Secured Action Authorization for Industrial Mobile Robots

Sarah Haas
Development Center Graz
Infineon Technologies Austria AG
Graz, Austria
sarah.haas@infineon.com

Thomas Ulz
Institute for Technical Informatics
Graz University of Technology
Graz, Austria
thomas.ulz@tugraz.at

Christian Steger
Institute for Technical Informatics
Graz University of Technology
Graz, Austria
steger@tugraz.at

***Abstract*—The increased deployment of mobile robots in industry poses higher risks for safety issues and damage of production material caused by abnormal robot behavior. Abnormal behavior can be caused by issues such as operational errors or security weaknesses. Security weaknesses entail attacks that might cause serious harm for humans and production material. One possibility to mitigate such attacks is action authorization. In this paper, we propose an authorization mechanism for actuator actions on mobile robots to prevent adversaries from executing false commands on a robot. The mechanism uses a ticketing system and relies on sensor data as well as plausibility checks. The tickets are issued to specific actuators to authorize the requested action. The sensor data adds to the understanding of the current context and the plausibility checks are used to check that specific actions are allowed to be performed in a given order. Security is further supported by secure elements that protect confidential data from being revealed.**

***Index Terms*—Authorization, Plausibility Checks, Ticketing, Mobile Robot.**

I. Introduction

In recent years, the trend towards increased automation entailed an increased application of mobile robots in industry. Strategies such as *Industry4.0* [1] and its centerpiece *Smart Factories* [2] will promote mobile robots in industry even more. Typical tasks for mobile robots in factories will include interaction with machines, other robots, production material or some server such as fetching or delivering production material, providing data or manipulating production material. As a research testbed for smart factories, the RoboCup Logistic League [3] was invented. In this testbed, a mobile robot basically consists of a central computing unit, a communication unit such as a router, one or more sensors such as a laser scanner or camera, and one or more actuators such as a gripper or motor. Furthermore, the communication partner of the mobile robot is a backend system. The basic structure of a mobile robot is shown in Fig. 1. The purposes of the different components are:

- **Actuators**: Actuators enables the robot to interact with the physical world. In case of the represented robot, the gripper is used to grab production material, and the motor is necessary to move the robot's wheels.
- **Central computing unit (CCU)**: The central computing unit triggers and coordinates all components on the robot. It uses the sensor measurements to calculate paths and trajectories as well as receiving processes commands from a central server.
- **Sensors**: Sensors are used to detect the robot's environment. The laser scanner, for example, is necessary to detect obstacles or measure distances. The camera is used when detecting specific objects or structures in the robot's operating range.
- **Communication Unit**: The communication unit, mostly a router, is used to communicate with other robots, factory equipment, or a central server. The unit can be equipped with different communication technologies for short and long-range communication such as Industrial WiFi, Near Field Communication, Bluethooth, or Zigbee. Furthermore, it acts as a router to distribute the data between the robot's components.
- **Backend**: The backend is the central planning unit that provides commands to the mobile robot. It decides which task a robot should execute to achieve maximum performance. The backend sends a command and signature of the command to the robot.

As all components on the robot are connected to the router via Ethernet, it would be easy for an adversary to access sensors and actuators. The adversary is not forced to hijack the CCU when trying to manipulate a sensor or actuator. The possibility of this kind of manipulations could cause safety issues for human workforce and might cause damage to production material. The damage could range from total destruction of workpieces or harm for humans to inaccuracies that would lower the product quality. To prevent such scenarios, security mechanisms need to be introduced [4], [5].

One step towards security is the authorization [6] of actions. Authorization is used to prevent adversaries from injecting malicious commands or changing any settings on sensors and actuators. In this paper, a two-staged authorization mechanism for industrial mobile robots will be presented. In the first stage, the mechanism performs a context-aware plausibility check on the incoming command. In the second stage, the sensor data of the robot is used to check the correct location and distance from a machine or workpiece, and an expiring authorization ticket is generated in case of a correct position. The authorization ticket can be validated by the actuator to check the source and correctness of the issued command. To protect the ticket generation and validation as well as the necessary secret keys, the mechanism uses secure elements (SE). Due to the fact, that those secure elements are much slower than traditional CPUs, significant overhead is caused

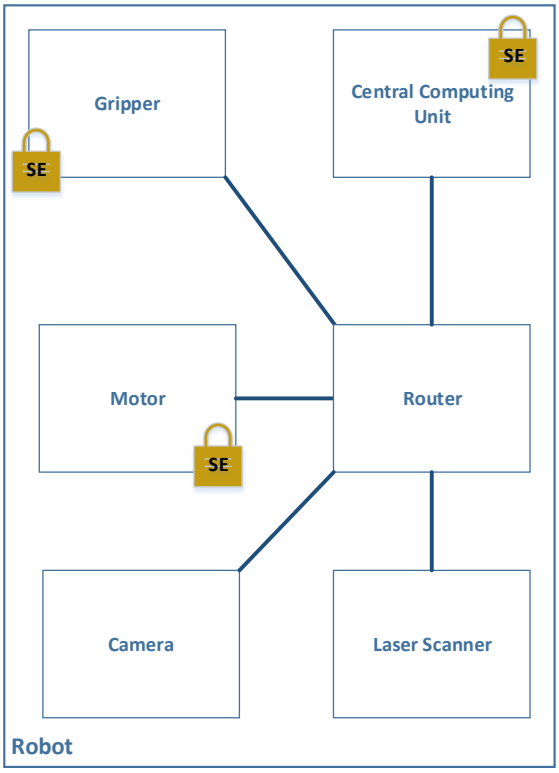

Fig. 1: This blockdiagram shows the basic structure of a mobile robot and how the components are connected. The components are connected by traditional Ethernet cables. The image further shows that the gripper, central computing unit and motor are equipped with secure elements (SE). The central planning unit is not in the picture since it is simply a remote server.

when each action needs to be validated by the actuator. To overcome this issue, a plausibility check is performed in the first stage of the authorization process. The check indicates whether the received command is realistic in the given context. In case of a realistic command, the CCU might not check the command's signature for its validity to reduce the overhead caused by the validity checks as the verification of the signature is rather expensive on a secure element.

To summarize, the contribution of this paper is a two-staged authorization mechanism for industrial robots, relying on context-aware plausibility checks and expiring authorization tickets. The mechanism is supported by secure elements to protect the ticket generation and validation, and the key material of the authorization tickets.

The remainder of this paper is structured as follows. In Section II, the background information and related work regarding the proposed approach are covered. Section III describes the authorization approach, as well as the ticket generation and validation, and the plausibility checks in detail.In Section IV, the threat analysis is discussed to show that the mechanism overcomes critical security issues. Finally, the paper is concluded in Section V.

II. Related Work and Background

A. Secure Element (SE)

An SE is a tamper-resistant device capable of performing cryptographic operations such as signature generation and storing confidential data such as key material in a secured environment. Tamper resistance means that the devices are protected against physical attacks such as side-channel attacks that try to reveal the confidential data. Side channel attacks are a typical problem for general purpose microcontrollers or CPUs as it can be possible to spy on calculation times depending on the input or try to manipulate calculations. SEs, in contrast, are built to withstand such attacks and are therefore used for security critical applications such as bank cards or Trusted Platform Modules [7].

B. One-Time Passwords (OTP)

The concept of One-Time Passwords (OTPs) was introduced by Lamport [8] in 1981. OTPs are passwords that become invalid after its first use. The motivation included attacks on plain text passwords such as interception of the password and later, replay attacks where the user's login credentials are captured and later used to access a system [9]. OTPs rely on non-invertible hash functions to create passwords and also to verify them. Therefore, both, client and the host have to use the same non-invertible hash function for OTP generation and verification.

As OTPs are a very basic concept, M'Raihi et al. [10] proposed *HOTP* (HMAC-based OTP). HOTP extends OTPs with Hashed Message Authentication Codes (HMAC). To extend the concept, a counter is introduced that is combined with the secret key or password to enable the use of the same secret key. The counter and secret key are forwarded to the hash function to generate an OTP. To enable a verification, client and server need to know the correct counter as well as the secret key. The counter is synchronized with a trusted entity such as a server. Each time an OTP is generated, the counter is incremented by a specific amount known by the client and server. When validating an OTP, the receiver increases its counter and then verifies the OTP. The server detects if the counter values diverge. To synchronize the counter, the server calculates several OTPs with increasing counter values and matches them to the received client OTP. If one of the calculated OTPs matches the client's OTP, the server sets his counter to the corresponding value; otherwise, the client cannot authenticate on the server anymore.

C. Authorization

The notion authorization was defined by Fraser in 1997 [6] as granting privileges to processes or users. This term is widely used in any operating system, company network or production system. Research in the area of authorization mostly includes access rights, access policies or group authorization concepts [11], [12]. Currently, research topics focus, amongst other things, on authorization and access rights in the cloud [13] or Internet of Things (IoT) systems [14].

A very basic authorization concept was shown by Gonçalves et al. [15]. The proposed approach uses a realistic sensor and actuator model for wheeled mobile robot simulations. In the simulation, a register whose value was checked before executing an action on a robotic arm is included. The register simply contained '1' if the movement was allowed to be executed, otherwise '0'. This solution does not contain any security concepts to protect the register from misuse.

The only more advanced approach known to the authors that includes authorization related to robotics and other mobile devices was proposed by Popovici et al. in 2003 [16]. The authors proposed a middleware platform for mobile devices in which the environment is enabled to adopt an executed application rather than performing self-adaptation of the application depending on the environment. The included authorization mechanism prevents unauthorized clients such as users or applications from executing actions on, e.g., a robotic arm. This mechanism checks clients privileges to ensure that they are authorized to execute actions in a physical system but does not include any current environmental information. Furthermore, this paper states that authorized clients must be defined by a trusted party before the middleware is started. The authorized clients and their privileges are fixed during runtime to prevent modifications by adversaries.

D. Execution Monitoring

In the field of robotics, plausibility checks are not considered as research topic yet; however, execution monitoring is related to it. In traditional execution monitoring, the model-based state of the execution and the real world state computed from sensing data are compared and checked for discrepancies. Plausibility checks aim at checking that the execution of an action is valid in the given context before it is executed. Plausibility checks are related to security which is important but not widely covered yet in robotic research, and therefore no such concepts exist yet. Plausibility checks are necessary to prevent adversaries from executing actions or injecting false commands in a mobile robot. Furthermore, they reduce the impact of hardware secure elements on the timing behavior of the mobile robot as the secure element is not necessarily used to check each incoming command or action.

The fact, that plausibility checks are not relevant can be seen in the paper by Norelis et al. [17]. The authors show a possible architecture for plan execution monitoring and control for mobile robots. They divide the architecture into three levels, the functional level, the control system level and the planning level. In the functional level, they describe the parts of the robot responsible for sensing, effecting and basic processing functions. The control system level is responsible for executing plans, organizing the functional level's behavior and reporting of failures. The planning level uses abstract world models to generate plans and passes them to the control system level that executes the plans and reacts to environmental changes. This concept never checks where the input for the planning level comes from or if it is allowed to be executed.

Another approach by Bouguerra et al. [18] for execution monitoring that is based on semantic knowledge also monitors the execution of actions and gathers data from the environment to improve the monitoring but never checks where the command is coming from or if the execution is valid at this point in time.

III. Action Authorization

Unauthorized action executions is a serious issue for mobile robots as the robots might harm themselves, humans or production material. To overcome this issue, the approach presented in this paper provides a two-staged authorization mechanism that protects mobile robots from such unauthorized executions by using a ticketing system supported by secure elements. The first stage is necessary to check if incoming commands are allowed to be executed. These commands can consist of several actuator actions. Due to the structure of the mobile robot, it is possible that malicious commands are sent by adversaries. The plausibility checks prevent the execution of such malicious actions. The second stage is necessary as due to the mobile robot's structure it might be possible to send commands directly to the actuator. An adversary could, therefore, execute actions without the permission of the plausibility checker. To mitigate this problem, the second stage uses authorization tickets to check if the requested action is permitted. To perform action authorization, the following preconditions need to be fulfilled:

- Backend, CCU, and actuators are equipped with an SE to store key material and other secret information, and supports secured computation and verification of OTPs.
- The CCU's and actuator's SEs share a secret key.
- The CCU's and backend's SEs hold each other's public key.
- The CCU's and actuator's SEs share a counter value. The CCU's SE holds individual counter values for each actuator. The CCU's counter value is increased by a specific amount every time a ticket is generated. The actuator's counter value increases by the same value every time a received ticket was valid. In the case, that the actuator does not check the ticket, it commands the SE to increase the counter to preserve the counter synchronization even when the ticket is not checked after generation.

A. Plausibility Checks

The context-aware plausibility check is the first of the two stages in the authorization mechanism to check if the incoming command is allowed to be executed. In principle, the list of possible commands is stored in the CCU's SE, and the SE simply confirms or denies the execution of a command based on the list. Each entry in the list consists of the command and several preconditions that need to be met. For example, a very simple precondition for delivering a product is that the product was picked before. The preconditions might, of course, be much more complex and highly dependent on the context and executed actions. For the RoboCup Logistics League [3],

TABLE I: Examples for commands executed by mobile robots in the RoboCup Logistics League. Each command needs to fulfill a number of preconditions to be authorized for execution.

Command	Pre-conditions
Pickup Product	– Gripper empty – Laser scanner detected correct distance from machine – Camera detected product on machine – Robot is not driving – Addressed machine matches production steps
Drop off product	– Gripper filled – Laser scanner detected correct distance from machine – Camera detected drop off space – Robot is not driving – Addressed machine matches production steps
Drive to machine	– Machine necessary for current product – Global path plan complete – Obstacle avoidance active

the list of executable actions is rather limited. In TABLE I, an example of commands with preconditions based on the RoboCup Logistics League setting is given. The preconditions are derived from the robot state and also from the incoming commands.

The plausibility check simply returns 'True' if the command is plausible in this context, otherwise 'False'. In case of a 'False', the CCU's SE can validate the command's signature by using the backend's public key, the command, and signature as input. If the signature is valid, the command can still be executed; otherwise, an alert is sent to a server to log the failure. The validation of a signature in an SE is a rather expensive process. If the plausibility check returns 'True' the CCU decides randomly if the signature is still checked in order to detect small modifications. The plausibility check is a simple comparison and less expensive. This mechanism reduces the overhead compared to a signature check for each incoming command significantly and is still able to identify wrong or malicious commands if they do not fit the context.

B. Authorization Mechanism

The authorization mechanism uses internal states and the sensor data to check if the robot is in the correct context and generates an authorization ticket that can be used by the actuator to approve the execution. In detail, the authorization mechanism consists of the following eight steps:

① If the plausibility check returns a positive result or the validity of the incoming command's signature was confirmed, the CCU requests the sensor data from one or more sensors.
② The sensor(s) compute a signature over their values and return the sensor value and signature to the CCU.
③ The CCU verifies the signatures to check if the sensor values were not modified during transmission. If the signatures are valid, the CCU sends the command to the SE and instructs it to compute an authorization ticket.
④ The SE computes the authorization ticket from the command, secret key, and counter value. The generated ticket is returned to the CCU.
⑤ The authorization ticket and command are sent to the corresponding actuator's microcontroller for execution.
⑥ The authorization ticket and command are sent to the actuator's SE for verification.
⑦ If the ticket is valid for the given command, the SE returns a success message to the microcontroller.
⑧ In case of a success message from the SE, the microcontroller executes the requested action. In case of an error message from the SE, the micro controller logs this incident and sends an encrypted alert message directly to a specific server. It also informs the CCU about the failed verification.

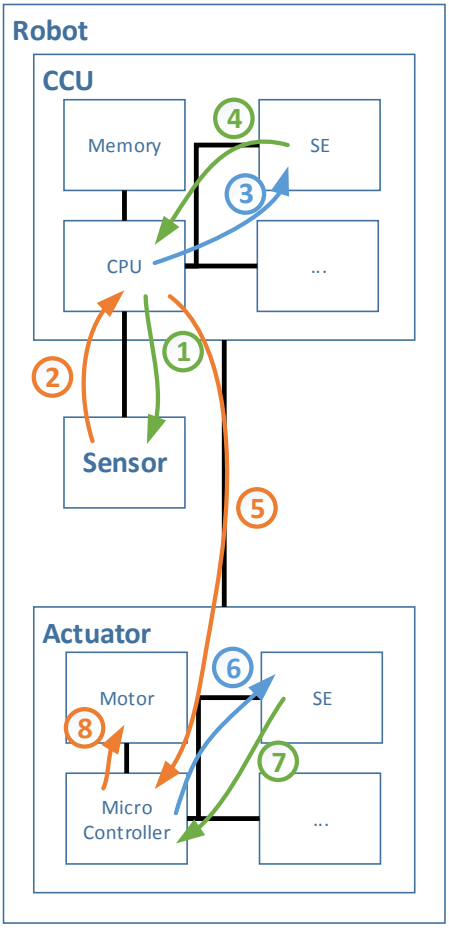

Fig. 2: The data flow when authorizing an action is shown in this figure. The CCU, senor and actuator are connected via Ethernet to a router that distributes the messages accordingly. The router is not depicted in this figure to simplify the illustration.

TABLE II: Generation and validation of authorization tickets

	CCU's SE		**Actuator's μC**		**Actuator's SE**
1 :	$otp_g \leftarrow \mathrm{HKDF}(K_S, cnt_g, cmd, pl)$				
2 :	$cnt_g \leftarrow cnt_g + 1$				
3 :		$\xrightarrow{otp_g, cmd, pl}$			
4 :			**if** $pl = True$		
5 :			$random = rand(0, 1)$		
6 :			**if** $random > limit$		
7 :				$\xrightarrow{otp_g, cmd, pl}$	
8 :			**else**		$otp_v \leftarrow \mathrm{HKDF}(K_S, cnt_v, cmd, pl)$
9 :			$executeAction(cmd)$		**if** $otp_v = otp_g$
					$cnt_v \leftarrow cnt_v + 1$
					$return$ success
10 :					$return$ failure

C. Authorization Ticket Generation and Validation

The generation and validation of authorization tickets are simple but efficient. TABLE II shows the steps from the generation of the OTP to the validation. The value for $limit$ is configurable and can be changed depending on the user's requirements. In step (1), the SE generates the password opt_g by using a hashed key-derivation function(HKDF) from the command cmd, a secret key K_S, the counter cnt_g and plausibility value pl. Afterward, the count cnt_g is incremented (2). The generated OTP, the command and plausibility value provided by the plausibility check is sent to the actuator's microcontroller (3). In step (4), the plausibility value is checked. If the value is True, a random number is generated (5). This number indicates if the generated OTP is checked even though the command was plausible. If the random number was smaller or equal than the given limit, the command is directly executed. In case of a random number larger than the limit, the OTP otp_g, command cmd and plausibility value pl are sent to the actuator's SE (7). In step (8), the verification OTP is generated by the actuator's SE with the secret key k_S, the command cmd, the counter cnt_v and the plausibility value pl. If the OTPs otp_g and otp_v match (9), a success message is returned to the actuator's microcontroller and the command is executed. Otherwise, an error message is returned, and the command execution is not permitted by the microcontroller.

IV. Threat Analysis

A threat analysis [19] will be performed as an evaluation and to highlight the security features of the proposed mechanism. The threat analysis describes the most important scenarios that constitute a threat and also discusses how the proposed mechanism solves the resulting threats.

The threat analysis lists all Entities (E), Assets (A) worth protecting, possible Threats (T), Assumptions (As) as well as Countermeasures (C) to mitigate threats and Residual Risks (R) that cannot be solved by the proposed mechanism.

The entities involved in this authorization mechanism and assumptions regarding their trustworthiness are:

(E1) Robot: (As1) trustworthy, but curious
(E2) Secure Element (SE): (As2) trustworthy
(E3) Adversary: (As3) not trustworthy

Before discussing the assets and threats, necessary assumptions need to be stated:

(As4) The shared secret key between the CCU's and actuator's SE are already stored on the SEs.
(As5) The counter values between CCU's and actuator's SE are synchronized.
(As6) The backend's public key is known to the CCU's SE.

The assets worth protecting can be derived from the entities, assumptions and mechanism steps:

(A1) Secret key: Loss of key would enable the generation of valid tickets by an adversary.
(A2) Counter: Loss of counter value might ease attacks that aim to reveal the secret key.
(A3) Actuator: Unauthorized action executions must be prevented to protect staff and production material from damage.

After listing the entities, assumptions and assets, the threats can be discussed:

(T1) Intentional or unintentional backdoors in the SE could ease attacks by adversaries.
Entities: (E1), (E2), Assets: (A1), (A2)
(C1) SEs are certified for specific security levels. The certification, e.g. by Common Criteria, proves that there are no backdoors.
(T2) Wrongly implemented or weak cryptographic algorithms on SE.
Entities: (E1), (E2), Assets: (A1), (A2)

(C2) Certification proves that strong and correct cryptographic algorithms are implemented on the SE.

(T3) Side-channel attacks or other physical attacks to reveal confidential data such as secret key or plausibility list. Entities: (E1), (E2),(E3) Assets: (A1), (A2)

(C3) SE is tamper-resistant and designed to make attacks infeasible.

(T4) Manipulations of command or injection of false command transmitted to the robot. Entities: (E1),(E2),(E3) Assets: (A3)

(C4) Plausibility check would reveal false command or manipulated commands as they do not fit the conditions.

(T5) Manipulations of authorization command or injection of false authorization commands sent to actuator. Entities: (E1), (E2),(E3) Assets: (A3)

(C5) Actuator's SE would detect false or manipulated authorization command as verification of the OTP fails.

(T6) Replay attacks on commands to the robot or authorization commands to the actuator. Entities: (E1), (E2),(E3) Assets: (A3)

(C6) Replay attacks capture previously sent traffic and sent them again later. The used counter value prevents such replay attacks as the counter changes the OTP for every computation even if all other parameters stay the same.

(T7) Denial-of-Service attacks on router. Entities: (E1), (E2),(E3) Assets: (A3)

(R7) These attacks would prevent the robot or actuator from performing any data exchange or operation. These attacks cannot be mitigated by any security measure as they simply try to shut a service down.

(T8) Manipulation of data from sensor or injection of false data. Entities: (E1), (E2),(E3) Assets: (A3)

(C8) Sensors compute signatures that protect the integrity of the sent data.

The threat analysis lists the most crucial threats identified by the authors, and is not exhaustive.

V. Conclusion and Future Work

This paper presented a two-staged authorization mechanism for actions on industrial mobile robots. The first stage is a plausibility check that ensures the correctness of incoming commands depending on the current context. The second stages authorizes the action execution by using authorization tickets. The whole process is supported by hardware security to support the protection of confidential data and cryptographic computations. The approach aims to reduce the overhead caused by the used hardware security but still prevent unauthorized action from execution. The conducted threat analysis highlights the security features and shows that only one residual risk in eight threats remains. A possibility for future work would be a more advanced plausibility algorithm that uses machine learning instead of the static list proposed in this paper. For our approach, we assume the sensors to be trustworthy. However, malicious modifications could disrupt the authorization process. Securing the sensors is out of scope for this paper, therefore, it is postponed to future work.

Acknowledgment

This work has been performed in the project Power Semiconductor and Electronics Manufacturing 4.0 - (Semi40), under grant agreement No 962466. The project is cofunded by grants from Austria, Germany, Italy, France, Portugal and - Electronic Component Systems for European Leadership Joint Undertaking (ECSEL JU).

References

[1] T. Bauernhansl, M. Ten Hompel, and B. Vogel-Heuser, *Industrie 4.0 in Produktion, Automatisierung und Logistik: Anwendung· Technologien· Migration*. Springer-Verlag, 2014.

[2] D. Zuehlke, "SmartFactory–Towards a Factory-of-Things," *Annual Reviews in Control*, vol. 34, no. 1, pp. 129–138, 2010.

[3] T. Niemueller, D. Ewert, S. Reuter, A. Ferrein, S. Jeschke, and G. Lakemeyer, "RoboCup Logistics League Sponsored by Festo: A Competitive Factory Automation Testbed," in *Automation, Communication and Cybernetics in Science and Engineering 2015/2016*. Springer International Publishing, 2016, pp. 605–618.

[4] L. A. Grieco, A. Rizzo, S. Colucci, S. Sicari, G. Piro, D. Di Paola, and G. Boggia, "IoT-Aided Robotics Applications: Technological Implications, Target Domains and Open Issues," *Computer Communications*, vol. 54, pp. 32–47, 2014.

[5] M. B. Line, O. Nordland, L. Røstad, and I. A. Tøndel, "Safety vs Security?" in *Proc. 8th International Conference on Probabilistic Safety Assessment and Management (PSAM 2006), New Orleans, USA*, 2006.

[6] B. Y. Fraser, "Site Security Handbook," 1997, RFC2196.

[7] "ISO/IEC 11889-1 Trusted platform module library - Part 1: Architecture," 8 2015.

[8] L. Lamport, "Password Authentication with Insecure Communication," *Communications of the ACM*, vol. 24, no. 11, pp. 770–772, 1981.

[9] N. Haller, "The S/KEY One-Time Password System," 1995, RFC 1760.

[10] D. M'Raihi, M. Bellare, F. Hoornaert, D. Naccache, and O. Ranen, "HOTP: An HMAC-Based One-Time Password Algorithm," Tech. Rep., 2005.

[11] R. K. Thomas and R. S. Sandhu, "Task-Based Authorization Controls (TBAC): A Family of Models for Active and Enterprise-Oriented Authorization Management," in *Database Security XI*. Springer, 1998, pp. 166–181.

[12] M. Islam, A.-e. Taha, and S. Akl, "A Survey of Access Management Techniques in Machine Type Communications," *IEEE communications Magazine*, vol. 52, no. 4, pp. 74–81, 2014.

[13] W. Sun, S. Yu, W. Lou, Y. T. Hou, and H. Li, "Protecting Your Right: Attribute-Based Keyword Search with Fine-Grained Owner-Enforced Search Authorization in the Cloud," in *INFOCOM, 2014 Proceedings IEEE*. IEEE, 2014, pp. 226–234.

[14] S. Cirani, M. Picone, P. Gonizzi, L. Veltri, and G. Ferrari, "IoT-OAS: An OAuth-Based Authorization Service Architecture for Secure Services in IoT Scenarios," *IEEE sensors journal*, vol. 15, no. 2, pp. 1224–1234, 2015.

[15] J. Gonçalves, J. Lima, H. Oliveira, and P. Costa, "Sensor and Actuator Modeling of a Realistic Wheeled Mobile Robot Simulator," in *IEEE International Conference on Emerging Technologies and Factory Automation, 2008. ETFA 2008.* IEEE, 2008, pp. 980–985.

[16] A. Popovici, A. Frei, and G. Alonso, "A Proactive Middleware Platform for Mobile Computing," in *Proceedings of the ACM/IFIP/USENIX 2003 International Conference on Middleware*. Springer-Verlag New York, Inc., 2003, pp. 455–473.

[17] F. R. Noreils and R. G. Chatila, "Plan Execution Monitoring and Control Architecture for Mobile Robots," *IEEE transactions on robotics and automation*, vol. 11, no. 2, pp. 255–266, 1995.

[18] A. Bouguerra, L. Karlsson, and A. Saffiotti, "Semantic Knowledge-Based Execution Monitoring for Mobile Robots," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 3693–3698.

[19] S. Myagmar, A. J. Lee, and W. Yurcik, "Threat modeling as a basis for security requirements," in *Symposium on requirements engineering for information security (SREIS)*, vol. 2005. Citeseer, 2005, pp. 1–8.