boot ist ein unveränderlicher Boot-ROM, in dem der erste Bootloader enthalten ist. Der Boot-ROM kann außerdem das Schlüsselmaterial enthalten, das zur Überprüfung der Integrität und Authentizität weiteren Codes benötigt wird [38].

3.4 Secure Services

Die bisher beschriebene Technologie ermöglicht soweit die Ausführung von vertrauenswürdigem Code in einer isolierten Ausführungsumgebung. Die Anwendungen, die in dieser Umgebung ausgeführt werden, werden auch als Secure Services bezeichnet. In [10,17,29] werden verschiedene Implementierungen von Trusted Execution Environments beschrieben, die zum Teil unterschiedliche Secure

Services unterstützen. Die umfangreichste dieser Implementierungen ist Arm Trusted Firmware-M (TF-M). Abbildung 1 zeigt die Architektur eines Trusted Execution Environments mit den Secure Services am Beispiel von TF-M.

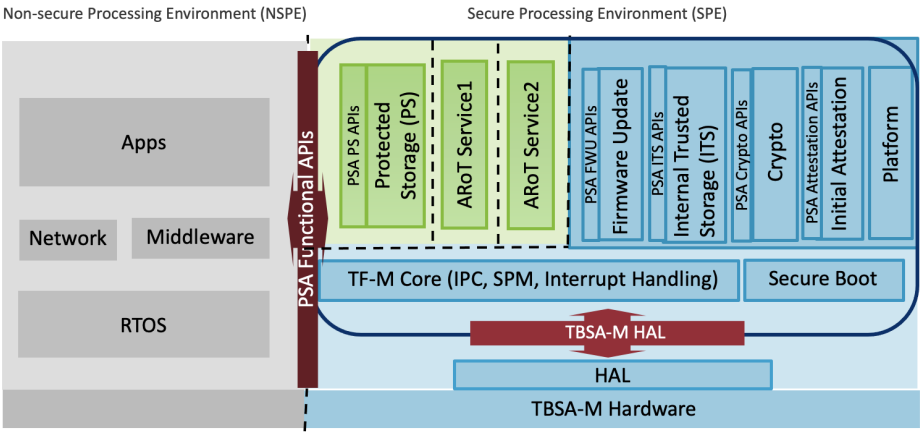

Abb. 1. Architektur von Arm Trusted Firmware-M (Auszug aus [17])

Secure Storage Ein Secure Service, der von vielen Trusted Execution Environments zur Verfügung gestellt wird, ist ein Secure Storage Service. Es existieren Anwendungen, die persistenten Speicher für die Speicherung von Anwendungsdaten verwenden. Zu solchen Anwendungsdaten gehören zum Beispiel Logdaten oder Konfigurationsdaten, welche auch Schlüsselmaterial enthalten können. Befinden sich solche Daten auf einem externen Flash-Baustein, können sie von einem Angreifer ausgelesen und manipuliert werden. Darüber hinaus könnte ein Angreifer den Inhalt des Speichers mit einer älteren Version überschreiben.

Die Aufgabe des Secure Storage Service ist es, die Vertraulichkeit, Integrität, Authentizität und Aktualität der Daten zu garantieren [16]. Die Daten müssen also nicht nur verschlüsselt und authentifiziert oder signiert werden, sondern es ist zusätzlich ein Replay-Schutz erforderlich. So wird sichergestellt, dass eine Anwendung nur die Daten aus dem Speicher ausliest, die sie auch zuletzt in den Speicher hinein geschrieben hat.

Remote Attestation Der Remote Attestation Service ist ein weiterer Secure Service, der von Trusted Execution Environments bereitgestellt wird. Die Integrität eines IoT-Nodes selbst ist zwar von großer Bedeutung. Aus Sicht des Gesamtsystems, in das ein IoT-Node eingebunden ist, muss der IoT-Node jedoch auch in der Lage sein, seine Integrität und Authentizität gegenüber Dritten (z.B. einem Cloud-Dienst) zu beweisen, da ansonsten, wie in Abschnitt 2.2 beschrieben, die Gefahr des Node Capturing besteht.

Die Grundlage für Remote Attestation ist ein Attestation-Protokoll zwischen einem IoT-Node und einem Drittsystem. Das Drittsystem sendet eine Challenge an den IoT-Node, in der Beweise für die Integrität und Authentizität der Plattform angefordert werden. Diese Beweise können beispielsweise den Inhalt des Speichers oder den Zustand des Prozessors beinhalten [19]. Der IoT-Node signiert die angeforderten Beweise mit einem Attestierungs-Schlüssel. Dieser Schlüssel ist die Root-of-Trust für die Remote Attestation. Er ist nur innerhalb des Trusted Execution Environments zugänglich und identifiziert den IoT-Node eindeutig. Der IoT-Node antwortet auf die Challenge mit den signierten Beweisen [21].

Kryptographie Ein weiterer Service, den Trusted Execution Environments unterstützen können, ist ein Softwarestack für kryptographische Operationen. Funktionen wie Secure Boot, Secure Storage und Remote Attestation benötigen kryptographische Funktionen, die innerhalb des Trusted Execution Environments bereitgestellt werden müssen. Ein Keystore zur Speicherung von Schlüsselmaterial und ein Zufallszahlengenerator können ebenfalls vom Crypto-Service bereitgestellt werden [3].

Secure Enclaves Implementierungen von Trusted Execution Environments bieten darüber hinaus die Möglichkeit, beliebige Software in einer isolierten Umgebung, in Form von sogenannten Enclaves, auszuführen. Anwendungsentwickler können selbst entscheiden, welchen Code sie getrennt von der normalen Ausführungsumgebung ausführen möchten. Allerdings kann es erforderlich sein, den Code anzupassen, um die Anwendungssoftware in einer Enclave ausführen zu können [10].

4 Herausforderungen

4.1 Implementierung auf Low-end Hardware

Die Bereitstellung von Trusted Execution Environments auf Constrained IoT-Nodes stellt eine besondere Herausforderung dar. Sabt et al. [34] nennen niedrige Kosten und ein geringer Overhead als Anforderungen an die Isolationsmechanismen. Darüber hinaus weisen Pinto et al. [31] darauf hin, dass Echtzeitverhalten und geringe Latenzen in der Interruptbehandlung eine wichtige Anforderung an Mikrocontroller-basierte Systeme sind. Diese Anforderungen müssen daher auch bei der Entwicklung von Trusted Execution Environments für IoT-Nodes eingehalten werden.

4.2 Sicherheit

Die Sicherheit von Trusted Execution Environments stellt eine weitere Herausforderung dar. Der Secure Monitor und die Secure Services eines Trusted Exucution Environments sind Softwarekomponenten, welche Sicherheitslücken

enthalten können. Diese Sicherheitslücken haben es in der Vergangenheit bereits ermöglicht, Speicherinhalte oder Schlüsselmaterial aus Trusted Execution Environments auszulesen [27] oder unerwünschten Code im Secure Execution Environment auszuführen [31]. Eine Lösung für diese Probleme ist die formale Verifikation aller Sicherheitskritischen Softwarekomponenten, welche in der Gesamtheit als Trusted Computing Base (TCB) bezeichnet werden [23,35].

Die formale Verifikation der Trusted Computing Base schützt jedoch nicht vor Seitenkanalangriffen. So können beispielsweise Timing-Attacken auf Caches zum Auslesen von Informationen genutzt werden [31]. Hat der Angreifer physischen Zugriff auf einen IoT-Node, sind auch Angriffe durch die Analyse der Stromaufnahme oder Fault Injection denkbar [20]. Ein physischer Angreifer kann außerdem Zugriff auf den Speicherbus des Systems erlangen [43]. Daraus folgt, dass je nach Art der Bedrohung zusätzlich der Einsatz von manipulationssicherer Hardware erforderlich sein kann [24].

5 Implementierungen für RISC-V

RISC-V ist eine neue und quelloffene Befehlssatzarchitektur (ISA), die in 32-bit Mikrocontrollern eingesetzt werden kann [39]. Die RISC-V Spezifikation enthält seit 2017 einen Mechanismus namens Physical Memory Protection (PMP) [40], der die Erstellung von Trusted Execution Environments auf Mikrocontrollern ohne virtuellen Speicher ermöglicht. Basierend auf PMP existieren verschiedene Implementierungen von Trusted Execution Environments für RISC-V Mikroprozessoren [23].

5.1 Keystone

Eine dieser Implementierungen ist Keystone [23]. Keystone ist ein Forschungsprojekt an der UC Berkeley, dass ein quelloffenes Framework für Trusted Execution Environments auf RISC-V Prozessoren entwickelt. Es unterstützt die Erzeugung von Enclaves sowie Secure Boot und Remote Attestation. Keystone unterstützt 32-bit RISC-V Plattformen, die einen Supervisor-mode und virtuellen Speicher implementieren. Mikrocontroller werden von Keystone derzeit noch nicht unterstützt. Das Projekt arbeitet jedoch an einer Portierung für Mikrocontroller unter FreeRTOS [22].

5.2 MultiZone Security

Eine weitere Implementierung von Trusted Execution Environments für RISC-V ist MultiZone Security [12] von dem Unternehmen Hex Five Security. MultiZone unterstützt die Aufteilung der Anwendungssoftware in mehrere Enclaves (Zones). Außerdem unterstützt MultiZone Secure Boot. Anders als Keystone ist MultiZone auch auf Low-end Mikrocontrollern ausführbar [12]. Es unterstützt jedoch selbst keine Secure Services. Desweiteren können Enclaves nicht dynamisch zur Laufzeit erstellt werden.

6 Zusammenfassung

Gegenstand dieser Ausarbeitung waren Trusted Execution Environments im Anwendungsbereich Internet der Dinge. Zunächst wurde anhand der Bedrohungen im IoT motiviert, warum der Einsatz von Trusted Execution Environments auch in ressourcenbeschränkten IoT-Nodes sinnvoll ist. Es wurde gezeigt, dass Isolationsmechanismen die Aufteilung der Anwendung in eine sichere und eine unsichere Ausführungsumgebung ermöglichen und dass diese Aufteilung genutzt werden kann, um Anwendungsfälle wie Secure Storage, Remote Attestation, kryptografische Operationen und weitere zu realisieren. Die Ressourcenbeschränktheit von IoT-Nodes stellt jedoch eine Herausforderung bei der Implementierung von Trusted Execution Environments im IoT dar. Außerdem bieten Trusted Execution Environments keine absolute Sicherheit, da weiterhin Bedrohungen durch Schwachstellen und physische Angreifer bestehen.

Zusammenfassend lässt sich jedoch festhalten, dass Trusted Execution Environments trotz der verbleibenden Risiken eine wichtige Schutzmaßnahme für IoT-Nodes sind, insbesondere gegen Softwareangreifer, die angesichts der steigenden Bedrohungslage im IoT flächendeckend eingesetzt werden sollte.

7 Ausblick

Ein Ziel der Masterprojekte soll die Entwicklung von Trusted Execution Environments für RISC-V Mikrocontroller unter RIOT sein. Mögliche Anwendungsfälle für TEEs unter RIOT sind Remote Attestation und Keystores mit der neuen PSA Crypto API [7].

In diesem Zusammenhang muss zunächst untersucht werden, welche RISC-V Mikrocontroller derzeit am Markt verfügbar sind und welche Teile der RISC-V Spezifikation diese implementieren. Mit diesem Wissen können im nächsten Schritt Anforderungen an das Trusted Execution Environment und Schnittstellen zwischen TEE und RIOT definiert werden. Auf Basis dieser Anforderungen kann dann nach existierenden Softwarelösungen gesucht und eigene Software entwickelt werden.

Relevante Fragestellungen, die beantwortet werden können, sind der Performance Overhead von Trusted Execution Environments für RISC-V Mikrocontroller sowie mögliche Weiterentwicklungen der RISC-V Spezifikation zur besseren Unterstützung von TEEs auf Low-end Mikrocontrollern.

Die Herausforderungen werden voraussichtlich darin bestehen, den Ressourcenverbrauch sowohl in Bezug auf die Performance als auch den Speicherverbrauch so gering wie möglich zu halten. Außerdem muss die Isolation zwischen dem Trusted- und dem General-Purpose Execution Environment getestet oder anderweitig verifiziert werden.

Literatur

1. Abomhara, M., Køien, G.M.: Cyber Security and the Internet of Things: Vulnerabilities, Threats, Intruders and Attacks. Journal of Cyber Security and Mobility **4**(1), 65–88 (2015). https://doi.org/10.13052/jcsm2245-1439.414
2. Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J.A., Invernizzi, L., Kallitsis, M., Kumar, D., Lever, C., Ma, Z., Mason, J., Menscher, D., Seaman, C., Sullivan, N., Thomas, K., Zhou, Y.: Understanding the Mirai Botnet. In: 26th USENIX Security Symposium (USENIX Security 17). pp. 1093–1110. USENIX Association, Vancouver, BC (Aug 2017), https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis
3. Arm Limited: PSA Cryptography API (2022), https://developer.arm.com/documentation/ihi0086/latest/, (Zugriff: 2023-09-01)
4. Atamli, A.W., Martin, A.: Threat-Based Security Analysis for the Internet of Things. In: 2014 International Workshop on Secure Internet of Things. IEEE (Sep 2014). https://doi.org/10.1109/siot.2014.10
5. Baccelli, E., Hahm, O., Gunes, M., Wahlisch, M., Schmidt, T.: RIOT OS: Towards an OS for the Internet of Things. In: 2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). IEEE (Apr 2013). https://doi.org/10.1109/infcomw.2013.6970748
6. Betts, D., Shahan, R., Lamos, B., Petersen, T., Meadows, P., Sherer, T., Pettibone, G., et al.: Security architecture for IoT solutions (2018), https://learn.microsoft.com/en-us/azure/iot-fundamentals/iot-security-architecture, (Zugriff: 2023-09-01)
7. Boeckmann, L., Kietzmann, P., Lanzieri, L., Schmidt, T., Wählisch, M.: Usable Security for an IoT OS: Integrating the Zoo of Embedded Crypto Components Below a Common API (2022). https://doi.org/10.48550/ARXIV.2208.09281
8. Bormann, C., Ersue, M., Keranen, A.: Terminology for Constrained-Node Networks. RFC 7228, RFC Editor (May 2014), http://www.rfc-editor.org/rfc/rfc7228.txt
9. Chui, M., Collins, M., Patel, M.: The Internet of Things: Catching up to an accelerating opportunity. McKinsey & Company, New York (Nov 2021), https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/iot-value-set-to-accelerate-through-2030-where-and-how-to-capture-it, (Zugriff: 2023-09-01)
10. Costan, V., Lebedev, I., Devadas, S.: Sanctum: Minimal Hardware Extensions for Strong Software Isolation. In: 25th USENIX Security Symposium (USENIX Security 16). pp. 857–874. USENIX Association, Austin, TX (Aug 2016), https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/costan
11. Garcia-Morchon, O., Kumar, S., Sethi, M.: Internet of Things (IoT) Security: State of the Art and Challenges. RFC 8576, RFC Editor (April 2019), https://www.rfc-editor.org/rfc/rfc8576.txt
12. Garlati, C., Pinto, S.: Secure IoT Firmware For RISC-V Processors. In: Embedded World Conference 2021 (Mar 2021)
13. Greer, C., Burns, M., Wollman, D., Griffor, E.: Cyber-Physical Systems and Internet of Things. Tech. Rep. 1900-202, National Institute of Standards and Technology (Mar 2019). https://doi.org/10.6028/nist.sp.1900-202

14. Hasan, M.: State of IoT 2022: Number of connected IoT devices growing 18% to 14.4 billion globally (2022), https://iot-analytics.com/number-connected-iot-devices/, (Zugriff: 2023-09-01)
15. Hassija, V., Chamola, V., Saxena, V., Jain, D., Goyal, P., Sikdar, B.: A Survey on IoT Security: Application Areas, Security Threats, and Solution Architectures. IEEE Access **7**, 82721–82743 (2019). https://doi.org/10.1109/access.2019.2924045
16. Hosam, O., BinYuan, F.: A Comprehensive Analysis of Trusted Execution Environments. In: 2022 8th International Conference on Information Technology Trends (ITT). IEEE (May 2022). https://doi.org/10.1109/itt56123.2022.9863962
17. Hu, D.: Trusted Firmware-M Documentation (2023), https://tf-m-user-guide.trustedfirmware.org/, (Zugriff: 2023-09-01)
18. Humayed, A., Lin, J., Li, F., Luo, B.: Cyber-Physical Systems Security — A Survey. IEEE Internet of Things Journal **4**(6), 1802–1831 (Dec 2017). https://doi.org/10.1109/jiot.2017.2703172
19. Johnson, W.A., Ghafoor, S., Prowell, S.: A Taxonomy and Review of Remote Attestation Schemes in Embedded Systems. IEEE Access **9**, 142390–142410 (2021). https://doi.org/10.1109/access.2021.3119220
20. Kocher, P., Lee, R., McGraw, G., Raghunathan, A., Ravi, S.: Security as a New Dimension in Embedded System Design. In: Proceedings of the 41st annual Design Automation Conference. ACM (Jun 2004). https://doi.org/10.1145/996566.996771
21. Lebedev, I., Hogan, K., Devadas, S.: Invited Paper: Secure Boot and Remote Attestation in the Sanctum Processor. In: 2018 IEEE 31st Computer Security Foundations Symposium (CSF). IEEE (Jul 2018). https://doi.org/10.1109/csf.2018.00011
22. Lee, D.: Keystone Enclave Documentation, https://docs.keystone-enclave.org/en/latest/Getting-Started/How-Keystone-Works/RISC-V-Background.html, (Zugriff: 2023-09-01)
23. Lee, D., Kohlbrenner, D., Shinde, S., Asanović, K., Song, D.: Keystone: An Open Framework for Architecting Trusted Execution Environments. In: Proceedings of the Fifteenth European Conference on Computer Systems. ACM (Apr 2020). https://doi.org/10.1145/3342195.3387532
24. Lemke, K.: Embedded Security: Physical Protection against Tampering Attacks. In: Embedded Security in Cars, pp. 207–217. Springer-Verlag. https://doi.org/10.1007/3-540-28428-1_12
25. Lin, J., Yu, W., Zhang, N., Yang, X., Zhang, H., Zhao, W.: A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications. IEEE Internet of Things Journal **4**(5), 1125–1142 (Oct 2017). https://doi.org/10.1109/jiot.2017.2683200
26. Makhdoom, I., Abolhasan, M., Lipman, J., Liu, R.P., Ni, W.: Anatomy of Threats to the Internet of Things. IEEE Communications Surveys & Tutorials **21**(2), 1636–1675 (2019). https://doi.org/10.1109/comst.2018.2874978
27. Ning, Z., Zhang, F., Shi, W., Shi, W.: Position Paper: Challenges Towards Securing Hardware-assisted Execution Environments. In: Proceedings of the Hardware and Architectural Support for Security and Privacy. ACM (Jun 2017). https://doi.org/10.1145/3092627.3092633
28. Ojo, M.O., Giordano, S., Procissi, G., Seitanidis, I.N.: A Review of Low-End, Middle-End, and High-End Iot Devices. IEEE Access **6**, 70528–70554 (2018). https://doi.org/10.1109/access.2018.2879615
29. Oliveira, D., Gomes, T., Pinto, S.: uTango: An Open-Source TEE for IoT Devices. IEEE Access **10**, 23913–23930 (2022). https://doi.org/10.1109/access.2022.3152781

30. Papp, D., Ma, Z., Buttyan, L.: Embedded Systems Security: Threats, Vulnerabilities, and Attack Taxonomy. In: 2015 13th Annual Conference on Privacy, Security and Trust (PST). IEEE (Jul 2015). https://doi.org/10.1109/pst.2015.7232966
31. Pinto, S., Santos, N.: Demystifying Arm TrustZone. ACM Computing Surveys **51**(6), 1–36 (Jan 2019). https://doi.org/10.1145/3291047
32. Rizvi, S., Kurtz, A., Pfeffer, J., Rizvi, M.: Securing the Internet of Things (IoT): A Security Taxonomy for IoT. In: 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE). IEEE (Aug 2018). https://doi.org/10.1109/trustcom/bigdatase.2018.00034
33. Rose, K., Eldridge, S., Chapin, L.: The Internet of Things: An Overview. Internet Society (Oct 2015), https://www.internetsociety.org/resources/doc/2015/iot-overview/, (Zugriff: 2023-09-01)
34. Sabt, M., Achemlal, M., Bouabdallah, A.: The Dual-Execution-Environment Approach: Analysis and Comparative Evaluation. In: ICT Systems Security and Privacy Protection, pp. 557–570. Springer International Publishing (2015). https://doi.org/10.1007/978-3-319-18467-8_37
35. Sabt, M., Achemlal, M., Bouabdallah, A.: Trusted Execution Environment: What It is, and What It is Not. In: 2015 IEEE Trustcom/BigDataSE/ISPA. IEEE (Aug 2015). https://doi.org/10.1109/trustcom.2015.357
36. Schiller, E., Aidoo, A., Fuhrer, J., Stahl, J., Ziörjen, M., Stiller, B.: Landscape of IoT security. Computer Science Review **44**, 100467 (May 2022). https://doi.org/10.1016/j.cosrev.2022.100467
37. SonicWall: 2023 SonicWall Cyber Threat Report (2023), https://www.sonicwall.com/2023-cyber-threat-report/, (Zugriff: 2023-09-01)
38. Tiburski, R.T., Moratelli, C.R., Johann, S.F., Neves, M.V., de Matos, E., Amaral, L.A., Hessel, F.: Lightweight Security Architecture Based on Embedded Virtualization and Trust Mechanisms for IoT Edge Devices. IEEE Communications Magazine **57**(2), 67–73 (Feb 2019). https://doi.org/10.1109/mcom.2018.1701047
39. Waterman, A., Asanović, K.: The RISC-V Instruction Set Manual Volume I: Unprivileged ISA (Dec 2019), https://github.com/riscv/riscv-isa-manual/releases/tag/Ratified-IMAFDQC, version 20191213, (Zugriff: 2023-09-01)
40. Waterman, A., Asanović, K., Hauser, J.: The RISC-V Instruction Set Manual Volume II: Privileged Architecture (Dec 2021), https://github.com/riscv/riscv-isa-manual/releases/tag/Priv-v1.12, version 20211203, (Zugriff: 2023-09-01)
41. Zainuddin, N., Daud, M., Ahmad, S., Maslizan, M., Abdullah, S.A.L.: A Study on Privacy Issues in Internet of Things (IoT). In: 2021 IEEE 5th International Conference on Cryptography, Security and Privacy (CSP). IEEE (Jan 2021). https://doi.org/10.1109/csp51677.2021.9357592
42. Zhao, S., Zhang, Q., Hu, G., Qin, Y., Feng, D.: Providing Root of Trust for ARM TrustZone using On-Chip SRAM. In: Proceedings of the 4th International Workshop on Trustworthy Embedded Devices. ACM (Nov 2014). https://doi.org/10.1145/2666141.2666145
43. Zhao, S., Zhang, Q., Qin, Y., Feng, W., Feng, D.: Minimal Kernel: An Operating System Architecture for TEE to Resist Board Level Physical Attacks. In: 22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019). pp. 105–120. USENIX Association, Chaoyang District, Beijing (Sep 2019), https://www.usenix.org/conference/raid2019/presentation/zhao