Sicherheit für ROS-basierte Applikationen auf Anwendungsebene

Bernhard Dieber[1] · Marc Pichler[1] · Stefan Rass[2] · Peter Schartner[2]

[1]Joanneum Research
Institut für Robotik und Mechatronik
{Bernhard.Dieber | Marc.Pichler}@joanneum.at

[2]Alpen-Adria-Universität Klagenfurt
Institut für Angewandte Informatik
{Stefan.Rass | Peter.Schartner}@aau.at

Zusammenfassung

Die starke Vernetzung in Industrie 4.0 Szenarien bedeutet auch, dass bisher durch Isolation gesicherte Produktionsnetzwerke nun potentiellen Angriffen ausgesetzt sind. Daher sind Sicherungsmaßnamen an den Anwendungen in solchen Netzwerken unabdingbar. Das Robot Operating System (ROS) erfreut sich großer Beliebtheit im Umfeld der Robotik. Allerdings finden sich in diesem Kommunikations-Framework einfach ausnutzbare Sicherheitslücken, welche unter anderem durch die Klartext-Natur der Datenübertragung im Publish-Subscribe-Mechanismus von ROS hervorgerufen werden. Dieser Umstand birgt nicht nur Risiken für Sachgüter, sondern – wie speziell im Falle der kollaborativen Robotik – auch beträchtliche Gefahren für die körperliche Gesundheit von Personen. In diesem Artikel schlagen wir eine Sicherheitsarchitektur auf Anwendungsebene vor. Während kryptographische Methoden Datensicherheit sowie -integrität gewährleisten, stellt ein dezidierter Authentifizierungsserver sicher, dass nur berechtigten Knoten der Zugriff auf die Anwendung gewährt wird. In einer Demonstration mit einem kollaborativen Roboter zeigen wir, wie eine solche Architektur praktisch eingesetzt werden kann, um Risiken einer Kompromittierung zu minimieren.

1 Einleitung

Bis vor einigen Jahren noch konnte die Systemsicherheit in Produktionsstätten großteils durch Isolation von Netzwerken gewährleistet werden. In einem Industrie 4.0 Umfeld wird jedoch der vermehrte Austausch zwischen Produktions-Netzen und Backend-Services wie Enterprise Resource Planning und ähnlichen Systemen nötig und daher ist es nicht mehr möglich diese Netze isoliert zu betreiben. Dies erhöht die Anfälligkeit für Cyber-Angriffe auf solche nicht mehr isolierten Netzwerke [ChDV13].

Am Beispiel von Stuxnet ist zu sehen, dass Industrienetzwerke – selbst wenn diese durch Air-Gaps abgekapselt agieren – nicht zwangsläufig vor Malware geschützt sind [Karn11]. Netzwerk- und Anwendungssicherheit in der Industrie sind Themen, die schon seit geraumer Zeit diskutiert werden [ChDV13, SPLA+15, ByDH04, DNHC05]. Eine grundlegende Übersicht über Sicherheitsproblematiken in Publish-Subscribe-Systemen wurde in [WCEW02] präsentiert. Kürzlich wurde eine auf künstlicher Intelligenz basierende Methode zur Eindringlings-

P. Schartner (Hrsg.) · D•A•CH Security 2016 · syssec (2016) 363-370.

Erkennung für SCADA-Systeme vorgestellt [MaJi14]. Shin et al. [SKJP+10] beschreiben verschiedenste Ansätze zur Eindringlings-Detektion in drahtlose Industrienetzwerke und schlagen in ihrer Arbeit Verbesserungen derselben vor. In [ÅGLN+11] wird von den Autoren eine Methode gezeigt, wie bestehende, verkabelte Infrastruktur mit drahtlosen Komponenten erweitert werden kann, ohne die Sicherheit zu beeinträchtigen. Ein kürzlicher Angriff auf die Energieversorgung in der Ukraine zeigt auf, dass auch heute noch die Sicherheit in industriellen Systemen alles andere als gewährleistet ist und dass es noch sehr viel Bedarf an Verbesserungen gibt [Fair16].

Auch ist die Industrie geprägt von Falschauffassungen der Gefahren, die von solchen Systemen ausgehen können. Die Fixierung auf unbedingte Geheimhaltung von Fertigungsplänen und Prozessen führt dazu, dass mögliche andere Angriffsszenarien übersehen werden.

Ein hypothetisches Szenario: in der Fertigungsstraße eines Automobilherstellers wurden Steuerungen von Industrierobotern mit Malware infiziert. Diese bewirkt, dass weniger Schweißpunkte an kritischen Teilen des Fahrzeugs gesetzt werden. Ein solcher Vorfall könnte zu verkürzten Service-Intervallen und – im Extremfall – zu Unfällen mit Todesfolge führen.

Seit kurzer Zeit gewinnt eine neue Generation industrieller Robotersysteme an Bedeutung, die sich durch ihre Eignung zur direkten Zusammenarbeit mit Menschen auszeichnet. Diese Roboter müssen nicht mehr länger in Käfigen oder hinter Lichtschranken betrieben werden, sondern können neben und mit Menschen arbeiten. Dies wird meist durch eine Leichtbauweise sowie zusätzliche Sensorik zur Erkennung von Kollisionen erreicht. In solchen kollaborativen Szenarien muss spezielles Augenmerk auf die Sicherheit der beteiligten Personen gelegt werden. Dafür werden die Robotersysteme meist mit zusätzlichen Sensoren ausgerüstet um z.B. die Annäherung eines Menschen und dessen Aktionen zu erkennen. Denn trotz der kollaborativen Eigenschaften dieser Roboter können sie bei hoher Geschwindigkeit oder ruckartigen Bewegungen immer noch beträchtlichen Schaden anrichten. Um in einer kollaborativen Arbeit nutzbringend zu sein, müssen diese Roboter direkt auf die beteiligten Personen reagieren, es kann also nicht mehr nur mit vorgegebenen Trajektorien und Zeitintervallen gearbeitet werden. Daher sind diese Systeme auf einen konstanten Informationsstrom einer Vielzahl an Sensoren angewiesen, um ihre Umgebung und die momentane Situation richtig zu erkennen und zu interpretieren. Hierzu zählen die Daten von Gelenkkräften, Kameras sowie 3D-Sensoren. Um diese komplexen Abhängigkeiten zwischen einzelnen Systemteilen beherrschbar zu machen, werden oft Mechanismen zur Entkopplung der einzelnen Module verwendet.

Security ist in Robotik-Systemen bisweilen keineswegs immer berücksichtigt, wie die Übersicht in [ElSo12] zeigt. Gleichwohl diverse Ansätze und Ideen für Sicherheitsarchitekturen in der Literatur existieren (siehe etwa [FiKi11] oder [ElSo12]), so ist Security häufig ein sekundäres Ziel dem die korrekte Funktionalität des Roboters mit höherer Priorität vorangeht. Im Umkehrschluss erweist security sich jedoch als notwendige Voraussetzung für eine korrekte Funktion des Roboters, wie wir durch praktische Beispiele weiter unten belegen. Zuweilen ist eine gewisse baseline-security sehr einfach erreichbar, wie wir am Beispiel des Robot Operating System (ROS) demonstrieren werden.

2 Das Robot Operating System

ROS, das Robot Operating System [QCGF+09], gewinnt in den letzten Jahren an Popularität. Seine Publish-Subscribe Architektur [WCEW02] ermöglicht es starke Abhängigkeiten zwi-

schen einzelnen Software-Modulen in Robotik-Anwendungen aufzulösen. Das grundlegende Konzept von Publish-Subscribe besagt, dass einzelnen Module Daten bereitstellen (sie sind Publisher) und diese Daten von anderen Modulen ”abonniert”werden können (die Subscriber). Darunterliegende Infrastruktur ist dafür verantwortlich diese Datenströme zu verwalten und Daten von Publishern zu Subscribern zu transportieren. ROS speziell implementiert topic-based publish-Subscribe. Hier können Publisher und Subscriber Themen für Daten vergeben, die sie produzieren beziehungsweise konsumieren. Für jedes Kommando oder jeden Sensorwert gibt es also ein eigenes Topic (das durch eine eindeutige Zeichenkette repräsentiert wird). Publish-Subscribe sorgt dafür, dass Publisher ihre Subscriber nicht kennen und vice versa. Einzelne Module können also separat voneinander entwickelt und getestet und unter Verwendung von ROS (oder eines anderen Publish-Subscribe Frameworks) zu immer neuen Anwendungen kombiniert werden.

In Publish-Subscribe gilt generell, dass jeder Publisher für beliebig viele Themen Daten erzeugen und ein Subscriber beliebig viele Themen abonnieren darf.

Diese Entkopplung birgt jedoch Risiken im Bereich der Sicherheit, welche von ROS nur wenig und oft sogar gar nicht berücksichtigt werden [MSFM13, FiKi11]. Auch eine kürzliche Sicherheitsüberprüfung hat gravierende Mängel in ROS festgestellt [StPo14]. Neben Schwächen bei Authentizität, Vertraulichkeit und Integrität wird beispielsweise darauf hingewiesen, dass es in ROS möglich ist, bereits laufende (und üblicherweise berechtigte) Knoten aus der Anwendung auszuschließen indem man einen Knoten mit demselben Namen startet. Auch eine XML DoS Lücke durch eine XML-Bombe im ROS Master wird aufgezeigt.

Ein Subscriber in ROS hat beispielsweise keine Möglichkeit, die Integrität der verarbeiteten Daten zu verifizieren oder festzustellen, ob die Daten aus der richtigen Quelle stammen. Ein Publisher wiederum kann nicht verhindern, dass unautorisierte Subscriber seine Daten verarbeiten. Zusätzlich dazu können Publisher oder Subscriber, ohne eine Authentifizierung durchlaufen zu müssen, in ein laufendes System eingreifen. Diese Eigenschaften ermöglichen verschiedene Angriffsszenarien auf eine ROS-basierte Applikation: es können falsche Sensordaten bzw. Bewegungsbefehle eingeschleust werden, die den Roboter zu falschen Bewegungen veranlassen. Dies kann entweder den Produktionsprozess stören oder auch zu Sachschäden oder Verletzungen von Menschen führen. Ein unautorisierter Subscriber könnte alle Nachrichten einer Applikation abfangen und die Bewegungsbefehle zum Reverse-Engineering des Produktionsprozesses verwenden. Dies hat potentiell erheblichen wirtschaftlichen Schaden zur Folge.

Während die Verwendung eines VPN hier als eine schnelle und einfache Lösung scheint, führt diese jedoch durch Verschlüsselung von nicht-sensitiven Daten zusätzlichen Overhead ein. Dieser wirkt sich sowohl im Echtzeitbetrieb des Systems (durch langsameren Verbindungsaufbau) aus, als auch in dessen Management, da die für das VPN notwendigen Zertifikate geschützt und im laufenden Betrieb gewartet werden müssen. Beides stellt einen Mehraufwand für das System und dessen Betreiber dar, welcher nach Möglichkeit minimiert werden sollte. Das in diesem Beitrag vorgeschlagene Verfahren beruht gleichermaßen auf asymmetrischer Kryptographie, verlagert das Management der Schlüssel jedoch in die Komponenten des Roboters, sodass der damit verbundene Aufwand sich auf die Hersteller der einzelnen Module verteilt. Darüber hinaus wurde in [ByDH04] vorgeschlagen, Sicherheit für Industrienetzwerke in Schichten aufzubauen. In einer solchen mehrschichtigen Sicherheitsarchitektur kann ein solches VPN natürlich zusätzlich zur Absicherung beitragen, jedoch sollte auch auf die Sicherheit der Anwendung selbst großes Augenmerk gelegt werden. Ein VPN würde lediglich Verschlüsselung und Netz-

werkzugang abdecken. Die Berechtigungen innerhalb der Anwendung (wer darf welche Daten publishen und wer darf sie erhalten) werden dadurch nicht berücksichtigt.

Der Rest dieser Arbeit ist wie folgt strukturiert: in Abschnitt 3 schlagen wir eine Architektur auf Anwendungsebene vor, welche die größten Sicherheitslücken in typischen ROS-Anwendungen adressiert, in Kapitel 4 demonstrieren wir diese Architektur anhand einer praktischen Applikation.

3 Sicherheitsarchitektur auf Anwendungsebene

In dieser Arbeit beschränken wir uns auf die Anwendungsebene. Wir behandeln ROS als Black-Box und implementieren Sicherheitsmaßnahmen wie einen Authentifizierungs-Server (AS) und Funktionen in den ROS-Knoten selbst.

Diese Herangehensweise erlaubt es uns zwei schwerwiegende Sicherheitsprobleme zu lösen:

1. Das Verändern der Trajektorie der Roboter-Bewegung. Damit einhergehend wäre eine unerlaubte Abänderung im Produktionsablauf möglich (z.B. Änderung der Anzahl an Schweißpunkten) sowie eine Gefährdung der Sicherheit von Menschen in der Nähe und mögliche Schäden an der Umgebung.
2. Das Ausschleusen von Informationen durch unautorisierte Subscriber. Dies könnte potentiell zum Reverse-Engineering von geschützten Produktionsprozessen verwendet werden.

Die hier für Publisher und Subscriber beschriebenen Mechanismen können auch für ROS-Services und die weiteren vorhandenen Datenverarbeitungsmechanismen eingesetzt werden.

3.1 Registrierung von neuen Knoten

Jeder ROS-Knoten (Publisher oder Subscriber) muss vor Aufnahme seiner Tätigkeiten eine Authentifizierung durchlaufen. Hierfür existiert eine eigene Komponente, welche die Rolle eines Authentifizierungs-Servers (AS) übernimmt. Mittels gängiger kryptografischer Methoden werden die Identität und die Berechtigungen jedes einzelnen Moduls festgestellt. Die Authentifizierung beginnt mit einem Challenge-Response Verfahren auf Basis digitaler Signaturen, in welchem ein Knoten seine Identität bestätigt sowie ein Kommunikationsschlüssel mit dem Authentifizierungsserver ausgehandelt wird.

Danach wird für jedes Topic, für welches ein Publisher Daten erzeugt bzw. welches ein Subscriber abonniert, ein eigener Schlüssel für symmetrische Verschlüsselung vergeben (K_i für das Topic i). Dieser Topic-Schlüssel wird vom Authentifizierungsserver erzeugt und nach erfolgreicher Authentifizierung per asymmetrischer Verschlüsselung an die relevanten Publisher und Subscriber verteilt. Man beachte, dass für die direkte Kommunikation zwischen Publisher und Subscriber asymmetrische Kryptographie nicht zum Einsatz kommt, da der Publisher seine Subscriber und damit auch deren öffentliche Schlüssel a priori nicht kennt und auch den Datenstrom nicht beeinflussen kann (also z.B. Daten nicht an ausgewählte Subscriber schicken kann). Zusätzlich wird der öffentliche asymmetrische Signaturschlüssel pk_P eines Publishers an den AS übertragen bzw. vom AS an einen Subscriber weitergegeben. Die eigentliche Kommunikation zwischen Subscriber und Publisher läuft somit symmetrisch verschlüsselt mit dem Topic key K_i, welcher vom AS (in der Rolle eines online Trust-Centers) ausgehandelt wird (das Verfahren ist analog zu herkömmlichen/bekannten Verfahren des Schlüsselaustausches unter Verwendung von Trust Centern; vgl. [Schn96]).

1. Zusammen mit der Registrierungsanfrage übermittelt der Publisher dem Zertifizierungsserver ein Public-Key-Zertifikat [a] $Z = (P, pk_P, S, \texttt{sign}(P \| pk_P \| S, sk_S))$. Dieses Zertifikat Z ermöglicht es dem AS zu ermitteln, wer die Quelle S des neuen Publishers ist.
2. Der AS vergleicht den authentischen Public-Signature-Verification-Key von S, und verifiziert den erhaltenen Key indem gegen $\texttt{verify}(Z, pk_S) \stackrel{?}{=} \texttt{true}$ geprüft wird. Nur wenn dies zutrifft, sendet der AS einen zufälligen Wert r zu P, welcher von P mit seinem Secret-Key sk_P, welcher zu Public-Key pk_P gehört, digital signiert wird.

   Der neue Publisher-Kanidat P antwortet dem AS mit der Signatur $sig = \texttt{sign}(r, sk_P)$
3. Wie zuvor verwendet der AS den authentifizierten Public-Key pk_P um zu verifizieren dass die Signatur an r korrekt ist. Der neue Publisher P wird also nur dann akzeptiert wenn $\texttt{verify}(sig, pk_P) = \texttt{true}$ ist.

[a] Es ist zu beachten, dass hier der Inhalt des Zertifikats absichtlich auf das Wesentliche beschränkt wurde. Das tatsächliche Zertifikat würde natürlich eine komplexere Struktur aufweisen.

Abb. 1: Zertifikat-basierte Challenge-Response Authentifizierung

3.2 Daten-Dissemination durch Publisher

Nachdem ein Publisher P innerhalb der Anwendung gestartet ist und die Authentifizierung und Schlüsselgenerierung durchlaufen hat, kann er beginnen, Daten Subscribern zur Verfügung zu stellen. Für jedes Daten-Topic i gibt es einen eigenen symmetrischen Schlüssel K_i, welcher während der Authentifizierung ausgehandelt wurde. Daten werden in einzelne Nachrichten-Pakete unterteilt. Jede Nachricht wird mit der Publisher-Identität P konkateniert und das Ergebnis mit dem symmetrischen Schlüssel K_i verschlüsselt: $c = E(P \| m, K_i)$. Das erhaltene Chiffrat c wird mit dem privaten Schlüssel sk_P signiert. Chiffrat und Signatur werden konkateniert und das Ergebnis wird dann über ROS an alle Subscriber veröffentlicht.

Ein Subscriber wiederum entschlüsselt die erhaltene Nachricht mit dem symmetrischen Schlüssel K_i um den Klartext bestehend aus Publisher Identität P und der ursprünglichen Nachricht m zu erhalten: $P \| m = D(c, K_i)$. Der Subscriber verifiziert die Signatur mittels des öffentlichen Schlüssels pk_P und kann die erhaltenen Daten m nach erfolgter Verifizierung weiterverarbeiten.

4 Demonstration

Um unsere Herangehensweise zu veranschaulichen, wurde eine Anwendung in ROS implementiert. In unserem Experiment wurde ROS Indigo verwendet, alle kryptographischen Funktionalitäten wurden mit CryptoPP [1] realisiert.

Die zur Demonstration entwickelte ROS-Applikation sendet periodisch gewünschte Gelenks-Positionen an einen KUKA iiwa[2]. Der Roboter sendet periodisch die aktuelle Position und Rotation des End-Effektors sowie die aktuellen Gelenksstellungen zurück. Abbildung 2 gibt einen Überblick über die Anwendung.

[1] http://cryptopp.com
[2] http://www.kuka-lbr-iiwa.com

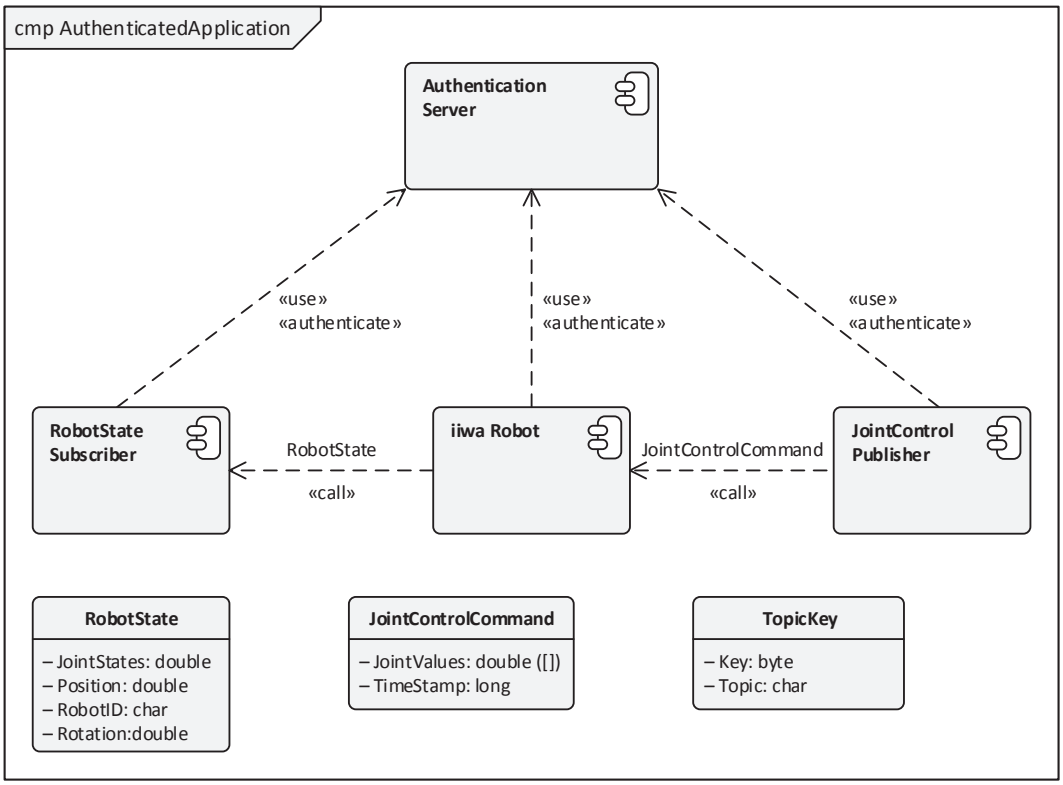

Abb. 2: Überblick über die Architektur und den Datenfluss der Applikation.

Die Informationsflüsse in dieser Anwendung entsprechen normalen Gegebenheiten in einer Roboter-Applikation. Hier ist es für einen Angreifer sowohl möglich unvorhergesehene Bewegungen auszulösen indem falsche Befehle eingeschleust werden oder auch aus den aufgezeichneten Status-Meldungen des Roboters die Bewegungs-Trajektorien rückzurechnen.

Da der KUKA iiwa in die Gruppe der kollaborativen Roboter einzuordnen ist, muss davon ausgegangen werden, dass sich Menschen in seinem Arbeitsumfeld befinden. Hier sind also ungeplante Bewegungen als sehr gefährlich zu werten.

Die Kommandos, welche die Winkel an den Gelenken direkt beeinflussen, bewegen den Roboter dazu, einer einfachen Trajektorie zu folgen. Damit ist es auch möglich, visuell darüber informiert zu werden, ob der Roboter seinem aktuellen Pfad folgt, oder ob bereits eine Störung durch einen Gegenspieler vorliegt.

In einem echten kollaborativen Szenario ist jedoch eine solche visuelle Feststellung nicht möglich, da der Roboter auf den Menschen reagieren muss und daher keine vorprogrammierte Trajektorie abfährt. Eine solche Situation ist daher wesentlich kritischer im Falle einer Fremdsteuerung des Roboters als es dies in einem klassischen Produktionsszenario wäre.

Im ersten unserer Testfälle, werden alle in Abschnitt 2 beschriebenen Sicherheitsmaßnamen ignoriert und lediglich ROS in seinem Ursprungszustand verwendet. Keine zusätzlichen Sicherheitsfeatures wurden implementiert. Dieser Testfall dient als Referenz für ein ungesichertes System, die dann mit dem zweiten unserer Testfälle verglichen werden kann.

Im zweiten Testfall werden alle Hinweise aus Abschnitt 2 umgesetzt und aktiv verwendet.

Wir verwenden einen Authentifizierungsserver, an dem sich neue Knoten authentifizieren müssen und der für das Schlüsselmanagement zuständig ist. Zusätzlich dazu werden die einzelnen Werte, welche die Winkel des Roboters bestimmen mit dem Topic-Key verschlüsselt. Dadurch entstehen ROS-Nachrichten, in welchen nur die Werte selbst, nicht jedoch die gesamte Message verschlüsselt wird. Es ist also für einen lauschenden Knoten im Netzwerk möglich, die Frequenz unterschiedlicher Nachrichtentypen zu erfassen.

Verschlüsselung des aktuellen Status des Roboters stellt sicher, dass ein böswilliger Subscriber-Knoten keine Informationen über die tatsächliche Trajektorie des Roboters sammeln kann.

4.1 Anwendungsverhalten mit inaktiver Sicherheit

Sind keine zusätzlichen Sicherheitsmechanismen vorhanden, so kann ein unautorisierter Publisher-Knoten die geplante Trajektorie des Roboters beeinflussen. Jegliche Kommandos, die der Knoten aussendet, werden ausgeführt. Dies ruft unvorhersehbare Bewegungen des Roboter-Arms mit hohen Geschwindigkeiten und der daraus resultierenden hohen kinetischen Energie hervor. Dies könnte Verletzungen an Menschen sowie Sachschäden an anderen Robotern oder sonstigem Equipment hervorrufen.

4.2 Anwendungsverhalten mit aktiver Sicherheit

Sind die zusätzlichen Sicherheitsmechanismen aus Abschnitt 3 implementiert, so ist es dem böswilligen Knoten nicht möglich sich bei dem AS zu authentifizieren. Der Topic-Key, welcher benötigt wäre, um gültige Gelenks-Positionen an den Roboter zu senden, wurde nicht an den Knoten vergeben. Der Roboter-Knoten überprüft, ob erhaltene Messages richtig verschlüsselt bzw. signiert wurden und führt daher nur autorisierte Kommandos aus. Somit folgt der iiwa nur der Trajektorie, die auch vom authentifizierten Knoten beabsichtigt ist.

5 Schlussfolgerungen und Zukünftige Arbeiten

Die in dieser Arbeit vorgestellte Sicherheitsarchitektur adressiert auf Anwendungsebene Sicherheitsinsuffizienzen, die in ROS aber auch in vielen anderen Publish-Subscribe Systemen existieren. Dazu gehören das Einschleusen von unautorisierten Daten und Kommandos sowie das Abhören von Befehlen und Status-Meldungen. Dennoch existieren weiterhin Schwächen, die nur durch Änderungen in ROS selbst ausgemerzt werden können. Dazu zählt beispielsweise eine mögliche Frequenzanalyse zu bestimmten Nachrichten, welche trotz verschlüsseltem Inhalt Aufschluss über bestimmte Kommunikationsmuster der Anwendung geben. Zusätzlich können Module nicht an einer Teilnahme in der laufenden Applikation gehindert werden. Daher könnte ein Publisher zwar nur Kommandos aussenden, die wegen der fehlenden Verschlüsselung von anderen Modulen ignoriert werden, die aber dennoch in hoher Frequenz zu einer Denial-of-Service Attacke genutzt werden können. Schlussendlich muss auch noch eine Methode entwickelt werden, die verhindert, dass Topic-Schlüssel zwischen Knoten ausgetauscht werden.

Die Adressierung verbleibender Sicherheitsprobleme erfordert Eingriffe in ROS selbst und wird Teil unserer zukünftigen Arbeiten sein. Ein erster Schritt ist es, die Aushandlung von Topics mit dem Master mit einer Authentifizierung und Autorisierung zu unterlegen, wobei hier große Teile des bereits implementierten Verfahrens in den ROS-Core übernommen werden können.

Literatur

[ÅGLN+11] J. Åkerberg, M. Gidlund, T. Lennvall, J. Neander, M. Björkman: Efficient integration of secure and safety critical industrial wireless sensor networks. In: *EURASIP Journal on Wireless Communications and Networking*, 2011, 1 (2011), 1–13, .

[ByDH04] E. Byres, P. E. Dr, D. Hoffman: The myths and facts behind cyber security risks for industrial control systems. *In: In Proc. of VDE Kongress* (2004).

[ChDV13] M. Cheminod, L. Durante, A. Valenzano: Review of Security Issues in Industrial Networks. In: *Industrial Informatics, IEEE Transactions on*, 9, 1 (2013), 277–293.

[DNHC05] D. Dzung, M. Naedele, T. von Hoff, M. Crevatin: Security for Industrial Communication Systems. In: *Proceedings of the IEEE*, 93, 6 (2005), 1152–1177.

[ElSo12] A. Elkady, T. Sobh: RoboticsMiddleware: A Comprehensive Literature Survey and Attribute-Based Bibliography. In: *Journal of Robotics* (2012), 15p, hindawi Publishing Corporation, doi:10.1155/2012/959013.

[Fair16] P. Fairley: Cybersecurity at U.S. utilities due for an upgrade: Tech to detect intrusions into industrial control systems will be mandatory [News]. In: *IEEE Spectrum*, 53, 5 (2016), 11–13.

[FiKi11] M. Finnicum, S. T. King: Building Secure Robot Applications. *In: Proceedings of the 6th USENIX Workshop on Hot Topics in Security* (2011).

[Karn11] S. Karnouskos: Stuxnet Worm Impact on Industrial Cyber-Physical System Security. *In: 37th Annual Conference of the IEEE Industrial Electronics Society (IECON 2011)* (2011), 4490–4494.

[MaJi14] L. Maglaras, J. Jiang: Intrusion detection in SCADA systems using machine learning techniques. *In: Science and Information Conference (SAI)* (2014), 626–631.

[MSFM13] J. McClean, C. Stull, C. Farrar, D. Mascareñas: A preliminary cyber-physical security assessment of the Robot Operating System (ROS) (2013), .

[QCGF+09] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng: ROS: an open-source Robot Operating System. *In: ICRA workshop on open source software* (2009), Bd. 3, 5.

[Schn96] B. Schneier: Applied Cryptography. Wiley, 2nd Aufl. (1996).

[SKJP+10] S. Shin, T. Kwon, G.-Y. Jo, Y. Park, H. Rhy: An Experimental Study of Hierarchical Intrusion Detection for Wireless Industrial Sensor Networks. In: *Industrial Informatics, IEEE Transactions on*, 6, 4 (2010), 744–757.

[SPLA+15] K. Stouffer, V. Pillitteri, S. Lightman, M. Abrams, A. Hahn: Guide to Industrial Control Systems (ICS) Security. Tech. Rep., National Institute of Standards and Technology (2015), NIST Special Publication 800-82, Revision 2.

[StPo14] L. Stroetmann, H. Pohl: Robot Operating System (ROS): Safe & Insecure. Technical report, SoftScheck GmbH (2014).

[WCEW02] C. Wang, A. Carzaniga, D. Evans, A. Wolf: Security issues and requirements for Internet-scale publish-subscribe systems. *In: System Sciences. HICSS. Proceedings of the 35th Annual Hawaii International Conference on* (2002), 3940–3947.