A Novel Simplified Framework to Secure IoT Communications

Sairath Bhattacharjya[1,2] and Hossein Saiedian[1,2]
[1]*Electrical Engineering & Computer Science, The University of Kansas, Lawrence, Kansas, U.S.A.*
[2]*Information & Telecommunication Technology Center, The University of Kansas, Lawrence, Kansas, U.S.A.*

Keywords: IoT, Security, Zero-trust, Key Generation, Plug-and-Play, Elliptic Curve Cryptography (ECC), Zero Interaction Pairing (ZIP), Zero-Interaction Authentication (ZIA), Communication.

Abstract: IoT devices are already in the process of becoming an essential part of our everyday lives. These devices specialize in performing a single operation efficiently. To maintain the privacy of user data, securing communication with these devices is essential. The plug-pair-play (P3) connection model uses the ZIP (zero interaction pairing) technique to set up a secured key for every pair of user and device so that the user doesn't have to remember a complicated password. The command execution model provides an authentication mechanism for every transaction. Routing the transactions through the gateway allows for auditing, providing a zero-trust environment. The zero-trust (ZT) model described in this paper addresses confidentiality, integrity, and authentication triad of cybersecurity while ensuring that the interactions with these devices are seamless. The architecture described in this article makes security a backbone. The model described in this paper provides an end-to-end framework to secure the communication with these smart devices in a cloud-based architecture respecting the resource limitation of these devices. A novel simplified framework to secure IoT communication.

1 INTRODUCTION

Internet of Things (IoT) have changed the direction to modern technological development. With its intrusive nature it has already penetrated our lives with wearable devices and smart objects for home automation systems. These devices are dealing with our personal information as well as performing micro transactions to make our lives easier. With this advantage comes the concern of privacy. It is imperative to say that we need a way to securely communicate with these devices.

Users have voiced their privacy concern with using these devices. To remove any doubt on this fact, there have been numerous experiments to prove that these devices can be easily hacked with readily available equipment (Ronen and Shamir, 2016). In many implementations it is seen that the manufacturers delegate the responsibility of securing the devices in the hand of the user by providing default credentials and expecting the user to change it. Such implementations have been heavily exploited by malware like Mirai and EchoBot to convert them into bots. The author in this article (Uslaner, 2004) talks about the concept of trust which very much applies to IoT ecosystems. Eliminating trust will help us focus more towards privacy and security.

In this paper we focused on a techniques to setup an end-to-end security framework for IoT devices in a cloud-based architecture. Concepts like fog computing has brought the cloud closer to the appliances and services (Puliafito et al., 2019). It has also portrayed that the bulk of the heavy lifting of computing and data processing can be shifted to the cloud to account for the limitation of resources in the devices. In this model, the *plug-pair-play* (P3) connection model sets up a secret key for every pair of user and device automatically without user's intervention. It uses the zero interaction pairing (ZIP) technique to avoid the user form remembering complicated passwords or the system having to use default credentials. The command execution model uses these unique keys to setup a zero-trust channel of communication between them over the untrusted internet. The heartbeat ensures that the device is alive and functional at all times. Overall, using this model, security can act as a backbone for any design architecture for these IoT devices. The model is flexible and can be used with a plethora of device types. It works in the background and does not provide additional overhead to the users to enable security of the device. In cybersecurity, the CIA triad addresses the three crucial aspects, namely confidentiality, integrity and availability. This model address all these aspects and ensures a secured functioning of the device.

Bhattacharjya, S. and Saiedian, H.
A Novel Simplified Framework to Secure IoT Communications.
DOI: 10.5220/0010210003990406
In *Proceedings of the 7th International Conference on Information Systems Security and Privacy (ICISSP 2021)*, pages 399-406
ISBN: 978-989-758-491-6

This article is organized as follows. We start by exploring the common security issues in IoT devices and how they are being exploited in Section 2. Here we also look into the potential solutions provided by the research community to secure IoT communications. Then we explain our security model in details in Section 3 and talk briefly about the implementation setup before exploring the performance of the model in terms of data security, memory utilization and time of operation in Section 4. We conclude with our findings and opportunity for future research in Section 5.

2 SECURITY ISSUES OF THE DEVICES

From the business reports we see a phenomenal growth in the market of IoT devices (Goasduff, 2019; Columbus, 2019). It is predicted that there will be 5.8 billion IoT endpoints by the end of this year and by 2022 the worldwide technology spending on smart devices would reach USD 1.2 trillion. The advent of modern technologies like artificial intelligence, machine learning and real time data streaming combined with the high-speed connectivity with the cloud helped businesses look at these devices as a potential solution to their specific problems. More and more organizations are relying on them to remodel and optimize their business needs.

With this unprecedented growth in demand for these smart objects, manufacturers are not getting enough time to perform adequate security testing on them. It is noticed that smaller players are not even providing options to patch the vulnerabilities. These issues are taken advantage of by attackers. Malware like Mirai are using these loop holes to convert these devices into botnets which are used to cause massive DDoS attacks (Bertino and Islam, 2017). At its peak, Mirai caused a 1.1 Tbps attack using 148,000 IoT devices. With its source code made public, the number of infected devices has doubled. The attack on Dyn Inc. DNS servers in 2016 is one of the most notable attack using IoT botnet.

Extensive surveys have been conducted to identify the security issues in IoT devices that lead to these massive attacks. One study noted that in ZigBee Light Link (ZLL) based connected lighting system manufacturers rely on an NDA (non-disclosure agreement) protected shared key to secure communications. Here are the common vulnerabilities of IoT devices that makes it a easy target for attackers (Neshenko et al., 2019; van Oorschot and Smith, 2019; Trappe et al., 2015; Bhattarai and Wang, 2018).

- **Resource Limitation.** Every research article pointed out that constrained resources in the device is a setback when it comes to implementing cryptographic techniques. It is possible that an attacker might drain the memory of the device by sending thousands of requests to the open port in a device.
- **Lack of User Authentication.** lack of available memory in the device restricts the implementation of complex authentication techniques. Thus, to maintain the legal standards, manufacturers end up using default credentials and common shared keys.
- **Inadequate Encryption.** Encryption is an effective tool to defuse the data so that an unauthorized user is not able to make sense of it. Cryptographic systems depend on the randomness of the algorithm as well as the key size to effectively morph the data. Due to insufficient storage in the device, it becomes difficult to store large keys. This is taken advantage of by the adversary who perform brute force attack to break a smaller key size.
- **Efficient Access Control.** It is often noticed that a proper access control mechanism is not maintained on the device. Many manufacturers allow use of default credentials on the device and the same user is entrusted with admin privileges on it. With higher privilege on them, the attacker can perform more damage not only to the device, but also to the network that they are installed in.

2.1 Proposed Solutions to Bridge the Gap

Creating an identity of a device and its user in a cloud-based architecture is essential in a heterogeneous ecosystem. It forms the baseline to tackle all the security issues that we have pointed out in the previous section. Researchers have taken different perspective to solve the issue of identity.

Bluetooth Low Energy (LE) can play a important role in securing the IoT devices. One research showed the potential of using IPv6 over Bluetooth LE (Nieminen et al., 2014). Wireless communication with the device is protected using the Bluetooth LE Link Layer security which supports both encryption and authentication by using the Cipher Block Chaining Message Authentication Code (CCM). OpenConnect was proposed to automate the integration of these devices in a cloud-based architecture (Pazos et al., 2015). The platform uses REST API endpoints to integrate the devices with the central command center. Security of the implementation is placed inside the integration

service. Another research showed an approximation arithmetic computer-based information hiding technique to provide features like IP watermark, digital fingerprinting and lightweight encryption for ensuring energy efficiency to low power equipments (Gao et al., 2017).

Some solutions have been proposed to effectively tackle the authentication issue for the resource constrained devices. A certificate-based authentication technique was proposed to redress the issue of password-based authentication (Atwady and Hammoudeh, 2017). A certificate is awarded to every entity in the system by a trusted certification authority. Another solution was proposed to use a One Time Password (OTP) scheme using elliptic curve cryptography. This solution depends on the Lamport algorithm to secure the generate OTP. Authentication of smart devices using the physical properties of the device for smart home environment was provided as a potential solution (Huth et al., 2015). The security mechanism used in this technique uses a random set of challenges along with symmetric key cryptography.

3 THE ZERO-TRUST (ZT) MODEL TO SECURE COMMUNICATION

Each proposal by the research community provided a unique perspective to the solution. Bluetooth LE becomes effective for low energy devices and provides a much smaller attack vector being a PAN (personal area network) network. Similarly, public key cryptography helps provide an identity for an entity in a network. The private-public key pair helps provide authentication as well as integrity check to messages send between two parties. In our proposed solution we combined these ideas to generate an efficient solution that would work for any resource constrained device.

In a cloud-based architecture, there are three primary components in the IoT ecosystem, namely, device, user, and gateway.

- **Device** represents the endpoint that specializes in performing a specific task (which we also refer to as an IoT device).
- **User** provides the commands and instructions to the device. In our implementation, we have used a mobile app to work as a user interface. In this model, we have categorized the user group into users and delegates. Each device can have at most one user who is the primary owner and has total control over it. The delegates represent other users to the device, like another person or a home assistant like Google Home or Amazon Alexa. They can access the device only when the user approves the pairing. The user has the right to grant access to a delegate to perform specific operations on a device.
- **Gateway** works as a middleman helping the user and device to communicate with each other over the internet. It consists of API endpoints that coordinate the communications between them. It also acts as a data store to record the details about the user, device, registration, and transaction logs. The gateway takes away the computation and memory-intensive operations like data analytics and forensics away from the device.

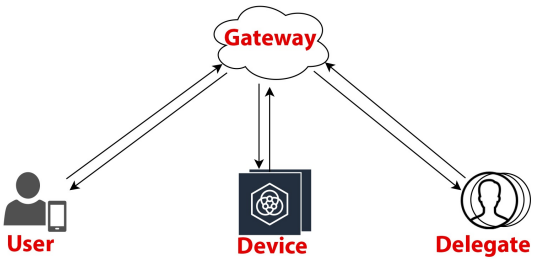

Figure 1: Proposed architecture for IoT ecosystem.

Figure 1, shows the different entities in the proposed architecture. Once the device and user are paired with one another using the P3 connection model, all further communications between them get routed through the gateway for logging the transactions.

3.1 Prebuilt Security in the Model

As described in the architecture, the user is responsible for providing commands and instructions to the device. To enable the user to perform its function, the framework comes with a few prebuilt security mechanisms. The user has to be registered with the gateway for the gateway to know their identity. All post-logon operations are to be accompanied by an Authentication header that contains a JWT (JSON web token) token to verify the identity of the user. All transactions to and from the gateway are protected by TLS/SSL. This ensures data security in transit.

To provide authentication to the device and gateway to one another, they are enabled with their own public-private key pairs. To respect the resource limitation of the device, we used elliptic curve cryptography (ECC) for the implementation. ECC is a preferred choice for public key cryptography rather than RSA because of the key size. A 32 bytes key can provide the same level of security as the 2048 byte RSA

key. The operational time for signing and verification is comparable. The private key is used to verify the identity of the device to the gateway:

<device_id, current_timestamp, raw_data>
→ data
<data, Enc{H(data), $PrivKey_{device}$}> →
package

The raw_data along with the device_id and the current_timestamp of the device forms the data to be sent to the gateway. The data is hashed and signed using the private key of the device. This provides both authentication as well as integrity check on the data since the private key is only available to the device. The timestamp provides protection against replay attacks. The gateway holds the public key of the device. On receiving the package, it extracts the data and verifies the given signature to make sure it is from the device who it claims to be. The same technique is used when sending information from the gateway to the device.

3.2 Plug-pair-play (P3) Connection Model

The P3 connection model falls in the category of the zero interaction pairing (ZIP) (Fomichev et al., 2019). The ZIP technique provides many advantages over the traditional password based security. Due to the lack of user interactions, this scheme can be implemented at large scale for a many-to-many combinations of users and devices. The P3 connection model requires the user to provide the WiFi credentials to the device and post that the secrets are created autonomously without any user intervention. The technique described in this paper helps distinguish each pair of user and device from another.

3.2.1 Setting up Shared Key for User

The user and delegates of a IoT device can change frequently. It is necessary to generate a key on the first instance the user wants to interact with the device. This avoids the need for a password based authentication. The shared key can be used to secure all future communications between the user and device pair. The same technique can be used to refresh the key at a regular interval. Figure 2 shows the steps to validate the identity of the user and device to one another and setting up the shared key.

For connection initiation, we propose using Bluetooth 4.0 or Bluetooth LE (Nieminen et al., 2014). Bluetooth LE has been designed for ultra-low power application yet keeping similarities with classic Bluetooth. All modern mobile phones and smart devices are enabled with Bluetooth LE. Another reason to use Bluetooth in setting up secret keys is the area of access. Since the Bluetooth connection can be established only in proximity of the device, the attack vector becomes smaller. Here are the steps of the P3 connection model for user:

- **Pairing.** The first step of the connection is the pairing between the user and the device. The user from his mobile app searches to find available device. In Bluetooth the connection happens between a master and a slave. In this case, the user's phone acts as a master and the device acts as a slave. Once the user finds the device, it pairs with it using the default pairing key embedded in the app and initiates a connection.
- **Generate Session Key.** Curve25519 is an elliptic curve offering 128 bit of security and designed for use with the elliptic curve Diffie-Hellman (ECDH) key arrangement scheme. Here, both the device and user, generate a key and share the public part with one another. Both generate the session key K_s using Diffie-Hellman and use it to secure the remaining transactions during the process.
- **Connect to Wi-Fi.** Once the session key is established, the next step is for the device to connect to the internet. For this, the user sends the Wi-Fi SSID and password encrypted with the session key Enc{<Wi-Fi SSID, Wi-Fi password>, K_s}. On receiving this information, the device tries to connect to the internet and ensures a successful connection. Once connected, it saves the information into its memory till the entire process terminates. It returns a "success" to the user.
- **User Verification.** After connecting to the internet, the device needs to verify the identity of the user. The user sends his user_id to the device encrypted Enc{user_id, K_s}. The devices send this identifier to the gateway along with the device's digital signature for verification. On receiving this information, the gateway ensures the validity of both the device and the passed user identifier. On successful verification, it creates a partial registration record.
- **Device Verification.** On receiving a green light from the gateway, the device returns a device_mac to the user encrypted with the prior session key Enc{device_mac, K_s}. The user forwards this information to the gateway along with the JWT token for user identity. The gateway verifies the user and then checks the device_mac to verify it against the partial verified registration

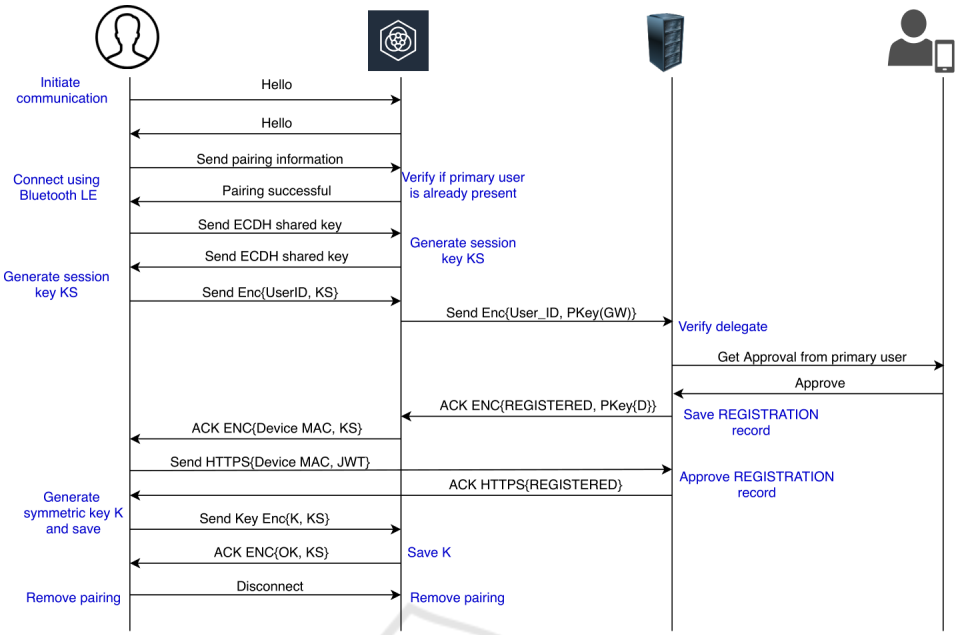

Figure 2: P3 connection between user and device.

record. Once verified, the gateway completes the transaction and returns success to the user.

- **Generate and Share the Symmetric Key.** On receiving a positive response back, the user generates 256 bits symmetric key along with a 128 bits initialization vector, saves it locally and shares it with the device $\texttt{Enc}\ \{K, K_s\}$. The device saves the same along with the user identifier and acknowledges the user that the key is saved securely. The device also saves the WiFi credentials in permanent storage.
- **Disconnect.** The Bluetooth interface is only used to help connect and verify the user and device. Once this connection is established, there is no need to hold on to the connection. The user initiates a termination and the device comply.

As mentioned before the gateway acts as a data store. It saves the registration record for command execution. After the shared key is generated and saved by the device and user, the future communications can be secured using this key. When the transaction goes over the internet, the gateway acts as a middleman to connect the two parties. In doing so, the gateway verifies the validity of the transaction using the registration record that was generated during the P3 connection model. The registration record provides an access control on who can access which device.

3.2.2 Setting up Shared Key for Delegate

The user is the primary owner of the device. In the previous section we described the process when the device is being connected for the first time and there are no prior users added to the device registration. Here we will describe the situation where the device is already registered with the primary user. When another user or device wants to communicate with the device, it is important for the primary user to be aware of it. The P3 connection model accounts for this scenario.

The steps for a delegate to connect to the device is detailed in Figure 3. The steps are similar to the connection with a user. The only step that is different is the user verification. When a delegate initiates a connection with a device and the device sends the user's identifier to the gateway for verification, the gateway checks the registration records and finds that there is a primary user already assigned to the device. The gateway notifies the user in the app asking for the approval to create the partial registration record. If the user approves, the transaction continues the same as for the user. If the user rejects, the transaction is terminated.

This approach gives an option for the user to intervene as to who can talk to the device. This ensures access control on the device. From the perspective of the delegate, they come to know that they are not the primary user in the response and whether it is approved by the owner or user. On evolving on this

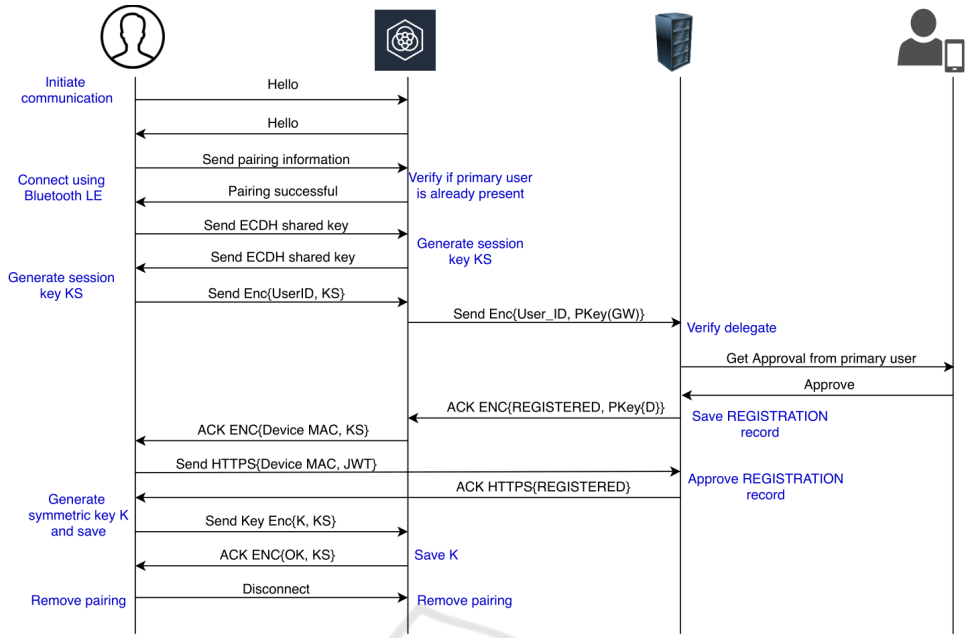

Figure 3: P3 connection between delegate and device.

idea, the user can define a role-based permission as to which delegate can perform which operation. We will not delve deeper into this idea since it is beyond the scope of this paper.

The secret key that is generated identifies each pair of user and device. This process eliminates the need to have a default credential or common predefined secret. This process works in the background and the user doesn't have to provide or remember any additional details to enable security. It also plays well with the plug-and-play model that the users are well accustomed with.

3.3 Command Execution on Device

The internet is an untrusted medium. When a communication flows from one system to another, it goes through multiple routers and it is practically impossible to secure each and everyone of them from being wiretapped. So, to maintain confidentiality of information between each pair of user and device, we would be utilizing the shared key generated in the P3 connection model described above. To audit all transactions, we mandated that all communications beyond the P3 connection model goes through the gateway. This way, the gateway can act as a keeper to not only log every transaction, but also verify that the request going to a device is legitimate. In this technique the device has to only verify that the request is coming from the gateway to verify that its authentic and any other request can be responded with a 401 unauthorized access response.

In most scenarios, the communications between a user and device would be of two main categories, namely,

- The user would request some data or information from the device
- The user would request the device to perform some action

In some cases, the device would proactively want to communicate with the user to pass some critical information. In the ZT model, we keep the command generalized and beyond the scope of the model. Each device can operate with its own set of commands and can be determined by the respective manufacturer. Figure 4 details the process of securing the communication between the user and device. The process of executing command is similar between the user and delegate. The below steps describe a request send by a user to get some information from the device:

- The user initiates the communications by generating a package to be sent to the gateway. The package contains the command and the current timestamp encrypted with the shared key K generated in the P3 connection. This package is sent along with the device MAC that the the user intends to communicate with and the JWT token in the authentication header.

  $\text{Enc}\{\text{<current\_timestamp,command>},K\} \rightarrow \text{package\_to\_gateway}$

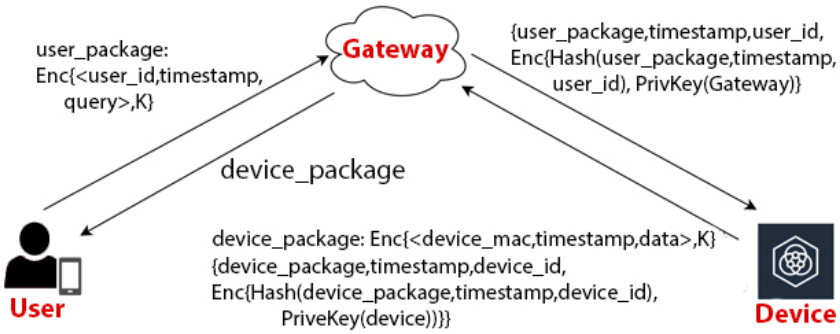

Figure 4: Communication between user and device via gateway.

- The gateway on receiving the package verifies the identity of the user and checks the registration records to ensure that the user has legitimate access to the device it is requesting to communicate with. On successful verification, the gateway creates a package for the device. It extracts the `package_to_gateway` from the request and adds the `user_id` and current server timestamp. Then using its own private key, the gateway signs the `package` and adds the verifies along with it and sends it to the device.

  $\{$`package_to_gateway, user_id, current_timestamp`$\}$ $\rightarrow$ `package`

  $\{$`package, Enc` $\{$`Hash(package, SHA256),` $\text{PrivKey}_{gateway}\}\}$ $\rightarrow$ `package_to_device`

- The device on receiving the `package_to_device`, verifies the signature of the gateway. Using the `user_id` it extracts the shared key of the user and extracts the command. It also verifies the server timestamp and the user timestamp to ensure that they are in sync and within an acceptable difference.
- On successful verification the device gathers the requested data and generates a package for the user. The data along with the device timestamp is encrypted using the shared key K. This user's package is appended with the `device_id` and device timestamp and the whole is signed with with the private key of the device. This `package_for_gateway` is sent back to the gateway

  `Enc` $\{$`<data, current_timestamp>, K`$\}$ $\rightarrow$ `package_for_user`

  $\{$`package_for_user, device_id, current_timestamp`$\}$ $\rightarrow$ `package`

  $\{$`package, Enc` $\{$`Hash(package),` $\text{PrivKey}_{device}\}\}$ $\rightarrow$ `package_for_gateway`

- The gateway on receiving the response from the device, verifies the signature and extracts the `package_for_user` and sends it to the user along with the server timestamp.
- The user on finally getting the response back, decrypts the `package_for_user` using the shared key K and verifies the time difference between the server and device. On successful verification, the cycle completes and the user is able to communicate with the device in a secured way.

One thing to note in the above step is that the request and response between the user and device is obfuscated from the gateway as well. The command execution technique follows the principle of zero-trust of "never trust, always verify". Every transaction is authenticates. The device gets a single point of verification to legitimize a request. We used two-layer encryption to secure the transaction. The two timestamp ensures that the request is flowing through the right path. On verifying the timestamp against the current timestamp of the user, replay attacks can be prevented.

4 IMPLEMENTATION PLAN

A temperature and humidity sensor is being build using a NodeMCU v3 ESP8266 microcontroller to implement the model. A DTH-22 sensor recorded the reading of the environment, and an HC-05 Wireless Bluetooth RF transceiver acts as a Bluetooth communication endpoint. We added a UCTRONICS 0.96 inch OLED module for the device display. The setup helped us simulate a low energy IoT device with its 512 KB of EEPROM storage, 64 KB of instructional RAM, and 96 KB of data RAM. The gateway was setup on AWS API Gateway using lambda functions to support the REST calls. We stored the user registration using AWS Cognito service, and DynamoDB acted as a data storage for the gateway. The user was

simulated using a mobile app build using React native on an Android platform.

5 CONCLUSION

The threat to the IoT devices is real and with the growing number of IP connected devices the attack vector is ever increasing (Bhattarai and Wang, 2018). Building a standardized end-to-end security model is essential to protect the privacy of the users. We can build trust among the users by eliminating trust from the security framework. To do so a proper mechanism like zero-interaction pairing (ZIP) (Fomichev et al., 2019) is needed that can integrate large numbers of devices and their owners into the system without much manual intervention. The P3 connection model described in this paper uses this technique to securely set up a secret key for the parties to communicate. Utilizing these keys in the command execution model verifies the identity of the parties on every request and response. We also demonstrated how we can off load the memory and processor intensive work to the gateway to let the device focus on its primary objective. The ZT model shows how a security framework can work in the background to protect our privacy as well as respect the limitation of memory and computing power of the device.

IoT is the next major breakthrough in the world of technology. These devices perform one specific operation, but it is specialized in doing it. They are slowly turning out to be an essential part of our everyday lives. With home automation systems and home assistants on the rise, we are starting to communicate with these devices with natural language and they are also transacting with our personal and financial data. However, this is just the tip of the iceberg for the potential of these gadgets. Proper security infrastructure is essential to control the activities of these devices. To ensure security and privacy P3 connection approach provides a zero-trust architecture that will verify the authenticity of every transaction.

REFERENCES

Atwady, Y. and Hammoudeh, M. (2017). A survey on authentication techniques for the internet of things. In *Proceedings of the International Conference on Future Networks and Distributed Systems*, ICFNDS '17, New York, NY, USA. Association for Computing Machinery.

Bertino, E. and Islam, N. (2017). Botnets and internet of things security. *Computer*, 50(2):76–79.

Bhattarai, S. and Wang, Y. (2018). End-to-end trust and security for internet of things applications. *Computer*, 51(4):20–27.

Columbus, L. (2019). 2018 roundup of internet of things forecasts and market estimates. shorturl.at/qMPTU. accessed January 21, 2020.

Fomichev, M., Maass, M., Almon, L., Molina, A., and Hollick, M. (2019). Perils of Zero-Interaction Security in the internet of things. *Proc. ACM Interactive Mobile Wearable Ubiquitous Technology*, 3(1).

Gao, M., Wang, Q., Arafin, M. T., Lyu, Y., and Qu, G. (2017). Approximate computing for low power and security in the internet of things. *Computer*, 50(6):27–34.

Goasduff, L. (2019). Gartner says 5.8 billion enterprise and automotive IoT endpoints will be in use in 2020. shorturl.at/jlosS. accessed January 21, 2020.

Huth, C., Zibuschka, J., Duplys, P., and Guneysu, T. (2015). Securing systems on the internet of things via physical properties of devices and communications. In *2015 Annual IEEE Systems Conference (SysCon) Proceedings*, pages 8–13.

Neshenko, N., Bou-Harb, E., Crichigno, J., Kaddoum, G., and Ghani, N. (2019). Demystifying IoT security: An exhaustive survey on iot vulnerabilities and a first empirical look on internet-scale IoT exploitations. *IEEE Communications Surveys Tutorials*, 21(3):2702–2733.

Nieminen, J., Gomez, C., Isomaki, M., Savolainen, T., Patil, B., Shelby, Z., Xi, M., and Oller, J. (2014). Networking solutions for connecting Bluetooth low energy enabled machines to the internet of things. *IEEE Networks*, 28(6):83–90.

Pazos, N., Muller, M., Aeberli, M., and Ouerhani, N. (2015). Connectopen - automatic integration of IoT devices. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pages 640–644.

Puliafito, C., Mingozzi, E., Longo, F., Puliafito, A., and Rana, O. (2019). Fog computing for the internet of things: A survey. *ACM Transactions on Internet Technology*, 19(2).

Ronen, E. and Shamir, A. (2016). Extended functionality attacks on IoT devices: The case of smart lights. In *2016 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 3–12.

Trappe, W., Howard, R., and Moore, R. S. (2015). Low-energy security: Limits and opportunities in the internet of things. *IEEE Security & Privacy*, 13(1):14–21.

Uslaner, E. M. (2004). Trust online, trust offline. *Communications of the ACM*, 47(4):28–29.

van Oorschot, P. C. and Smith, S. W. (2019). The internet of things: Security challenges. *IEEE Security & Privacy*, 17(5):7–9.