ntry]none/global pages5 pages4
pages6 pages12
pages-1 pages6 pages17
pages7 pages13 pages9
pages14 pages4 pages10 pages13
pages10 pages8 pages6 pages8
pages10 pages15 pages15
pages6 pages7

The Wolf in Sheep's Clothing Subversive Adversarial Agent Identification

Jean P. Castillo*, Matthew W. Swartwout†, Russel C. Jackson‡ and Murat C. Cavusoglu§
Department of Electrical Engineering and Computer Science, Case Western Resere University
Cleveland, OH 44106
Email: *jpc87@case.edu, †mws85@case.edu, ‡rcj33@case.edu, §mcc14@case.edu

***Abstract*—The abstract goes here.**

I. Introduction

Security has been a hot topic in the world of software with millions of dollars being poured into the industry year to year. Software is now starting to become more and more responsible of commercial robots, bringing forth the question regarding the security of cyber-physical systems. We see these cyber systems taking off in self-driving vehicles, drones, and even rescue robots such as the firefighting robots in []. There have been forward thinking papers such as [] implicating the repercussions of autonomous drones in war, a task that has particular potential for infiltration of adversarial agents.

Chain of trusts has been established in the previous works to combat the issue of impersonation, many providing approaches to protecting against malicious alterations to the system. This paper provides three elements of a chain of trust expands on the trust acquired by a robot's individual assessment of new and existing agents by means of communication, state detection and distributed state estimation. The root of trust however relies in a HSE unit which fingerprints specific physical properties and uniquely identifies an agent.

II. Related Work

Chain of trusts, as pertaining to cyber systems, have been introduced in works such as [] where a DNA clone resistant approach is taken throughout; a Physical Unclonable Function (PUF) was introduced which obtains physical differences between other electronic devices. Similar to this work the extended chain of trust this paper is part of a larger work which incorporates a Hardware Secure Enclosure (HSE) which is used to provide a unique hardware signature used to verify other parts of the system much like [].

A. State Estimation

State estimation is a fundamental problem for robotics. Without knowing its location, a robot cannot be very useful. One common technique for robot localization is to use a Kalman Filter (KF). The KF is one of the most studied and most heavily utilized Bayesian filter [, pp. 39-81]. At its core, a KF represents the state of a linear system as a multivariate normal distribution. By representing the state of the system as a normal distribution the filter can compactly represent the degree of belief of any possible state of the system.

The application of the KF to the task of robot localization has been studied intently In [], the KF is explained to be so popular for three main reasons. First, it is optimal under certain assumptions, second that it is recursive which is memory efficient, and finally that it relies only on having knowledge of noisy sensor data and not being able to measure certain state variables directly.

However, the KF is best used for a linear system. Thus, plain KFs are not generally used for localization. The two most popular options for real-world localization are the Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF) which both extend the KF to better handle non-linearity. The UKF was Julier and Uhlmann, and presented in []. The UKF has two primary advantages over the EKF. The first is that it is more accurate than the EKF and this accuracy difference increases as the system becomes less linear. And second that it is computationally less complex than the EKF.

A UKF was chosen to be implemented for this paper, however many of the statements here regarding the filters can be applied to EKFs as well. Our chosen software package also makes it trivially easy to switch between a UKF and UKF. For the remainder of this article, we will use the generic term KFs, demonstrating the fact that these statements apply to both UKFs and EKFs.

B. Secure State Estimation

There are three core goals for the security of any cyber-physical system: integrity, availability, and confidentiality []. Those can be thought of as three simple questions: "Can I trust the data?", "Do I have access to the data when I need it?", and "Is the data hidden from those who shouldn't have it?". Secure state estimation requires a focus on all three of these pieces.

1) Integrity: The accuracy of a KF when the sensors are functioning property is not in question, so we must look at when the sensors are not functioning properly. This leads to two possible scenarios, sensor failure and false data injection.

In a sensor failure scenario one or more of the sensors will stop providing all feedback. Luckily, this is something that KFs can handle without problem. The KF is always using all available information to generate its state estimate, and if one sensor fails the only result is that the accuracy of the filter will decrease. Obviously, if 100% of sensors fail, the filter will become non-functional, but as long as at least one sensor remains operational the filter will not have difficulties. It is also worth nothing that, due to the design of the filter, the KF does not have issues with sensors going in and out of a failure state. When information is available it is used, and when it is not available the sensor is ignored.

Knowing that sensor failure is not a huge risk for the KF, sensor interference is a much more complex problem. If an attacker is somehow able to inject false data into the filter the pose estimate can be skewed. And this is a very real problem. The number of ways that false data could be added to a filter are too numerous to count, ranging from physically blocking a lidar sensor to hijacking the network packets of a wirelessly communicating sensor.

[] and [] both demonstrate that KFs are susceptible to false data injection attacks, and that even with failure detectors a clever adversary can craft their attack to bypass these detectors. [] and [] both attack this problem by adding extra steps into their KF algorithm. Both successfully show that the filter output can be shielded from the effects of the attack.

2) Availability: In a distributed system, Denial of Service (DoS) attacks become a real threat. While there are many ways to execute such attacks and also many ways to defeat them ([], []) such systems are outside the scope of this TurtleBot. Rather than showing that the system can defeat DoS attacks, we will show that it can function sufficiently in the event that there is a network interruption and inter-robot communication is prevented.

3) Confidentiality: Secure Ad-Hoc Channel Communication: There are various works that cover a communications for adhoc networks. [] uses an extension of the ElGamal algorithm in order to limit the access of agents in the network. In the event that an agent is compromised the information within the decrypted message remains protected through steganography, a method used for hiding information within a text []. The implementation uses direct agent to agent communication and thus no hierarchy is used; no encrypted broadcast messages are sent and it seems unnecessary to encrypt the information further as it would lead to potential DOS attacks.

Another protocol introduced in [] provides forward secrecy through registration, authentication and password changing involving communication between user, sensor and gateway nodes. A long term secret is used for the session and a change in password occurs once the connection is established; however, in a moving environment there may instances when only two nodes will be able to communicate at one time.

In the field of automobiles [] developed a protocol named PBA that is well guarded against high traffic and lossy wireless networks. Work in [] addresses DOS attack resistance and stores shortened MAC address signatures for memory consumption and rapid verification. [] takes a private public key pair approach which may be prone to further attacks as the base station could update their keys. Connectivity to the central station after the release is not taken to be necessary in this paper's approach.

C. Distributed State Estimation

As wireless networks become more and more omnipresent in our world, and given the increasing number and decreasing costs of mobile robots, the concept of a distributed state estimation system is very desirable. Given the rapid rise of these technologies though, the amount of research done into distributed systems is quite small compared to that for single robots. This was true years ago ([]), and is still true today.

Traditionally, in order to increase the accuracy of state estimation, more sensors are added to a robot. However, increasing the number of sensors on a robot will increase the cost and complexity. And, where a robot operating by itself might need n sensors for a sufficiently accurate state estimate, when operating in an environment surrounded by m other robots that can share sensor data there is the possibility of needing far fewer sensors on each individual robot.

Most research into such distributed systems relies on the concept of cooperative positioning. That is, the robots coordinate their motions in order to use their sensors to determine the state of the other robots. Most often this involves one robot acting as a stationary landmark while the other robot moves. [], [], [], and [] all explore the idea of cooperative positioning. This is a valuable and effective concept when working with a team of coordinated robots. However many systems will not allow for such close cooperation. For example, autonomous cars on a highway will not be able to start and stop for each other and still get to their destination on time. And in rush hour in a city the processing power needed to coordinate a plan for the thousands of cars on the road would be incredible.

Other work into distributed systems still makes the assumption of a cooperative team. [] and [] both demonstrate a system where a KF is implemented that estimates the state of multiple robots in a system. However these states are computed relative to each other, considering the network of robots to be essentially one large robot with different parts. Continuing the highway example, cars will be moving in and out of communication with each other as some enter the highway and others exit. Thus the group of robots will be constantly changing and evolving, a challenge not tackled by these works.

The previously mentioned works are all examples of a decentralized, rather than distributed KF problem. [] makes clear the distinction between these two, saying:

> ”...[decentralized] methods require a complete network with all-to-all links. This solution is not scalable for large-scale sensor networks... Thus, decentralized Kalman filtering and distributed Kalman filtering are two separate problems. In the latter one, each node only is allowed to communicate with its nieghobrs on a graph G that is connected but rather sparse.”

[] presents a solution to the distributed problem using consensus filters and multiple KFs. However this work still requires adaptation of the KFs and extra computation, which is something we are trying to avoid.

D. State Detection

In order to obtain behavior that is expected and not expected different types of planning have been introduced in literature. Complex environments can be made up of many possible paths which can be computed with strategies such as that of [], where each iteration of path planning uses a more complex predictor. Each of these predictors' path can be used to build a map of states and corresponding behavior that an agent is able to perform. Other planning algorithms use specialized planning based on a sub process task such as []'s vertical control system, where one module is independently responsible to avoiding objects while more abstract modules can plan according to behavior of objects. Such planning can also be implemented into a map of acceptable behaviors for each state. Multiple goal resolution is attractive as it may still be used in the evaluation of an observed behavior discussed in III.B .

III. Methodology

The method in which the three strategies of interest are implemented are now outlined. The reader should make note the intended chain of trust commences with hardware integrity at its core moving through the various levels of authentication for self-verification. Once the agent can verify its own state it then proceeds to estimate the likelihood that other agents are taking an appropriate actions.

A. Secure Ad-Hoc Channel Communication

The communication between agents must be encrypted with secret and unique keys. In addition future releases of a robot must be able to communicate with the existing version. There are of course more requisites much as data integrity, privacy of information, and authenticity of the message. We utilize the trust chain's hardware signature in the following protocol.

A cryptographically random challenge is produced for which there is a globally known one way function to compute a response by means of the originating party's hardware signature, this response is what will be used as input to the encryption cypher of choice. Naturally the receiver will not have the hardware signature of the originator, and thus we must use a unique table for each agent in the system that is created upon creation of the agent itself mapping each MAC with a CR pair. In order to account for extreme table sizes the table can be kept outside of the secure hardware and be encrypted by the hardware key. These keys are very important and therefore, in order to comply with privacy we must be able to change keys as to avoid decryption attempts. For this we will create a churn of keys for every packet that is sent; that is, the sender of a packet sends a new cryptographically generated random C-R pair that the each agent's table may be updated with for the next packet to be sent.

In order for the receiving party to know the message decrypted is in fact legitimate there must be text that can be verified. There is also need to know who the message came from due to our authenticity requirement. These can both be satisfied by placing the source MAC in the encrypted portion of the message exchange. To further protection we place the MAC of an agent inside the HSE. Replay attacks can also be a danger, and the method of guarding is having an accurate timestamp.

New agent authentication work in a similar manner to normal communication but must be initiated by the party with the appropriate and working CR pair. Last but not least, this communication must be implemented like a tunnel where each agent can be uniquely identified by the system.

B. State Detection

We start our state detection by defining our state and control:

State	Control
$x, y, \theta, v_f, v_theta$	a_f, a_θ

Where subscript f refers to the local frame forward direction of the agent. The following outlines how the process of obtaining the probability of being compromised from the sensory inputs and the desired method of path planning.

Position and velocity are included in the state in order to make future conjectures about the behavior of the robot. Now that we have a defined state we must gather the state from a combination of sensor data and information obtained from the robot. We use a particle filter for this purpose which we initialize to cover the full area of interest. If an initial position is provided a particle is given that claimed position, but all others are kept in their random position to accommodate for a false given position.

We obtain the average of the particles which is utilized in order to obtain an approximate point nearest to some neighbors on two maps: a compromised map and an uncompromised map. Both maps are comprised of state points along the path planned that are considered compromised or uncompromised as illustrated in figure 1.

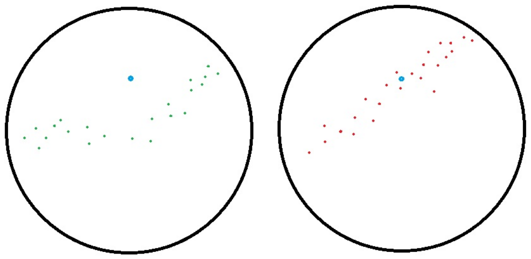
Fig. 1: The left shows the uncompromised map made up of states avoiding some area and the right shows a compromised map where the states do cross the same area. In this case the state obtained from the particle filter mean, demonstrated in blue, would likely yield a more compromised probability

A KD tree is used on both maps to find the closest approximate points to each mean found. The few particles gathered by this process are used, by the compromised and uncompromised maps, to form probability density function (PDF) by means of a Kernel density estimation (KDE). This distribution is used to obtain the probability the mean of the particle filter approximation falls within each map. The product of these two probability biases the input to the next iteration to the Bayes equation 1 where b = behavior of the robot, s = state of the robot and c = the probability the robot is compromised.

$$\beta el = \frac{(P(b,s|c)P(c)}{(P(b,s|c)P(c) + P(b,s|c')P(c'))} \quad (1)$$

This implementation provides a way a way for the change in behavior, that is from uncompromised to compromised or vice versa, to impact the current belief.

C. Distributed State Estimation

[INSERT DISTRIBUTED STATE ESTIMATION METHODOLOGY HERE]

IV. Experiments

The three different concepts were implemented and are detailed below.

A. Secure Ad-Hoc Channel Communication

In order to communicate between agents within the proximity range the wifi communication channel is analyzed. In order to keep the agent to agent principle we render the IP layer of communication unfit and to a more conservative approach of omitting the IP layer to the network stack.

Use of the data link layer pushed the use of raw sockets; the socket class for python was used instead. The downside however remains in that the socket class requires sudo, or root, access in order to run the node since permissions do not allow sockets to be ran under a regular user. Since giving admin privileges to a node can be considered a threat, there is push to either change the permissions for saw sockets or to change to a new library all together.

The python code was first loaded onto two different laptops, both running Ubuntu 14.04 LTS with a full ROS install, and connectivity was established. We first start with obtaining the database from the 'central base' module created as seen in figure 2. The directory where the database file resided was located and moved to the correct agent where the robot_proxy was executed.

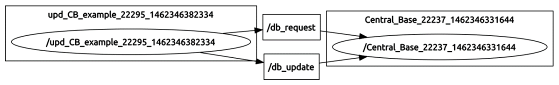
upd_CB_example_22295_1462346382334
/upd_CB_example_22295_1462346382334
/db_request
/db_update
Central_Base_22237_1462346331644
/Central_Base_22237_1462346331644

Fig. 2: The initial communication with the central frame is made here. Each database, in its encrypted form, is created to be placed with each corresponding agent.

A new connection is detected and notified via a list in topic /Host_lists which signifies a working connection. Now, in order to test the appropriate data the testing node Simple_calc.py was used as seen in Figure 3.

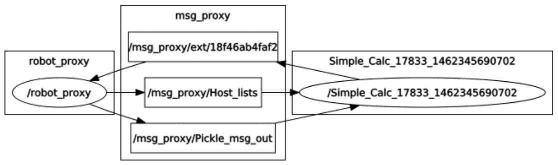
msg_proxy
/msg_proxy/ext/18f46ab4faf2
robot_proxy
/robot_proxy
/msg_proxy/Host_lists
Simple_Calc_17833_1462345690702
/Simple_Calc_17833_1462345690702
/msg_proxy/Pickle_msg_out

Fig. 3: Topic Interaction with node- As new agents are populated, the external host lists are updated. Hence, when the topic handler retrieves the Host Lists from all the connection objects, the nodes in the current system will have an accurate overview of which topics are being published elsewhere based on the MAC of each external agent.

Pickle_msg out is the received messages from the outside world that are rebroadcasted. Each external agent connected has a subscriber created through /msg_proxy/ext/. Scoping is utilized throughout to address each agent outside the current robot. The pickle_msg contains a MAC field After setting up the correct interface to transfer frames over wifi things behaved like a regular wire. The OS was used in to set up a local network; with the secure nature of the messages exchanged this network can be left open so that future releases will not have difficulty communicating with older models.

The results were, as expected, the node could not publish another type of message on our specified topic. Then, after making the appropriate PickleSend Type we are able to send the message to the subscriber callback in the robot proxy. From there we learn our transmission was successfully encrypted and transmitted through, received decrypted and unicasted to the receiving host. The process is carried on to the test bench of two zed boards where the implementation of response computation must be extendable to both the laptop and board despite architecture. We are able to verify the authentication and individual agent assignment by verifying the test data itself at each end of the connection.

B. State Detection

[INSERT RESULTS HERE]

V. Conclusion

We have presented three components from a larger chain of trust that is designed to ensure security of not only the agent itself, but also of the group of agents surrounding it.

Acknowledgment

We would like to thank Professors Murat C. Cavusoglu, Russell C. Jackson and Tipakorn Greigarn and for their time, support and guidance.