International Journal of Information Security
https://doi.org/10.1007/s10207-021-00562-7

REGULAR CONTRIBUTION

Establishing and validating secured keys for IoT devices: using P3 connection model on a cloud-based architecture

Sairath Bhattacharjya[1] · **Hossein Saiedian**[1,2]

Abstract

IoT devices are slowly turning out to be an essential part of our everyday lives. These devices perform one operation, and they specialize in doing so. When communicating with these devices, we need to set up a secured key preventing unauthorized communications. We have been using the plug-and-play model for electronic devices for decades. These IoT devices fall into the same realm. The plug–pair–play connection model follows the same principle so that the user does not feel the added pressure of remembering a complex password or rely on a default credential. It helps to generate a secret that is only known to the device and its user. We used elliptic curve cryptography to circumvent the resource limitations on the device. The model establishes a zero-trust pattern where all requests and responses are validated and verified before being processed. This paper provides a unique way to set up a secret key for each user and device pair without much user interaction. The model sets the path to end-to-end secured communication.

Keywords IoT · Security · Zero-trust · Key generation · Plug-and-play · Elliptic curve cryptography (ECC) · Zero interaction pairing (ZIP) · Zero-interaction authentication (ZIA)

1 Introduction

Internet of Things (IoT) has changed the direction of 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 question of privacy. Establishing a secured communication channel with these devices is crucial.

Users have voiced their privacy concerns with using these devices. There have been numerous experiments to prove that these devices can be easily hacked with readily available equipment [15]. In many implementations, the manufacturers delegate the responsibility of securing the devices to the user by providing default credentials and expecting them to change the password. Malwares like Mirai and EchoBot have exploited these vulnerabilities to convert them into bots. The author in this article [18] talks about the concept of trust, which very much applies to IoT ecosystems. Establishing a zero-trust framework will help us concentrate more on privacy and security.

Zero-trust is a strategic initiative that helps prevent successful data breaches by eliminating the concept of trust. Rooted in the principle, "never trust, always verify" zero-trust is designed to protect the modern digital environment. It leverages network segmentation, prevents lateral movement, provides threat prevention, and simplifies granular user access control. The zero-trust model recognizes trust as a vulnerability. The concept of zero-trust is particularly important in the heterogeneous ecosystem of smart devices. With the huge growth in the number of connected endpoints, it is difficult to have trust in a request or response that is coming from an unknown source over an untrusted medium.

In this paper, we focused on techniques to set up secured keys for communicating with IoT devices without trusting any entity. We wanted to eliminate the need for default credentials or predefined secret keys.

The *plug-and-play* model has been popular with electronic devices for decades. IoT devices fall in the same genre. Manufacturers have adopted a similar pattern for getting the

---

✉ Hossein Saiedian
saiedian@ku.edu

[1] Electrical Engineering and Computer Science, The University of Kansas, Lawrence, KS 66045, USA

[2] Information and Telecommunication Technology Center (ITTC), The University of Kansas, Lawrence, KS 66045, USA

device up and running. The paper describes a pairing step, which establishes a secret for each pair involved in the communication. We termed it the *plug-pair-play* model or the P3 connection model. Here, we focused on the most popular architecture for IoT devices, *i.e.* cloud architecture. Concepts like fog computing have brought the cloud closer to the appliances and services [14]. Cloud architectures can be utilized to shift the compute-intensive operations away from the device. In this paper, we explored the techniques to validate the identity of the user and device using the cloud gateway before setting up a secret.

We organized the article as follows: We start by exploring the common security issues in IoT devices and security threats related to them in Sect. 2. Here, we also look into the potential solutions provided by the research community to secure IoT communications. Then, we explain our P3 connection model in detail in Sect. 3 followed by its usage in Sect. 4. Then, we briefly discuss the implementation setup before exploring the performance of the model in terms of data security, memory utilization, and time of operation in Sect. 5. We conclude with our findings and the opportunity for future research in Sect. 6.

2 Security issues of the devices

From the business reports, we see the phenomenal growth of IoT devices [5,8]. It was predicted that there will be 5.8 billion IoT endpoints by the end of 2020, 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 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. Smaller players are not even providing options to patch the vulnerabilities. These issues are taken advantage of by attackers. Malware like Mirai uses these loopholes to convert these devices into bots. Perpetrators used such botnets to cause massive DDoS attacks [3]. 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 endpoints has doubled. The attack on Dyn Inc. DNS servers in 2016 is one of the most notable attacks using IoT botnet, which brought down the internet for many parts of the USA as shown in Fig. 1 [9].

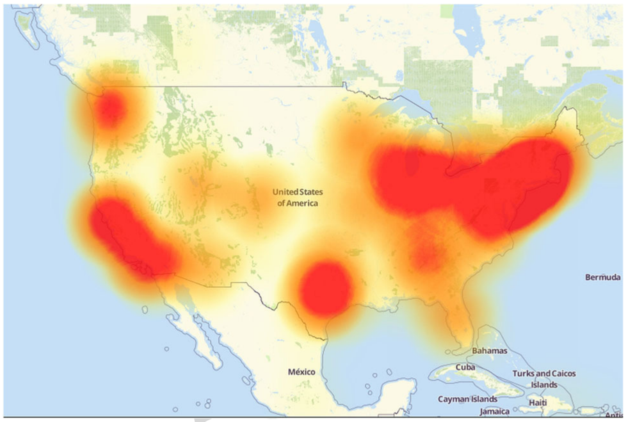


Fig. 1 Attack on Dyn DNS servers brought down the internet in many parts of US

Extensive surveys are 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 make them an easy target for attackers [4,11,17,19].

- **Resource limitation:** Every research article pointed out that constrained resources in the device are a setback when implementing cryptographic techniques. An attacker might drain the device's memory by sending thousands of requests to the open port in a device.
- **Lack of user authentication:** Limited memory on the device restricts the implementation of complex authentication techniques. Thus, to maintain the legal standards, manufacturers end up using default credentials and commonly shared keys.
- **Inadequate encryption:** Encryption is an effective tool to defuse the data. Thus preventing an unauthorized user from making sense of it. Cryptographic systems depend on the randomness of the algorithm and the key size to effectively morph the data. Due to insufficient storage in the device, it becomes difficult to store large keys. An adversary takes advantage of it by performing a brute force attack to break a smaller key size.
- **Efficient access control:** A proper access control mechanism is not maintained on these devices. Many manufacturers allow the use of default credentials on the device, and the same user is entrusted with admin privileges on it. With higher privilege on the default accounts, 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 noticed in the previous section. Researchers have taken different perspectives to solve the problem of identity.

Bluetooth Low Energy (LE) can play a significant role in securing IoT devices. One research showed the potential of using IPv6 over Bluetooth LE [12]. Wireless communication with the device is protected using the Bluetooth LE Link Layer security. This technique supports both encryption and authentication by using the Cipher Block Chaining Message Authentication Code (CCM). OpenConnect proposed to automate the integration of these devices in a cloud-based architecture [13]. The platform uses REST API endpoints to integrate the devices with the central command center. Security of the implementation is 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 equipment [7].

Researchers came up with multiple proposals to tackle the authentication issue for resource-constrained devices. A certificate-based authentication technique was put forward to redress the problem of password-based authentication [2]. 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 their physical properties was provided as a potential solution for the smart home environment [10]. The security mechanism used in this technique uses a random set of challenges along with symmetric key cryptography.

3 The P3 connection model

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

In a cloud-based architecture, there are three primary components in the IoT ecosystem:

- **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 owners and delegates. Each device can have at most one owner who has total control over it. The delegates represent other users to the device, including another person or a home assistant like Google Home or Amazon Alexa. They can access the device only when the owner approves the pairing. The owner has the right to grant access to a delegate to perform specific operations on a device. Throughout this paper, we have addressed the owner and delegate separately when needed and collectively called them as users on concepts that apply to both.
- **Gateway** works as a middleman provided by the manufacturer to help 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 hold information about users, devices, registrations, and transaction logs. When a new device is manufactured, a record is created in the gateway database. The gateway holds the identity and public key of the device to communicate with it after the initiation. It takes the computation and memory-intensive operations like data analytics and forensics away from the device.

Figure 2 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. However, in the P3 workflow, the user directly interacts with the device to set up the secret. This is the only operation where the device and user communicate directly.

In this architecture, we have used a combination of Bluetooth and WiFi technologies to enable a secure communication channel. The P3 connection model uses Bluetooth for pairing the user and device in a secured way. As explained in Sect. 3.2, during the pairing the user provides the WiFi credential for the device to create a registration record in the gateway. Once the pairing is complete and the secret is stored successfully, all further communications happen over WiFi.

3.1 Prebuilt security in the model

As described in the architecture, the user is responsible for providing commands and instructions to the device. The framework comes with a few prebuilt security mechanisms to enable the user to perform its operations. The user is registered with the gateway to generate an identity. An authentication header accompanies all post-login operations.

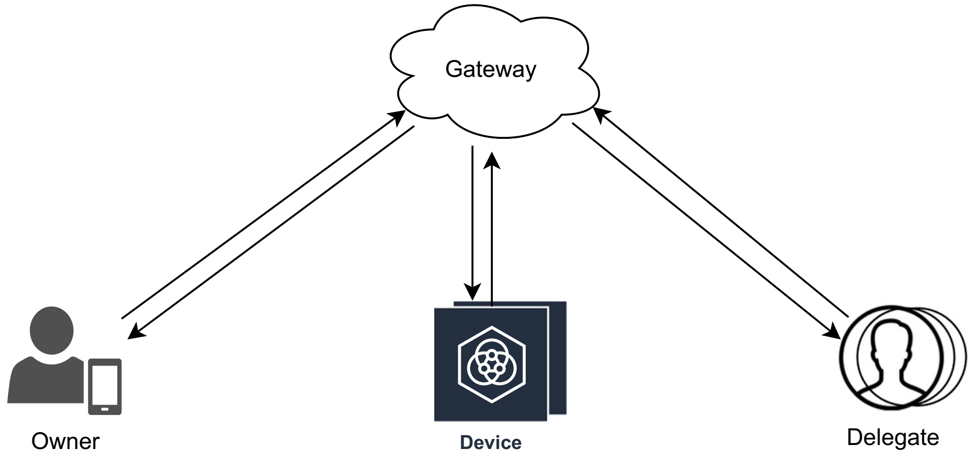


Fig. 2 Proposed architecture for IoT ecosystem

It contains a JWT (JSON web token) token to verify the identity of the user. All communications from the user to the gateway are protected using TLS to ensure data security in transit.

To provide authentication to the device and gateway, each has its public–private key pairs. During the manufacturing of each device, a unique public–private key pair is generated for each device. The gateway holds the public key with itself, and the private key is embedded in the device's EEPROM. We used elliptic curve cryptography (ECC) on the device to respect the resource limitation. ECC is a preferred choice for public-key cryptography for IoT devices rather than RSA due to the smaller key size. A 256 bits key can provide the same level of security as the 2048 bits RSA key. The operational time for signing and verification is comparable. The private key verifies the identity of the device to the gateway:

$$\begin{aligned}&\texttt{<device\_id, current\_timestamp, raw\_data>}\\&\rightarrow \texttt{data <data, Enc\{H(data), PrivKey}_{device}\texttt{\}>}\\&\rightarrow \texttt{package}\end{aligned}$$

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 authentications as well as an integrity check on the data since the private key is only available to the device. The timestamp protects 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 that it claims to be. The same technique is used when sending information from the gateway to the device.

3.2 Setting up shared key for owner

The users of an 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 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 3 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 [12]. Bluetooth LE has been designed for ultra-low power applications 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 the proximity of the device, the attack vector becomes smaller:

- **Pairing:** The first step of the connection is the pairing between the owner's mobile and the device. The owner from his mobile app searches to find the available device. This is easily possible since both the app and the device are provided by the same manufacturer. The manufacturer provides the device with a unique name, and the same is searched by the app. Once found, the pre-defined pairing key can be used to connect to the device. In Bluetooth, the connection happens between a master and a slave. In this case, the owner'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 owner generate a key and share the public part. Both generate the session key $\mathrm{K_S}$ using Diffie–Hellman and use it to secure the remaining transactions of the flow.
- **Connect to Wi-Fi:** Once the session key is established, the next step is for the device to connect to the internet.

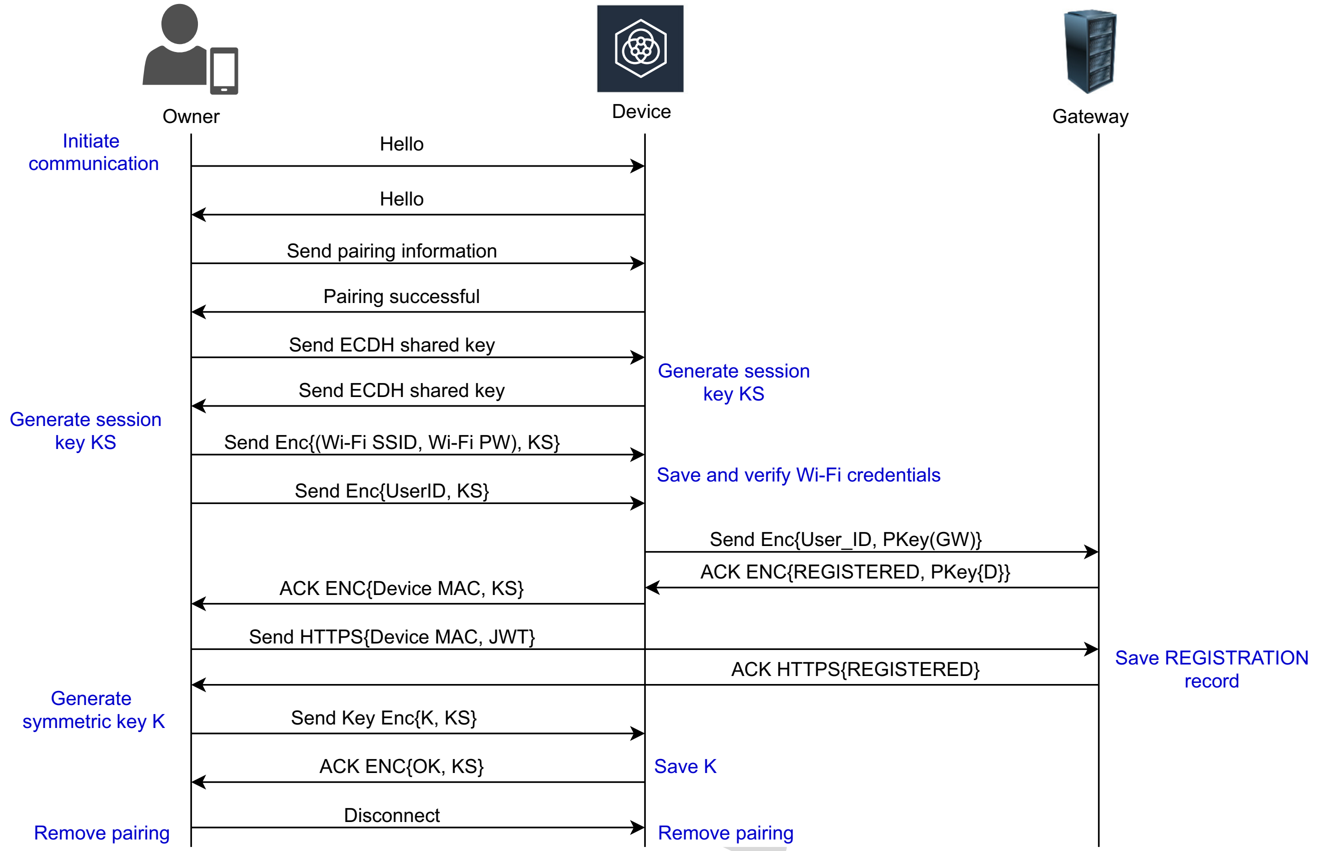


Fig. 3 P3 connection between owner and device

For this, the owner 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 owner.

- **User verification:** After connecting to the internet, the device needs to verify the identity of the owner. The owner 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 owner encrypted with the prior session key Enc{device_mac, K_S}. The owner 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 record. The gateway also checks to verify that the device is not registered against another owner. Once verified, the gateway completes the transaction and returns success to the user.
- **Generate and share the symmetric key:** On receiving a positive response, the owner generates a 256 bits symmetric key along with a 128 bits initialization vector, saves it locally, and shares it with the device Enc {K,K_S}. The device saves the same along with the user identifier recognizing it as the owner 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 owner initiates a disconnect request and the device complies.

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 owner, 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 reg-

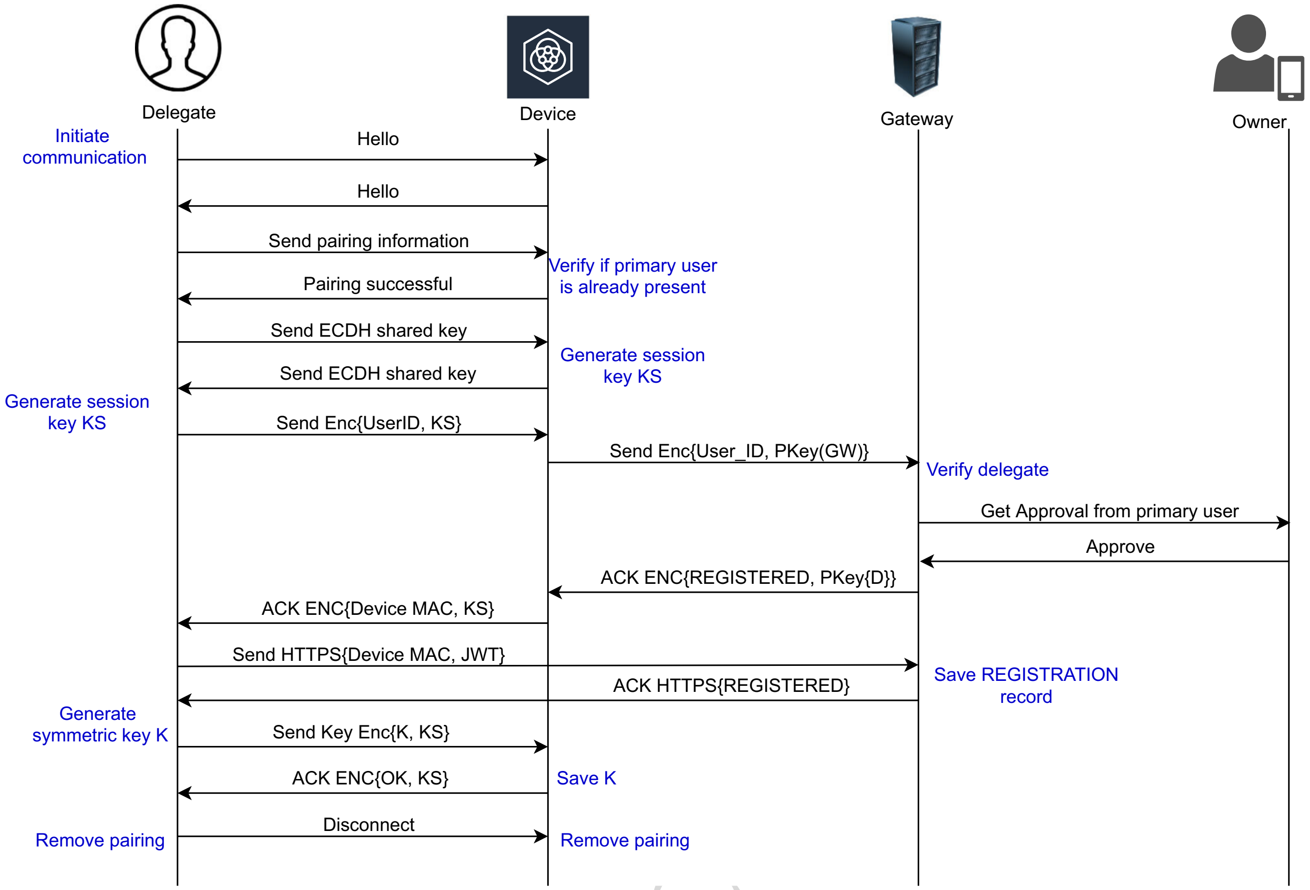


Fig. 4 P3 connection between delegate and device

istration record that was generated during the P3 connection model. The registration record provides access control on who can access which device. However, the gateway cannot interpret the data exchange between the owner and the device.

3.3 Setting up shared key for delegate

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

The steps for a delegate to connect to the device are detailed in Fig. 4. The steps are similar to the connection with a user as described in Sect. 3.2. The user verification process is different for them. 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 an owner already assigned to the device. The gateway notifies the owner in the app asking for approval to create the partial registration record. Once the owner approves, the transaction continues the same as for the user.

If the owner rejects, the transaction is terminated. This ensures that the owner is in control of the device and can track who has access. In this article, we concentrated strictly on secure communication protocols. We have provided equal authorization for all delegates. Another approach to have a fine-grain control on the delegates is to implement role-based access control (RBAC). That would give more control to the owner and they can define what operations can be performed by a delegate.

This approach gives an option for the owner of the device to intervene as to who can talk to the device. The secret key generated in the P3 connection model identifies each pair of user and device. This process eliminates the need to have a default credential or predefined secret. This process works in the background, and the user does not have to configure or remember any additional details to enable security. It also plays well with the plug-and-play paradigm that the users are well accustomed to.

4 Using the secret key for communication

The internet is an untrusted medium. When communication flows from one system to another, it goes through multiple routers, and it is practically impossible to secure every one of them from being wiretapped. To maintain the confidentiality of information between each user and device pair, we would be utilizing the shared key *K* generated in the P3 connection model described above.

Various techniques have been utilized over the year by the industry to communicate with the device. One of the most common patterns is the heartbeat approach. In this, the device sends out a pulse at a regular interval to the gateway to indicate that it is active and functioning. If the gateway receives a message from a user for a device, the gateway utilizes this pulse to forward it. Once the secret key is generated between the user and device using the P3 connection model described in Sect. 3, it becomes easy to maintain confidentiality and integrity.

The user sends out a command to the device encrypted using *K* and the JWT token to identify itself to the gateway. The gateway identifies the user and the registration record. It sends the request to the device along with the user's identifier. The device verifies the gateway's certificate to authenticate the sender and then extracts the key using the user's identifier. The device uses *K* to decrypt the command. Then, it formulates the response and encrypts it with the same key. It sends it back to the gateway, which returns the encrypted message to the user. On a similar approach, the user decrypts the response using *K* and completes the cycle.

One of the advantages of utilizing the key is that the command and device response is hidden from everyone including the gateway. Every pair can securely communicate with each other. The P3 connection model helps generate a key in an automated way and can help maintain privacy during communication.

5 Performance of model

A temperature and humidity sensor was 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 acted 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 set up 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.

For validating the performance of the model, we focused on three primary aspects, namely data security, operational time, and device memory utilization.

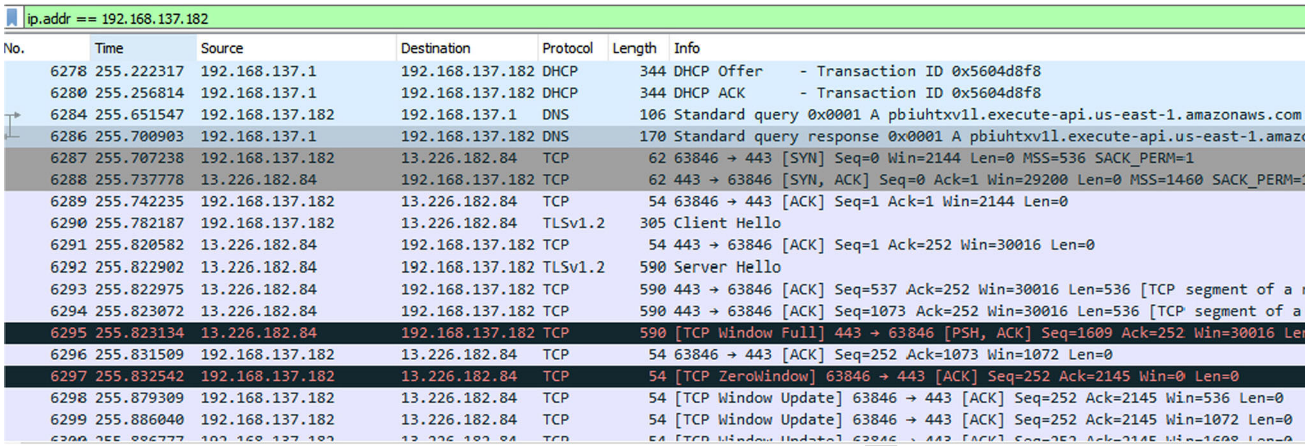
5.1 Data security

The framework proposes a security model that can seamlessly work in the background and protect the user's privacy without manual intervention. In the mentioned architecture, we used different cryptographic techniques that enhance the strength of the platform respecting the limitations available.

In the gateway, we used the TLS certificate to protect all communications directed towards it. The API gateway provided by the cloud providers is by default associated with HTTPS endpoints. We utilized this setup to our advantage. Figure 5 shows the Wireshark output showing the encrypted communication from the device. Both the user and device utilize the API endpoints that are exposed publicly by the gateway during verification. The device stores the fingerprint of the certificate in its storage and uses it to perform the three-way handshake. The app framework provides the same facility for mobile devices.

For the device, having an RSA certificate was expensive. The certificate would consume 2048 bits and would need an additional 1024 bits for the private key. To compensate for space and maintain the same level of secrecy, we utilized elliptic curve cryptography (ECC) with 256 bits key length. We used ED25519 as the choice of asymmetric cryptography to create the device identity. We enabled the device with a signing key, and the corresponding verification key was kept available to the gateway. When communication generates from the device to the gateway, the information was signed using the signing key. The gateway used the verification key to verify the identity of the device. The public–private key pair helps create an identity for each device.

For the user to communicate with the device, the P3 connection model helps set up a shared K. This key protects all communications between the paired user and device. We used symmetric key encryption to make the cryptographic process faster. We utilized AES 256 as the choice of encryption technique with a 128-bit initialization vector (IV) for CBC mode. It made the encryption processes faster when comparing to asymmetric encryption. Each of the secret keys is maintained by the respective user and device preventing any unauthorized access. This key is kept only with the entity that participated in the P3 connection. As explained earlier, after the secret is established, the user talks to the device via the gateway. The gateway uses the JWT token present in the user session to verify the identity of the user, and also sends the user identifier to the device to extract the correct key.



```
ip.addr == 192.168.137.182
No.      Time         Source           Destination      Protocol Length Info
   6278 255.222317  192.168.137.1    192.168.137.182 DHCP      344 DHCP Offer    - Transaction ID 0x5604d8f8
   6280 255.256814  192.168.137.1    192.168.137.182 DHCP      344 DHCP ACK      - Transaction ID 0x5604d8f8
   6284 255.651547  192.168.137.182  192.168.137.1   DNS       106 Standard query 0x0001 A pbiuhtxv1l.execute-api.us-east-1.amazonaws.com
   6286 255.700903  192.168.137.1    192.168.137.182 DNS       170 Standard query response 0x0001 A pbiuhtxv1l.execute-api.us-east-1.amaz
   6287 255.707238  192.168.137.182  13.226.182.84   TCP        62 63846 → 443 [SYN] Seq=0 Win=2144 Len=0 MSS=536 SACK_PERM=1
   6288 255.737778  13.226.182.84    192.168.137.182 TCP        62 443 → 63846 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=
   6289 255.742235  192.168.137.182  13.226.182.84   TCP        54 63846 → 443 [ACK] Seq=1 Ack=1 Win=2144 Len=0
   6290 255.782187  192.168.137.182  13.226.182.84   TLSv1.2   305 Client Hello
   6291 255.820582  13.226.182.84    192.168.137.182 TCP        54 443 → 63846 [ACK] Seq=1 Ack=252 Win=30016 Len=0
   6292 255.822902  13.226.182.84    192.168.137.182 TLSv1.2   590 Server Hello
   6293 255.822975  13.226.182.84    192.168.137.182 TCP       590 443 → 63846 [ACK] Seq=537 Ack=252 Win=30016 Len=536 [TCP segment of a
   6294 255.823072  13.226.182.84    192.168.137.182 TCP       590 443 → 63846 [ACK] Seq=1073 Ack=252 Win=30016 Len=536 [TCP segment of a
   6295 255.823134  13.226.182.84    192.168.137.182 TCP       590 [TCP Window Full] 443 → 63846 [PSH, ACK] Seq=1609 Ack=252 Win=30016 Le
   6296 255.831509  192.168.137.182  13.226.182.84   TCP        54 63846 → 443 [ACK] Seq=252 Ack=1073 Win=1072 Len=0
   6297 255.832542  192.168.137.182  13.226.182.84   TCP        54 [TCP ZeroWindow] 63846 → 443 [ACK] Seq=252 Ack=2145 Win=0 Len=0
   6298 255.879309  192.168.137.182  13.226.182.84   TCP        54 [TCP Window Update] 63846 → 443 [ACK] Seq=252 Ack=2145 Win=536 Len=0
   6299 255.886040  192.168.137.182  13.226.182.84   TCP        54 [TCP Window Update] 63846 → 443 [ACK] Seq=252 Ack=2145 Win=1072 Len=0
```

Fig. 5 Wireshark logs showing use of TLS

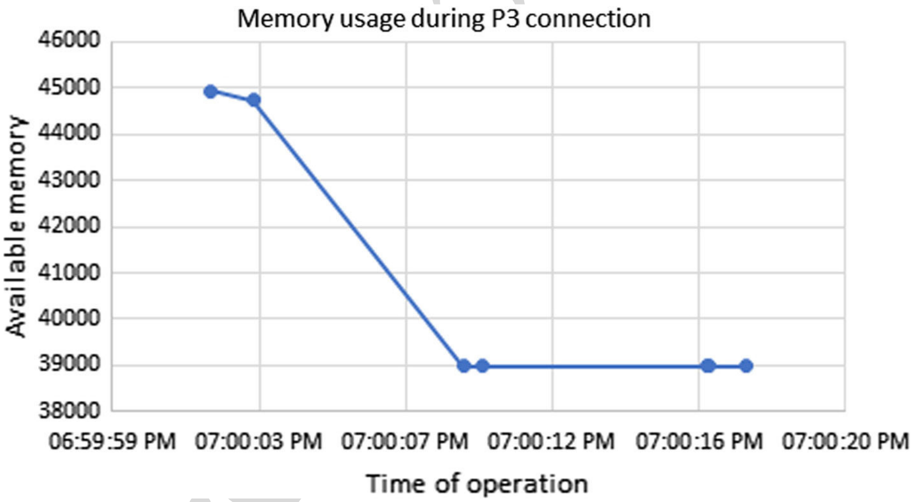


Fig. 6 Memory usage during P3 connection model

The P3 model can also be utilized to refresh the keys at a regular interval.

5.2 Memory utilization

As we had seen in Sect. 3, the P3 connection model has six steps. We tracked the operational time for each of those steps using an inbuilt ESP library. We created a wrapper around the *ESP.getFreeHeap()* function to print the available memory on the Arduino console:

```
//Function to print current memory usage

void availableMemory(){
Serial.print("Memory available: ");
Serial.println(ESP.getFreeHeap());
}
```

Figure 6 shows the memory utilization of the model. As we notice, in the first two steps of getting the hello message and sending an encrypted hello reply, the device uses only an extra 200 bytes. In the next section, we see a drop of five kilobytes to decrypt the message sent by the user containing the Wi-Fi credentials. Post that we do not see any further change in memory usage. The process utilizes around 60% of the available data memory of the device to perform the different steps of the model. The memory gets released for other operations after the function is terminated.

5.3 Operational time

Time of operation becomes vital when it comes to user interactions. For any request generated by a user, they expect a fast response. In the P3 connection model, we wanted the operations to be optimized. From the CloudWatch logs in AWS, we see that the operation time for the lambda functions (that are behind the API endpoints) takes 170 ms to create the partial registration record and 193 ms to complete the registration process. Table 1 shows the execution time on the device.

As we see in the table, it takes around 14 s to complete the whole operation. User verification takes a maximum time

Table 1 Operational time for each step in P3 connection model

Operation	Time (ms)
Pairing and generating the session key	1210
Connect to Wi-Fi	3249
User verification	6527
Device verification	2032
Generate and share the symmetric key	1140
Total	14,158

of 6 s to communicate with the gateway. We will have to consider the cold start of the lambda functions. Cold start happens when the lambda executes for the first time when no other instances exist. The lambda is brought in the server memory for processing for the first time. We built the application with buffer time for unforeseen situations like network delays. For setting up a connection, the user app gave the device 5 s to respond. As we see in the table, it took around 1.2 s to respond. Similarly, for allowing the user to enter the WiFi credentials, the device waits for 60 s (1 min). The timing is not the most optimized but within the acceptable range, considering the entire validation and verification process before establishing the secret key.

6 Conclusions

The P3 connection model provides a groundwork for ensuring a secured communication channel with the IoT devices. The process can seamlessly integrate millions of users and devices. The framework provides the first step for an end-to-end security model that relies on the principle of zero-trust. More research in the area of zero-interaction authentication (ZIA) can provide the required solution to protect the privacy of data [6]. In our P3 connection model, we have expanded the idea of using a Bluetooth connection to perform the pairing. We can utilize LTE or cellular network, and many researchers are looking at potential alternatives like cellular IoT [16], and LPWAN [1] technologies to avoid the dependency of home routers and the risks associated with their hardening. These technologies expand the scope of IoT implementation to multiple sectors like healthcare, industrial use, and other large-scale implementations.

The threat to the IoT devices is real and with the growing number of IP-connected devices, the attack vector is ever-increasing [4]. We can build trust among the users by eliminating trust from the security framework. The P3 connection model described in this article provides a mechanism to securely set up a secret key for the parties to communicate. Both the parties involved in the conversation are verified by the other to eliminate the threat of unauthorized access. The model shows the technique to provide a secure channel of communication respecting the limitation of memory and computing power of the device. The same technique can also be utilized to refresh the shared key on a regular interval to avoid side-channel attacks to predict the key.

The technique described here helps maintain the security triad of integrity, confidentiality, and authentication. The secret key generated by the user and device will protect the confidentiality of the data from other parties including the gateway. The model ensures that the gateway or cloud server is not able to interpret the information exchange between the user and the device. Generating a unique key for each pair ensures authentication. The device explicitly knows whom it is communicating with; also separating the primary owner from other users (delegates) enables accountability to the owner. The model demonstrates the importance of access control on the device by the user. Data integrity is enforced by the data format that is being transacted with. After decryption, if the data is not in the correct JSON format the request is rejected. This process establishes the foundation for a zero-trust model, that works on the principle “never trust, always verify.” Every request and response is verified for authentication and authorization before any other action is taken.

IoT is the next 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.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Animals performed This research does not contain any studies with human participants or animals performed by any of the authors.

Informed consent Informed consent was obtained from all individual participants included in the study. (The authors were the only individuals who participated in the study.)

References

1. Adame, T., Bel, A., Bellalta, B.: Increasing lpwan scalability by means of concurrent multiband iot technologies: an industry 4.0 use case. IEEE. Access **7**, 46990–47010 (2019)

2. Atwady, Y., Hammoudeh, M.: 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 (2017)
3. Bertino, E., Islam, N.: Botnets and internet of things security. IEEE Computer **50**(2), 76–79 (2017)
4. S. Bhattarai and Y. Wang. End-to-end trust and security for internet of things applications. Computer, 51(4), 20–27, 2018
5. Columbus, L.: 2018 roundup of internet of things forecasts and market estimates. shorturl.at/qMPTU, 2019. Accessed 21 Jan 2020
6. Fomichev, M., Maass, M., Almon, L., Molina, A., Hollick, M.: Perils of zero-interaction security in the internet of things. In: Proceedings of ACM Interactive Mobile Wearable Ubiquitous Technology, vol. 3(1) (2019)
7. Gao, M., Wang, Q., Arafin, M.T., Lyu, Y., Qu, G.: Approximate computing for low power and security in the internet of things. IEEE Computer **50**(6), 27–34 (2017)
8. Goasduff, L.: Gartner says 5.8 billion enterprise and automotive IoT endpoints will be in use in 2020. shorturl.at/jlosS, 2019. Accessed 21 January 2020
9. Hilton, S.: Dyn analysis summary of Friday October 21 attack. https://bit.ly/39mqJl6, 2016. Accessed 28 January (2020)
10. Huth, C., Zibuschka, J., Duplys, P., Guneysu, T.: Securing systems on the internet of things via physical properties of devices and communications. In: 2015 Annual IEEE Systems Conference (SysCon) Proceedings, pp. 8–13 (2015)
11. Neshenko, N., Bou-Harb, E., Crichigno, J., Kaddoum, G., Ghani, N.: 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 (2019)
12. Nieminen, J., Gomez, C., Isomaki, M., Savolainen, T., Patil, B., Shelby, Z., Xi, M., Oller, J.: Networking solutions for connecting Bluetooth low energy enabled machines to the internet of things. IEEE Networks **28**(6), 83–90 (Nov 2014)
13. Pazos, N., Muller, M., Aeberli, M., Ouerhani, N.: Connectopen: automatic integration of IoT devices. In: 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT), pp. 640–644 (2015)
14. Puliafito, C., Mingozzi, E., Longo, F., Puliafito, A., Rana, O.: Fog computing for the internet of things: A survey. ACM Transactions on Internet Technology **19**(2), 1-41 (2019)
15. Ronen, E., Shamir, A.: Extended functionality attacks on IoT devices: the case of smart lights. In: 2016 IEEE European Symposium on Security and Privacy (EuroS&P), pp. 3–12 (2016)
16. Sharma, S.K., Wang, X.: Toward massive machine type communications in ultra-dense cellular iot networks: Current issues and machine learning-assisted solutions. IEEE Communications Surveys Tutorials **22**(1), 426–471 (2020)
17. Trappe, W., Howard, R., Moore, R.S.: Low-energy security: Limits and opportunities in the internet of things. IEEE Security & Privacy **13**(1), 14–21 (Jan 2015)
18. Uslaner, E.M.: Trust online, trust offline. Communications of the ACM **47**(4), 28–29 (April 2004)
19. van Oorschot, P.C., Smith, S.W.: The internet of things: Security challenges. IEEE Security & Privacy **17**(5), 7–9 (2019)