Addressing Security And Privacy Issues By Analyzing Vulnerabilities In IoT Applications

Francisco Javier Candelario Burgoa

University of Texas at El Paso

Follow this and additional works at: https://scholarworks.utep.edu/open_etd

Part of the Cognitive Psychology Commons, and the Computer Engineering Commons

Recommended Citation
https://scholarworks.utep.edu/open_etd/3393

This is brought to you for free and open access by ScholarWorks@UTEP. It has been accepted for inclusion in Open Access Theses & Dissertations by an authorized administrator of ScholarWorks@UTEP. For more information, please contact lweber@utep.edu.
ADDRESSING SECURITY AND PRIVACY ISSUES BY ANALYZING VULNERABILITIES IN IOT APPLICATIONS

FRANCISCO JAVIER CANDELARIO BURGOA

Master of Science in Computer Engineering

APPROVED:

_____________________________
Sai Mounika Errapotu, Ph.D., Chair

_____________________________
Yuanrui Sang, Ph.D.

_____________________________
Robert C. Roberts, Ph.D.

_____________________________
Jingyi Wang, Ph.D.

_____________________________
Stephen L. Crites, Jr., Ph.D.
Dean of the Graduate School
Dedication

I dedicate this work to my family and friends that have supported me throughout my career and made it possible for me to be here.
ADDRESSING SECURITY AND PRIVACY ISSUES BY ANALYZING VULNERABILITIES IN IOT APPLICATIONS

by

FRANCISCO JAVIER CANDELARIO BURGOA, B.S.E.E.

THESIS

Presented to the Faculty of the Graduate School of
The University of Texas at El Paso
in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE

Department of Electrical and Computer Engineering
THE UNIVERSITY OF TEXAS AT EL PASO
December 2021
Acknowledgements

I would like to acknowledge Dr. Sai Mounika Errapotu for guiding me throughout my research and thesis. She was always available whenever I experienced any roadblock and answered every question I encountered. Dr. Errapotu was a true mentor and was able to lead me to a successful and enriching research experience. If it was not for her, I would have never done any cybersecurity investigation, which turned out to be thought-provoking and fascinating.

I would also like to acknowledge the Texas Instruments Foundation Endowed Scholarship Program for Graduate Students for their economical support throughout my master’s degree which allowed me to complete my degree without worrying about money to pay for tuition.

Finally, I would like to acknowledge the UTEP College of Engineering for their continuous support throughout my bachelors and master’s degree.
Abstract

The Internet of Things (IoT) environment has been expanding rapidly for the past few years into several areas of our lives, from factories, to stores and even into our own homes. All these new devices in our homes make our day-to-day lives easier and more comfortable with less effort on our part, converting our simple houses into smart homes. This increase in inter-connectivity brings multiple benefits including the improvement in energy efficiency in our homes, however it also brings with it some potential dangers since more points of connection mean more potential vulnerabilities in our grid. These vulnerabilities bring security and privacy concerns into our day-to-day lives. The more devices inside a house, the more information can be collected about the people living there. Naturally, people can be concerned about how attackers could acquire their information and feel hesitant about the inclusion of IoT devices inside their home despite the clear benefits it can bring to them. Even though this is a clear issue for the community, smart home cybersecurity is still in its infant stage. The purpose of this thesis is to improve upon smart-home cybersecurity techniques to provide a more secure environment that will protect the users’ privacy while maintaining a fast response time. This is aimed to be done with a dual approach; a cryptographic approached based on elliptic curve cryptography and Blake2 hashing, and a non-cryptographic approach based on synthetic data generation using a specific type of neural network called Generative Adversarial Network (GAN), which will create adversarial traffic for an Intrusion Detection System (IDS) to train on and improve its accuracy and decrease the number of false negatives. This encryption technique is tested upon a smart home environment with a Raspberry Pi as a control hub. Once these techniques are refined, they can be fully implemented on smart home devices and even moved into an industrial control system environment.
Table of Contents

Dedication ........................................................................................................................................ iii
Acknowledgements ........................................................................................................................ v
Abstract ........................................................................................................................................... vi
Table of Contents ........................................................................................................................ vii
List of Tables ..................................................................................................................................... ix
List of Figures .................................................................................................................................... x
Chapter 1: Introduction ....................................................................................................................... 1
  1.1 Organization ............................................................................................................................ 2
Chapter 2: Motivation and Purpose .................................................................................................... 4
  2.1 Energy efficiency need due to climate change ........................................................................ 4
  2.2 Research on IoT security is on infant stage ............................................................................ 4
  2.3 Limit invasive collection of personal information ................................................................. 5
  2.4 Moving towards a smart grid ................................................................................................. 6
  2.5 Scale up from smart homes to industry control systems ....................................................... 7
Chapter 3: Smart Home and Smart Grid ............................................................................................. 8
  3.1 Smart home definition ............................................................................................................. 8
  3.2 Smart grid definition .............................................................................................................. 9
  3.3 Motivation for change to smart grid ..................................................................................... 10
Chapter 4: Security and Privacy Issues ............................................................................................. 12
  4.1 Types of attacks ..................................................................................................................... 12
    4.1.1 Taking Control Attacks ................................................................................................. 13
    Replay Attack ......................................................................................................................... 13
    Man-in-the-Middle Attack ....................................................................................................... 14
    Impersonation Attack ............................................................................................................ 15
    4.1.2 Steal information attacks ............................................................................................ 16
    Message Modification Attack ............................................................................................... 16
    Eavesdropping Attack ........................................................................................................... 17
    4.1.3 Disrupt services attacks ............................................................................................... 18
<table>
<thead>
<tr>
<th>Denial-of-service Attack</th>
<th>.................................................................</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>Masquerade Attack</td>
<td>..................................................................................</td>
<td>19</td>
</tr>
<tr>
<td>4.1.4 Based on attacker location</td>
<td>........................................................................</td>
<td>20</td>
</tr>
<tr>
<td>Outside the network</td>
<td>...............................................................................</td>
<td>20</td>
</tr>
<tr>
<td>Inside the network</td>
<td>...............................................................................</td>
<td>20</td>
</tr>
<tr>
<td>4.2 Current security measures against attacks</td>
<td>...........................................................</td>
<td>21</td>
</tr>
<tr>
<td>Chapter 5 – Technologies to Use</td>
<td>..........................................................................</td>
<td>24</td>
</tr>
<tr>
<td>5.1 Cryptographic Approach: ECC + BLAKE</td>
<td>........................................................................</td>
<td>24</td>
</tr>
<tr>
<td>5.1.1 Elliptic Curve Cryptography</td>
<td>.........................................................................</td>
<td>24</td>
</tr>
<tr>
<td>5.1.2 BLAKE2 hashing</td>
<td>..................................................................................</td>
<td>29</td>
</tr>
<tr>
<td>5.2 Non-Cryptographic Approach: GAN data</td>
<td>.......................................................................</td>
<td>32</td>
</tr>
<tr>
<td>5.2.1 Generative Adversarial Networks</td>
<td>.......................................................................</td>
<td>32</td>
</tr>
<tr>
<td>5.2.2 - GAN’s generator and discriminator</td>
<td>.........................................................</td>
<td>33</td>
</tr>
<tr>
<td>Chapter 6 – Proposed Hybrid Approach</td>
<td>..........................................................................</td>
<td>39</td>
</tr>
<tr>
<td>6.1- Encryption experimentation</td>
<td>..........................................................................</td>
<td>39</td>
</tr>
<tr>
<td>6.2 Data generation with GAN</td>
<td>............................................................................</td>
<td>45</td>
</tr>
<tr>
<td>Conclusions</td>
<td>..................................................................................</td>
<td>49</td>
</tr>
<tr>
<td>References</td>
<td>..................................................................................</td>
<td>50</td>
</tr>
<tr>
<td>Glossary</td>
<td>..................................................................................</td>
<td>53</td>
</tr>
<tr>
<td>Appendix A</td>
<td>..................................................................................</td>
<td>54</td>
</tr>
<tr>
<td>Encryption with BLAKE2 and ECDSA using Raspberry Pi 4 and sensors</td>
<td>..........................</td>
<td>54</td>
</tr>
<tr>
<td>Appendix B</td>
<td>..................................................................................</td>
<td>59</td>
</tr>
<tr>
<td>Synthetic data generation using Generative Adversarial Networks</td>
<td>.......................</td>
<td>59</td>
</tr>
<tr>
<td>Vita</td>
<td>..................................................................................</td>
<td>67</td>
</tr>
</tbody>
</table>
List of Tables

Table 5.1 – RSA vs ECC key size [12] ............................................................. 27
Table 6.1 – Algorithm performance metrics .................................................. 44
List of Figures

Figure 3.2 – Smart home connected to the smart grid ................................................................. 10
Figure 4.1 - Types of attacks ........................................................................................................ 13
Figure 4.2 – Replay Attack ........................................................................................................... 14
Figure 4.3 – MITM attack ............................................................................................................ 15
Figure 4.4 – Impersonation Attack ............................................................................................. 16
Figure 4.5 – Message Modification Attack ................................................................................ 17
Figure 4.6 – Eavesdropping attack ............................................................................................ 18
Figure 4.7 – DOS attack ............................................................................................................. 19
Figure 4.8 – Masquerade attack ............................................................................................... 20
Figure 4.9 – How RSA Encryption Works [8] .......................................................................... 21
Figure 4.10 – How Hashing Algorithms Work [10] .................................................................. 23
Figure 5.1 – Elliptic curve graph and formula ........................................................................... 25
Figure 5.2 – Digital signing process ......................................................................................... 26
Figure 5.4 – ECC vs RSA key length growth [14] ................................................................... 28
Figure 5.5 – ECDSA vs RSA performance test screenshot [15] ................................................. 28
Figure 5.6 – Speed comparison of hashing algorithms [18] ....................................................... 30
Figure 5.7 – Hashing algorithms throughput comparison [19] .................................................. 31
Figure 5.8 – GAN counterfeiter vs. police officer analogy ......................................................... 33
Figure 5.9 – Generator multiple outputs until matching the real sample [21] ......................... 34
Figure 5.10 – Discriminator heatmap [21] ............................................................................... 35
Figure 5.11 – GAN’s architecture [23] ...................................................................................... 37
Figure 5.12 – a) MNIST dataset samples and GAN output b) TFD dataset samples and GAN output. [24] ......................................................................................................................... 36
Figure 5.13 – Picture of a person created by a GAN [25] .......................................................... 36
Figure 6.1 – Hardware setup for smart home simulation .......................................................... 40
Figure 6.2 - Sense HAT pin usage diagram [26] ........................................................................ 41
Figure 6.3 – Picture of hardware setup ..................................................................................... 41
Figure 6.4 – User authentication algorithm using BLAKE and ECDSA ..................................... 42
Figure 6.5 – Appendix A code output ....................................................................................... 43
Figure 6.6 – Packet labeling process – Fig. modified from [27] .................................................. 46
Figure 6.7 – Final labeled CSV file .......................................................................................... 47
Figure 6.8 – Data distribution of real data vs generated data on SYN-flooding attack packets... 48
Chapter 1: Introduction

Computers have evolved greatly in the past decades. They keep getting smaller and with a greater computing power every year. This has allowed us to implement embedded systems in multiple objects that we interact with on a daily basis. This implementation of computers made our things capable of performing more tasks and collecting data from its environment to improve their performance. This converted simple objects into smart devices. As we connected all these devices to the internet and simplified them to work with each other they became capable of executing more complex tasks and help humans solve complex problems.

One of the main problems humanity faces today is climate change. One big factor that contributes to climate change is the lack of efficiency in the utilization of electricity. The implementation of smart devices is an opportunity to regulate the energy consumption with constant and real-time communication between devices and the grid. However, with more communication and more collection of data, there comes some security and privacy concerns. The new smart devices may be the target for cybersecurity attacks, where the attacker tries to collect information from the users without their authorization. In order to prevent this, several technologies have been implemented. Data encryption allows the users to share messages and if the information is intercepted, it will not have any meaning to the attacker. Also, intrusion detection systems have been developed to catch whenever an attack is attempted and alert the user.

As new techniques to keep systems safe are developed, attackers also develop new methods to get around the security measures and get access to the network. This becomes a constant race between attackers and defenders to keep improving upon their techniques. In this thesis we aim our focus on proposing a hybrid approach. The first part of the approach will have an encryption perspective, combining Elliptic Curve Cryptography (ECC) with BLAKE hashing. This will
provide a high level of security comparable or higher than current standards, while using less resources and being faster than other current encryption techniques used. The second part of the approach is a non-cryptographic approached based on artificial intelligence. A type of neural network called Generative Adversarial Network (GAN) will be used to generate synthetic packet traffic data which can later be used to train an Intrusion Detection System (IDS) to detect possible attacks. The new data will be useful since network traffic datasets with Internet of Things devices are scarce. Allowing an IDS to train with big amount of data will allow it to improve its performance and diminish the number of false negatives by the IDS whenever there is an attack on the system.

1.1 Organization

The remainder of this document is organized as follows:

Chapter 2 talks about the motivation and purpose for this research. How climate change is an imminent problem that requires us to use whatever tools we can to fight it. It also talks about how the research on smart home cybersecurity is still on its early stages and in need of greater focus. We discuss the collection of private information done by technology companies and the aim to limit these practices. Finally, we discuss how the increase in smart home devices leads us to moving towards a smart grid and how the security practices in this thesis can be transposed from homes to industry in the future.

Chapter 3 discusses the definitions and implications on smart home and smart grid. We also discuss the benefits of a smart grid over our current infrastructure and how this motivates the change towards a smart grid.
Chapter 4 has a description of the most popular cyberattacks on smart home environment and a brief description about the current encryption techniques and their functionality as well as description on hashing and their implementations.

Chapter 5 starts describing our proposed hybrid approach of Elliptic Curve Cryptography alongside with BLAKE hashing. We describe the ECC algorithm and the benefits it provides over other encryption algorithms like RSA. We also do a similar description for BLAKE and benefits over other hashing algorithms. On the second half, we describe Generative Adversarial Networks and how they can be used to generate data to train an Intrusion Detection System.

Chapter 6 includes details on the experimentation process, hardware setup and GAN training process alongside with results.

Appendix A contains the code used with the Raspberry Pi and the sensors to test BLAKE2 hashing and ECDSA encryption as well as the timing metrics of each of the algorithms.

Appendix B contains the code used to train the GAN network and generate synthetic network data.
Chapter 2: Motivation and Purpose

2.1 Energy Efficiency Need Due to Climate Change

Climate change is a problem that affects all humans and changes among several sectors of industry need to happen before we can see a positive impact towards solving this issue. One main contributor to this problem is the increase in power consumption and the lack of energy efficiency. Power consumption continues to increase globally, for this reason many countries have tried to increase their energy efficiency, making better use of the electricity they produce. The Internet of Things has introduced a lot of devices that require energy into our lives, however it has also increased our ability to be more energy efficient with sensors that stop energy consumption whenever it is not needed like motion sensor lights to avoid wasting electricity when a light bulb is on, but no one is near it. Thirty-seven percent of total power consumption in the US comes from the residential sector, therefore if homes become more energy efficient, it will have a massive impact in the reduction of energy wastage in the country.

2.2 Research on IoT Security is on Infant Stage

With the expansion of IoT into smart homes, cybersecurity has become a main concern. There is a lot of information to get in our homes. Devices inside our home track us and our activities, what we like and dislike, when we are inside or outside. There is a lot that an attacker can get out of accessing the information in our smart home grid. Simple things like when our lights are on or off could be an indicator of if we are inside the house or not and if our valuables are unattended for. All this data could be used to create a profile of each user that can be used by an attacker to use that information by itself or to plan a physical attack towards the user or its home. Considering all of this and the fact that more and more houses have smart home devices inside them every day, we would think that security is pretty advances on these devices, however smart home cybersecurity research is a relatively new field with much more to explore and improve.
Timely response is very crucial when talking about smart home devices. Whenever a user gives an order to its smart hub, they expect an immediate response. Performing actions through a smart device should be as quick or quicker than being able to do them yourself. Asking your smart hub for the temperature outside should be quicker than looking it up on your phone. All this means that quick response times are crucial when dealing with smart home devices. Unfortunately, quick response time and high security cryptography do not tend to go hand in hand. The more complex the encryption is, the more time it will take to encrypt and even more to decrypt. This is why some of the algorithms used currently are not as secure as they could possibly be. Nevertheless, new algorithms are created continuously and with every new advance, they become more secure and quicker. The focus of the cryptographic approach in this thesis is to use two state-of-the-art algorithms together which will increase the level of security greatly, while maintaining quick response times that will keep device use normal.

2.3 Limit Invasive Collection of Personal Information

The current control hubs available to manage your smart home collect plenty of information about its users which they later use for their own benefit in practices like targeted advertising. By creating a profile of the user's personality, likes and dislikes, the company is able to create a better advertising market which they can profit from. The better the model of the user, the better advertising service they can provide to whoever wants to advertise on their platforms. This intrinsically means the more information they can get from their users, the better. This puts the user’s privacy in danger since the less privacy the user has, the more information they can get to build a better model and sell more advertising. This is one of the examples of how companies profit from lessening the privacy of their users. Another goal of this thesis is to provide an independent central control hub which has the user’s privacy as a priority. All communication between users, control hub and devices will be encrypted and the only collection of data from big companies would be through each of their devices instead of being able to collect every single
order given to the control hub. This dissipates information and would create a less accurate model of the user, giving it more privacy and control over the information it shares and the advertising it sees.

2.4 MOVING TOWARDS A SMART GRID

Implementing IoT into smart homes has allowed us to use energy more strategically. Energy prices change dynamically according to demand. Using sensors and devices that can automatize certain tasks and schedule them according to the change of energy price. For example, the user can load its smart washer and dryer and set it up to start whenever energy prices are lower in the day. This scheduling of tasks takes load off from the grid which allows it to generate electricity more uniformly throughout the day and not having to compensate for energy surges. The same concept can be applied to industry. Implementation of smart devices and sensors in industry can significantly increase their energy efficiency, consequently decreasing their energy costs. As homes and industries become smarter, we will be able to move towards a smart power grid. A smart grid consists of two-way communication between the grid and the customers that is able to respond in real time to the energy demands.

Moving towards a smart grid will have multiple benefits. More efficient transmission of electricity, since there will be a better understanding of the energy needs of each area of the city. Quicker restoration of electricity after power outages. With less load on the grid, it will be easier to restore power whenever an outage happens. Reduced operations and management costs for the utility provider since they can produce energy more uniformly throughout the day. Reducing costs for the utility provider translates to lower power costs for consumers. Reduced peak demand. Currently during the afternoon around 4pm to 9pm the power consumption peaks due to multiple users consuming high amounts of electricity during that period, this also means higher production prices during those times. With the use of smart grids this peak can be flattened, and the consumption divided throughout the day. This integration will also facilitate the use of renewable
energies both by the utility provider and customer-owned renewable energy systems. The user will be able to use the power they produce during peak hours and consume from the grid the rest of the day, with this electricity will be more economical since they will be paying for off-peak prices, and they will also help flattening the peak of power consumption in their communities.

2.5 Scale up from smart homes to industrial control systems

These energy and security technologies are being explored on a small scale in the smart home environment, however they can be scaled up to industrial systems as well. Power efficiency applied to industries will have great economic and environmental effects. On the other hand, security and privacy can be much more important when talking about industries than homes since it can be sensitive data from hundreds or thousands of people or proprietary information that could cost greatly to the company if the information is leaked. Therefore, security is a top priority when transmitting information inside an industry setting. Fast and secure protocols are a must. Using the proposed protocols will allow the company to take advantage of the IoT capabilities while maintaining ultimate security and privacy.
Chapter 3: Smart Home and Smart Grid

3.1 Smart home definition

The concept of smart home is understood by most people, however when dealing in the scientific community, different authors define a smart home differently. The Oxford dictionary defines a smart home as: “A home equipped with lighting, heating, and electronic devices that can be controlled remotely by phone or computer.” Another widely accepted definition from the Smart Home Association is: “the integration of technology and services through home networking for a better quality of living”. [1] For this thesis we will merge both of these definitions to make up our own to bring qualities mentioned in both. It is important to consider that most of the electronic devices that make up a smart home can be controlled remotely, this is noted in the dictionary definition. However, it is also important to remark that the integration of this technology is to improve the quality of living of the people inside that smart home. With all this in mind, the proposed smart home definition for this thesis is the following: “A smart home is a home which integrates technology and services through electronic devices that can be controlled remotely by a user with the purpose of bettering its quality of living” This definition attempts to integrate all major qualities of a smart home while keeping in mind that after weighting pros and cons of adding smart devices to a home, the users’ quality of life should increase.

Smart homes have evolved greatly in the past decades. More and more devices are added into our homes making them smarter and smarter. The mechanisms to connect devices between each other have also evolved, making the experience more cohesive. Nowadays you can set up routines or triggers so that if a condition is true, then a series of actions from several different devices happen at the same time. In the case of a house intrusion, not only will the alarm ring, but the security cameras will start recording, the alarm system will call the police department, the lights will turn on, your emergency contact can be notified and any other task that you seem fit. This is just a drastic example of how a smart home can help during an emergency, but also on a day-to-day basis a smart home can make its user’s life easier. For example, when a bedtime routine
gets started, the house security alarm is armed, all the locks are closed, the lights will be turned off, you will be read your calendar activities for the next day while you get reminded to brush your teeth and finally relaxing music can be played to help you sleep better. Even your sleep patterns can be monitored to help you keep track of your sleep habits and stay healthy. This and many more uses can be how user’s benefit from adding sensors and devices to their home to increase their quality of life.

3.2 SMART GRID DEFINITION

The creation of smart homes is tightly related with the possibility of having a smart grid. A smart grid is defined as: “A self-healing network equipped with dynamic optimization techniques that use real-time measurements to minimize network losses, maintain voltage levels, increase reliability, and improve asset management.” [2] A smart grid is full of sensor and devices that continuously check the status of the grid. They collect data of the grid’s behavior and after analyzing this data we can solve several of the current issues of the power grid. If we break down the definition of the grid, first of all, is a self-healing network. The grid is considered self-healing due to the fact that it can minimize failures. By analyzing problems and abnormalities that present on the grid, it can isolate the failures and react to protect the network, consequently preventing blackouts. Next, the smart grid is equipped with dynamic optimization techniques, which means that depending on the current status of the network, it can evolve and use their resources more efficiently to be more environmentally friendly and cost-effective. It also uses real-time measurements which means that since homes are also becoming smarter, the grid has a better understanding of the energy consumption on real time as the consumption happens and with the analysis of energy consumption it can predict consumption behavior, being more prepared for spikes of electricity usage. This readiness makes it possible to maintain voltage levels supplied to homes, making the grid more reliable and safer while also being better for the environment since the resources are used more effectively.
On figure 3.2 it can be seen how a smart home is made up of different smart devices interconnected making up a network. This may or may not include a power generation system like solar panels on the house or any other form of power generation which will be feeding the electricity requirements of the house and even selling extra electricity back into the grid through a smart power meter that can keep track of electricity both ways, from the grid into the house or from the house into the grid. These elements make up what is called a Home Area Network. Whenever multiple houses have a similar setup, it makes up the Neighborhood Area Network. Finally, the neighborhood is connected to the smart grid, which monitors the power needs of all the neighborhoods in the community it services.

3.3 Motivation for Change to Smart Grid

There are different motivations towards the implementation of a smart grid. Much of the existing power infrastructure is in need of renovation since it is reaching its useful life. There is the option of just renovating the infrastructure keeping the same design but renewing it with new
materials or changing the design to a more modern one adding sensors and devices to make our grid “smarter”. With this comes one of the first drawbacks of smart grids, high initial cost. Renovating our whole power distribution system has a very high initial cost with all the addition of new devices. Our new renovated system should also be capable of supplying more power than the previous one since the electricity demand in the world keeps increasing with time as population increases. The current infrastructure is already overstressed which gives us an extra incentive to upgrade our grid. This stress could be partially released by decentralizing some of the power generation, luckily the smart grid is also more friendly with renewable energies since with more control on generation is easier for the grid to take advantage of these intermittent sources of electricity. The damage done to the environment by higher use of power can be counteracted with the increase in the use of renewable energies which would also make it easier and more cost-effective for users to implement their own renewable energy generation systems in their smart homes. [3]

In many cities and countries, power generation is monopolized by a single organization and users have no other option than to pay the prices set by said organization. The introduction of smart grids and smart homes will also benefit the consumer with clear and transparent prices and usage on their consumption. With a smart meter users will be able to get data on their hourly power consumption and see how much electricity they are consuming and by which devices it is being consumed, being able to regulate more effectively their power consumption. The user will have much more control over their power usage or power generation. [4] This will also allow for smaller power generation stations to be incorporated into the grid and distribute the stress in the generation of electricity throughout the network. It even opens the possibility for smaller companies to generate great amount of electricity and deal as independent generators for which consumers can buy their power directly, allowing for greater price competition bringing the prices down for the consumer.
Chapter 4: Security and Privacy Issues

We have discussed lot of the benefits of smart grids and smart homes. However, one of their main areas that need further research and development is privacy and security. With the addition of sensors and devices, comes the increase in information and data patterns. All this information is very useful when used as intended, to help both the electricity provider and the consumer work together keeping demand and generation hand in hand. However, there is a valid concern regarding all this new data, this data could be used inappropriately and reveal a lot of information on its user. Not only are you able to tell what devices that consumer is using, but also for how long and even when did they use them. By analyzing patterns in this data, you can get a clear picture of the user’s day to day routine and behavior profile. A leak on this data could signify a threat to the user since the attacker can use this information against the user in multiple ways. It is clear how it is in everyone’s best interest for this information to be protected and security measures taken to prevent possible attacks into smart home networks. Next, we will discuss some of the different types of cyberattacks that could pose a risk to smart home networks.

4.1 Types of Attacks

There are multiple ways of catalogizing attacks on a smart home network, however for the purpose of this thesis we will divide them into four categories: Taking control, steal information, disrupt services and location of the attacker. Each of these categories has specific attacks within them as it can be seen on Figure 4.1.
4.1.1 Taking Control Attacks

Taking Control attacks refer to when the attacker attempts to take over a device to achieve a desired goal. In this category we have replay attacks, man in the middle attacks and impersonation attacks. We will further go into details about each of these attacks:

**Replay Attack**

The replay attack is that in which the attacker intercepts the transfer of data between two nodes, for example the communication between the control hub and a smart device. The message that was intercepted is recorded and then replayed to pose as the authentic node. Once the attacker has masqueraded as an authentic user, it can easily monitor the data transfer in those nodes. [5] A graphical representation of a replay attack can be seen on Figure 4.2. In a smart home context, the attacker could try to do a replay attack on your smart lock by replaying your “lock” command. Your lock would perform the action, but also authenticate the attacker as a valid user. The attacker then would be able to lock and unlock your smart lock willingly since it is now an authorized user.
A Man-in-the-Middle (MITM) attack has some similarities with the replay attack in the sense that there is a third-party sniffing in the communication between two nodes or two users without them knowing. Two users could be sharing messages back and forth communicating in what they believe is a private setting, however there is an attacker listening to their communication. Not only can the attacker listen to the messages sent, but also modify the information contained in those messages, therefore the recipient would receive a different message than the one that was originally sent for them. In figure 4.3 it can be seen a graphical representation of a MITM attack. In a smart home environment this can be seen as the connection between the user and one of the smart devices, for example in a smart oven, the wrong temperature could be recorded and sent to cloud, making the oven not turn off automatically. This could pose an obvious safe hazard for the user since the food inside the oven could burn and cause a fire in its home.
**Impersonation Attack**

An impersonation attack is that in which the attacker gets information from the actual user to pose and successfully authenticate as the real user. The attacker could get the user’s IP address and the MAC address of a device and with this data it can pose as a legitimate node in the network by stealing their identity. [5] The purpose of enhancing our authentication protocols and techniques is to prevent this from happening. Nowadays dual-factor authentication is very prominent in cybersecurity, and this helps minimize impersonation attack since the attacker would need to get access to multiple devices at once. In figure 4.4 you can see a graphical representation of an impersonation attack. In a smart home environment, the attacker runs mutual authentication alongside the legitimate user, to get access to the system without the target knowing there is an intruder. [6]
4.1.2 Steal information attacks

The next classification in which we will divide the cyber-attacks are the ones which purpose is to steal information from the target. We will be exploring further into two types of attacks under this category: Message modification and eavesdropping.

**Message Modification Attack**

In a message modification attack, the intruder alters the message sent by one party. The attack can delay, reorder, reroute or change the message to get some sort of access. In some instances, the new message can contain malicious code to infiltrate even deeper in the network. This type of attacks is popularly done on email due to weak security protocols, inserting malicious content into the email. A message modification graphical representation can be seen on figure 4.5.

In a smart home environment, the intruder could intercept communication between the user and a device to make it perform a desired action later than scheduled or to perform a different action instead. For example, by intercepting the user’s message to a smart lock attempting to close the
lock, the message could be changed to open the lock, therefore leaving a security breach in the user’s home.

**Message Modification Attack**

![Message Modification Attack Diagram]

Figure 4.5 – Message Modification Attack

**Eavesdropping Attack**

Eavesdropping is a type of Man-in-the-middle attack where the intruder devotes solely to listening to the network traffic and read or steal data. Eavesdropping attacks happen predominantly on wireless network communications, which make up most of the communications happening in a smart home environment. An eavesdropping attack graphical representation can be seen on figure 4.6. In a smart home environment, an eavesdropping example can be seen when an attacker snoops or sniffs into the network traffic between security cameras and the control hub. By checking this traffic an attacker could infer when there is movement detected by the cameras and when it is not, which could be used to determine the blind spots in the camera system.
4.1.3 Disrupt services attacks

This type of attacks focuses on saturating the services of the server or device to incapacitate the legitimate user from connecting to the server. We will be focusing on two attacks: Denial-of-Service (DOS) and Masquerade attacks.

**Denial-of-service Attack**

A DOS attack is that in which the attacker is able to disable a machine or server, thus making it inaccessible to its intended users. The attacker floods and saturates the machine with traffic requests or by sending it data to make it crash. A graphical representation of a DOS attack is shown in figure 4.7. In a smart home environment, an attacker could saturate the requests of the control hub, making the user unable to communicate with any of the smart devices connected to that hub.

Figure 4.6 – Eavesdropping attack
Masquerade Attack

In a masquerade attack, the intruder uses a fake identity to gain unauthorized access to the network through legitimate access identification. [5] One of the identities used by attackers can be a network identity or network ID, which is the portion of the TCP or IP address that identifies a host’s network. [7] The attacker can use stolen log-in information to access a system which they should not be able to access. In figure 4.8 a graphical representation of a masquerade attack is shown. In a smart home environment, a masquerade attack could happen when the attacker uses a network identity to bypass the access identification system of the control hub and be able to control the network’s devices as an apparently legitimate user.
4.1.4 Based on attacker location

Outside the network

The final classification in which we will divide attacks is based on the attacker location. Most attacks happen when the user is outside the network. Intrusion detection systems (IDS) were invented to alert users whenever an unauthorized entry to the network happens. Cybercriminals can attack all types of networks, from industries to small homes, and if they are able to get access to the network by bypassing the IDS, they may stay inside the network undetected for months, collecting information and maybe even preparing for a planned physical attack. Therefore, it is crucial that we develop strong IDS and encryption algorithms to keep these attackers outside our network.

Inside the network

Even though these attacks may be more uncommon, they still happen frequently. These attacks happen when someone that was provided access to the network, knowing or unknowingly provides access or data to another entity that is not supposed to have access to this system. This can happen inside a company when an employee accidentally discloses private information, when...
a disgruntled employee makes public company secrets or even attempts to access another portion of the same network where he shouldn’t be accessing. In a smart home environment this can happen when the user shares his internet password with a neighbor or another entity that later on tries to make an attack on his system.

4.2 CURRENT SECURITY MEASURES AGAINST ATTACKS

There are many encryption algorithms used today. One of most popular ones is Rivest-Shamir-Adleman (RSA) encryption. RSA is a public-key encryption algorithm. Public key encryption work with two different keys, a public key and a private key. The public key is available for everyone, the private key is used by a specific user. The data is encrypted with the public key, and it can only be decrypted using the corresponding private key. The keys can have different sizes, the bigger the key, the more secure the encryption is. Typical key sizes for RSA today are 1024 and 2048 bits. This type of encryption is great to keep secure communications, however the time to encrypt and decrypt the information, depending on the size of the message, can take a couple of seconds. Figure 4.9 shows a graphical representation of how a public key encryption algorithm like RSA works.

Figure 4.9 – How RSA Encryption Works [8]
Another really popular cryptographic approach used is hashing. Hashing is the process of converting a given key into another value. A hash function is used to generate a new value that is the result of a mathematical algorithm. [9] Performing a hash is a one-way operation. This means that once the hash of a message has been calculated, there is no way to go back to the original message. The result of performing a hash function on a key is called hash value or just hash for simplicity. Usually, the resulting hash is much shorter in length than the original message, which allows for quicker handling of data. The hash is also a fixed length which also makes handling long and varied length data easier. For these reasons, hashing is popular to improve efficiency in data processing by making tables and indexing the full message and the hash value attached to it. One example of a popular application of hashing is in the use of passwords. A log in service can store the hash of a user’s password instead of the plaintext of the password. This is with the purpose that if there is a breach in the system and an attacker is able to retrieve the information where the passwords were stored, they encounter a hash of the passwords which can’t be undone, therefore that information is useless to the attacker. Nevertheless, since a hash is quick to compute, whenever a user tries to use the log in service, the portal will take the string that the user just input, compute its hash, and then compare it to the stored hash in their file. If the hashes match, then the password is valid. If they do not match, then the access is denied.

A good hashing function will have very rare collisions. A collision happens when two different values are put into the hashing function and the same hash is turned as a result. Another characteristic of a good hashing algorithm is that even if two messages differ by very little, such as a few bits, then the resulting hashes of both messages should be very different from each other, differing in almost every bit. A good hashing algorithm will also not have preimage attacks done on them. A preimage attack is when given a specific hash, an attacker is able to figure out the original key that was given to the hash function. Some examples of popular hashing algorithms are MD5, SHA-1, SHA-2, SHA-3, and BLAKE2.
SHA-3 is one of the most popular hashing algorithms at the moment. This is due to its high level of security. No collision attacks or preimage attacks have been found with this algorithm. Nevertheless, there is another hashing algorithm that has a security level as high as SHA-3 but with better speed and performance. This hashing algorithm is called BLAKE. BLAKE was developed in 2013 as an improved version over SHA-3. This is why in our proposed approach we suggest the implementation of BLAKE as a hashing protocol.
Chapter 5 – Techniques

In this chapter we discuss the functionality and benefits of certain algorithms. We discuss the way Elliptic Curve Cryptography and BLAKE2 cryptography work separately and together to enhance the security of a system by adding two layers of protection. We discuss the speed and throughput of BLAKE2 as well as the key sizes of ECC. On the second section we discuss the use of GANs, its architecture and how it is used for data generation.

5.1 Cryptographic Approach: ECC + BLAKE

The first portion of our hybrid approach is a cryptographic approach. The innovative portion of our methodology is the combination of two state-of-the-art algorithms that are faster than its current counterparts. The fact that they are so lightweight allows us to encrypt more of the communications that would otherwise stay unprotected while maintaining quick response times since real-time interaction is crucial when dealing with human-device interactions on smart home devices.

5.1.1 Elliptic Curve Cryptography

The first algorithm we are using is our encryption algorithm which is based on Elliptic Curve Cryptography (ECC) more specifically, we will be using its digital signature algorithm called Elliptic Curve Digital Signature Algorithm (ECDSA). ECDSA is one of the most complex public key cryptography encryption algorithms. ECC is often compared to the RSA algorithm which is used for one-way encryption of data like emails and software, and it is based on the complexity of prime factorization. ECC is based on the complexity of algebraic structure of elliptic curves over finite fields. In figure 5.1 it can be seen an example of a basic elliptic curve and its algebraic formula.
ECDSA comes into play by generating digital signatures. A digital signature is a mathematical technique used to validate the authenticity and integrity of a message, software or digital document. [11] The way a digital signature works is that two keys are created. The sender uses a private key to encrypt the message and the only way the receiver can decrypt and open the message is by using the sender’s public key. If there was a modification in the message by a third party and the message was re-encrypted with another key, then the receiver will not be able to open the document since the public key will no longer work on this new encryption. For all this to work, the private key needs to be maintained as a secret, otherwise whatever entity gets ahold of
the private key could intercept the original message and create a new message signed with the stolen private key and the receiver would open it as normal without any knowledge that the original message was not the one received. Figure 5.2 depicts the process of signing and verifying a message using a digital signature.

![Digital Signing Process Diagram](image)

Figure 5.2 – Digital signing process

The reason why ECC has been gaining popularity over RSA is mainly because of its smaller key size. As discussed before RSA has a typical key size of 1024 and 2048 bits, the longer the key size the stronger the encryption and therefore harder to break. Nevertheless, with an algorithm based on elliptic curves, it is possible to have much smaller key sizes while still maintaining the same level of security. In figure 5.3 it can be seen how an elliptic curve key of only 160 bits is equivalent to a 1024 bit key on RSA while maintaining the same security, meaning the key size in ECC is 6.4 times smaller than on RSA.
Not only is the key size smaller, but also the growth of the key size in ECC in comparison to the growth in RSA, is much slower. This means that more complex encryptions can be achieved with much less computational power in the future than if we continue using RSA. The key size ratio on a 1024-bit RSA signature compared to ECC is 1:6, however the ratio on a 15360-bit RSA signature compared to ECC is 1:29. This means that a the ECC key size will be 29 times smaller than the RSA key while maintaining security. Not only that but the equivalent key size in ECC would only be 521 bits, which is about half of the size of the currently used 1024-bit RSA keys, while being 15 times safer. Figure 5.4 displays a graph of the key size growth comparison between RSA and ECDSA algorithms. It is clear to see how RSA key sizes grow much faster and ECDSA key sizes grow more linearly. This behavior has big advantages one of the main ones is that as more safe and complex signatures are needed, key size is contemplated to grow. Currently, RSA 15,360-bit signatures are too computationally expensive to compute, meaning they become unusable with current computing power for real-time applications since the computing time of a signature this size moves from milliseconds into seconds or even minutes depending on the computing power of the system used. [13]
Figure 5.4 – ECC vs RSA key length growth [14]

On figure 5.5, a screenshot of a performance test comparing RSA vs ECDSA signatures on a MacBook Pro is shown. On this screenshot it can be seen how 1,864 RSA signatures were performed in 10 seconds while in the same time frame 42,874 ECDSA signatures were performed. Not only did the ECDSA algorithm performed 23 times as many signatures as the RSA algorithm, but also the ECDSA signatures were 256-bit while the RSA were 2048-bit signatures, meaning that the ECDSA signatures performed were also 1.5 times safer than the RSA signatures.

Doing 256 bit sign ecdsa's for 10s: 42874 256 bit ECDSA signs in 9.99s
Doing 2048 bit private rsa's for 10s: 1864 2048 bit private RSA's in 9.99s

Figure 5.5 – ECDSA vs RSA performance test screenshot [15]
5.1.2 BLAKE2 hashing

The second part of our cryptographic approach is hashing. Hashing allows us to get a string and converted to a set length hash value that has been changed from the original in a one-way transformation which is irreversible as explained on chapter 4. The algorithm selected to perform hashing was BLAKE2. BLAKE2 was introduced in 2013 as the winner of the SHA-3 hashing competition. BLAKE2 is often compared to MD5, SHA-2 and SHA-3. SHA-2 was the hashing standard, however there was a need for faster hashing in hardware. SHA-3 is more secure than SHA-2 however it can take longer to perform than SHA-2. MD5 is a faster algorithm than the two previously mentioned, however MD5 is famously vulnerable to collision and length-extension attacks. [16], [17]

Different generations of hashing algorithms have variations within themselves. Each variation has a different hash bit length. For example, SHA-3 has 224-, 256-, 384- and 512-bit variations, and they each get their name on the output size of their hash. The longer the hash output, the greater the security level of that hash function. Nevertheless, the bigger the output hash, the algorithm has also a slower computing time. BLAKE2 has 2 versions: BLAKE2b, which is optimized for 64-bit architectures, and BLAKE2s which is aimed for smaller architectures. Figure 5.6 shows SHA-1, SHA-3, MD5 and BLAKE2 algorithms and their respective speeds measured in cycles per byte hashed. On the graph it is clearly shown how BLAKE2 is the fastest of all the algorithms compared. BLAKE2s has a very similar speed to MD5 and BLAKE2b has a faster speed, additionally BLAKE2 has a greater security level than MD5. BLAKE2 features no collision or length extension attacks, providing the same security level of SHA-3. [18]
As shown on figure 5.7, BLAKE2b has the highest throughput from all the compared hashing algorithms, being able to process just short of one gigabyte of data per second. SHA-1 is a close second with 909 Mebibytes per second (MiBps), however SHA-1 has a lower security standard. In third place comes BLAKE2s with 648 MiBps. With this throughput we can also see how using BLAKE2 should provide more than enough throughput to handle encryption of communication between smart home devices, which messages should be fairly small, making the hashing time only a fraction of a second.
Among the benefits of BLAKE2 are the faster performance than MD5 on 64-bit architecture systems, 32% less RAM required over the first generation of this algorithm (BLAKE), minimal padding which translates on faster speeds and easier implementation and direct support with no overhead of parallelism, tree hashing, and personalization. [18] This last benefit is crucial when implementing a hashing algorithm on a smart home device. Minimizing overhead becomes decisive since repetitive hashing will occur whenever any communication happens between nodes. With minimal overhead, the encryption times can be diminished making it viable to maximize the places where communication gets encrypted without disrupting performance and a real-time response from the smart device. BLAKE2 has been adopted by many projects due to its high speed, security, and simplicity.

Figure 5.7 – Hashing algorithms throughput comparison [19]
5.2 NON-CRYPTOGRAPHIC APPROACH: GAN DATA

5.2.1 Generative Adversarial Networks

The second half of our proposed approach is the generation of synthetic data using artificial intelligence to train an IDS. There are many types of neural networks, and each has better applications on specific fields. The neural network selected for this task was a Generative Adversarial Network. A GAN is an approach to generative modeling using deep learning. Generative modeling is a way of unsupervised learning where the network discovers and learns regularities or patterns in input data. Once it find this patterns, the model is able to create and output new samples that could have plausibly been drawn from the original dataset. [20] GAN are characterized by being two neural networks that train in competition with each other.

An analogy to further understand the behavior of GANs is to think of the first network as a counterfeiter that creates fake money, and the second network is a police officer which tries to distinguish between the real money and the fake money. A graphical representation of the GAN analogy can be seen on figure 5.8. The first network is called a generator, which is fed noise and attempts to generate fake instances of money that aim to look as realistic as possible. The second network is called a discriminator and receives both, real money and counterfeited money. The discriminator’s task is to differentiate between the real money and the fake money. The counterfeiter and the police officer compete against each other. With every iteration, the counterfeiter creates money that looks more and more similar to real money, and the police officer polishes his technique to notice little details that allow him to be better at telling which is the real money.
5.2.2 - GAN’s generator and discriminator

Most machine learning systems work taking a complex input, like an image, and output a simpler product like a label of what is pictured in the image. However, GANs work the opposite way. A Generative Adversarial Network is especially effective type of generative model. It takes in a simpler input and outputs a complex product like a picture of a human face. One important input for the generator is randomness. Without some sort of randomness added to the system, the generator would create the same output every single time it ran. Nevertheless, the system also needs to have an input of the real samples for it to learn what is likely to be a human face versus just random noise. The error function that gives the generator feedback comes from the discriminator. This error tells the generator how far off the generated sample is of a real sample. Then the generator uses more of the random noise to affect its output during each epoch, until the output resembles a real like sample and the error is minimal. Figure 5.9 shows the progression of
the output of the generator throughout the different epochs. The first output begins being random noise and after several iterations, it approaches more and more the shape of the real sample.

![Diagram of generator and discriminator](attachment:generator_diagram.png)

**Figure 5.9 – Generator multiple outputs until matching the real sample [21]**

The GAN’s discriminator has a different objective. The discriminator aims to distinguish the synthetic samples from real samples. GAN Lab is an online tool which helps visualize the training progress of a GAN. It creates a visual representation of the discriminator’s progress by creating a heatmap with one color showing where the real samples are more likely to be and with another color where the fake samples are more likely to be. In figure 5.10, it is shown a heatmap from a discriminator that has trained for several repetitions and is able to distinguish fairly well between real and fake samples. It shows in green the areas where real samples are likely to be and in purple the areas where fake samples probably are. As it can be seen in the picture most real samples, pictured as green dots, are indeed in the area marked green and the purple dots are also located in its respective color. This distribution of the heatmap means the discriminator will be able to distinguish the samples with a high effectivity. As the generator trains and get better at creating more real-like samples, the heatmap of the discriminator will start to turn grey, meaning the discriminator is not able to tell where the real or fake samples could be. [21]
5.2.3 - Current Applications of GANs

GANs are currently used in many applications. However, most of these applications revolve around images. The most popular GAN application is where the network is fed with a dataset of human faces and the network produces an image of a human face that is not a real person, however the result is so similar that it could fool a human into believe that they are looking at a picture of another human. Figure 5.11 shows two early demonstrations of GAN in action. The first image (a) shows the network being fed with the Modified National Institute of Standards and Technology (MNIST) dataset which contains samples of handwritten digits. Then the second image (b) shows the network being fed with the Toronto Faces Dataset (TFD) which contains pictures of faces of humans. Then on the right, highlighted in yellow, it can be seen the samples created by the GAN network.
Moreover, GANs have evolved being able to produce fake outputs that are very closely related to the real dataset samples. In Figure 5.12 is shown a photograph of a person that is completely created by GANs and that looks very closely to what a picture of a man would be. This gives allows us to think of using this powerful technology in other aspects that are not images. In the focus of this thesis, the GAN will be applied towards network traffic, attempting to generate real-like network packet data that could be used as real traffic to train an IDS.

Figure 5.12 – Picture of a person created by a GAN [23]
5.2.4 - Mechanism

In Generative Adversarial Networks, both neural networks (generator and discriminator) train at the same time and against each other. The generator tries to fool the discriminator and the discriminator tries to always beat the generator by distinguishing the fake sample it created. When the discriminator predicts if a sample is real or fake, it gets feedback of whether its guess was correct or not. Depending on this, a loss function is created. This loss function is provided to the generator as feedback for the generator to learn and create more real-like samples. The objective of the discriminator is to minimize the loss function, while the objective of the generator is to maximize it. The training of a GAN is a min-max game between the generator and discriminator. The objective is to achieve binary cross-entropy, meaning the generator maximizes the discriminator loss and the discriminator minimizes its loss in attempting to classify the fake samples. [24] Figure 5.13 shows a graphical representation of a Generative Adversarial Network, and it can be seen how both networks are connected and get feedback to achieve its objective.

![GAN’s architecture](image)

Figure 5.13 – GAN’s architecture [25]
5.2.5 – Relevance

The GAN’s dual nature of generator versus discriminator makes it optimal for cybersecurity applications. On cybersecurity there are two fronts, the attacker who tries to get access to a system and the defender who attempts to keep the attacker out of the network. Usually whenever a defense system is developed, the attacker portion of the environment is assumed to behave a certain way and some defense tactic is designed in response to that. Using a generative adversarial network allows us to get a real-like attack model using the generator that would be able to produce benign and malicious traffic network. At the same time, work on a possible IDS by using the trained discriminator which is able to distinguish between normal network packets and attack packets.

The use of ECC instead of RSA is a long-term solution since, even though RSA remains unbroken until this day, ECC will be able to adapt quicker for higher security levels whenever RSA is broken. [26] Smaller key sizes as well as a slower growth rate of key sizes allow for ECC to scale up its security without compromising performance. Current key sizes of ECC are nine times smaller than RSA’s keys. As key sizes grow, ECC keys would perform 20 or 30 times better than RSA keys. Likewise, BLAKE2 performance outmatches SHA3 hashing by allowing six times more mebibytes per second. This translates to hashing performing six times faster and allowing for faster response times in smart home devices.

The GAN is one of the most efficient types of neural networks for data generation. By standardizing this process, we can generate adversarial traffic for all type of network attacks. The bigger and more diverse the datasets created with this technique, it will be easier to train intrusion detection systems and have them be more effective. With IDS that are able to acknowledge whenever there is traffic that is out of the ordinary and most likely an attack, it would be able to alert the user to take action against this unknown traffic. Therefore, reducing the attacks that go unnoticed and that are able to steal data for long periods of time from the user without getting caught.
Chapter 6 – Proposed Hybrid Approach

In this chapter we will discuss the approach proposed to mitigate the risk of the previously mentioned attacks. We will be explaining both sections of our approach, the cryptographic approach, and the non-cryptographic approach. In the cryptographic approach, we used a hardware setup using a Raspberry Pi as a control hub. This microcontroller has several sensors connected to it and the communication is encrypted using Elliptic Curve Cryptography and BLAKE2 hashing.

In the non-cryptographic approach, we used a Generative Adversarial Network and we feed this neural network with network attack samples coming from a IEEE smart home network attacks database. This database was previously labeled, separating the normal traffic network packets and the attack packets. Once the GAN was trained with these packets, it was able to generate adversarial traffic that could be used to train an IDS.

6.1- Encryption experimentation

In order to test our encryption, approach we designed a hardware setup. This hardware setup has as a main control unit a Raspberry Pi 4. This Raspberry Pi has several sensors connected to it. We chose a motion sensor, a flame sensor, and a sense Hat which contains a temperature sensor, a barometric pressure sensor, a humidity sensor, and a LED screen. The Raspberry Pi is designed as a substitution to the control hub in a smart home, and all the sensors connected to it are simulating the smart devices connected to the control hub. Figure 6.1 shows the components connected to the Raspberry Pi.
The sense HAT covered all the pins in the Raspberry Pi; therefore, we had to use a pin extender and use the diagram shown in figure 6.2 to tell what pins could still be used over the sense HAT to connect the rest of the sensors. Then on figure 6.3 a picture of the setup with all the sensors and connectors can be seen. A breadboard was added to distribute 5V to all the sensors from the Raspberry Pi.
Figure 6.2 - Sense HAT pin usage diagram [27]

Figure 6.3 – Picture of hardware setup
Whenever a major change happens in the system, user authentication is requested to verify the correct user is managing the system. For example, if a flame is detected by the system, will generate an alert, and request a pin number will be requested. The user will enter a pin number using a keypad. This pin will be hashed using BLAKE2 and then fed into the ECDSA algorithm alongside the generated signature during the setup of the sensor to validate the pin entered. If the pin entered is the incorrect pin, then whenever we perform the hash function of the pin the outcome will be different, therefore the validation of the signature will not pass.

Figure 6.3 shows a graphical representation of the algorithm to validate the user’s pin. In this particular example a sensor x is connected to pin # 21, therefore during the setup of the sensor, the hash of number 21 alongside the private key generated is used as a message to create the digital signature using ECDSA. This algorithm uses a signature and the message to validate if the signature is valid or not, if the message is different than the original one then the signature will not be approved.

Figure 6.4 – User authentication algorithm using BLAKE and ECDSA
Using BLAKE2 and ECDSA allowed us to have a high security threshold while maintaining a low overhead and keeping response times pretty brief. We tested our hardware setup using the “hashlib” and “ECDSA” python libraries. The code related to this testing can be found on Appendix A. Figure 6.5 shows the output of the code in Appendix A.

![Code Output](image)

**Figure 6.5 – Appendix A code output**

In this code we have included timers to measure the time it takes for the hashing and signing portions of the code. This allows us to get performance metrics on the algorithms used. We took the timestamps where each of the events happened and subtracted them to get the time it took for each action to get computed. Table 6.1 shows the results which translate to the performance metrics of each step in each algorithm. In this table it can be seen how the time to generate the keys is only
0.046 seconds. The time to generate the signature is 0.0036 seconds. The time to hash the code input is 0.00049 seconds. Finally, the time to verify the signature using elliptic curve and the hash message is only 0.0318 seconds. Looking at these times we can see how even applying two encryption algorithms to the user authentication, it is still fast enough to keep a real-time experience with its user.

Table 6.1 – Algorithm performance metrics

<table>
<thead>
<tr>
<th>TIMESTAMPS CAPTURE</th>
<th>PERFORMANCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security system ready</td>
<td></td>
</tr>
<tr>
<td>PROGRAM STARTS:</td>
<td>4.29153E-06 seconds</td>
</tr>
<tr>
<td>KEYS GENERATED:</td>
<td>0.046298504 seconds</td>
</tr>
<tr>
<td>beginning signature:</td>
<td>0.04693222 seconds</td>
</tr>
<tr>
<td>SIGNATURE FINISHED:</td>
<td>0.050561428 seconds</td>
</tr>
<tr>
<td>Temperature= 33.9 C</td>
<td></td>
</tr>
<tr>
<td>Pressure= 881.6 milibars</td>
<td></td>
</tr>
<tr>
<td>Humidity= 30.7%</td>
<td></td>
</tr>
<tr>
<td>Motion detected!</td>
<td></td>
</tr>
<tr>
<td>Please enter the code:</td>
<td>15</td>
</tr>
<tr>
<td>CODE RECEIVED. HASH BEGINS:</td>
<td>5.876016378 seconds</td>
</tr>
<tr>
<td>HASH COMPLETE:</td>
<td>5.876508951 seconds</td>
</tr>
<tr>
<td>SIGN VERIFICATION STARTS:</td>
<td>5.878648996 seconds</td>
</tr>
<tr>
<td>Motion signature good</td>
<td></td>
</tr>
<tr>
<td>SIGNATURE VERIFIED:</td>
<td>5.910476446 seconds</td>
</tr>
<tr>
<td>TIME TO GENERATE KEYS:</td>
<td>0.046294 seconds</td>
</tr>
<tr>
<td>TIME TO GENERATE SIGNATURE:</td>
<td>0.003629 seconds</td>
</tr>
<tr>
<td>TIME TO HASH:</td>
<td>0.000493 seconds</td>
</tr>
<tr>
<td>TIME TO VERIFY SIGNATURE:</td>
<td>0.031827 seconds</td>
</tr>
</tbody>
</table>

Static Security Analysis

Many works in recent literature investigated on authentication and protection mechanisms in IoT environments. Secret key cryptosystems are easier to implement but have single point of failure and do not ensure non-repudiation. Therefore, most of the IoT applications focused on public key cryptosystems, but different from other applications IoT needs authentication and protection algorithms with low computation and communication overhead since they should be implemented on low power devices. ECC scheme-based authentication guarantee a high level of security. But static security analysis of ECC based applications shows the potential of breach when vulnerabilities inherent in the system are exploited. To enhance the complexity of existing ECC based authentication, we investigated on ECC based authentication with an additional level of
protection using BLAKE2 to make it resilient to the exploitable vulnerabilities in smart home environment. Deterministic and non-deterministic signatures with BLAKE2 make system resilient to the attacks (taking control, tampering, disruption) mentioned in smart home threat model, guarantee end-to-end protection without loss of information. The hybrid encryption based on ECC and BLAKE2 is effective compared to the existing ECC based authentication and is faster in computation.

6.2 Data Generation with GAN

Synthetic network traffic data generation is needed in order to create a vast and labeled dataset with packet information that can be used in order to train an IDS. To create data, we need data. The GAN needs some dataset from which it can learn and then generate its own fake data. For this task we looked into several different IoT datasets. The datasets we considered were:

The IEEE IoT dataset, the DARPA 1999 dataset, KDD 1999, and the KITSUNE dataset. The paper “A Generative Adversarial Approach for Packet Manipulation Detection” has a similar goal of synthetic data creation with GANs, however the dataset used by them is over 20 years old, they used the DARPA 1999 dataset. Since IoT in smart homes has greatly evolved in the past two decades, we decided to use a more recent dataset and selected the IEEE IoT dataset. This dataset consisted of 42 PCAP files, which contain network packet data. One file contains benign network traffic. The remaining 41 files contained network traffic which included different types of attacks towards an IoT device. These attacks included the following: syn-flooding, ack-flooding, host-brute-force, http-flooding, and UDP-flooding.

For the GAN to be able to learn, labels are needed indicating which are attack packets and which are normal traffic packets. In order to label the packets contained in the IEEE dataset, several steps had to be followed. First, we opened the PCAP files using Wireshark. Wireshark allows you to see the network packets and apply filters to them. We used Wireshark to apply a display filter with a rule provided by the IEEE dataset to separate the normal packets from the attack packets.
Then we saved those filtered packets into a CSV file where we added a new column. In this new column we added a 1 for the attack packets and a 0 for the benign packets. After the labeling, we filtered all the labeled packets by timestamp to reorganize the packets in their original order and saved this as a new CSV file with all the packets arranged and labeled. Figure 6.6 shows the process of labeling the packets in the PCAP files in order to be able to feed them into a GAN. Figure 6.7 shows an example of the final CSV file with that extra column labeling each network packet as normal or as an attack.

Figure 6.6 – Packet labeling process – Fig. modified from [28]
This final CSV file is fed into a GAN which learns the patterns in network data and understands how normal network traffic looks versus traffic where an attack on a smart device looks. After running the GAN on 5000 epochs, the neural network is able to produce packet data that behaves similar to the sample data provided to it. The code used to generate the synthetic data can be found on Appendix B. On this code the matplotlib is also used to generate a graphical representation of the packet distribution on different epochs of the GAN. Figure 6.8 shows the data distribution based on a DOS attack, specifically a SYN-flooding attack that was fed into the GAN. It can be seen how on zero epoch the data produced by the GAN is basically just random noise, however as epochs pass, the data distribution looks more and more similar to the actual data samples that were originally fed into the network.
Figure 6.8 – Data distribution of real data vs generated data on SYN-flooding attack packets
Conclusions

In conclusion, there is a need for better security and privacy techniques applied to smart home devices to prevent attacks like the ones described in our threat model. Our proposed cryptographic approach offers low overhead, high performance, low use of computing resources, and high security. This cryptographic approach performed remarkably during our hardware tests, having hashing and signature times of less than one tenth of a second combined. Elliptic Curve Cryptography alongside BLAKE2 hashing provides lightweight and secure techniques to protect user data while maintaining a real-time response.

A Generative Adversarial Network seems to be capable of learning network data patterns to distinguish between benign traffic and attack packets. On top of that whenever it was fed with SYN-flooding attack data, it was able to generate data that would approximate the behavior of an attack on a smart home network. This data can be used further to train an Intrusion Detection System that could outperform the IDS that are currently trained with limited attack samples.

In future work, the discriminator portion of the IDS could be trained with the generated samples of the generator and its performance should be tested against current detection systems. Other datasets should also be explored, to expand the attack variety for the GAN to learn different types of attacks and its discriminator be able to get an even better performance. The cryptographic approach based on ECC and BLAKE2 could be extended to test the performance and resilience to vulnerabilities in a dynamic control environment.
References


Glossary

Cybersecurity - the state of being protected against the criminal or unauthorized use of electronic data, or the measures taken to achieve this.

False Negative - is a test result which wrongly indicates that a condition does not hold.

Overhead - any combination of excess or indirect computation time, memory, bandwidth, or other resources that are required to perform a specific task.

Request - request methods are the assets that indicate the specific desired action to be performed on a given resource.

Sniffing - is a process of monitoring and capturing all data packets passing through given network.

Spoofing - is the act of disguising a communication from an unknown source as being from a known, trusted source.

Synthetic data - is information that's artificially manufactured rather than generated by real-world events.
Appendix A

ENCRYPTION WITH BLAKE2 AND ECDSA USING RASPBERRY PI 4 AND SENSORS

```python
#SenseHAT with motion sensor, flame sensor and keypad
from sense_hat import SenseHat
import time
import RPi.GPIO as GPIO
from keypadBCM import keypad
from hashlib import blake2b
import board
from ecdsa import SigningKey, VerifyingKey, BadSignatureError

sense = SenseHat()
sense.clear()

GPIO.setmode(GPIO.BCM)
PIRpin = 15  #motion sensor - BOARD PIN 10 - GPIO PIN 15
GPIO.setup(PIRpin, GPIO.IN)
FLAMEpin = 21  #flame sensor - BOARD PIN 40 - GPIO PIN 21
GPIO.setup(FLAMEpin, GPIO.IN)

if __name__ == '__main__':
    # Initialize
    kp = keypad(columnCount = 3)

    # password = "123#"
    passwordhash =
    "0f31952b34795e0dbdd8c698a13779e407dbd0f4c967f2cf38f3d1d34f0544ec7816bfc06f6d9840df9f91d504d70c870c3766cb9b42731c8e47841b29bd25"
    #flame pin number = 21
    flamehash =
    "8c715c0b894785852fbc391d6e2131bf0f0c703852f25b1c07429f356d7e48df5998ac0df4f1ef7019ebfda0877f79d6b91c1b98084efbb7314258608c"  #hash of 21
    #motion pin number = 15
    motionhash =
    "4a9bb12a4834e77430779ea675900f4e84b5a8b9400a67b81985cd4b85e0a28b5d6b59f896cc72cd6aad3390b51b2c76d4eb8f8d0ce205f425697e5180b35ae"  #hash of 15

print('Security system ready')
start_time = time.time()
#printing for performance metrics
print("PROGRAM STARTS:--- %s seconds ---" % (time.time() - start_time))
```
sk = SigningKey.generate()
print("KEYS GENERATED:--- %s seconds ---" % (time.time() -
start_time)) #Time to generate key pair
vk = sk.verifying_key
# with open("private.pem", "wb") as f:
#   f.write(sk.to_pem())
# with open("public.pem", "wb") as f:
#   f.write(vk.to_pem())

with open("messageFlame.txt", "rb") as f:
    messageFlame = f.read()
    print("beginning signature:--- %s seconds ---" %
(time.time() - start_time))
sig = sk.sign_deterministic(messageFlame)
print("SIGNATURE FINISHED:--- %s seconds ---" % (time.time() -
start_time)) #Time to generate signature
with open("signatureFlame", "wb") as f:
    f.write(sig)

with open("messageMotion.txt", "rb") as f:
    messageMotion = f.read()
    sig = sk.sign_deterministic(messageMotion)
with open("signatureMotion", "wb") as f:
    f.write(sig)

def PINflame():
    print("Please enter the code:")

    ####### 2 Digit wait #######
    code = ""
    for i in range(2):
        digit = None
        while digit == None:
            digit = kp.getKey()
        code = code + str(digit)
    print(code)
    time.sleep(0.4)

    print("CODE RECEIVED, HASH BEGINS:--- %s seconds ---" %
(time.time() - start_time))
    codeBytes = bytes(code, 'utf-8')
    h = blake2b()
    h.update(codeBytes)
    print("HASHED FINISHED:--- %s seconds ---" % (time.time() -
start_time)) #Time to compute the hash
    usercode = h.hexdigest()
    with open("userFlame.txt", "w") as f:
with open("messageFlame.txt", "rb") as f:
    message = f.read()
print("SIGN VERIFICATION BEGINS:--- %s seconds ---" % (time.time() - start_time))
with open("signatureFlame", "rb") as f:
    sig = f.read()

with open("userFlame.txt", "rb") as f:
    messageuser = f.read()
try:
    vk.verify(sig, messageuser)
    print("Flame signature good")
    print("SIGNATURE VERIFIED: --- %s seconds ---" % (time.time() - start_time))  # Time to verify the signature
except BadSignatureError:
    print("FLAME SIGNATURE FAILED")

def PINmotion():
    print("Please enter the code:")

        2 Digit wait
    code = ""
    for i in range(2):
        digit = None
        while digit == None:
            digit = kp.getKey()
        code = code + str(digit)
    print(code)
    time.sleep(0.4)
    print("CODE RECEIVED- HASH BEGINS:--- %s seconds ---" % (time.time() - start_time))
    codeBytes = bytes(code, 'utf-8')
    h = blake2b()
    h.update(codeBytes)
    print("HASH COMPLETE:--- %s seconds ---" % (time.time() - start_time))
    usercode = h.hexdigest()
    with open("userMotion.txt", "w") as f:
        f.write(usercode)

with open("messageMotion.txt", "rb") as f:
# Function for flame interrupt

def FLAMEcallback(FLAMEpin):
    print("Flame detected!\n")
    print(FLAMEpin)
    time.sleep(1)
    PINflame()

GPIO.add_event_detect(FLAMEpin, GPIO.BOTH, bouncetime=5000)  # let us know when the pin goes HIGH or LOW
GPIO.add_event_callback(FLAMEpin, FLAMEcallback)  # assign function to GPIO PIN, Run function on change

# Function for motion sensor interrupt

def MOTIONcallback(PIRpin):
    print("Motion detected!\n")
    print(PIRpin)
    time.sleep(1)
    PINmotion()

GPIO.add_event_detect(PIRpin, GPIO.BOTH, bouncetime=5000)  # let us know when the pin goes HIGH or LOW
GPIO.add_event_callback(PIRpin, MOTIONcallback)  # assign function to GPIO PIN, Run function on change

# Infinite loop for the system to always keep and printing the temperature, pressure and humidity on the LED screen.
while True:
time.sleep(1)
# Take readings from all three sensors
t = sense.get_temperature()
p = sense.get_pressure()
h = sense.get_humidity()

# Round the values to one decimal place
    t = round(t, 1)
p = round(p, 1)
h = round(h, 1)

print("Temperature= " + str(t) + " C")
print("Pressure= " + str(p) + " milibars")
print("Humidity= " + str(h) + "%")

    # Create the message
    # str() converts the value to a string so it can be
    # concatenated

message = "Temperature: " + str(t) + "C  Pressure: " + str(p) + "mb  Humidity: " + str(h) + "%"

# Display the scrolling message
    sense.show_message(message, scroll_speed=0.05)

    sense.clear()

    time.sleep(10)
Appendix B

SYNTHETIC DATA GENERATION USING GENERATIVE ADVERSARIAL NETWORKS

import pandas as pd
import numpy as np
import os
import tensorflow as tf
from tensorflow.keras.layers import Input, Dense, Dropout
from tensorflow.keras import Model
from tensorflow.keras.optimizers import Adam

df = pd.read_csv('csv/dos-synflooding-1-dec-mix.csv')
df.drop(['Attack', 'No.', 'Time', 'Source', 'Destination', 'Protocol', 'Length', 'Info'], axis=1, inplace=True)
print(df.columns)

file_name = "csv/dos-synflooding-1-dec-mix.csv"
columns_to_drop = ['Info']
categorical_features = ['Attack', 'Source', 'Destination', 'Protocol', 'Length']
continuous_features = ['No.', 'Time']

# training configuration
noise_dim = 32
dim = 128
batch_size = 32
log_step = 100
epochs = 5000+1
learning_rate = 5e-4
models_dir = 'model'

df = pd.read_csv(file_name)
df.drop(columns_to_drop, axis=1, inplace=True)
print(df.columns)

for column in categorical_features:
    df[column] = df[column].astype('category').cat.codes

df.head()

for column in continuous_features:
    min = df[column].min()
max = df[column].max()
feature_bins = pd.cut(df[column], bins=np.linspace(min, max, 21), labels=False)
df.drop([column], axis=1, inplace=True)
df = pd.concat([df, feature_bins], axis=1)
print(df)

from sklearn.preprocessing import PowerTransformer

df[df.columns] = PowerTransformer(method='yeo-johnson',
standardize=True, copy=True).fit_transform(df[df.columns])
print(df)
pw = PowerTransformer(method='yeo-johnson', standardize=True, copy=True)
pwt = pw.fit_transform(df[df.columns])
print(df)
df[df.columns] = pwt

class GAN():
    def __init__(self, gan_args):
        [self.batch_size, lr, self.noise_dim, 
        self.data_dim, layers_dim] = gan_args

        self.generator = Generator(self.batch_size).
            \build_model(input_shape=(self.noise_dim,),
        dim=layers_dim, data_dim=self.data_dim)

        self.discriminator = Discriminator(self.batch_size).
            \build_model(input_shape=(self.data_dim,),
        dim=layers_dim)

        optimizer = Adam(lr, 0.5)

        # Build and compile the discriminator
        self.discriminator.compile(loss='binary_crossentropy',
            optimizer=optimizer,
            metrics=['accuracy'])

        # The generator takes noise as input and generates imgs
        z = Input(shape=(self.noise_dim,))
        record = self.generator(z)
# For the combined model we will only train the generator
self.discriminator.trainable = False

# The discriminator takes generated images as input and determines validity
validity = self.discriminator(record)

# The combined model (stacked generator and discriminator)
# Trains the generator to fool the discriminator
self.combined = Model(z, validity)
self.combined.compile(loss='binary_crossentropy', optimizer=optimizer)

def get_data_batch(self, train, batch_size, seed=0):
    # random sampling - some samples will have excessively low or high sampling, but easy to implement
    np.random.seed(seed)
    x = train.loc[np.random.choice(train.index, batch_size)].values
    # iterate through shuffled indices, so every sample gets covered evenly
    start_i = (batch_size * seed) % len(train)
    stop_i = start_i + batch_size
    shuffle_seed = (batch_size * seed) // len(train)
    np.random.seed(shuffle_seed)
    train_ix = np.random.choice(list(train.index), replace=False, size=len(train))  # wasteful to shuffle every time
    train_ix = list(train_ix) + list(train_ix)  # duplicate to cover ranges past the end of the set
    x = train.loc[train_ix[start_i: stop_i]].values
    return np.reshape(x, (batch_size, -1))

def train(self, data, train_arguments):
    [cache_prefix, epochs, sample_interval] = train_arguments

    data_cols = data.columns

    # Adversarial ground truths
    valid = np.ones((self.batch_size, 1))
    fake = np.zeros((self.batch_size, 1))
for epoch in range(epochs):
    # ---------------------
    #  Train Discriminator
    # ---------------------
    batch_data = self.get_data_batch(data, self.batch_size)
    noise = tf.random.normal((self.batch_size, self.noise_dim))

    # Generate a batch of new images
    gen_data = self.generator.predict(noise)

    # Train the discriminator
    d_loss_real =
    self.discriminator.train_on_batch(batch_data, valid)
    d_loss_fake =
    self.discriminator.train_on_batch(gen_data, fake)
    d_loss = 0.5 * np.add(d_loss_real, d_loss_fake)

    # ---------------------
    #  Train Generator
    # ---------------------
    noise = tf.random.normal((self.batch_size, self.noise_dim))
    # Train the generator (to have the discriminator label samples as valid)
    g_loss = self.combined.train_on_batch(noise, valid)

    # Plot the progress
    print("%d [D loss: %f, acc.: %.2f%%] [G loss: %f]" %
          (epoch, d_loss[0], 100 * d_loss[1], g_loss))

    # If at save interval => save generated events
    if epoch % sample_interval == 0:
        # Test here data generation step
        # save model checkpoints
        model_checkpoint_base_name = 'model/' +
        cache_prefix + '_{}model_weights_step_{}.h5'

        self.generator.save_weights(model_checkpoint_base_name.format('generator', epoch))

        self.discriminator.save_weights(model_checkpoint_base_name.format('discriminator', epoch))

        # Here is generating the data
        z = tf.random.normal((432, self.noise_dim))
gen_data = self.generator(z)
print('generated_data')

def save(self, path, name):
    assert os.path.isdir(path) == True, \\
    "Please provide a valid path. Path must be a directory."
    model_path = os.path.join(path, name)
    self.generator.save_weights(model_path)  # Load the generator
    return

def load(self, path):
    assert os.path.isdir(path) == True, \\
    "Please provide a valid path. Path must be a directory."
    self.generator = Generator(self.batch_size)
    self.generator = self.generator.load_weights(path)
    return self.generator

class Generator():
    def __init__(self, batch_size):
        self.batch_size = batch_size

        def build_model(self, input_shape, dim, data_dim):
            input = Input(shape=input_shape,
            batch_size=self.batch_size)
            x = Dense(dim, activation='relu')(input)
            x = Dense(dim * 2, activation='relu')(x)
            x = Dense(dim * 4, activation='relu')(x)
            x = Dense(data_dim)(x)
            return Model(inputs=input, outputs=x)

class Discriminator():
    def __init__(self, batch_size):
        self.batch_size = batch_size

        def build_model(self, input_shape, dim):
            input = Input(shape=input_shape,
            batch_size=self.batch_size)
            x = Dense(dim * 4, activation='relu')(input)
            x = Dropout(0.1)(x)
            x = Dense(dim * 2, activation='relu')(x)
            x = Dropout(0.1)(x)
            x = Dense(dim, activation='relu')(x)
            x = Dense(1, activation='sigmoid')(x)
            return Model(inputs=input, outputs=x)
data_cols = df.columns

# Define the GAN and training parameters
df[data_cols] = df[data_cols]

print(df.shape[1])

gan_args = [batch_size, learning_rate, noise_dim, df.shape[1], dim]
train_args = ['', epochs, log_step]

model = GAN

# Training the GAN model chosen: Vanilla GAN, CGAN, DCGAN, etc.
synthesizer = model(gan_args)
synthesizer.train(df, train_args)

# You can easily save the trained generator and load it afterwards

synthesizer.save('model/gan/saved', 'generator_patients')

synthesizer.discriminator.summary()

models = {'GAN': ['GAN', False, synthesizer.generator]}

# Setup parameters visualization parameters
seed = 17

import matplotlib.pyplot as plt

# number of fraud cases
noise_dim = 32

test_size = 492

np.random.seed(seed)
z = np.random.normal(size=(test_size, noise_dim))

real = synthesizer.get_data_batch(train=df, batch_size=test_size, seed=seed)
real_samples = pd.DataFrame(real, columns=data_cols)

model_names = ['GAN']

models = {'GAN': ['GAN', False, synthesizer]}
```python
# Actual fraud data visualization
model_steps = [0, 100, 200, 300, 400, 500, 1000, 2000, 3000, 4000, 5000]
rows = len(model_steps)
columns = 5
axarr = [[]]*len(model_steps)

fig = plt.figure(figsize=(14, rows*3))

for model_step_ix, model_step in enumerate(model_steps):
    axarr[model_step_ix] = plt.subplot(rows, columns, model_step_ix*columns + 1)
    for group, color, marker in zip(real_samples.groupby('Protocol'), colors, markers):
        plt.scatter(group[1][[col1]], group[1][[col2]], marker=marker, edgecolors=color, facecolors='none')
        plt.title('Actual Network Traffic')
        plt.xlabel(col1) # Only add y label to left plot
        plt.ylabel(col2)
        xlims, ylims = axarr[model_step_ix].get_xlim(), axarr[model_step_ix].get_ylim()

    if model_step_ix == 0:
        legend = plt.legend()
        legend.get_frame().set_facecolor('white')

    i=0
    [model_name, with_class, generator_model] = models['GAN']
    generator_model.load_weights( base_dir + '_generator_model_weights_step_' + str(model_step) + '.h5')
    ax = plt.subplot(rows, columns, model_step_ix*columns + 1 + (i+1))
    g_z = generator_model.predict(z)
gen_samples = pd.DataFrame(g_z, columns=data_cols)
    # gen_samples.to_csv('Generated_sample.csv')
    plt.scatter(gen_samples[[col1]], gen_samples[[col2]], marker=markers[0], edgecolors=colors[0], facecolors='none')
    plt.title("Generated Data")
    plt.xlabel(data_cols[0])
```
```python
ax.set_xlim(xlims), ax.set_ylim(ylims)
plt.suptitle('Comparison of GAN outputs', size=16, fontweight='bold')
plt.tight_layout(rect=[0.075,0,1,0.95])

# Adding text labels for traning steps
vpositions = np.array([i._position.bounds[1] for i in axarr])
vpositions += ((vpositions[1] - vpositions[1]) * 0.35)
for model_step_ix, model_step in enumerate(model_steps):
    fig.text(0.05, vpositions[model_step_ix], 'training\nstep\n' + str(model_step), ha='center', va='center', size=12)

plt.savefig('Comparison_of_GAN_outputs.png')

print("g_z before inverse transform", g_z)
g_z=pw.inverse_transform(g_z)
print("g_z after inverse transform", g_z)
gen_samples = pd.DataFrame(g_z, columns=data_cols)
gen_samples.to_csv('Generated_sample.csv')
gen_samples.drop('Unnamed: 0', axis=1, inplace=True)
print(gen_df.columns)
print(df.shape, gen_df.shape)

from table_evaluator import load_data, TableEvaluator
print(len(df), len(gen_samples))
table_evaluator = TableEvaluator(df, gen_samples)
table_evaluator.visual_evaluation()
Vita

Francisco Javier Candelario Burgoa was born in Ciudad Juarez, Mexico where he studied through high school. During middle school and high school, he participated and then was an organizer of the Math Olympics in the state of Chihuahua, Mexico. He graduated valedictorian from “Colegio San Patricio” high school. Thanks to this he was awarded the University of Texas at El Paso (UTEP) Presidential Excellence Scholarship. Mr. Candelario pursued a degree in Electrical Engineering at UTEP. During these four years, Mr. Candelario was a member of the University Honors Program, the Institute of Electrical and Electronic Engineers (IEEE), the Society of Hispanic Professional Engineers (SHPE) and became an officer of Engineers for a Sustainable World (ESW).

Mr. Candelario graduated from UTEP during the Fall 2020. Mr. Candelario graduated as an honors student having completed multiple honor courses and projects, graduated as Summa Cum Laude and was also awarded the Distinguished Senior Award upon graduation. As an undergraduate student, Mr. Candelario participated in UTEP’s Fast Track program, starting with his master’s degree coursework, where he took a computer architecture course and met who later became his mentor and advisor Dr. Sai Mounika Errapotu. Mr. Candelario started his cybersecurity research activities with Dr. Errapotu on Spring 2020 where he became more knowledgeable on neural networks and encryption algorithms.

During his master’s degree, he was awarded the Texas Instruments Scholarship and maintained a 4.0 GPA. In 2020 and 2021, Mr. Candelario participated in two internships as a software engineer with Cummins Inc. Mr. Candelario will join Cummins in January of 2022 as a Tool Component Development Engineer.

Contact email: candelarioburgoajavier@gmail.com