How P2P framework can help mitigate trust and security risks of IoT applications

***Abstract*—The rapid growth of IoT devices and its impact in our day to day is one of the biggest challenges for information security nowadays. Privacy in IoT is one of these challenges since IoT devices collect massive amounts of personal data about users and its environment unbeknownst to them and, due to the hardware limitations of the devices, they cannot apply traditional encryption schemes and other privacy centered architectures. We must search new ways to connect these devices without forfeit user privacy and to guarantee same security standards as other current user based applications. Currently, there are many ways to provide user privacy in several contexts but there is no standard to provide users and developers ways to achieve it IoT applications. This paper presents a framework proposal to connect one or several IoT devices to create applications with user privacy in its design through The Onion Router (TOR) to send data over the net anonymously. Our main contribution is to adapt a device whose capabilities to use a heavy encryption-based technology are limited into a fully extendable system that closes the gap between anonymity and IoT.**

***Index Terms*—Internet of Things (IoT), privacy, The Onion Router (TOR), Anonymous Communication.**

I. Introduction

INTERNET of things (IoT) have gained lots of presence in many different aspects of our lives and a produced huge impact with a great amount of devices connected to the internet whose applications are embedded in its infrastructure and to be reused for different purposes. These devices form a network of physical sensors constantly monitoring and sending high volumes of data over internet. However, privacy in these communication schemes presents a challenge for its applications since IoT devices gather lots of personal user data and have a specific design lacking of required processing capabilities to execute security tasks. Therefore, there is a considerable quantity of sensible data flowing over the internet without a privacy assurance which presents a big risk for these devices and its users.

This work focuses on the analysis of the current private and non-private communication architectures between IoT devices, identify its limitations and develop a new product from there. We make a *.onion* based framework to address trust and privacy issues for the IoT capable of establishing communications between IoT devices having privacy by design. The proposed framework lays the groundwork for the development of applications that can guarantee privacy in between device communication with low computation cost. Finally, we elaborated a POC with a simulated health device that tracks health information and sends it to a remote hospital application.

II. TOR & Hidden services

TOR project is a non-lucrative organization researching privacy and anonymity online. This project offers a technology that routes the internet traffic over several relays maintained by thousands of volunteers around the world making really hard to identify the origin and destination of a user's connection. The TOR project develops a product called onion services and can only be accessed from the TOR network, this is why they are also called hidden services.

The hidden service protocol makes use of the TOR network so a rendezvous point between client and service can be created and they can exchange messages with end-to-end encryption and authentication. In order to establish a hidden service the service will select 3 random relays from the TOR network to act as introductory points and create a anonymized circuits to them. These introductory points are the only ones that know how to connect to the hidden service and the connection can only be done through them. After that, the hidden service will publish a privately signed service descriptor containing the list of its introduction points that can only be decrypted with the hidden service address which is also the public key of the before mentioned private signature. The descriptor will be uploaded to a distributed hash table available to clients. When a client wants to connect to a hidden service gets the hash table and asks for the service descriptor. By using the public key encoded in its *.onion* address it verifies the signature of the descriptor itself; inside this descriptor there are the introductory points that can provide access to the hidden service. Then, the client will pick an introductory point and establish a circuit to it, it will ask to the relay to become an introductory point by providing a nonce that will be used later. Now, the introductory point is ready to notify the service that a client wishes to establish a connection by providing the nonce and a rendezvous address to continue the exchange, to which the service will run some verifications and ultimately decide if it wants to connect or not. If the service finally decides to establish a connection it will connect to the rendezvous point and send the nonce to it. The rendezvous point finally will relay messages between the client and the service by verifying the same nonce in both sides [11].

III. State of the art

One of the biggest challenges for IoT is to successfully establish a connection with some degree of privacy. These devices also have limited capabilities to perform computational costly tasks such as encryption in a regular basis making hard to integrate these new system in actual privacy based systems. However, there are solutions that allow to establish communications between devices with some privacy degree,

for instance, in [6] Hoang, N. P., and Pishva, D. consider different privacy related IoT challenges and make a list of multiple well-known vulnerabilities applied to IoT systems, however, by connecting these devices to a more powerful client such as a laptop it is possible to use the powerful anonymous capabilities of TOR and thus it is possible to send all home appliances data anonymously. Similarly, in [**?**] it shows the limitations of IoT to establish secure communications due to its hardware limitations and to solve this problem they propose to use more powerful clients capable to route communications anonymously, again, over TOR in different manners and mitigate the problem.

Currently, IoT technology is a vibrant and constantly studied field; many IoT tools and development frameworks emerge offering different solutions to developers that want to make use of this technology. Currently there are several frameworks and middleware solutions to develop IoT apps but they don't usually include user or location privacy and other aspects of privacy, they are left to the developer to consider. However, there are some guidelines for the creation of a development framework with privacy at its core. In [3] P. Porambage, M. Ylianttila, C. Schmitt, P. Kumar, A. Gurtov and A. V. Vasilakos establish a set of attributes to include in a development framework for IoT. They propose a series of guidelines to determine the quality of a framework and present a novel *Privacy by Design* (PbD) concept that exposes several characteristics to focus such as transparency and purpose, identity privacy data minimization and more. Also, 7 principles to consider in the creation of an privacy-focused IoT framework were defined making privacy a top priority and anticipating security issues.

In Privacy by Design concept in [5] a framework guideline it is also proposed in order to assess privacy capabilities for IoT and introduces the lifecycle of a communication between devices and a methodology to evaluate privacy in IoT frameworks:

1) Consent and data acquisition
2) Data preprocessing
3) Data processing and Analysis
4) Data storage
5) Data Dissemination

Similarly, in [7] E. Al Alkeem, C. Y. Yeun and M. J. Zemerly expose how Man In The Middle (MITM) attack affect IoT devices such as wearables and propose a framework which overcomes the most security requirements along with the privacy issues that mitigates those attacks.

A more domain suited framework proposal exists in [8] where Juacaba Neto, Mérindol and Theoleyre tackle the multidomain applications problem and how meet its privacy needs by introducing and applying the Named Data Networking (NDN) technology to bring scalable privacy. The concept of multidomain applications is assumed by Onu, Mireku, Kwakye and Barker in [9] and introduce an IoT framework for the smart environments with privacy policies and taxonomy that can be applied in multiple environments.

IV. Contribution

IoT devices typically don't have enough encryption power to establish a connection through TOR themselves, this is why we use an auxiliary entry device, a powerful client, to gain access to the TOR network making a device-gateway pair where the latter will send data anonymously. The gateway can create a TOR circuit and deliver device communications to a hidden service. Hidden services addresses will be stored in the gateway and can be dynamically selected to suit the application and device needs. When the IoT device wishes to transmit will indicate a previously known 'id' of a recipient device, much like a traditional communication structure, and the gateway will map this 'id' to a hidden service address to build the circuit and relay the same information anonymously. By doing this we can considerably reduce the overhead of an IoT device and delegate computational costly tasks to a more powerful client.

A hidden service, unlike a centralized service, may not have the same availability and other benefits of reliability, this is why it is important to make a service agreement before hand in order to accomplish successful transmissions. In our case, a gateway may be ready to send but the hidden service may not be available, to avoid this, it is crucial to establish a connection schedule or any other sort of window to guarantee a successful transmission attempt for both parties. On the other hand, an asynchronous scheme may be another choice by signaling the end device when a device is ready to transmit and to make sure the hidden service will be available to receive the sensible payload.

Using Privacy by Design [5] guidelines we propose a framework with same priorities that focuses on communication between IoT devices.

1) **Consent and data acquisition.** It is implicit in our contribution and it's left to the developers to implement solutions to authorize other channels to gather data.
2) **Data preprocessing.** There is no data preprocessing in our contribution. Data minimization is accomplished transmitting the minimum amount of information
3) **Data processing and analysis.** Our contribution is not constrained to a single type of data and further analysis made to the transmitted data is out of our scope
4) **Data storage.** Our contribution will gather data regarding acknowledgement of sending, a logging system.
5) **Data dissemination.** We send data anonymously to another device leaving the data encapsulated in a single transmission between both parties.

To comply with local and temporal privacy we send the data through the TOR network and access control can be accomplished through hidden services.

We will leave the data lifecycle and store system against loss, destruction or revelation open to implementation.

A. Introducing a privacy framework for IoT

Current IoT devices have multiple embedded solutions for many uses cases and many technologies can work in one single device but for our case we will focus in data transmission over the internet.

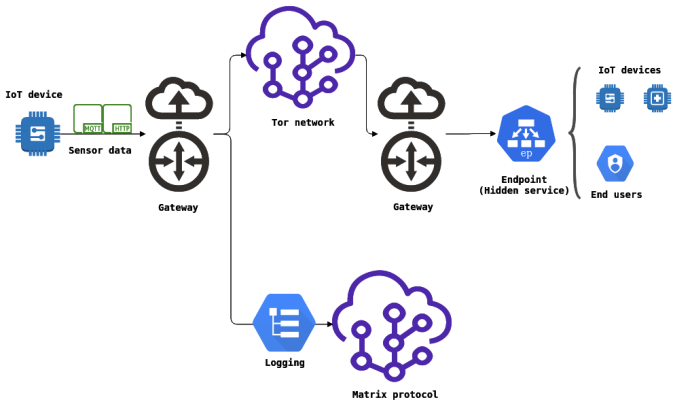

Fig. 1. Framework architecture

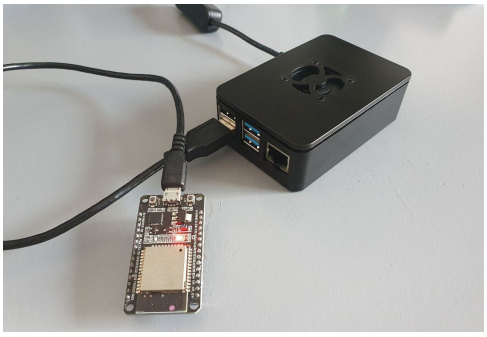

Fig. 2. POC: ESP32 as IoT device and Raspberry as gateway

For our experiment we have programmed a simulation of a sanitary device that sends a patients health information such as arterial pressure and beats per minute. This information is sent to a MQTT broker installed on a Raspberry. Another program is subscribed to all broker topics and sends all information to the main application hub.

MQTT is an OASIS lightweight message protocol standard for IoT. It works with a publish subscribber architecture where a broker receives a message and sends them to all devices subscribed to a topic [12]. The architecture, though, is protocol independent and can work with multiple technologies negotiated in the before mentioned service agreement phase; for instance, it could also send the health information in a more traditional HTTP server-client architecture.

We keep a logging register when data gets transmitted but the framework is implementation independent as well so developers may choose another logging technology if any. To register an IoT device transmission we will use an external logging service whose main goal is also privacy. Matrix is decentralized communication protocol for IP in real time. It consist in a federated ecosystem for instant messaging and VoIP whose messages are stored in a distributed network where users participate in conversations called rooms [13].

Our framework divides the connection and data transmission as below:

1) The IoT device sends data to its connected gateway with a selected destination 'id'.
2) The gateway resolves the hidden service address with the given 'id' and will establish a TOR circuit to transmit.
3) An acknowledgement of send will be registered in the Matrix homeserver.
4) A webapp running behind the hidden service address gathers and shows the data. This webapp is accessed through a TOR browser.

To handle MQTT connections a mosquitto broker [14] listens for connections and relayed to a previously subscribbed gateway. The gateway module consist in a modified Python built-in server receiving both HTTP PUT connections and MQTT pub messages. When a data transmission is received, the gateway will resolve its *.onion* address through the mandatory 'id' and relay the message through TOR.

To integrate the gateway with TOR the requests library [20] was modified in order to use the SOCKS protocol instead of plain HTTP.

Authorized users can the access anonymously to the *.onion* address with TOR browser and get to a webpage presenting the sensor data to evaluate the current state. We created a simple hidden service with docker, Flask microframework [15] and Redis cache system [16] in order to present this information. The TOR service creates a connection to the TOR network and a hidden service with it that listens to default port 80 for new connections and forwards to port 5000 where the app is running. The web service receives HTTP PUT requests sent by the main app and registers them in the cache system. Data can then be accessed with HTTP GET requests through anonymous connection via TOR browser.

B. Experimental methodology and results

The simulation consists in a ESP32 chip programmed with Arduino that publishes beats per minute and arterial tension in a fixed rate of 10 seconds to the broker.

As mentioned in the previous section, we installed all needed software in a Raspberry Pi 4 8GB Model B with 1.5GHz 64-bit quad-core CPU (8GB RAM). The gateway is a collection of software tools and libraries running in the Raspberry:

- MQTT Eclipse mosquitto server. Acts as a broker for the ESP32 MQTT client.
- Matrix Synapse homeserver.
- TOR. We install TOR to get access to SOCKS protocol in order to properly send data over TOR network.
- Python3. We extended the built-in Python3 http server in order to receive HTTP requests and send them over a TOR relay.
- Python-Requests: A modified version of a python-request session object to work with SOCKS and send data to a hidden service.
- Paho-mqtt: An MQTT Python client library [17] to subscribe to the mosquitto broker and receive the device messages.

Synapse in a Matrix implementation in Python that executes a local server (*homeserver*) to generate the ecosystem. We have installed a Synapse homeserver in the Raspberry to run a Matrix local server to register all message sending devices.

These devices are registered as users and both devices and users have access to the room so the user can see what or when is being sent to the hidden service. A room can have multiple users registered, third party users can also join the conversation and keep a track of the patient health state (doctors, hospital staff, patient's relatives, etc.). Through Synapse command line interface we register new users to observe what is being logged by the system. Fig.3 shows a user registration.

Fig. 3. Registering new user through Synapse CLI

Element.io is a chat app that connects to Matrix homesever [18] and manages users and rooms. We used Element.io to create a private chat room for authorized users as seen in Fig.4 and Fig. 5.

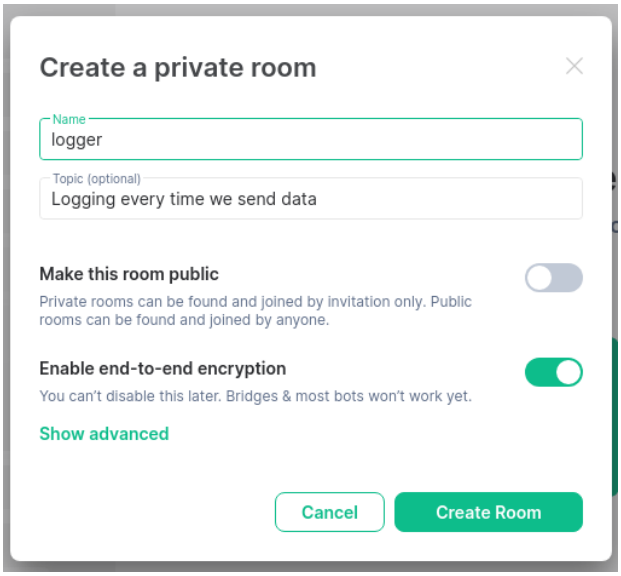

Fig. 4. Chat application: Creating a distributed room in Synapse homeserver

For the IoT device we used the ESP32 ESP-WROOM-32 chip. It runs an Arduino program to connect to the WLAN and an MQTT publisher to publish messages to the running mosquitto broker. Published data is a static simulated health constants of a patient pacemaker which sends beats per minute and arterial pressure 6.

The gateway stores *.onion* addresses matching with a hidden service in the TOR network. To generate an *.onion* address we need a running hidden service beforehand and store it in the gateway.

We use Docker virtualization tool to create a hidden service, it consists in a TOR client running in port 80 and a forwarded connection to port 5000 where a Flask app is running and storing in a Redis cache service the received data over HTTP PUT. The Docker virtualization is running in another computer, acting as a hospital computer where authorized users

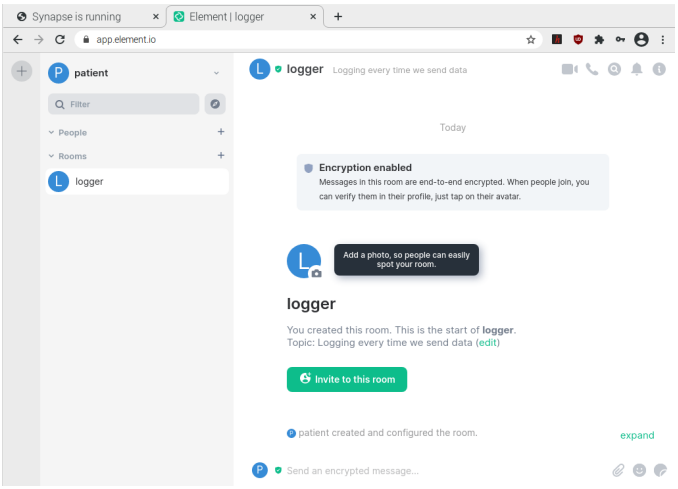

Fig. 5. Chat application: Using homeserver as a logger

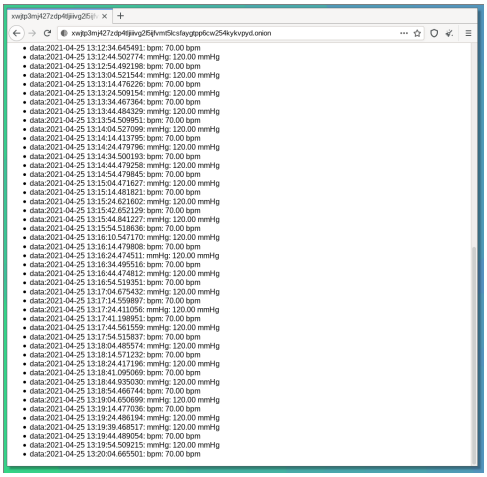

Fig. 6. Simulation results for the network.

have access. Finally, we can access the webapp through a regular connection with TOR browser as shown in Fig. 6.

Note this is a regular webapp running behind a TOR relay and therefore it can be easily modified to accept other HTTP requests, WebSockets or store or present the information in another way.

The IoT framework is publicly available in GitHub [19] to serve as blueprint for similar applications as shown in Fig.7.

V. Conclusions & Future Work

IoT devices and applications are more and more common. They generate high volumes of data unbeknownst to their users and can deal with pretty sensitive data such as location or health stats. These devices do not have needed capabilities to perform tasks to integrate into our current security systems hence they are not able to guarantee some privacy to its users transmitting unsecured over internet. In this paper, we studied the current security needs of IoT and IoT frameworks, some solutions to this problems and we presented a proposal of IoT framework with privacy concerns in its design following strategies and guidelines presented by other authors in what

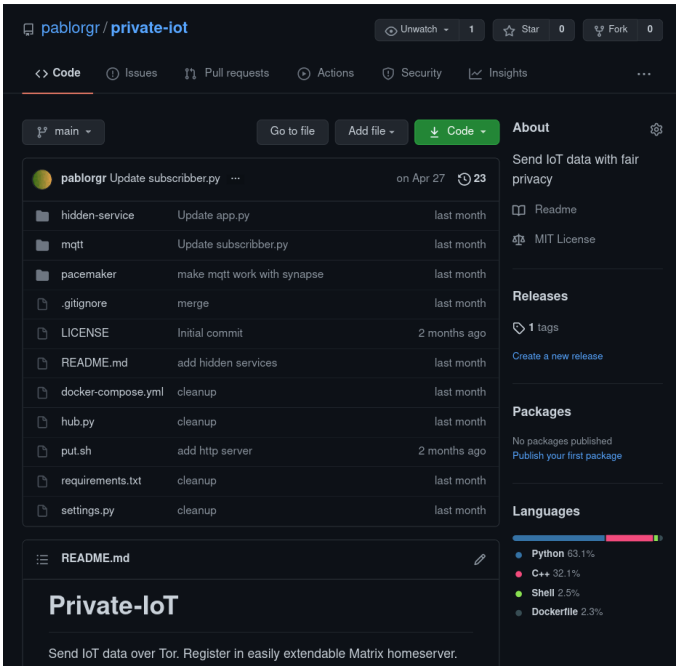

Fig. 7. Software framework publicly available in GitHub

a privacy by design IoT framework should be. This framework gives the blueprints to integrate several devices from different sources and technologies and transmit information anonymously over distributed networks such as TOR. An IoT application was developed following this new framework with ease to build and deploy. We use distributed architectures such as hidden services and the Matrix ecosystem to develop an example of IoT application that can send sensitive health data over the net and access that data in a secure manner.

In the future, we will perform experiments that can determine more precisely how secure the resulting apps are following previous studies that provide metrics and other tools to measure privacy.

Development frameworks are usually in constant growth and improvement and we plan to keep working in this presented tool in order to become a fully functional framework that can make use of the potential of distributed technologies to develop more secure IoT applications. A next iteration of the software will be able to integrate the Matrix ecosystem to give users a command and control (C&C) chat with third party apps that can be connected or federated to the ecosystem. We will also give users more tools to allow other apps, rooms and users in a more flexible way. Furthermore, we will also implement the Consent and data acquisition step in order to give the user a more controlled experience with its connected IoT devices. Finally, we will make the installation process more accessible to developers by providing a command line interface in order to interact in the first stages of the app development process and provide a plugin-like structure to integrate or disengage different modules allowing developers integrate their own.

ACKNOWLEDGMENT

The author would like to thank the SmartUIB institutional project of the University of the Balearic Islands (UIB), Silvia Puglisi (UPC) and Víctor García (UOC).

REFERENCES

[1] Hiller, J., Pennekamp, J., Dahlmanns, M., Henze, M., Panchenko, A., & Wehrle, K. (2019). Tailoring Onion Routing to the Internet of Things: Security and Privacy in Untrusted Environments. 2019 IEEE 27th International Conference on Network Protocols (ICNP), Network Protocols (ICNP), 2019 IEEE 27th International Conference On, 1–12. https://doi.org/10.1109/ICNP.2019.8888033

[2] Gunes Acar, Christian Eubank, Steven Englehardt, Marc Juarez, Arvind Narayanan, and Claudia Diaz. 2014. The Web Never Forgets: Persistent Tracking Mechanisms in the Wild. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS '14). Association for Computing Machinery, New York, NY, USA, 674–689. DOI:https://doi.org/10.1145/2660267.2660347

[3] P. Porambage, M. Ylianttila, C. Schmitt, P. Kumar, A. Gurtov and A. V. Vasilakos, "The Quest for Privacy in the Internet of Things," in IEEE Cloud Computing, vol. 3, no. 2, pp. 36-45, Mar.-Apr. 2016, doi: 10.1109/MCC.2016.28.

[4] A. Ukil, S. Bandyopadhyay and A. Pal, "IoT-Privacy: To be private or not to be private," 2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2014, pp. 123-124, doi: 10.1109/INFCOMW.2014.6849186.

[5] Charith Perera, Ciaran McCormick, Arosha K. Bandara, Blaine A. Price, and Bashar Nuseibeh. 2016. Privacy-by-Design Framework for Assessing Internet of Things Applications and Platforms. In Proceedings of the 6th International Conference on the Internet of Things (IoT'16). Association for Computing Machinery, New York, NY, USA, 83–92. DOI:https://doi.org/10.1145/2991561.2991566

[6] Hoang, N. P., & Pishva, D. (2015). A TOR-based anonymous communication approach to secure smart home appliances. 2015 17th International Conference on Advanced Communication Technology (ICACT), 517–525.

[7] E. Al Alkeem, C. Y. Yeun and M. J. Zemerly, "Security and privacy framework for ubiquitous healthcare IoT devices," 2015 10th International Conference for Internet Technology and Secured Transactions (ICITST), 2015, pp. 70-75, doi: 10.1109/ICITST.2015.7412059.

[8] R. C. Juacaba Neto, P. Mérindol and F. Theoleyre, "A Multi-Domain Framework to Enable Privacy for Aggregated IoT Streams," 2020 IEEE 45th Conference on Local Computer Networks (LCN), 2020, pp. 401-404, doi: 10.1109/LCN48667.2020.9314825.

[9] E. Onu, M. Mireku Kwakye and K. Barker, "Contextual Privacy Policy Modeling in IoT," 2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), 2020, pp. 94-102, doi: 10.1109/DASC-PICom-CBDCom-CyberSciTech49142.2020.00030.

[10] P. Panagiotou, N. Sklavos and I. D. Zaharakis, "Design and Implementation of a Privacy Framework for the Internet of Things (IoT)," 2018 21st Euromicro Conference on Digital System Design (DSD), 2018, pp. 586-591, doi: 10.1109/DSD.2018.00103.

[11] TOR Project - Onion Services
"How do onion services work?"
https://community.torproject.org/onion-services/overview/

[12] MQTT: The Standard for IoT Messaging
https://mqtt.org/

[13] Matrix - An open network for secure, decentralized communication
https://matrix.org/

[14] Eclipse Mosquitto
An open source MQTT broker
https://mosquitto.org/

[15] Flask
https://flask.palletsprojects.com/en/2.0.x/

[16] Redis.io
https://redis.io/

[17] paho-mqtt 1.5.1
https://pypi.org/project/paho-mqtt/

[18] Element.io
Secure collaboration and messaging
https://element.io/

[19] GitHub - pablorgr/private-iot
Send IoT data with fair privacy
https://github.com/pablorgr/private-iot

[20] Requests: HTTP for Humans
https://docs.python-requests.org/en/master/