2021 IEEE International Conference on Communications Workshops (ICC Workshops) | 978-1-7281-9441-7/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICCWorkshops50388.2021.9473818

On the Security of Cyber-Physical Robotic Systems Using Dynamic Modeling and Simulation

Joshua B. Hector[‡], Pantelis Katsiaris[†], Nicole E. Carey[†], Nick Cote[†] and Danda B. Rawat[‡]
[‡]Data Science and Cybersecurity Center (DSC2), Howard University, Washington DC, 20059, USA
[‡]E-mail: *joshua.hector@bison.howard.edu*, *danda.rawat@howard.edu*
[†]Autodesk Research, Pier 9 San Francisco California, 94111, USA
[†]Email: {*pantelis.katsiaris*, *nic.carey*, *nick.cote*}*@autodesk.com*

***Abstract*—This paper addresses the lack of cyber-physical security in robotics systems by implementing a new layer of security that monitors the torque values that the robot is exerting, while in motion, for possible anomalies. A simulated compliant robot (the Franka Panda) that used the Unity physics engine was adapted to create and test this anomaly-detection security layer. Anomalies observed in the torque signals are reported back to the operators and a resilience measure is enacted to create a layer of cyber-physical security for robotics systems. We provide a simulation of a possible attack and demonstrate the response of the system.**

***Index Terms*—Cyber physical systems, robotics, resilience, robotics security, simulating and modeling based security.**

I. Introduction

Cyber-physical security is of fundamental importance to robotic systems. Without it, cyber-attacks can cause damage or lead to the release of proprietary information. Attacks centered around hacking the movement of robots in a manufacturing setting can lead to altered and possibly dangerous products or a loss of production. Cyber-physical security protects against possible threats to people and their information, while also determining the next steps in the event that a security breach occurs [1].

The problem of cyber-physical security for robots is attracting more attention in the community. Previous work done in this area includes fault detection, isolation and modeling of robotic systems by applying dynamic models [1]–[5], [16]. Robots can also be designed to implement security from outside threats [6]–[12]. Security systems have been built and tested using robots to secure locations in buildings, and to provide additional protection to areas with a security architecture already in place. All these contributions are aimed at securing physical spaces with the use of robots. The approach of this paper can be considered a deeper look into the security of the robot itself as we address the importance of securing the very machinery meant to make human lives better.

The types of security threats that are being tackled in this paper are ones that inhibit or alter the motion of the robot. Cyber-physical attacks that alter control parameters, tamper with calibration, change production logic or alter personnel signals are examples of issues to be addressed. These kinds of attacks are easily seen in a production setting where robots oversee the manufacturing of products and require maintenance by human operators.

Currently there is a lack of security in robotic systems at the operations or GUI application level, however, a low-level security implementation is often included by the manufacturer to keep the robot secure [13]. This security implementation is heavily impacted as soon as the operators try to contextualize the robot for their needs, and there are no steps in place to secure a robot once it is under attack. The difficulty of implementing security on robots is that robotic programming is often highly specialized. This makes creating something tangible such as anti-virus software or a firewall difficult for all robots. A course of action to resolve this issue is to create a general approach that can be added to robots at the discretion of the creators.

The implementation of this security feature will be up to the robot operators to enact themselves. The middle-ware and low level portions of any robot are handled by the manufacturer, and because of this any security that is implemented on the robot prior to the operator's contextual adjustment is the responsibility of the manufacturer [13]. In order to add security to robotic systems this paper proposes a feature that can be added at the top level of a robot control stack.

This security feature monitors the torque values that the robot is exerting while in motion, and if the torque is outside of the expectations for that robot a resilience measure begins to move the robot to a safe position. This implementation will be called the security trigger throughout this paper. Applications of the security trigger may include settings where the precision and accuracy of the robot's movements are of vital importance to the success of a task, or any instance of robotic movement. Examples of these tasks would be surgical work, industrial manufacturing, or robotic security guards.

To test the implementation of this security feature, we used a simulation of a commercial robot platform (the Franka Panda code found here [14]) in the Unity 2020.14b physics and simulation environment [15]. This Unity release provides a fast (near real-time) approximation of fully articulated connected bodies, making it an excellent platform to use for dynamics dependent robotic experimentation. The security trigger proposed here is a general concept that is meant to be used across many different types of robots and dynamical systems. It adds

This project is funded in part by Autodesk Inc. and by the US NSF under grants CNS/SaTC 2039583 and 1828811 at Howard University.

TABLE I: This table compares the approach of this paper to a couple of the approaches already present in the field. The security protocol ITP can be found in [18] and the detection and mitigation approach is addressed in [17].

Approach	Problem Addressed	Key Feature	Major Differences
Security Trigger (Our Approach)	Securing robots from cyber-physical threats.	Dynamic Motion detection by analyzing the torque placed on each joint of a robot to ensure robot trajectory integrity.	This implementation is a solution for after an attack has been realized. It tackles identifying that a security threat is currently present and enacting resilience.
Detection and Mitigation [18]	Designing security for surgical robots.	Dynamic motion detection using a model to analyze the types of security attacks and their impact on surgical robots.	This model shows the possible types of security attacks on surgical robots and shows what the implications of these attacks are.
Security Protocol ITP [17]	Designing security for telesurgery robots and their controllers.	A list of protocols and procedures suggested to create security for telesurgery robots.	This is a security measure to avoid a cyber attack from happening in the first place. This outlines how a system should be set up to avoid as many attacks as possible.

security by detecting anomalies in the movement of machinery.

This paper is organized as follows. First Sec. II shares some of the work related to securing robotic systems and the importance of doing so. Next a complete rundown of the methods used to create the security feature is given in Sec. III. Sec. IV talks about the application of the security feature to our case study Panda robot. The results and algorithmic testing is provided in Sec. V, and finally we conclude the paper in Sec. VI.

II. Related Work

This section provides a literature review related this paper.

A. Dynamic Model-based Detection

Homa Almezadeh and *et. al.* demonstrated targeted cyber-physical attacks on teleoperated surgical robots in their work in [17]. They work with teleoperated surgical robots using dynamic model-based detection and mitigation. In their work, the authors described the nature of cyber-attacks that are typical for surgical robots and the consequences that come from unwanted access to critical robotic systems. They presented a defense model for detecting and mitigating possible attacks before the threat could compromise the integrity of the system. This is vital as a system malfunction during a surgery could result in injury or death of the patient. This work stresses the importance of avoiding a denial-of-service attack in our systems. Here a complete shutdown of the robotic system while operation is going on can cause human harm.

B. Cyber-physical Systems Security

Gregory S. Lee and Bhavani Thuraisingham created the Secure Interoperable Telesurgery Protocol (ITP) in their research work in [18]. The secure ITP defines a security framework to structure the communication between telesurgery robots and the controllers for them. This work was solely for telesurgery robots and to ensure that the strict guidelines for them are met. They stressed the importance of security for their robotics systems while also recognizing the issue that robot manufacturing does not have standardization that allows for a single software suite to be created. This framework approach is very much like the suggestion in this paper. Since the standardization of robots is not existent, these frameworks can provide security engineers with the tools they will need to secure robotic systems.

C. Assessment of ROS

Sergio Sandoval and et. al. tackle the issue of the current robot operating system that is widely used named ROS or the Robot Operating System [19]. The preliminary design of ROS did not have any security features to protect against cyber attacks. The authors created a new form of operating system (known as middleware in robots) called ROS2. This new system is designed on standards that are provided by the Data Distribution Service (DDS), and was designed to provide security to the areas where the original ROS failed.

D. Argument for Security in Robots

Nico Hochgeschwender and et. al. make an argument for the importance of security in autonomous robots [20]. They write about the current applications of robots in a social aspect such as being a tour guide, a receptionist or an office assistant and how these robots with close proximity to humans need to be secure for reasons of public safety. The authors also push for a security engineer to be present at all development stages of robots as the cost of implementing security after the product has been built is high and introduces complications.

E. Security for Robotic Platforms

Dr. Akashdeep Bhardwaj and et. al. study the impact of lacking robotics security as the world moves towards automation in the industry domains of manufacturing, agriculture, logistics, healthcare, and transportation [21]. They address the parallels between hardware, networking, applications and

platforms in relationship to robots to make justification for why robotic systems need security that is on par with larger scaled software systems that we use today.

III. Methodology

A. The Security Trigger

The security trigger is an algorithm that runs a comparison between expected motion and real motion while the robot is moving. To track this motion the algorithm was created to calculate the torque at each joint and compare it to the expected torque for that joint. The expected torque is calculated using a dynamic model of the robot. This model allows us to use the information we know about the robot and its trajectory such as, force due to gravity, the Coriolis forces, joint positions and so on to create a range of expected torque values for our robot. When the robot is in motion the torque that is observed from the robot should be within the expected range of values. Any deviation from this range prompts an alert that abnormal activity is present in the robot and another algorithm forces the robot into a safety position.

Algorithm 1 Security Trigger - Generalized

```
procedure DETECT THREAT AND RETURN ALERT
    while robotExpectedMotion == True do
        expectedTorque = value calc from eq(1)
        observedTorque = torque from robot motion
        for revolute joint in robot do
            jointError = (t_observed − t_expected)/t_expected
            if jointError >= threshold(.05) then
                // Print the error message and alert the system
                Debug.log(alert)
                robotExpectedMotion = False
```

B. Dynamic Model

To calculate the torques for a robot the symbolic inertia matrix $M(q)$ must be rearranged into a vector form by exploiting its symmetry. The robot has n joints, so we obtain a vector $m(q) \in \mathbb{R}^m$, with $m = n(n+1)$ components, containing all the lower triangular elements of $M(q)$ [22].

The simulated dynamic model of robots with elastic joints can be derived using the procedure in [23] by separating the torques of the motors from the link-side torques. Using these methods, a general model for a robot of n degrees of freedom can be written as

$$M(q)\ddot{q} + S(q,\dot{q})\dot{q} + g(q) = \tau, \tag{1}$$

where q, $\dot{q}$, $\ddot{q} \in \mathbb{R}^n$ are, respectively, the joint positions, velocities, and accelerations, $M(q) \in \mathbb{R}^{n\times n}$ is the inertia matrix, $g(q) \epsilon \mathbb{R}^n$ is the gravity vector and $S(q,\dot{q})\dot{q} = c(q,\dot{q}) \in \mathbb{R}^n$ is the vector of the Coriolis and centrifugal forces [22]. The motor torque from the inverse dynamics that are calculated produces the value that will be used to compare the expectations of the movement to the reality.

C. How to use the Dynamic Model for Comparison

The algorithm does a comparison of the observed torque and the expected torque by taking the difference of the two quantities and measuring it with respect to a margin of error that is deemed acceptable for that robot.

$$|\tau_{expected} - \tau_{real}| \div \tau_{expected} > error_{acceptable} \tag{2}$$

If the above equation becomes true while the robot is in motion, the security trigger alerts the users of abnormal activity and starts the resilience measure. An acceptable margin of error needs to be determined by the users for their specific robot, however, for the purpose of testing the functionality of the algorithm in this paper $\tau_{expected} \pm 5\%$ or $\tau_{expected} = \pm 5\%$ returns a security check.

D. Resilience

To implement resilience into a robot, the control modality of it can be switched when abnormal movement is detected. The robot can be controlled through different modalities depending on the user requirements. Torque mode, position mode and velocity mode are some common examples. Torque mode works by sending a torque vector to the robot motors, position mode by giving the joints a desired position, and velocity mode by sending a velocity to a joint [22]. In this paper, position was the input which was translated to torque through the controller. This allowed us to determine the robot's location, and to simulate the presence of an attack.

In our implementation when a threat is detected the control mode switches from position control to torque control in order to move the robot safely to its home position. By knowing the current and safe position of the robot a joint space trajectory can be derived. This joint space trajectory can be applied to eq. 1, and the set of torques that will drive the system to its desired configuration is derived. The safety position for the robot allows the users to decide next steps to secure it. This safety measure allows the users to still access the robot without having to shut down the entire system. In order to enable this switch in modes, a flag was set up to enact the switch of modalities whenever unexpected values were detected by the security trigger. This flag switch initializes the robot's torque mode to take control of how the robot moves and forces it into a safe position. This switch between modes can be a change of whatever modes the operators deem necessary for their specific robot. Algorithm 2 shows the logic behind this method.

IV. The Panda Robotic Environment in Unity

A compliant PANDA robot with 7 degrees of freedom that was simulated in Unity was used to conduct experimentation. The torque detection and resilience building was implemented in the top level of the robot control. This control for our simulated environment was a C# script that initialized the impedance-driven robot. Figure 1 depicts the Panda robot that was used for this study.

Algorithm 2 Modality Switch - Boolean Flag Example

procedure SWITCH MODES WHEN THREAT DETECTED
  while robotExpectedMotion == True **do**
    $positionMode = True$
    $torqueMode = False$
    $jointError = (t_{observed} - t_{expected})/t_{expected}$
    if $jointError >= threshold(.05)$ **then**
      //Alert users and switch modes
      Debug.log(alert)
      $positionMode = False$
      $torqueMode = True$
      robotExpectedMotion = False

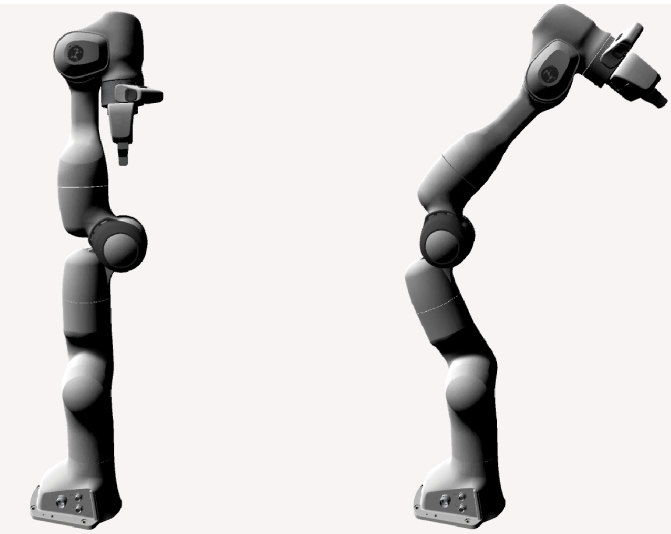

Fig. 1: The Panda robot that was simulated in Unity [15]. The left side of the image shows the Panda before the simulation has begun and the right-side image shows the target position or end location once the simulation has been run.

To test the functionality of the algorithms mentioned before the expected torque was calculated using the joint components that Unity provided for our simulation. This includes the velocity, joint position, forces and so on. To simulate an attack on the robot the observed torque was an alteration of the expected torque equation by changing the target position. This represents an attack that has altered the target position but leaves the controller settings and other H/W elements intact. The equations below depict the calculations done for expected and observed torque:

$$\tau_{expected} = (K\Delta q_{target}) + (D\Delta\dot{q}) \tag{3}$$

$$\tau_{observed} = (K\Delta q_{targetAltered}) + (D\Delta\dot{q}) \tag{4}$$

Where K is the joint stiffness, D is the joint damping, Δq_{target} and $\Delta\dot{q}$ are the target position error and calculated velocity error.

These torque values were compared to each other and evaluated for the Panda robot. This evaluation was referenced earlier in this paper under algorithm 1. After the detection was made the Panda robot used a Boolean flag, (as shown in algorithm 2), to switch from position mode to torque mode. Securing the robot in our case meant increasing the joint stiffness and joint damping while reducing the joint velocity to keep the robot locked in one position but still running.

V. PERFORMANCE EVALUATION AND DISCUSSION

A. Testing the Alert

In order to corroborate the analysis above, we performed the evaluation of the cyber-physical security for the robotic system. Testing included the security trigger's reaction when the observed torque was outside and inside the range of the expected torque. The implementation for our robot alerted the users through Unity's console log. The console log kept track of the observed torque in the system and printed the error messages. Figure 3 shows the output to the console when the robot was under attack and Figure 2 shows the output that was displayed when the robot was operating within expectations.

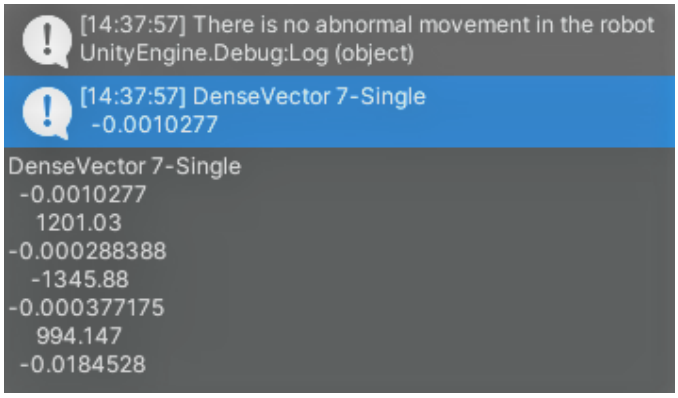

Fig. 2: This is an image of the Unity console while the robot is in motion. To keep track of the motion visually for the users, the observed torque was printed to the console log. A message denoting if there is abnormal behavior or not was also included in this log.

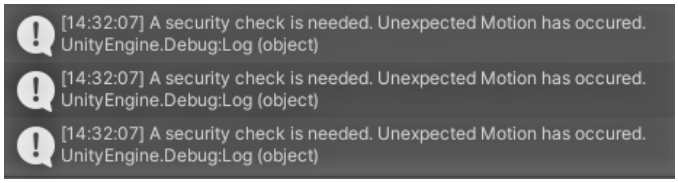

Fig. 3: This is the console log from Unity when an unexpected torque value has been detected in the robot. The program logs 3 different instances of this because the irregular torque was found in the 3 joints that are moving in the Unity simulation.

Figures 5 and 6 depict the comparisons of the expected torques to the observed torques of the Panda robot moving over a period. Each graphical figure depicts a moving joint in

the simulation environment and shows the differences between an acceptable deviation from movement and an unacceptable one.

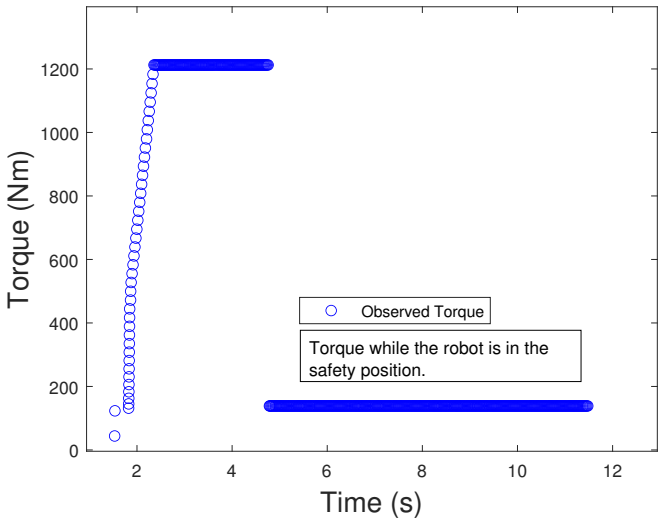

Fig. 4: This is a graph of Torque vs. Time when the resilience measure is active in our robot. The torque value is kept constant because the robot is locked into its safety position after the modality switch occurs.

B. Testing Resilience

To display the functionality of the resilience algorithm a similar graph of Torque vs Time was produced. This graph shows the robot initializing and moving to its target position, running into an unexpected torque, and then moving to its rest position via the torque mode control that was previously discussed in this paper.

C. Discussion

The security trigger is a feature meant to assist in the recovery of a robot that has already been attacked. It operates as a mechanism to give users the ability to secure the robot without stopping it to ensure that no further damage or cost is incurred. A complete stop in operations would make a denial-of-service attack completely successful. The proposed method mitigates the complete success of a denial-of-service attack by allowing the robot to remain functional while it is under the attack. At this point in time mitigation is the only thing that can be done if the robot controller is attacked. This is because the robot controller is the only way that the operators have to communicate with the robot.

However, the security trigger as a stand-alone method of security is not enough. This alert and resilience feature is best suited to accompany a security protocol for robots once an attack has been realized. Our team will be continuing progress on the expansion of the security project to incorporate the security trigger as just one piece of a protocol that can be enacted to secure robots of many different types. This expansion of work could include taking more information from the sensing apparatus of robots to categorize the difference between possible faults and cyber-physical threats. Information such as the temperature and vibrations would be helpful in this endeavor.

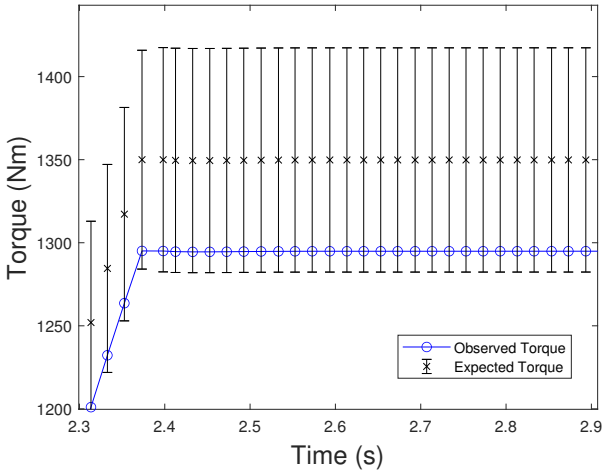

Fig. 5: This is a graph of Torque vs. Time of one joint as the robot is moving to its target position. This shows a case of the observed torque within the acceptable error range. The error bars denote $\pm 5\%$ the value of the expected torque at that time step. This is a small time slice of the overall motion of the robot.

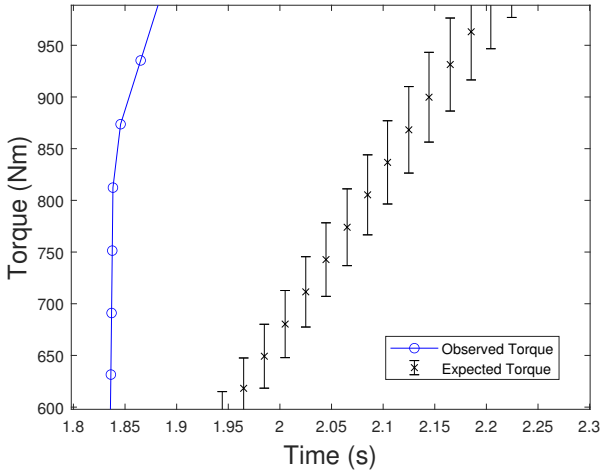

Fig. 6: This is a graph of Torque vs. Time of one joint as the robot is moving to its target position. This figure shows the robot traveling outside of the acceptable range of torque. The error bars denote $\pm 5\%$ the value of the expected torque at that time step. This is a small time slice of the overall motion of the robot.

VI. Conclusion

Cyber security is an area that needs more attention in the robotics field. Especially as the automation of human life progresses, cyber-physical threats and attacks could target the very machines meant to make our lives easier. The security trigger proposed in this paper is a contingency plan to a denial-of-service attack that targets the motion of robots. The alert and resilience portions of the security trigger were tested and found to be operational for the case study Panda robot that was simulated in Unity.

Security as it relates to robots will have to be a more custom process for each establishment and each type of robot. However, techniques like the security trigger that can be implemented across a wide range of machines and products could be what is needed to expand the toolbox of security engineers.

Acknowledgments

This project is funded in part by the US NSF under grants CNS/SaTC 2039583, CMMI 2036359 and 1828811, and Autodesk Inc. at Howard University. However, any opinion, finding, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of these funding agencies.

References

[1] Geers, Kenneth. Strategic cyber security. Kenneth Geers, 2011.

[2] S. Haddadin, A. De Luca and A. Albu-Schäffer, "Robot Collisions: A Survey on Detection, Isolation, and Identification," in IEEE Transactions on Robotics, vol. 33, no. 6, pp. 1292-1312, Dec. 2017.

[3] A. Albu-Schäffer, C. Ott and G. Hirzinger, "A unified passivity-based control framework for position torque and impedance control of flexible joint robots", Int. J. Robot. Res., vol. 26, pp. 23-39, 2007.

[4] S. Nicosia, P. Valigi and L. Zaccarian, "Dynamic modelling of a two link flexible robot and experimental validation," Proc. of IEEE Intl Conf. on Robotics and Automation, MN, USA, pp. 1953-1958 vol.3, 1996.

[5] M. W. Spong, "Modeling and control of elastic joint robots", ASME J. Dyn. Syst. Meas. Control, vol. 109, no. 4, pp. 310-319, 1987.

[6] T. S. Li et al., "Design and implementation of sensor fusion based behavior strategies for a surveillance and security robot team," 2008 SICE Annual Conference, Tokyo, 2008, pp. 2968-2972.

[7] P. K. Singh, R. Singh, S. Nandi, K. Z. Ghafoor, D. B. Rawat and S. Nandi, "An efficient blockchain-based approach for cooperative decision making in swarm robotics." Internet Technology Letters 3.1 (2020): e140.

[8] F. O. Olowononi, D. B. Rawat, and C. Liu. "Resilient Machine Learning for Networked Cyber Physical Systems: A Survey for Machine Learning Security to Securing Machine Learning for CPS." IEEE Communications Surveys & Tutorials, Vol. 23, No. 1, pp.524 - 552, 2020.

[9] R. Doku, et al. "LightChain: On the lightweight blockchain for the Internet-of-Things." 2019 IEEE International Conference on Smart Computing (SMARTCOMP). IEEE, 2019.

[10] A. Javaid, et. al. "Cyber security threat analysis and modeling of an unmanned aerial vehicle system," in 2012 IEEE Conference on Technologies for Homeland Security (HST), 585–590, 2012.

[11] J. Han et. al., "Collective robot behavior controller for a security system using open SW platform for a robotic services," 2011 Intl Conf. on Control, Automation and Systems, Gyeonggi-do, 2011, pp. 1402-1404.

[12] J. Park, T. Uhm, G. Bae and Y. Choi, "Stability Evaluation of Outdoor Unmanned Security Robot in Terrain Information," 2018 18th International Conference on Control, Automation and Systems (ICCAS), Daegwallyeong, 2018, pp. 955-957.

[13] Brogårdh, Torgny. "Present and future robot control development—An industrial perspective." Annual Reviews in Control 31.1 (2007): 69-79.

[14] Nicole E Carey and Joshua Hector, Franka Panda Unity GitHub Repository: $github.com/joshuahector86/FrankaPanda_UnityAdditions$

[15] Unity Simulation Tool, website: https://unity.com/products/unity-simulation

[16] D. B. Rawat, J. JPC Rodrigues and I. Stojmenovic, Cyber-physical systems: from theory to practice. CRC Press, 2015.

[17] H. Alemzadeh et al. "Targeted attacks on teleoperated surgical robots: Dynamic model-based detection and mitigation." Proc. of 46th IEEE/IFIP Intl Conf. on Dependable Systems and Networks, 2016.

[18] Gregory S. Lee, Bhavani Thuraisingham, "Cyberphysical systems security applied to telesurgical robotics, Computer Standards & Interfaces," Volume 34, Issue 1, 2012, Pages 225-229, ISSN 0920-5489.

[19] S. Sandoval and P. Thulasiraman. "Cyber Security Assessment of the Robot Operating System 2 for Aerial Networks". In: 2019 IEEE International Systems Conference (SysCon). 2019, pp. 1–8.

[20] N. Hochgeschwender, G. Cornelius, and H. Voos. "Arguing Security of Autonomous Robots". In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2019, pp. 7791–7797.

[21] Akashdeep Bhardwaj, Vinay Avasthi, and Sam Goundar. "Cyber security attacks on robotic platforms". In: Network Security 2019 (Oct. 2019), pp. 13–19.

docs.unity3d.com/2020.1/Documentation/ScriptReference/ArticulationBody.html

[22] Claudio Gaz et al., Dynamic Identification of the Franka Emika Panda Robot With Retrieval of Feasible Parameters Using Penalty-Based Optimization. IEEE Rob. & Automation Let., 2019, 4 (4), pp.4147-4154

[23] M. Spong, "Modeling and control of elastic joint robots," ASME J. Dyn. Syst. Meas. Control, vol. 109, no. 4, pp. 310–319, 1987.