future internet

MDPI

Article

Enabling the Secure Use of Dynamic Identity for the Internet of Things—Using the Secure Remote Update Protocol (SRUP)

Andrew John Poulter *, Steven J. Ossont and Simon J. Cox

Faculty of Engineering and the Environment, University of Southampton, Southampton SO17 1BJ, UK; sjj698@zepler.org (S.J.O.); s.j.cox@soton.ac.uk (S.J.C.)

* Correspondence: a.j.poulter@soton.ac.uk

Received: 20 July 2020; Accepted: 15 August 2020; Published: 18 August 2020

Abstract: This paper examines dynamic identity, as it pertains to the Internet of Things (IoT), and explores the practical implementation of a mitigation technique for some of the key weaknesses of a conventional dynamic identity model. This paper explores human-centric and machine-based observer approaches for confirming device identity, permitting automated identity confirmation for deployed systems. It also assesses the advantages of dynamic identity in the context of identity revocation permitting secure change of ownership for IoT devices. The paper explores use-cases for human and machine-based observation for authentication of device identity when devices join a Command and Control(C2) network, and considers the relative merits for these two approaches for different types of system.

Keywords: C2; command and control; identity; internet of things; IoT; MQTT; NFC; security; QR Code

1. Introduction

One of the key aspects of any real-world deployment of Internet of Things (IoT) devices is the security of the system. In the context of Internet of Things (IoT), security requires that messages from devices have not been modified on route, and that they originate from a valid sender, as well as often requiring that the messages are protected from eavesdroppers. The Secure Remote Update Protocol (SRUP) [1] proposes a mechanism to address these requirements. SRUP enables secure, and authenticated Command and Control (C2) communication for IoT devices. The protocol is built on top of Message Queuing Telemetry Transport (MQTT) and utilizes a signed, binary message pattern which is used to send operational messages (such as data, commands, or instructions to receive software or firmware updates) between a C2 server and a series of IoT devices.

In addition to these message types, the protocol also supports a number of *management* messages, including facilitating the secure joining of devices to a C2 network.

In order to ensure the authenticity of both the message and the sender, all messages are signed by the sender. Messages may also be optionally encrypted by using MQTT overTransport Layer Security (TLS). In the current implementation of SRUP, both operations rely on the use of Rivest-Shamir-Adleman (RSA) public/private key-pairs (although the protocol permits any asymmetric cryptographic system to be utilized, providing all parties in the C2 network agree on which is being used).

This paper builds upon a previous work [2] which described a scheme by which a human *observer* could confirm that a specified *physical* device was in possession of the *logical* identity it claimed to hold, and describes the implementation of a scheme to enable the automatic matching of a physical device, to its logical identity, via a trusted third-party: an observer node.

2. Background

2.1. Dynamic Device Identity

There are two approaches to defining the identity of a hardware device. The first of these is to issue the device with a fixed identity at the time of manufacture. This may be carried out either by permanently associating a key to the hardware by burning it into Read Only Memory (ROM) or storing it within a cryptographic storage device [3], or by deriving the identity using physical characteristics of the hardware—such as Physical Unclonable Functions (PUFs) [4].

The alternative method, which alleviates the need to rely on pre-determined fixed identities, is to use dynamically generated identities. This technique creates a unique identity at the time that a device is registered with the system, and allows that identity to be revoked and subsequently a new identity may be created.

Adopting this approach has a number of advantages. It eliminates the requirement to securely generate and distribute the keys associated with a fixed identity conferred at the time of manufacture. Given that it also enables the identity associated with a device to be revoked, and for a new identity to be assigned, this approach makes it much easier to ascertain that any previous access to a device has been revoked, since the identity to which any previous permissions applied no longer exists. This concept is especially important for high-value Internet-connected devices, such as cars, which may be expected to have more than one owner during their lifetime. When utilising a fixed identity it is not possible for the new owner of a device to be certain that all previous owners have relinquished all access to the device; but when dynamic identities are used, the new owner may simply generate a new identity for the device and delete the old identity, guaranteeing that no other user has the security credentials for the new identity.

Dynamic identity also has the advantage that a device may be joined to any compatible system—with the minimum prior knowledge of that system. Providing the device knows the end-point address for the system's key registration service, all other data can be bootstrapped, and the C2 service does not require any previous relationship with the device manufacturer or originator.

2.2. Dynamic Identity and SRUP

When initially registering with the system (or when replacing a previous identity with a newly created one), devices are required to establish contact with a web application overSecure Hyper-Text Transfer Protocol (HTTPS)—to request an identity and to perform key exchange with the system. By using HTTPS, the device is able to positively determine the identity of the web service, and prevent a *man-in-the-middle* attack versus the key exchange process. This may exploit public Certificate Authorities (CAs) for Internet-based resources (where the CA certificate is already present within the root of trust on the device); or by the a priori provision of a private CA certificate to the devices, for systems designed to operate on private networks. Using HTTPS also ensures that the traffic between the device and web application is encrypted against eavesdropping.

This process generates a SRUP key-pair to be used by devices and the C2 server when communicating via SRUP messages. In all cases the sending device signs the message—enabling the receiver to positively determine that the message in question has originated from a valid source, and that it has not been tampered with (or otherwise modified) in transit. Additionally, for systems electing to use MQTT over TLS for message security, the service also issues the device with an TLS certificate and key, enabling the device to participate in encrypted communications with the MQTT broker. This approach also permits the broker to restrict the device's access to topics, other than those associated with the device itself—and thus prevents a malicious device connecting in order to eavesdrop on the message traffic at the broker.

Within the SRUP protocol, identity revocation can be conducted either from the device (for example as a result of user interaction), or from the C2 server. Revocation from the device ensures that the old identity no longer exists, regardless of whether it remains registered within the C2 system:

this has the advantage that it does not require consent from the C2 system, and enables the device to join a new (or rejoin the previous) C2 system under a new identity. Devices should send a *deregister request* to inform the server that that identity is being revoked—but this is not mandatory. Revocation from the C2 server will result in the device being sent a *deregister command* message. On receipt of this the device should revert to an *unregistered* state—since it will no longer be able to communicate with servers using the identity it currently holds. Within the protocol it is not possible for a third-party to cause the revocation of an identity.

2.3. Dynamic Identity and C2 Systems

Once a device has an established identity, that device may elect to join a C2 network, either autonomously or as a result of a user-interaction. In low-security scenarios a C2 server may be configured to permit joining of any devices without further establishment of their credentials and as such, a *simple join* operation may be conducted.

Since any device joining the system may have just generated (or regenerated) its identity it is not possible for the C2 server (or a human operator of that C2 server) to ascertain the physical device to which that identity pertains. For low security systems (those where there are no sensitivities to the data and where the system itself is not controlling critical operations), this may be regarded as acceptable. However, for systems where there is the risk of an attacker injecting false data into the system, such an approach cannot be adopted.

The behaviour governing what types of join can be permitted (either globally, or on a device type basis) is determined in software by the C2 system.

The risk from a simple join is that an attacker could stage an attack against the system by intercepting the initial registration request from a device and then registering their own device and joining this instead. Since in the context of a simple join there is no validation of the physical identity of the device that has just requested to join, an attack may register their own device in place of the real device.

3. Validating Physical Identity Using Third-Party Observation

The solution to the attack mechanism identified in Section 2.3 is to require that the logical device joining proves its *physical* identity to the system at the time of the join operation.

The mechanism to enable this (described in previous work [2]) requires that the C2 server sends the device a message encrypted using its SRUP public key—containing a randomly determined 128-bit cryptographic nonce value. The device must then respond by *showing* the value back to the server—via a channel external to the protocol itself. This technique guarantees that only the logical device involved in the join request is in possession of the correct nonce value; and since the third-party is interacting with the physical device in the real-world, if the physical device is able to provide this value to the *observer* then the logical device must correspond to the physical device.

Whilst this approach is still theoretically susceptible to an attack scenario where an attacker is able to cause the joining of a malicious device to a C2 network, such an approach requires the device to be co-located with the deployed system. For critical systems, such as those posing a hazard to life (or systems for which an elevated threat is suspected), there would still therefore be a requirement to adopt traditional techniques of manual inspection to determine the authenticity of the physical device, before the join process is initiated. The automated observation approach does however prevent a remote adversary from being able to join the system: and as always in security, if the attacker has physical access to facilities or other locations, there are a great many other (simpler) forms of attack that they could exploit.

The third-party observer is required to be able to receive information from the device, via an external channel to the SRUP protocol. Previous work identified mechanisms for displaying 128-bit nonce values to enable easy comparison by a human observer—including pictographs, and word lists.

Implementations of both of these techniques were examined as a part of this work, and examples of each can be seen in Section 4).

Further work was conducted to examine a number of technologies to enable machine-based observation: including machine-readable visual codes, and very short-range Radio Frequency communications protocols.

4. Implementing Observation-Based Identity Confirmation, with a Human-Moderated Join

As described in more detail in [2], the concept of a human-moderated join is to require a human user to positively match a unique value sent to the device—with the corresponding display of that value by the C2 user-interface. This process is shown in Figure 1.

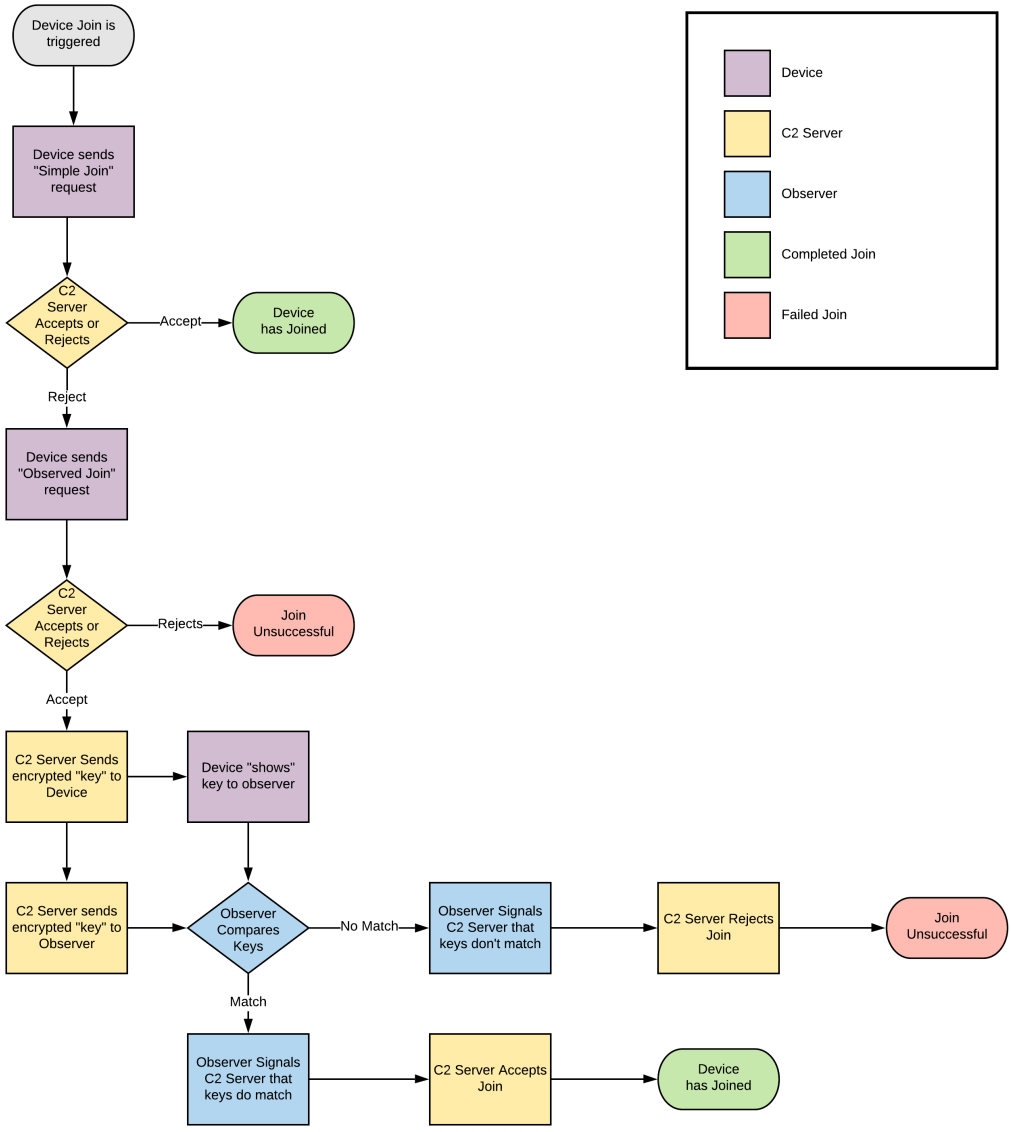


Figure 1. A flowchart illustrating the observed join process: showing the roles of each of the three entities involved.

Two simple example devices were constructed to demonstrate human-in-the-loop observation, based around the approaches described by [2]. Both devices are based around the Raspberry Pi Single-Board Computer.

The first (shown in Figure 2) consisted of a Raspberry Pi 3 fitted with a full-colour Liquid Crystal Display (LCD) graphics display panel.

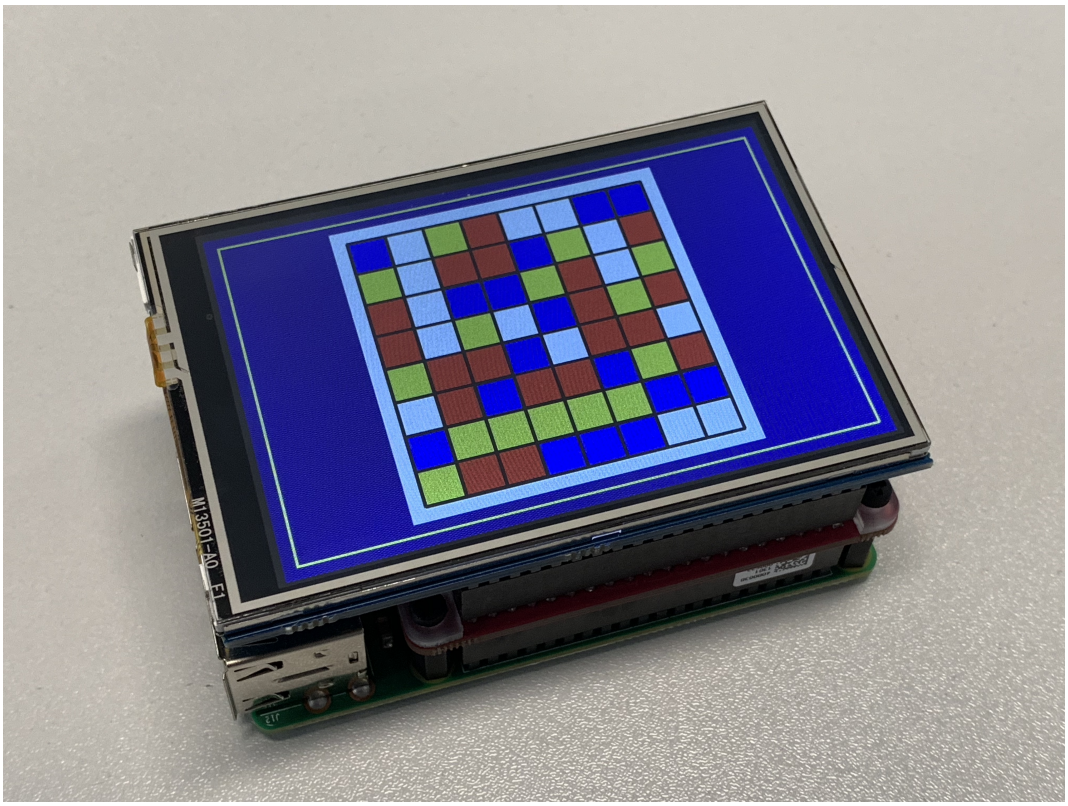


Figure 2. An example Internet of Things (IoT) device—built from a Raspberry Pi 3 fitted with a full-colour liquid crystal display (LCD) graphics display panel.

This device was used in conjunction with an implementation of a four-colour pictogram.

A second device (shown in Figure 3) consisting of a smaller Raspberry Pi Zero W and a three-colour Electronic Ink (eInk) display [5] was also built; this was used with a word-list observation.

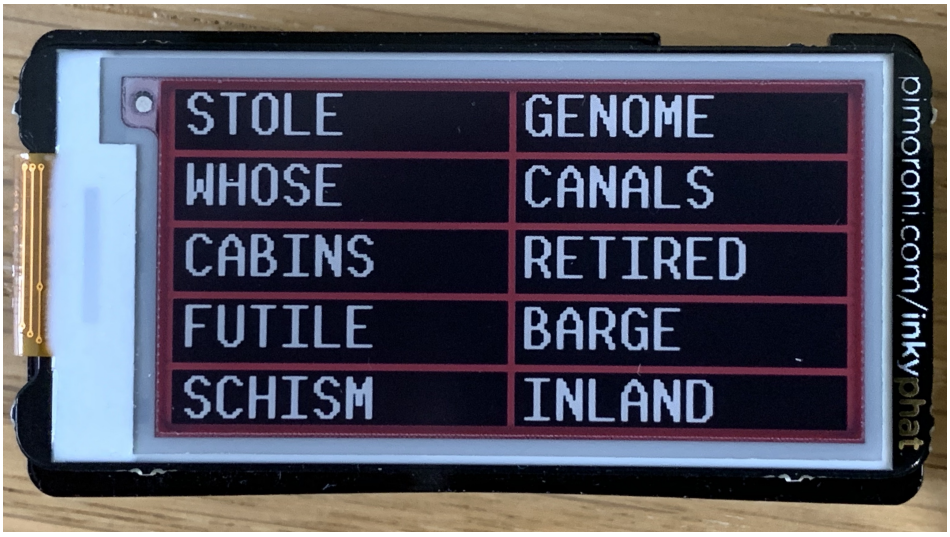


Figure 3. An example IoT device—built from a Raspberry Zero W fitted with a three-colour eInk display.

Both of these were demonstrated in combination with a simple web-based C2 server, which was configured to show either a two- or four-colour pictogram or the word list. An example of this is shown in Figure 4.

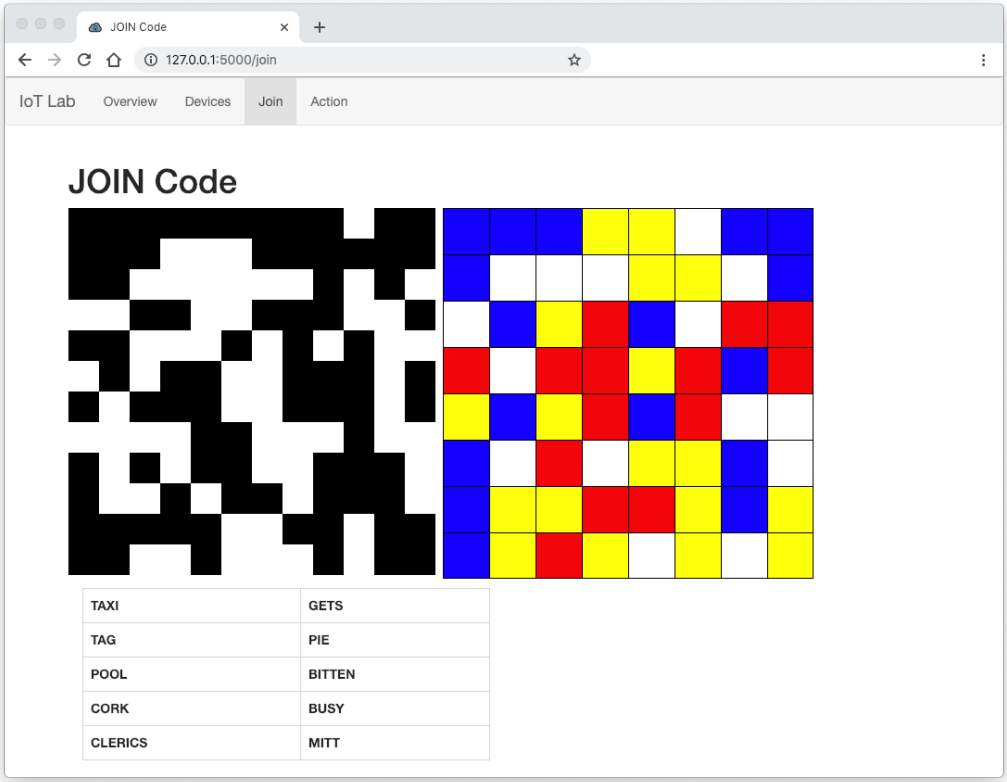


Figure 4. An example web-based C2 interface for an IoT system, showing an example display depicting a range of human-comparable display options.

5. Technologies for Automated Observation-Based Identity Confirmation

For scenarios with a large number of devices, or where devices are expected to be deployed autonomously without a human presence, it may be necessary or desirable to adopt machine-based observation.

For this to work, a trusted *observer node* must be present within the C2 network, and must have already securely joined (for example using a human-moderated observation—as described in the previous section). This observer will then utilize one or more techniques to observe the value presented by the joining device, and to confirm that this matches the value that the C2 server had transmitted.

Unlike a scheme adopting a human observer, where the C2 system may be able to directly present a representation of the value to the user for them to compare with the device, an automated observer node must also be securely sent a copy of the code. Specifically, once the C2 server receives an automated join request from a device, the server will respond by sending an `OBSERVED_JOIN_RESPONSE` message to the device, as well as an `OBSERVATION_REQUEST` message to the observer. Both of these messages contain a copy of the 128-bit nonce value to be used for the comparison, and each message is encrypted using the recipient's SRUP public-key, ensuring that only that recipient is able to decrypt the data and read the value.

Once the observer and the device each have their own copy of the nonce, the device should present the value to the observer for comparison and onwards signalling.

A flowchart illustrating the observed join process is shown in Figure 1.

In this work, two sets of technologies have been examined to provide this observation.

5.1. Visual Observation Technologies

One set of technologies that can be adopted for the observation is machine-readable visual codes—such as barcodes, QR or Data Matrix codes.

Conventional one-dimensional barcodes can store a maximum of around 100 characters (for example Code 128 is defined in [6] and can store 103 data symbols). However, two-dimensional barcodes such as the QR Code or Data Matrix (defined [7]) can store over 1000 symbols.

For SRUP observations, we need to encode a 128-bit value: here is a Universally Unique Identifier (UUID) rendered as a string of 32 hexadecimal characters [8]. As such, any display hardware capable of displaying either a one- or two-dimensional barcode could be adopted—along with any barcode technology capable of displaying 32 hexadecimal characters (e.g., Code-128, Code-93 or Code-39).

The disadvantage of linear codes is that for a 32-character code, the resulting barcode is quite long—exceeding the convenient aspect-ratio and dimensions of many displays without unduly squeezing the vertical height—resulting in the mark/space size of the code being reduced.

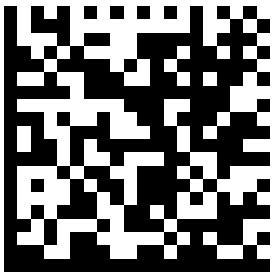
Figure 5a–c show the same 32-character hexadecimal value, rendered in a number of one-dimensional codes, and Figure 6a,b show the same data rendered as two-dimensional codes.

(**a**) Code-39

(**b**) Code-93

(**c**) Code-128

Figure 5. A 32-character hexadecimal value, rendered in a number of one-dimensional bar code types.



(**a**) Data Matrix Code

(**b**) QR Code

Figure 6. The 32-character hexadecimal value, rendered as two-dimensional bar code types.

Experimentation using codes for the same UUID value—rendered on an LCD display, and read using a smart-phone camera—showed that only Code-93 and Code-128 linear barcodes could be reliably read (and that Code-93 was most often read). Code-39 format barcodes were not readable.

Two-dimensional codes, on the other hand, are already more suitable to being rendered on a display, due to their square shape, and both Data Matrix and QR codes could be easily read by the app, with approximately the same accuracy.

The Data Matrix code has some potential advantages over the QR code, including the resulting size of the code required for a given length of data being smaller, and improved detection and error correction [9].

Further experimentation was conducted using a Raspberry Pi and camera, looking to automatically locate and extract a 32-character hexadecimal value encoded as a two-dimensional barcode. Unlike the previous example, here the code was not manually aligned with the reader: rather, the software was required to identify where in the image streamed from the camera, the barcode was located before it could be decoded.

In both cases the Pi (a Raspberry Pi 3B+), fitted with a Raspberry Pi Camera Module, was running Python code to process the image. For the Data Matrix code, the `pylibdmtx` library [10] was used, and the QR codes were read using the `pyzbar` library [11].

In both cases the code was tweaked to maximize performance. With the Data Matrix code, the best performance was achieved when processing a 320 × 240 grayscale. PNG file (taken from an video capture of the camera being shown the code): and it took an average of 12.7 s per image to detect and extract the code (in contrast to less than a second when the DM code is manually cropped from the image file). This is unacceptably slow for a system processing video frames, and is in contrast to the `pyzbar` library, searching for QR codes—which was shown to manage to process several frames per second.

Although additional image processing and manipulation techniques could be used (for example to detect and crop the DM code in the image), for the purposes of demonstrating the machine observation techniques QR codes were selected.

5.2. Radio Frequency Identification

In addition to a visual observation, a short-range RF link was also adopted to demonstrate a different class of observation node. Although a number of technologies were initially considered (including Bluetooth [12], and the IEEE 802.15.4 Zigbee standard [13]), the best guarantees as to the physical location of the device in question were achieved when using ultra short-range RF communications such as those used for Radio-Frequency Identification (RFID).

RFID technologies fall into two broad classes: low-frequency devices (operating at 125 kHz in Europe), and high-frequency devices operating at 13.56 MHz [14].

Low frequency systems typically have an operating range of several inches, but have very low data transfer capabilities and are only used with simple passive *tags* containing a static serial number determined at the time of manufacture. Such devices are typically used for asset tracking. High frequency systems may store up to 4 Kb of data [15], and may be writable as well as readable. This class of tag is often produced in a *credit-card sized* form-factor, and is often used to provide security access tokens, as well as cashless ticketing in public transportation systems.

Near-Field Communication (NFC) also operates on the 13.56 MHz frequency and supports active communication between two devices, over a range of a few centimetres.

Despite (or perhaps because of) its ubiquity within the security (RFID) and banking sector (NFC for contactless payments), the state of easily accessible and open-source software to support operations more complex than reading or writing to simple tags is somewhat limited.

Most devices capable of reading NFC data (including most mobile phones) are able to read static tags which have been formatted using the NFC Data Exchange Format (NDEF) [16] standard; but there is somewhat limited support for the active data exchange provided by the Simple NDEF Exchange Protocol (SNEP) [17,18].

A simple device was constructed to utilize Simple NDEF Exchange Protocol (SNEP), utilizing open-source example C code and based on the PN532 NFC [19] chipset connected using the

Inter-Integrated Circuit Protocol (I^2C). This was interfaced to a Raspberry Pi as a USB serial device, using an Arduino development board, based on the Atmel ATMega32U [20] microcontroller. This is shown in Figure 7.

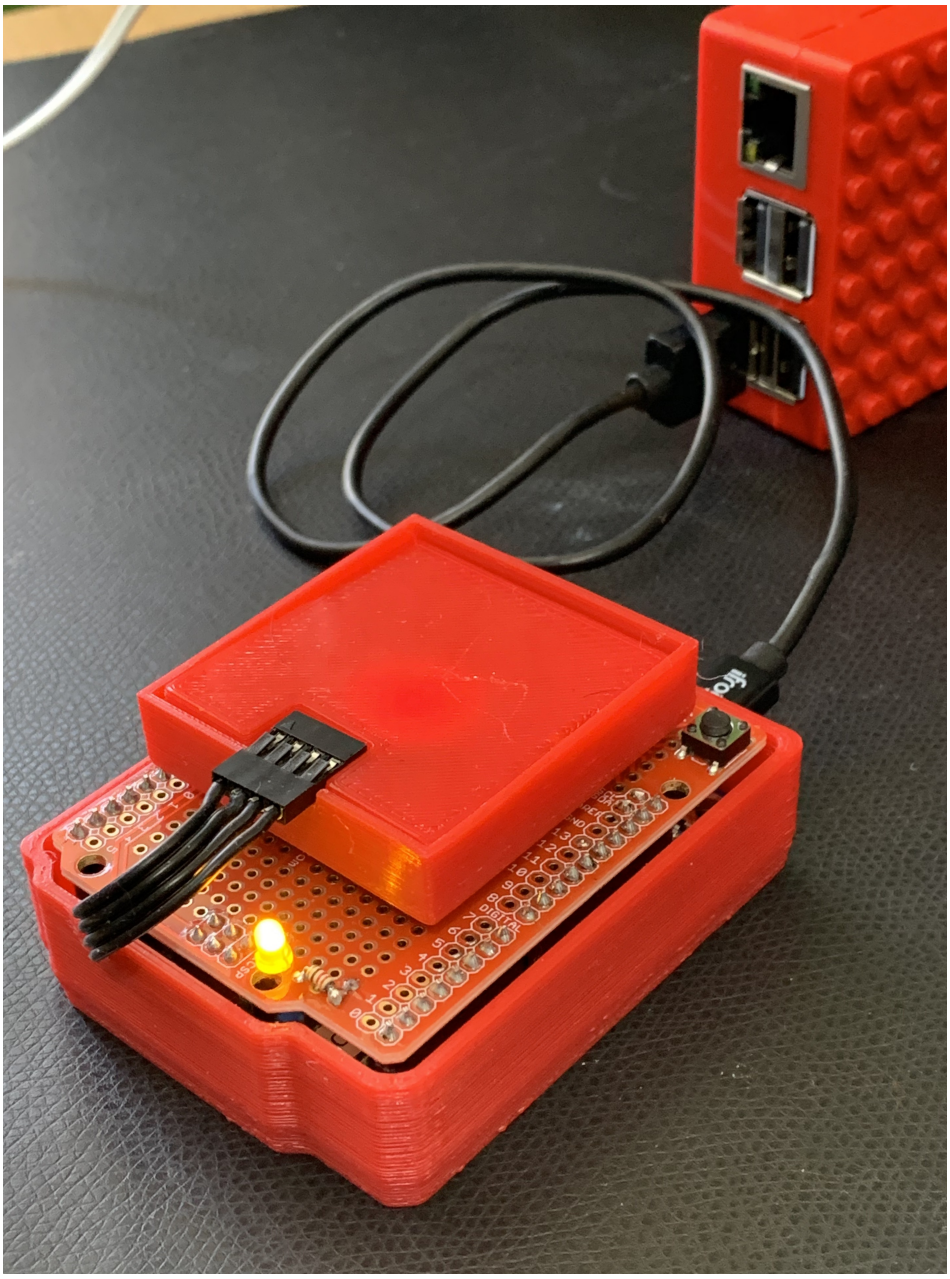


Figure 7. An near-field communication (NFC) Observer device, consisting of a PN532 NFC module, connected via Inter-Integrated Circuit Protocol (I^2C) to a development board based on an Atmel ATMega32U4 microcontroller, connected to a Raspberry Pi via Universal Serial Bus (USB).

6. Other Device Identity Validation Techniques

El-hajj et al. identify a taxonomy for IoT authentication schemes [21], identifying *Token-based* and *Procedurally based* techniques for device authentication, in addition to identification based around the static *hardware-based* approaches described in Section 2.1.

Token-based authentication schemes, such as OAuth2's Device Authorization Grant [22], are typically used to provide a secure mechanism for an individual user to validate their credentials for a given service on a specific device, which is carried out using a third-party device, such as using a

mobile-phone, to visit a Universal Resource Locator (URL) associated with the process. This approach may be seen, for example, when a user logs into a video-sharing website on a smart-television, which presents a QR code for the user to read with their phone, causing the phone's browser to access the page to complete the validation operation.

Although superficially similar, the Device Authorization Grant approach is very different to the one outlined in Section 5.1. This approach utilizes the OAuth2 [23] token-based approach and is designed to permit a user to associate a device with the identity that they have defined with an external identity provider (for example, Google, Facebook, etc.). Although a bespoke authorization service could be constructed within the specifications of the OAuth2 standard, the standard of this falls somewhat outside of the typical OAuth2 use-case. Additionally, such an approach requires a human-in-the-loop to conventionally log into the authorization service in question, and offers no fully automatic mechanism to pair the physical and logical device identities.

Moreover, using an OAuth-based approach requires that the device (in its deployed state) is able to make outbound HTTPS requests. In contrast, although SRUP does require the device to utilize HTTPS as a part of the initial registration phase (which may be performed prior to operational deployment), the remainder of the join operations take place wholly over the MQTT protocol—which is much more suited to potentially constrained network conditions which may be expected in an operational deployment of a IoT device.

Procedurally based approaches, such as Datagram Transport Layer Security (DTLS) [24], are utilized for devices adopting static identity and provide security for the messages, but do not themselves offer any guarantee that the physical device in question corresponds to the identity of the logical device and rely on secure on-device storage of the certificate and key.

7. Implementing Observation-Based Identity Confirmation, with a Machine-Moderated Join

A series of devices was constructed, based on the Raspberry Pi platform and utilizing (an updated version of) the pySRUP library (introduced in [25]). By using pySRUP, a SRUP IoT device can be quickly written in Python, which can exploit all of the features of the SRUP protocol.

During this work, the pySRUP library was modified to fully support human and machine-based observations. The web-based C2 interface was also modified to support observation.

7.1. Sequence Diagram

The full information exchange process that occurs during the machine-moderated observed join process is illustrated as a sequence diagram in Figure 8.

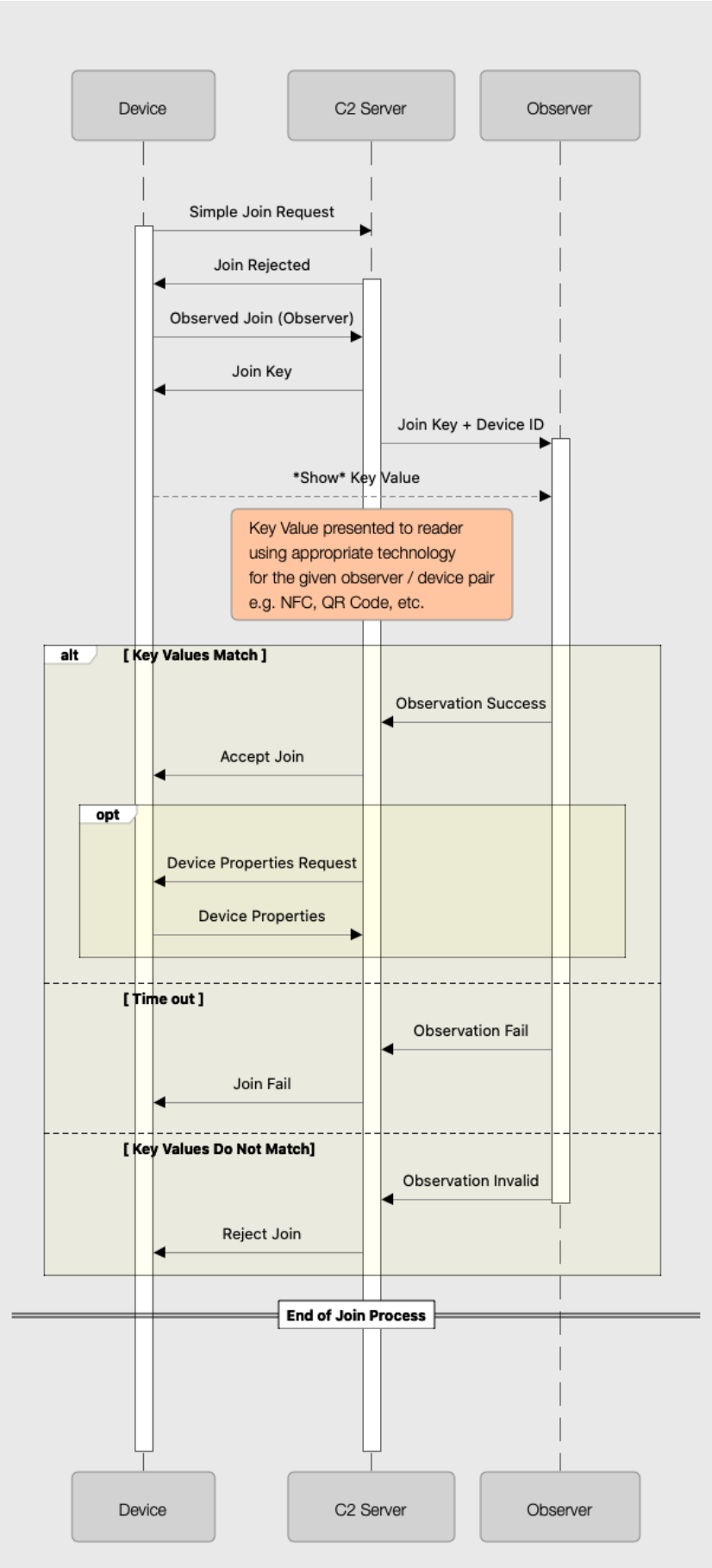


Figure 8. A sequence diagram showing the Machine-Moderated Join Process.

7.2. Hardware

Two sets of devices were constructed around the Raspberry Pi hardware: one using the visual recognition scheme and one to use NFC.

The visual device was identical to the one used for the human-readable pictograms, with a simple observer constructed from a Raspberry Pi fitted with a camera module.

The NFC hardware consisted of two Raspberry Pi's connected over USB to the PN532 modules described previously. One was configured as an NFC observer and a second as a joining device.

In the case of both the visual and the NFC devices, the same pySRUP library code was utilized. The only change required was that the device must specify the required operation in the call-back function relating to the observation and presentation of the identity confirmation.

7.3. Operation

With the visual observation scheme, the target device is required to be within the field of view of the observing camera. For the NFC-based observer, the device must be close enough to the observer for the observer to be able to read the data.

In either case, the outcome of an observation attempt is one of three states: either the code was read correctly (VALID), or an invalid code was read (INVALID), or no code could be read and the observation operation timed-out (FAILED). For this example system, in the event that the observer signalled back to the C2 server that it failed to read a value, the C2 server would simply reissue the observation request to both the device and the observer. In a real-world system, the implementation should cap the number of failed read requests by a given device to a (highly system- and implementation-specific) *reasonable number*; but for testing purposes this was not capped in the demonstration.

Although a logical observer device is only able to read one type of device communications, multiple logical observers may be combined into a single physical device.

8. Considerations for Real-World Use of Observed Join

8.1. Human versus Machine Observation

The question of which type of observer (human or machine)—and the larger question of whether an observed join is required at all—is highly dependant on the specifics of the operation and deployment of a given system.

A system utilizing wired sensors on a closed network (such as that found within a larger operational platform—such as a vehicle or factory) may not require any identity validation when joining (especially where physical access to an air-gapped network is controlled). A more typical deployment in a domestic or commercial setting may utilize an open Internet connection to facilitate the communications and as such could use a controlled join. For small-scale deployments, a human-moderated join is an ideal mechanism, especially since this has no requirement for specialist observation nodes. However, a smartphone application could be utilized as an observer (itself joining via a human-moderated join)—and then subsequently using either the smartphone's camera for visual recognition, or potentially utilising the phone's NFC capabilities to read values from devices without a user-interface capable of displaying a visual code. Although this implementation has not been explored in this work, the majority of modern smartphones support NFC for mobile payment, and Application Program Interfaces (APIs) exist within major smartphone operating systems to access some or all of the functionality of the NFC hardware within the phone.

Human moderated joins provide the highest guarantee of device identity, since the human observer is explicitly able to ascertain that the device in question is the device that is being joined (especially where the human installer has manually triggered the join operation themselves). It is however the least scalable to very large deployments, or for scenarios where there is a large number of join and leave operations expected to be conducted.

For such large-scale operations, installed devices could join the C2 network, and have their identity validated by a dedicated observation node. For example, within a factory setting an installer could utilize a hand-held observer device using NFC to positively validate devices as they were installed and joined into the system; or for a distributed sensor application, devices could be joined (and validated) as they are deployed.

8.2. Benefits of Machine Observation

In a real-world deployment, different scenarios have different requirements for a machine-moderated join. For devices where the deployment conditions prevent the observer node from physically coming into close proximity to the device (such as an observer covering an area deployment, it may be preferable to conduct an observation via a visual method, from a longer distance than may be possible with other technologies. This may require the device to be large enough that both the display, and the device to which it is mounted, are sufficiently visible to the remote observer.

Physical proximity can provide the best guarantee of the identity of the physical device, for scenarios where this is achievable. For example, devices being deployed via a conveyor belt-fed system, devices which are at a known and physically small location (such as passing through a door, or gateway), or devices where a human can physically access and *tap* the device could all utilize this type of approach.

Although designed around the IoT, and the idea of largely static devices, this approach (and the SRUP protocol in general) will work well to support broader classes of *smart* devices, including smart vehicles. However, for the purposes of identity validation, neither of the technologies employed in this example implementation work especially well for scenarios where the device is in motion.

For example, in a scenario where the device would only be in the observers field-of-view for a short period, careful and accurate timing would be required to ensure that the messages supporting the request have been sent (and arrived) with sufficient time for the observer to be ready to view the device. This may require, for example, that the device makes the request some period of time before it expects to be observable (although the longer the time period between the message and the observation, the greater the risk of another device being detected instead).

Similarly, both technologies require some finite time for the read operation to complete (a visual observer would require that the entire display is visible in at least one frame of video; and an Radio Frequency-based observer would require the device to be within the operating range of the reader for at least the duration of the read operation). As such, operations to perform the join would be required to take place during a time period when the device was static (such as at a control point or barrier—prior to entering the smart system's control).

8.3. Issues

In the present implementation of the system, a device is required to know the identity of the observer that it wishes to use. In the examples shown above, this is hard-coded into the device's source code. There is currently no *explicit* mechanism within SRUP to specifically enable transmission of an observer's identity to a device. This is deliberate, since until or unless a device joins a C2 network, it is not defined as to which servers may send SRUP messages to it. This is in contrast to the identity of the *default* C2 server for a given device, which is explicitly sent to the device as a part of its initial registration and key exchange process.

Since the device has the SRUP public key belonging to the default server (required so that it can be used as a part of the joining process to validate messages from the server), the server could use the extant `DATA` message within the SRUP protocol in order to send the identity of the observer that should be used, after the server has refused the initial simple join. This could be trivially implemented by a system using the pySRUP library by simply adding a data message handler callback function to the device code and using the existing `send_SRUP_Data` method from the library, to send the observer ID from the server.

9. Conclusion

This paper has identified how both human and machine-based observation techniques can be used to positively confirm the identity of a device joining a C2 network. We have shown that by using these techniques we can utilize a dynamic identity scheme for IoT devices, enabling both guaranteed revocation of prior access to a device by replacing the device identity with a new one, and simplifying the process of device key distribution. We have shown that a suitable device (running software built using the pySRUP library) can securely join a C2 network, positively establishing its identity to the network, with no prior knowledge of the device required by the network, and with only a single URL address for the key-management service required to be supplied to the device.

9.1. Future Work

A number of further experiments are planned to examine the utility of SRUP in conditions representative of real-world deployments. In particular we plan to construct a further extension of the protocol to support syndication of information between C2 networks exhibiting asymmetric trust relationships. We also plan to examine performance and efficiency of the protocol in a variety of network conditions, and to obtain performance metrics for comparison with other approaches.

9.2. Further Details

The software described here can be obtained as: `doi:10.5281/zenodo.3898242`. A video demonstrating the machine-moderated join process in action can be seen at: https://youtu.be/Vi135raj1LE.

Author Contributions: Conceptualization: A.J.P., S.J.O. and S.J.C.; methodology: A.J.P.; software: A.J.P.; validation: A.J.P., S.J.O. and S.J.C.; investigation: A.J.P.; writing—original draft preparation: A.J.P.; writing—review and editing: A.J.P., S.J.O. and S.J.C.; supervision: S.J.O. and S.J.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was wholly funded by the United Kingdom Defence Science and Technology Laboratory (Dstl). Dstl is a part of the U.K. Ministry of Defence.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Poulter, A.J.; Johnst, S.J.; Cox, S.J. SRUP: The Secure Remote Update Protocol. In Proceedings of the 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT), Reston, VA, USA, 12–14 December 2016; pp. 42–47. [CrossRef]
2. Poulter, A.J.; Johnston, S.J.; Cox, S.J. Extensions and Enhancements to "the Secure Remote Update Protocol". *Future Internet* **2017**, *9*, 59. [CrossRef]
3. Delaune, S.; Kremer, S.; Ryan, M.D.; Steel, G. A Formal Analysis of Authentication in the TPM. In *Formal Aspects of Security and Trust*; Degano, P., Etalle, S., Guttman, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 111–125.
4. Suh, G.E.; Devadas, S. Physical Unclonable Functions for Device Authentication and Secret Key Generation. In Proceedings of the 2007 44th ACM/IEEE Design Automation Conference, San Diego, CA, USA, 4–8 June 2007; pp. 9–14.
5. Chen, Y.; Au, J.; Kazlas, P.; Ritenour, A.; Gates, H.; McCreary, M. Flexible active-matrix electronic ink display. *Nature* **2003**, *423*, 136. Citation Key: chenFlexibleActivematrixElectronic2003. [CrossRef] [PubMed]
6. *Information Technology—Automatic Identification and Data Capture Techniques—Code 128 Bar Code Symbology Specification*; Technical Report ISO/IEC 15417:2007; International Organization for Standardization: Geneve, Switzerland, 2007.
7. *Information Technology—Automatic Identification and Data Capture Techniques—Data Matrix Bar Code Symbology Specification*; Technical Report ISO/IEC 16022:2006; International Organization for Standardization: Geneve, Switzerland, 2006.

8. Leach, P.J.; Mealling, M.; Salz, R. A Universally Unique IDentifier (UUID) URN Namespace; Technical Report RFC4122. 2005. Available online: https://www.hjp.at/doc/rfc/rfc4122.html (accessed on 18 August 2020).
9. Kulshreshtha, R.; Kamboj, A.; Singh, S. Decoding robustness performance comparison for QR and data matrix code. In *CCSEIT '12: Proceedings of the Second International Conference on Computational Science, Engineering and Information Technology, Coimbatore, India, 26–28 October 2012*; Association for Computing Machinery: New York, NY, USA, 2012; pp. 722–731. [CrossRef]
10. Hudson, L.; Kursancew, V.; Weston, J. Pylibdmtx. 2020. Available online: https://github.com/NaturalHistoryMuseum/pylibdmtx (accessed on 18 August 2020).
11. Hudson, L.; Newby, A. Pyzbar. 2020. Available online: https://github.com/NaturalHistoryMuseum/pyzbar (accessed on 18 August 2020).
12. Bluetooth Special Interest Group. *Specification of the Bluetooth System*; Bluetooth Special Interest Group: Kirkland, WA, USA, 2014.
13. 802.15 WG—Wireless Personal Area Network Working Group. *IEEE Approved Draft Standard for Low-Rate Wireless Networks*; Number IEEE 802.15.4-2020; IEEE: Piscataway, NJ, USA, 2020.
14. Weis, S.A. RFID (radio frequency identification): Principles and applications. *System* **2007**, *2*, 1–23.
15. Garcia, F.D.; de Koning Gans, G.; Muijrers, R.; Van Rossum, P.; Verdult, R.; Schreur, R.W.; Jacobs, B. Dismantling MIFARE classic. In Proceedings of the European Symposium on Research in Computer Security, Málaga, Spain, 6–8 October 2008; pp. 97–114.
16. NFC Forum. *NFC Data Exchange Format (NDEF) Technical Specification*; History: Cheltenham, UK, 2006.
17. NFC Forum Technical Specification. *SNEP: Simple NDEF Exchange Protcol*; NFC Forum: Wakefield, MA, USA, 2008.
18. Lotito, A.; Mazzocchi, D. OPEN-SNEP project: Enabling P2P over NFC using NPP and SNEP. In Proceedings of the 2013 5th International Workshop on Near Field Communication (NFC), Zurich, Switzerland, 5 February 2013; pp. 1–6. [CrossRef]
19. NXP Semiconductors B.V. *PN532/C1—Near Field Communication (NFC) Controller*; NXP Semiconductors B.V.: Eindhoven, The Netherlands, 2017.
20. Microchip Technology Inc. *ATmega16U4/ATmega32U4—8-Bit Microcontroller with 16/32K Bytes of ISP Flash and USB Controller*; Microchip Technology Inc.: Chandler, AZ, USA, 2016.
21. El-hajj, M.; Fadlallah, A.; Chamoun, M.; Serhrouchni, A. A Survey of Internet of Things (IoT) Authentication Schemes. *Sensors* **2019**, *19*, 1141. [CrossRef] [PubMed]
22. Bradley, J.; Denniss, W.; Tschofenig, H.; Jones, M. OAuth 2.0 Device Authorization Grant. 2019. Available online: https://tools.ietf.org/html/rfc8628 (accessed on 18 August 2020)
23. Hardt, D. The OAuth 2.0 Authorization Framework. 2012. Available online: https://tools.ietf.org/html/rfc6749 (accessed on 18 August 2020)
24. Kothmayr, T.; Schmitt, C.; Hu, W.; Brünig, M.; Carle, G. DTLS Based Security and Two-Way Authentication for the Internet of Things. *Ad Hoc Netw.* **2013**, *11*, 2710–2723. [CrossRef]
25. Poulter, A.J.; Johnston, S.J.; Cox, S.J. pySRUP—Simplifying Secure Communications for Command Control in the Internet of Things. In Proceedings of the 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), Limerick, Ireland, 15–18 April 2019; pp. 273–277. [CrossRef]