How to Prevent Misuse of IoTAG?

Bernhard Weber
Dept. Electrical Engineering and Information Technology
Ostbayerische Technische Hochschule
Regensburg, Germany
email:
bernhard1.weber@st.oth-regensburg.de

Lukas Hinterberger
Dept. Mathematics and Computer Science
Freie Universität
Berlin, Germany
email:
lukas.hinterberger@fu-berlin.de

Sebastian Fischer* and Rudolf Hackenberg†
Dept. Computer Science and Mathematics
Ostbayerische Technische Hochschule
Regensburg, Germany
email:
sebastian.fischer@oth-regensburg.de*
rudolf.hackenberg@oth-regensburg.de†

***Abstract*—Since IoT devices are potentially insecure and offer great attack potential, in our past research we presented IoTAG, a solution where devices communicate security-related information about themselves. However, since this information can also be exploited by attackers, we present in this paper a solution against the misuse of IoTAG. In doing so, we address the two biggest problems: authentication and pairing with a trusted device. This is solved by introducing a pairing process, which uses the simultaneous authentication of equals algorithm to securely exchange and verify each others signature, and by using the server and client authentication provided by HTTP over TLS. We provide the minimum requirements and evaluate the methods used. The emphasis is on known and already proven methods. Additionally, we analyze the potential consequences of an attacker tapping the IoTAG information. Finally, we conclude that the solution successfully prevents access to IoTAG by unauthorized clients on the same network.**

Keywords*—*Internet of Things; IoTAG; device pairing; device authentication; trusted connection.

I. Introduction

As more and more devices are connected to the Internet, the so-called Internet of Things (IoT), the risk that the devices will be misused by attackers is also increasing [1]. In order to obtain an overview of one's own devices in the home network, especially in the consumer sector, we have developed the *IoT* Device Identific*A*tion and Reco*G*nition (*IoTAG*) solution [2]. The devices provide security-relevant information about themselves to a central location (e.g., the router), which can use this information to make an analysis about security. The security analysis can be done once for each individual device and once for the complete network. IoTAG is made available to the device's network as a service which is accessible using Hypertext Transfer Protocol Secure (HTTPS) and uses the JavaScript Object Notation (JSON). IoTAG in its currently proposed version allows the access of this information by any device on the same network.

Since this information can also be useful to a potential attacker, we want to extend IoTAG so that the data is only shared between the device and a trusted central point (hub). This is to prevent an attacker from, for example, reading the firmware version of a device via IoTAG and thus finding a potential vulnerability if the firmware is no longer up to date.

In this paper we evaluate the various options for establishing a trustworthy connection that cannot be misused by a foreign device afterwards. Several aspects have to be taken into account. First, it may be possible that there is no hub in the network. In this case, the IoTAG should not be retrievable. However, if an IoTAG-capable hub is subsequently installed, it must be possible to activate it. In the second case, the IoTAG should only be read by a trusted hub. Each device must remember this hub. A subsequent change must be possible, but only with the explicit consent of the user. Otherwise, the entire mechanism is obsolete if an attacker can still find a way to get at the data.

At the end of the paper, we evaluate the risk if an attacker gets hold of the IoTAG information and what dangers result from this. If the IoTAG is used sensibly and, ideally, all devices are constantly provided with updates, then this security mechanism is not necessary, because the attacker cannot find any new attack surfaces even with the information.

The paper is structured as follows: Section II starts with the related work and similar approaches. Section III describes the security threat to our IoTAG solution. Section IV covers the solution to prevent attackers to misuse IoTAG. Section V contains the specified minimum requirements and in Section VI, we evaluate the security, if an attacker gets the IoTAG data. At the end, in Section VII, a brief conclusion is given.

II. Related Work

There are already a number of approaches for implementing a pairing process between IoT devices and a central hub.

J. Han et al. propose a method that enables pairing without human intervention [3]. Instead, the recordings of multiple devices within an infrastructure are matched to ensure that the devices are in physical proximity. This approach is based on the assumption that events within the infrastructure, such as the movements of a person, can be detected by multiple

 ISBN: 978-1-61208-845-7

devices with their respective sensors and thus the position of the device can also be determined This method assumes that the devices are physically shielded from the outside world and explicitly refers to smart home environments. It also assumes that a potential attacker has no access to this infrastructure. Since our approach is intended to enable the use of IoTAG in both industrial and private environments and both indoors and outdoors, Han et al.'s approach is not applicable.

The approach developed by X. Li et al. also relies on the combination of several sensor values to validate the pairing process [4]. The authors propose the use of wearables for this purpose. For example, when a button is pressed, it is possible to record the hand movement required for this via a smartwatch and to compare whether the button was actually pressed by the user. However, since it cannot be assumed that the end user of the IoTAG device has the necessary hardware, this approach is not suitable for use with IoTAG.

A similar approach is followed by S. Pan et al. [5]. This is based on the motion data collected by the device sensors and a camera that also evaluates these movements. This method is suitable for the use of devices that are both, portable and wireless, but not for wired or industrial devices.

III. Security Threat

IoTAG provides a network scanner with valuable information about the devices and their specifications in a IoT network. It supports the evaluation of the security of those networks, but the proposed IoTAG standard has no limitation to who could access the provided data. This allows it to be used by a variety of software programs and keeps the standard open to use for everyone. On the other hand, this could also be used by an attacker as an easy way of gaining knowledge about the targeted network and the exact devices and firmware versions used. Although, there are other relatively easy and reliable ways to accomplish that, as shown by V. Sivaraman et al. [6], this paper provides a solution to secure the IoTAG further against malicious use.

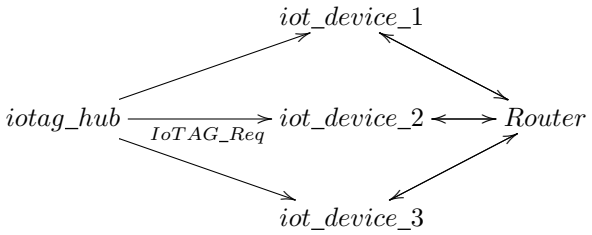

Figure 1: Example of a simple network using IoTAG.

The current published version of IoTAG has no limitation of which devices on the same network are allowed to access the device's metadata. This enables the use of IoTAG by simply scanning a network for devices which are offering the IoTAG on the specified port. Figure 1 shows a simple network topology containing IoT devices and a router, which acts as a gateway to the internet, and a device which accesses all available IoTAGs. Such a device or software is hereinafter called "IoTAG hub".

The target of the attack, which the solution proposed in this paper aims to solve, is to gain access to the IoTAG for malicious use. The attack is deemed successful if a device in the same network is able to access the meta data provided by IoTAG without the explicit permission by the network's owner. For example, a smart speaker, which is connected to the network and was programmed to collect all available IoTAGs and send them to the manufacturer for market studies, is considered malicious. In this paper we assume that the attacker already gained unrestricted access to the user's network. This, for example, could have happened by compromising an existing device, which is reachable from the internet, or by guessing or brute-forcing the pre-shared key of a wireless network. Additionally the attacker has the ability to capture all packets send over the network.

IV. Solution

To lock down the IoTAG against malicious use, its access must be limited to applications trusted by the user. This is achieved by only sending the tag to a requesting and authorized client. It splits the solution into two parts: Specifying the protocol used for authenticating a request and defining how the client gains the trust of the user and therefore gets credentials for the authentication process.

A. Authentication

For the communication with the clients, the current draft specifies the Hypertext Transfer Protocol (HTTP) over Transport Layer Security, short TLS, which combines to HTTPS (Hypertext Transfer Protocol Secure), as defined in RFC 2818 [7]. This limits the authentication to protocols which are based on HTTPS or alternatively to a self-developed protocol. As security is the main focus of this project, an own protocol is the least favorite of those option, because it would open an area for potential vulnerabilities in comparison to the use of well established and audited protocols. HTTP over TLS, as implied by the name, tunnels HTTP through a TLS encrypted connection. Both Protocols offer their own ways of authenticating a user. HTTP has two standardized ways, namely Basic Authentication and Digest Access Authentication, which are specified in the RFC 2617 [8]. The main difference is that Basic Authentication does only encode, but not encrypt, the transmitted password, while Digest Access Authentication hashes the password and, if implemented securely, also prevents replay attacks.

TLS on the other hand allows the client to provide the server with a client certificate during the handshake, as specified in RFC5246 7.4.6 [9], which allows for the authentication of the client. The client certificate has to follow the X.509 format, which is specified in RFC 5280 [10]. This format includes a signature which is unique and either signed by a trusted authority or using the certificate's private key. During the TLS handshake, the client has to proof that it holds the

 ISBN: 978-1-61208-845-7

private key, belonging to the certificate and the server. In this case, the IoTAG service has to check if provided certificate belongs to an authorized client. This can either be achieved by maintaining a certificate authority, which signs the signature of the certificates for the allowed clients, or, if it is signed by its own private key, by validating and comparing the certificate's signature to a list of allowed signatures.

The proposed solution for IoTAG is the use of the client certificate based authentication in the TLS protocol version 1.2 or above. The devices need to offer at least the cipher suite "TLS_RSA_WITH_AES_256_CBC_SHA256" for TLS 1.2 [9] or "TLS_AES_256_GCM_SHA384" when using TLS 1.3 [11] to be future proof and to ensure that the device supports both AES-256 and SHA-256, which is used during the pairing process. The hub needs to support both TLS versions and both ciphers and ideally should also support the use of the other cipher suites specified. This option is chosen over HTTP authentication because it is implemented in the security layer itself. Basic authentication's security is purely based on the secure channel which is used for communication. A potential attacker could silently break the SSL encryption during the initial pairing, as the server certificate used by the device is unknown to the hub at this point. The credentials are transmitted without any encryption and could easily be reused, once the attacker knows them. The Digest Access Authentication on the other hand is encrypted, but the credentials need to be sent to the hub at some point during the pairing. This initial transmission could be encrypted by using an additional pairing algorithm, but this would unnecessarily complicate the process. The TLS authentication is the best fit, because the private key of both sides are never shared with anyone else and the signature can easily be saved during the pairing process and then be verified on each connection.

B. Pairing

To ensure that the client connecting to an IoT device is trusted, an initial pairing is needed. This prevents potential malicious clients to gain access to the IoTAG of a device, which otherwise could help them to get useful information for an attack (see Section VI). The pairing has two main aspects it needs to achieve: The IoTAG service on the IoT device needs to be sure that the connecting hub is trusted by the user and that the secure connection is directly between the two devices without anyone listening (man-in-the-middle). Optionally, the hub should also be able to verify, if the server is paired with the actual device.

The proposed solution for the verification of the hub is the use of a key phrase, hereinafter called PIN, which is randomly generated for authentication purposes and limiting the ability to pair with a device to specific time slots. The complexity of the PIN is chosen by the device manufacturer with minimal requirements, as stated later in this section. The generated PIN needs to be provided to the end user together with the device. For example, it could be written onto a sticker on the device itself, or it could be printed onto a piece of paper which comes in the box.

The time slot limitation can be implemented by the manufacturer in the following two ways, depending on the type of device. The first option is to give the device a physical button which needs to be pressed during the pairing process and opens the IoTAG service for new connections for a limited time. This solution works for devices that already have a button or can easily integrate one in their hardware design. This is the preferred option as it needs an explicit action done by the user, which ensures that it is the user's intention to enable IoTAG. The second option provides a way to accommodate devices where the manufacturer does not want to or can not integrate a button into the design. It allows new connections to come in for a specific amount of time after each fresh boot of the device. In both cases, the option to pair with a new hub is disabled once a client is paired to the device and can only be enabled again by a factory reset of the device. This further secures the protocol against malicious use, as it prevents any third party access to the IoTAG in an already configured environment. The exact minimum requirements are specified in Section V.

To provide the hub with a way of verifying the identity of the device, the manufacturer can provide the user with a URL pointing to the public key of a certificate authority as specified in RFC 5246 [9] and RFC 8446 [11] in the X.509 format. The given certificate authority must be the one used to sign the server certificate of the IoT device. The user can provide the hub with this URL during the pairing process, which enables the hub to verify on each connection that the certificate used by the IoT device is a genuine one and approved by the manufacturer. Alternatively, the hub could already come with a list of those certificate authorities to further ease the pairing process for the user. This step is only an additional layer of security and is not required for a secure communication, as the hub can already verify that the other device knows the PIN.

In conclusion, the proposed pairing process has the following steps: the IoT device enables the access to the IoTAG service running on it. This is either done during the boot of the device, or by a button press. In each case, the IoTAG service becomes unavailable, if no pairing is done after a specific time period. During this time slot, the device broadcasts every second a "hello"-packet to the whole network, it is connected to. The hub receives those packets and lets the user know that a new device is available for pairing. The user is then prompted to input the PIN and the hub generates and saves a new client certificate. The signature of this certificate is later encrypted and sent to the IoT device. Once the process is initiated by the user, the hub sends a "hello"-packet back to the device. This communication is done using TCP on the port 27071.

The encryption is done using the Advanced Encryption Standard (AES), as described by V. Rijmen et al. [12], using Cipher Block Chaining (CBC), as described by M. J. Dworkin et al. [13], with a random initialization vector and a key length of 256 bit. Additionally, a random sequence with the length of

 ISBN: 978-1-61208-845-7

the key is added to the beginning of the certificate signature. This allows the IoT device to decrypt the whole signature without knowledge about the randomly generated initialization vector.

The key used for the encryption, is generated using the simultaneous authentication of equals algorithm (SAE), a password-authenticated key agreement protocol, which was developed by D. Harkins in 2008 [14] and was later updated and published as the Dragonfly Key Exchange in RFC 7664 [15]. Both, the 802.11 Wi-Fi specification by IEEE [16] and the newest Wi-Fi protected access version 3 [17], use SAE as part of their security. Due to this wide use, IoT devices should be capable of it already, or, at least have the processing resources needed for an implementation. SAE enables two devices to calculate the same, high-entropy secret, called PMK, while using a potentially low-entropy key as the shared secret. It prevents offline attacks, as the PIN cannot be guessed without contacting the device for verification, online attacks, as an attacker will not be able to guess the PIN or PMK by just observing, and replay attacks, as the knowledge of a PMK is of no use for each new pairing process. Additionally, once a hub is paired, even the knowledge of the PIN does not enable an attacker access to the IoTAG, as only one hub is allowed. Overall, the high-entropy key generated by SAE, allows the IoTAG service to use user-friendly and relatively easy keys (PIN) as a secure way to authenticate a hub. For compatibility with AES 256 and to ensure compatibility with the IoT devices, the used hash algorithm for IoTAG for SAE is set to SHA-256 [15] [16].

The key generation process for IoTAG access works as follows: if the device receives a “hello”-packet during the pairing time slot from a potential hub, it sends a SAE commit message, as specified in IEEE 802.11 [16], to the hub, which responds with his commit message. Once the device is done with the key generation, it sends the SAE confirm message to the hub, which answers with its confirm message. After receiving the message both, the hub and the device verify the validity of the calculated values and, if they are correct, the PMK is successfully determined and the hub uses this PMK as the key for the AES encryption of the signature. The whole pairing process, including the exchange of the signatures, is visualized in Figure 2.

The encrypted message containing the client certificate’s signature is afterwards send to the IoT device, using the same communication channel. Once the device receives the message from the hub, it uses the previously calculated PMK to decrypt the provided signature, while also using a random initialization vector and discarding the first block after the decryption. If the message is of a valid format, the signature is saved, the pairing gets disabled and connections to the device are limited to the provided client certificate. Additionally, the device responds to the server with an encrypted message containing its certificate signature using the same PMK and procedure as described before. The hub can then verify the format and save the signature, so it can verify the device’s identity later.

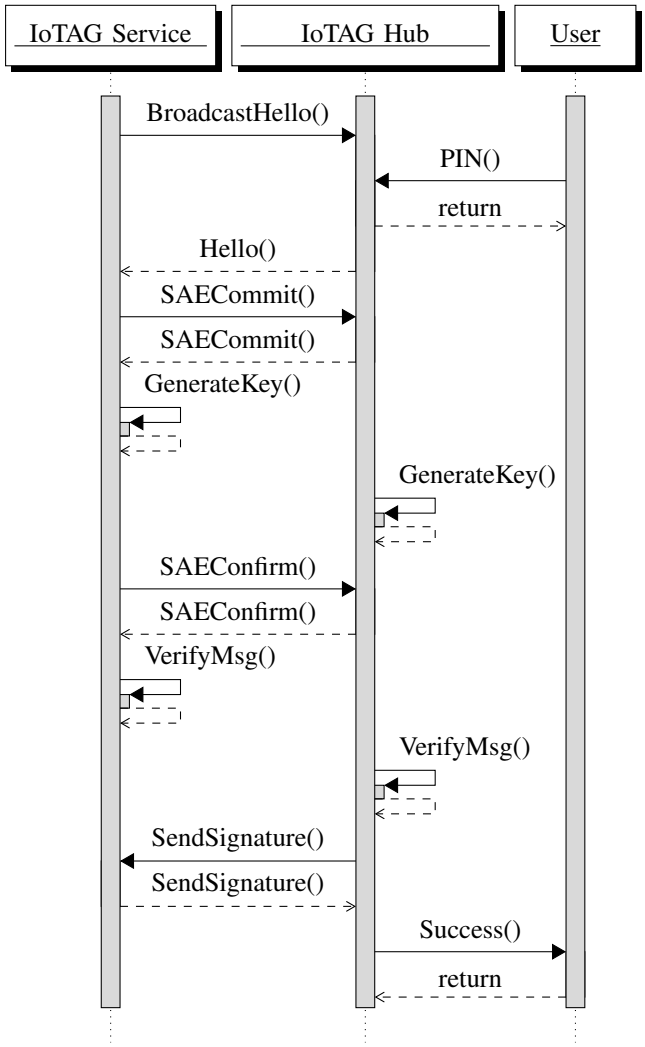

Figure 2: Pairing process between the device and the hub.

V. Specified Minimum Requirements

As a guideline for the manufactures, we define the following minimum requirements. The time slot chosen, should provide the user with enough time to comfortably configure the connection, while also providing additional security against external attacks. It should be at least 1 minute and shall not exceed 10 minutes. The manufacturer is free to choose a duration in between those limits, depending on the device type.

Concerning the security of the PIN, the following limits are specified: The PIN has to be at least decimal and 4 digits long. Alternatively it can be every other valid UTF-8 encoded string. This allows the manufacturer to use already existing unique and secret information, for example a pairing password which is used for the manufacturer’s app. Also, the device needs to limit the amount of failed pairing requests to three per pairing time slot. This ensures that the average time needed for guessing the PIN using a brute-force attack is long enough

 ISBN: 978-1-61208-845-7

TABLE I: IOTAG DATA [18]

1	Manufacturer
2	Name
3	Serial number
4	Type
5	ID
6	Category
7	Secure boot
8	Firmware
9	Client software
10	Updates
11	Cryptography
12	Connectivity
13	Services

that its success is unrealistic, as each pairing or restart can only be initiated by the user and not by the attacker.

VI. Security Evaluation

In this section, we consider the threats in the event that an attacker can query the IoTAG information from one or all devices on the network. This means that the attacker has access to the network and is able to retrieve the IoTAG. At first, we have to look at the provided data from IoTAG (Table I). If an attacker manages to get the data from all devices, he gets a complete overview of the IoT network (provided that all devices support the IoTAG). In more detail, some information can be used to attack single devices.

A. Device Information

The device information, like manufacturer, name and serial number can be used to find existing security vulnerabilities. In addition, security vulnerabilities can also be found for other products of the manufacturer, which are often found in similar form in several products. With this information, there is no need for black box analysis. The attacker does not have to laboriously search for information about the individual devices and thus try out a device detection. With this simple information, the time required for a successful attack on individual IoT devices is reduced.

B. Software and Updates

The current software version and the update link can be used to check for outdated software. This information can also be retrieved with only the device information, but as it is presented in the IoTAG, the information can be processed automatically. This can save some time for an attacker, but the benefits for a network administrator are greater, as it provides no secret information.

C. Cryptography

The Cryptographic Information contain the concrete encryption algorithms and key lengths. This can help an attacker identify easy-to-crack methods and short key lengths. This allows the weakest device to be found specifically. This can be a huge security risk, but hiding the algorithms (security by obscurity [19]) is not the goal of encryption. The security should only be dependent on the key strength.

D. Secure boot, Connectivity and Services

Other security relevant information, like secure boot, the different connectivity and services can be used to gain more information about potential attack vectors. Most of these information are not exclusively to IoTAG and can also be found out in other ways by an attacker.

E. Evaluation

In summary, an attacker can use the IoTAG information to accelerate his attack or to find weak devices. However, the point of IoTAG is that an administrator can find the same vulnerabilities in the network and thus close the potential gaps. Thus, the advantage of IoTAG is higher than the danger that an attacker can tap the information. Furthermore, the methods presented in this paper make sure that no unauthorized entity can get access to the IoTAG.

VII. Conclusion

In its previous form, IoTAG was vulnerable to misuse by an attacker who could use it to retrieve security-critical information about the IoT devices installed in a network and thus identify the weakest point. The reason for this was the fact that the devices could not distinguish to whom they were providing this information.

This vulnerability was eliminated by the method presented in this paper. Pairing the devices with a central hub, responsible for monitoring the devices and authorized by the network operator to use the IoTAG data ensures that the devices do not respond to arbitrary requests.

The pairing is realized by the central hub transmitting the signature of a TLS certificate it has created to a device. When the HTTPS connection is established later, the client can use this signature to validate that its communication partner is the hub.

In order to create a secure communication channel between the device and the hub for the pairing process, the user stores a device-specific PIN on the hub. This PIN is used as authentication during the key exchange process, which in our case is SAE. By means of the generated key, the communication is encrypted using AES. The time factor also plays a role. The pairing process cannot be carried out at will, but is limited to a period of time predefined by the device manufacturer. This prevents an attacker from guessing the PIN and restart the pairing process.

With the completion of this work, the focus for the further development of IoTAG can now be placed on practical testing and further improvements based on the findings.

 ISBN: 978-1-61208-845-7

REFERENCES

[1] M. Hogan and B. Piccarreta, "Interagency Report on the Status of International Cybersecurity Standardization for the Internet of Things (IoT)," Tech. rep., National Institute of Standards and Technology, 2018.

[2] L. Hinterberger, B. Weber, S. Fischer, K. Neubauer, and R. Hackenberg, "IoT Device IdentificAtion and RecoGnition (IoTAG)," CLOUD COMPUTING, 2020, p. 17, 2020.

[3] J. Han et al., "Do You Feel What I Hear? Enabling Autonomous IoT Device Pairing Using Different Sensor Types," in 2018 IEEE Symposium on Security and Privacy (SP), pp. 836–852, 2018, doi: 10.1109/SP.2018.00041.

[4] X. Li, Q. Zeng, L. Luo, and T. Luo, "T2Pair: Secure and Usable Pairing for Heterogeneous IoT Devices," in CCS '20, p. 309–323, Association for Computing Machinery, New York, NY, USA, 2020, ISBN 9781450370899, doi:10.1145/3372297.3417286.

[5] S. Pan et al., "UniverSense: IoT Device Pairing through Heterogeneous Sensing Signals," in HotMobile '18, p. 55–60, Association for Computing Machinery, New York, NY, USA, 2018, ISBN 9781450356305, doi:10.1145/3177102.3177108.

[6] V. Sivaraman, D. Chan, D. Earl, and R. Boreli, "Smart-Phones Attacking Smart-Homes," in Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks, WiSec '16, p. 195–200, Association for Computing Machinery, New York, NY, USA, 2016, ISBN 9781450342704, doi:10.1145/2939918.2939925.

[7] E. Rescorla, "HTTP Over TLS," RFC 2818, RFC Editor, May 2000, doi:10.17487/RFC2818.

[8] J. Franks et al., "HTTP Authentication: Basic and Digest Access Authentication," RFC 2617, RFC Editor, June 1999, doi:10.17487/RFC2617.

[9] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246, RFC Editor, August 2008, doi: 10.17487/RFC5246.

[10] D. Cooper et al., "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," RFC 5280, RFC Editor, May 2008, doi:10.17487/RFC5280.

[11] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," RFC 8446, RFC Editor, August 2018, doi:10.17487/RFC8446.

[12] V. Rijmen and J. Daemen, "Advanced encryption standard," Proceedings of Federal Information Processing Standards Publications, National Institute of Standards and Technology, pp. 19–22, 2001.

[13] M. J. Dworkin, "Recommendation for Block Cipher Modes of Operation: Methods and Techniques," National Institute of Standards and Technology, 2001.

[14] D. Harkins, "Simultaneous Authentication of Equals: A Secure, Password-Based Key Exchange for Mesh Networks," in 2008 Second International Conference on Sensor Technologies and Applications (sensorcomm 2008), pp. 839–844, 2008, doi:10.1109/SENSORCOMM.2008.131.

[15] D. Harkins, "Dragonfly Key Exchange," RFC 7664, RFC Editor, November 2015, doi:10.17487/RFC7664.

[16] I. . W. Group, "IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks–Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016), pp. 1–4379, 2021, doi:10.1109/IEEESTD.2021.9363693.

[17] Wi-Fi-Alliance, "WPA3 specification version 2.0," 2020.

[18] L. Hinterberger, S. Fischer, B. Weber, K. Neubauer, and R. Hackenberg, "Extended Definition of the Proposed Open Standard for IoT Device IdentificAtion and RecoGnition (IoTAG)," The International Journal on Advances in Internet Technology, vol. 13, pp. 110–121, 2020.

[19] R. T. Mercuri and P. Neumann, "Security by obscurity," Commun. ACM, 46, p. 160, 2003.

 ISBN: 978-1-61208-845-7