Technical Report
KN–2017–DISY–01

Distributed Systems Laboratory

HomeCA:
Scalable Secure IoT Network Integration

Robert Müller
Marcel Waldvogel **Daniel Kaiser**
Distributed Systems Laboratory
Department of Computer and Information Science
University of Konstanz – Germany

The sheer number of devices in the Internet of Things (IoT) makes efficient device integration into a user's home or corporate network a nightmare. More and more owners lose control over their devices, often due to badly chosen security defaults, software bugs, or broken protocols. The lack of user interface and the long period of device usage increase the plight. We identify several root causes, resulting in HomeCA, a comprehensive set of secure, vendor-neutral practices based on existing protocols and open standards. These practices avoid most of the common pitfalls, allow long-term permission management and secure usage, and include support for automatic device integration. We also present a protocol for ensuring secure key updates when acquiring device ownership.

Table of Contents

1 Introduction

The sheer number of devices in the Internet of Things (IoT) makes efficient device integration into a user's home or corporate network a nightmare. Not only is the number of incidents of lost control over IoT devices on the rise[1], which are then used to mount Distributed Denial of Service (DDoS) attacks; there is often also very little the users can currently do to prevent this[2]. To add insult to injury, the humongous problems caused by insecure and sometimes insecurable devices are complemented by poor pairing protocols, which are insecure by design or implementation: Many ZigBee applications profiles for smart homes allow by definition any malicious device to join the network and thus become trusted[3]. Wi-Fi is not much better, recommending insecure defaults[4] and opening barn doors with Wi-Fi Protected Setup (WPS)[5] allowing offline attacks[6].

The abuse of IoT devices is made possible through careless integration of the devices into the home or enterprise network. Web-enabled devices become accessible from anywhere in the world, rarely with secure passwords and fixed security problems. However, most manufacturers and users pay little attention to these facts or are not knowledgeable in the field. Some might not care about the security or privacy of the device and its data, forgetting the impact this device may have on its surroundings near and far.

The manufacturers have been asked to provide more security from the start[7][2]. We believe the core problems are as follows:

- Vendors deliver their devices with simple (savings in support calls) instead of secure setup.
- Vendors do rarely provide security updates, provide no easy way of installing them, remove functionality unnecessarily together with updates[8], or go out of business.
- Users do often leave passwords at the manufacturer's settings, if they can change them at all.
- Devices are often unnecessarily accessible from the Internet.

With HomeCA, we try to determine new ways of pairing home devices, try to get rid of passwords, and take advantage of new developments such as built-in security keys in microcontrollers[9]. At the same time, we are trying to remain vendor neutral and avoid unnecessary trust in the device keys, as manufacturing errors, database leaks, and low entropies have been known to cause problems in other publicized cases.

The HomeCA lifecycle consists of the following three workflows:

Manufacturing. Initial key pair creation, public key delivery *(optional)*.
Ownership change. Release *(active or passive)*, HomeCA discovery, key verification *(optional)*, key update, certificate creation, rights management *(optional)*.
Connection establishment. Service discovery, (D)TLS connection setup.
Refresh *(optional)*. Liveness verification, certificate lifetime extension, rights management update *(optional)*.

With these three processes, explained in Section 4, secure pairing and optional gatewaying can be provided, resulting in (1) removing insecure passwords and (2) reducing the dependency on updates, all in a vendor-independent layer with little overhead. HomeCA creates a protocol layer that (3) provides security,

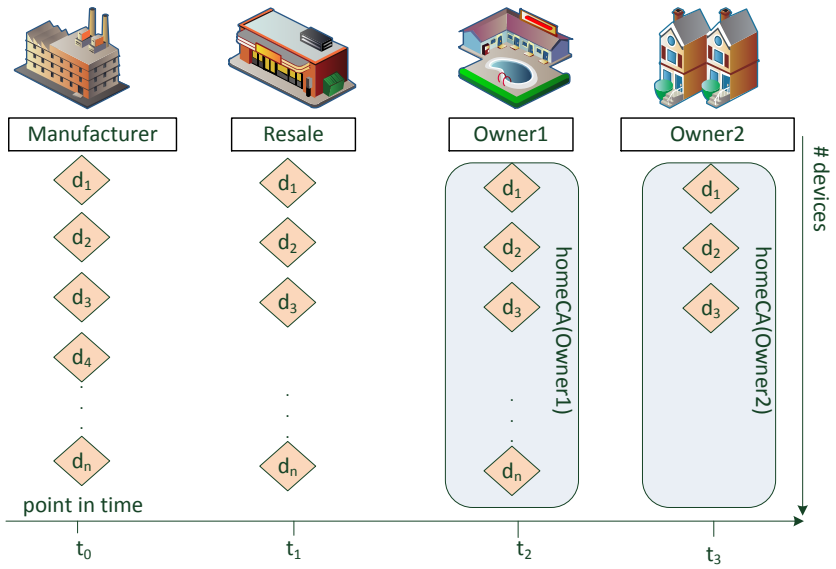

Fig. 1. HomeCA coverage on IoT devices life cycle

(4) does not require complex and potentially insecure user interaction, and (5) can help reduce the trust required in the manufacturer.

Figure 1 illustrates the lifecycle of IoT devices from manufacturing, resale, use in a home network to the possibility of a reuse at another home, e.g. a friends home. A fictional set of IoT devices $\{d_1, d_2, d_3, d_4, \ldots, d_n\}$ is created at a factory and being sold at a retail store. Once they are bought, they are integrated and operated in a first home network. Eventually, they are moved to a second home, showing disassociation and reassociation of the IoT device. We show the entire life cycle of an IoT device to cover the possible handling scenarios in application scenarios following.

HomeCA is a security protocol layer that covers the IoT devices within a trustworthy home network. It uses a machine-to-machine (M2M) authentication protocol for the automatic integration of new IoT devices. It is designed to protect the devices from unauthorized manipulation and access using a Certification Authority (CA) within the home network. Threats that are denied through HomeCA and the used *methods* are:

- lacking or corrupted (security) updates *(verification)*,
- corruption of the manufacturer itself *(verification)*,
- attacks from the Internet aimed at eavesdropping communication *(encryption)*, and
- attacks from the Internet aimed at taking control of the IoT device *(certificates)*.

2 Related Work

Existing research on the new challenges of IoT devices is available in [10], in the area of security of PKI encryption keys in [11], [12], and [4] on the key

management. A broad analysis of Authentication and Access Control for the IoT can be found in [13].

Research on secure and resource saving integration of WSN (wireless sensor Nodes) into the IoT as in [14] do not cover the use in WLAN networks. while Wi-Fi Protected Setup [5] and similar technologies do not provide the possibility to integrate a large number of devices.

Little is known about the actual steps behind the Atmel/AWS cooperation[9]. However, we assume steps similar to our Manufacturing workflow (overview in Section 1, details in Sections 4.1 to 4.5).

3 HomeCA Model

For the design of HomeCA, the major aspects security and the use cases are considered. We introduce our security considerations in Section 3.1 and present the HomeCA system and its use cases in Section 3.2.

3.1 Security Model

IoT devices are target of attacks that try to access the (maybe sensitive) sensor data or try to gain control over the device. Devices taken over are used for illegal actions like DDOS attacks. This type of attempts, usually originated from the Internet, trying to access IoT devices within the private network, are detected from HomeCA through their source. HomeCA can then dismiss those access attempts to its IoT devices. There is no unauthorized access to the IoT devices since traffic must pass the check of HomeCA and thus cannot reach the IoT device.

For privacy reasons it shall not be disclosed to the manufacturer, how many devices are within the personal/private network managed by homeCA. This is achieved by the signatures between the manufacturer, devices and homeCA.

Attackers try to find and access IoT devices connected to the Internet. For example they use scripts or tools to search with, i.e. the TELNET protocol. Other information about the devices may simply be stolen from a manufacturer or service website that the IoT device is eventually connected to. For this point it is sufficient to know the IP address the device within the home network is listening to.

Next, the attacker tries to log in with default credentials to collect all the devices that have not changed the factory-default username and password combination. Once successful, the attacker can do whatever he/she wants on the configuration of the IoT device. HomeCA protocol protects the access to the IoT devices and therefore prevents this type of direct access to the IoT devices, except they are legit.

Changes to the IoT device configuration requested by the owner are redirected to the HomeCA web-interface. As a consequence, multiple access attempts for IoT devices within the private network can be detected by the unusual high number within a definite time window. Allowed access attempts are additionally protected against maliciously configured bots by the use of i.e. CAPTCHA [15] or similar technologies capable of preventing bots from performing specific operations automatically.

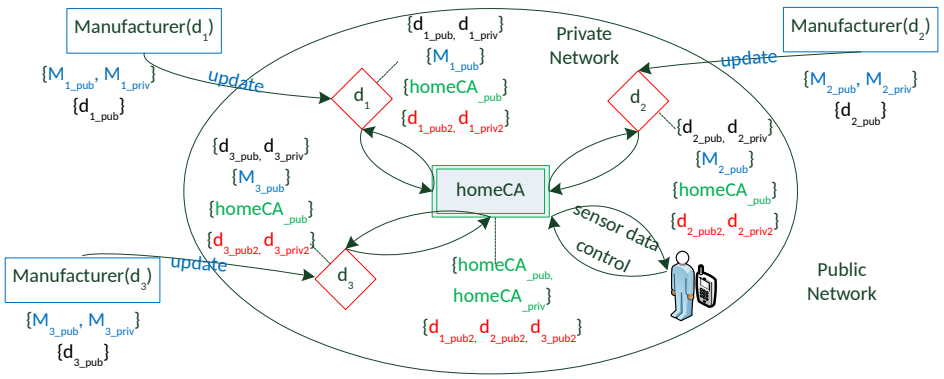

Fig. 2. HomeCA System Overview

3.2 System Overview

Figure 2 shows the protocol set-up for the secure communication of three devices with their three manufacturers and within a private/personal network. First, keys for encrypted communication are exchanged between the Manufacturers (M) $M_{1,2,3}$ and Devices (d) $d_{1,2,3}$. This allows HomeCA to verify an update issued by $M_{1,2,3}$ before it is applied by $d_{1,2,3}$. Second, secure keys for bi-directional communication are exchanged between HomeCA and $d_{1,2,3}$. This happens within the private/home network and provides a trusted communication channel between the IoT device and HomeCA for access and control of the IoT devices $d_{1,2,3}$. For the execution there is no user-interaction required.

In order to make the interface scalable for many devices we chose an interface that does not require human interaction and works completely automatic based on a trust relationship between manufacturer, device and home network. The protocol execution steps are shown in Figure 3 between the parties Manufacturer M, device d and our network certification authority HomeCA.

To motivate the HomeCA design decisions, we discuss different use cases that require the secure infrastructure to update its trust relationship due to:

- Device d is brought into a private wireless network for the first time. HomeCA recognizes the new device and has to authenticate against d to prove it is authorized to manage d. To do this, there are two ways:
  1.) Either directly at the manufacturer M when the device is purchased, the device ID is transmitted from M to HomeCA: $\{deviceID(d)\}$. After a handshake, HomeCA can authenticate and establish a secure communication with d.
  2.) The other way is used e.g. when d is bought at a retail store. d and HomeCA meet in a secure private network and no interference is being detected by d. HomeCA is actively asking for a new to be integrated device, when it knows i.e. from the application registry for sensor data from the sensor of the IoT device d. Diffie-Hellman key exchange between d and HomeCA is used to obtain k to create new public and private keys.
- Device d is sold /given away /defect: If HomeCA did not see its IoT Devices within a defined period of time t, the certificates for that specific device are

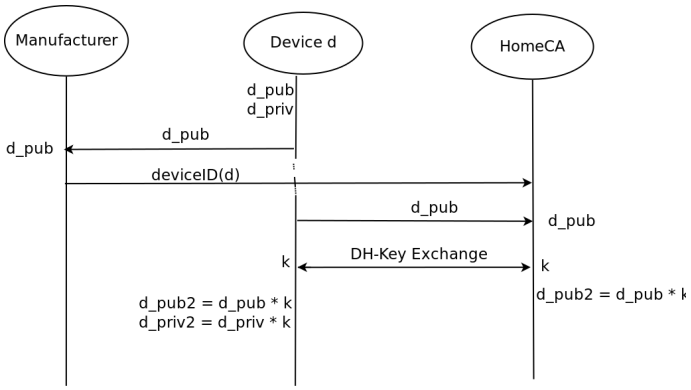

Fig. 3. HomeCA protocol integrating a new device

removed from HomeCA and must re-authenticate if it appears in the private network again at a later point in time. The initial time window is set to $t = 10$ days but shall be adopted based upon the dynamic of the actual user behavior of the private network, i.e. the rate at which devices are introduced and removed.

- Device software d has a security weakness or needs to be updated/patched: Only HomeCA can approve and sign a trustworthy update. Attackers without this certificate cannot create a valid update for the device. HomeCA monitors and keeps track of all updates the IoT devices receive from the manufacturer.
- If the device d is compromised and behaves strangely, the HomeCA can remove the certificate for the device d.
- If the manufacturer M is compromised and no longer trustworthy, the HomeCA can automatically inform the IoT devices assigned with HomeCA about this situation.

4 HomeCA Workflows

The goal of the HomeCA workflows is to create a trusted association between the device and the network and thus the other connected devices. Thereby, reasonable but minimal trust should be expended on the manufacturer of the device and the HomeCA itself. To achieve this, as little sensitive information as necessary should be processed by other devices. Key pairs should be solely managed by the device owning them, meaning private key material should not be shared or transmitted unless absolutely necessary.

4.1 Manufacturing: Initial Key Pair Creation

Key creation at manufacturing time comes with the following three options:

Local generation. This is the ideal, but requires a good internal source of reliable and confidential random bits[16], together with ample processing power. If the random number generator (RNG) does not meet these criteria, the key will be weak and can be guessed with little effort[6].

Table 1. IoT Device Protection Mechanisms and Access Rights

Service	Allowed Function	Required Protection	Use Gateway?
Heating	Report temperature	TLS 1.2, AES-GCM	—
Garage Door	Open/Close	SSLv3	—
Home Entertainment	Volume control	none	✓
Coffee Maker	Preheat machine	SSLv2	✓
Health	Query vitals	TLS 1.2	—

Key feeding. The device does not possess a good enough RNG. Therefore, a key pair is created by an external device and fed to the IoT, e.g. during burn-in test. As a result, the external device owned by the manufacturer knows the private key. This key can then be obtained by third parties[17] and therefore should not be considered secret.

Randomness feeding. An alternative way to cope with a bad on-chip RNG is to use an external random source, e.g. provided again during burn-in test. The device then calculates its own key pair based on that data and keeps the private key secret. However, anyone knowing the key generation algorithm (often public) and the random bits fed, can recreate the private key and verify it with the public key, duplicating the problems of *key feeding* above.[1]

Even if the key is generated on the device itself, the manufacturer may be able to extract the key using low-level device debug mechanisms such as JTAG[18].[2] Exemplary IoT devices and their protection mechanism are shown in Table 1.

4.2 Manufacturing/Resale: Public Key Delivery

After the key pair creation, optionally, the public key may be extracted from the device and associated with the device serial number. If this public key remains associated with the particular device during the entire sales chain, it can be used to simplify the following ownership change process (cf. Section 4.5).

4.3 Ownership Change: Release

A device first must be released by its previous owner, before it starts looking for a new home. This is best done using an explicit command to the device by the previous owner (which, on the first sale, is the manufacturer), called an *active release*. Fallback methods (*passive release*) should include a timeout (not having seen the HomeCA for a predefined period, e.g. a week) or could include some form of physical interaction, including a dedicated button. To some extents, this can be considered a variation of the Resurrecting Duckling[19].

4.4 Ownership Change: HomeCA Discovery

A released IoT device, whether actively or passively, will search for a new home network. After connecting to a network, the device will first use Multicast DNS

[1] Mixing in data from a weak local RNG is possible, but will not significantly increase the attack effort.

[2] A particularly malicious manufacturer might even include a fuse bit, which after activation would hide itself and the private key access possibility.

Service Discovery (mDNS-SD aka Zeroconf aka Bonjour)[20] to discover the HomeCA. But how will connect to the network in the first place?

1. Probe an open network. This is easiest to implement, but will
2. Connect to a (closed) WPA Enterprise network. The authentication phase is extended such that contacting the HomeCA in a limited fashion is already possible then. Later connections to a WPA Enterprise network will present the HomeCA certificate to safely connect in EAP-TLS mode.

As long as our device has not been accepted and granted full access by a HomeCA (cf. Section 4.6), it will keep probing.

4.5 Ownership Change: Key Verification

When the device public key has been delivered as part of the order, it may be pre-configured in the HomeCA, allowing later automatic joining as soon as the device comes within network range. This is of course the most comfortable and thus recommended operation.

Without public key delivery, the device will try to join the network and contact the HomeCA. The device will sit in this *unverified* state until the user has manually confirmed the device addition. In the simplest case, this is performed similar to the WPS Push-Button authentication[5]. However, we envision a control application on the owner's smartphone, which displays device information (name, type, join time, ...) to verify it is actually the correct device. This application may then also be used for rights management (cf. Section 4.7).

4.6 Ownership Change: Key Update

The key currently associated with the device might be known to the manufacturer (cf. Section 4.1) and/or previous owner. Anyone knowing the key can directly impersonate the device or perform an undetectable man-in-the-middle (MITM) attack. Thus, reusing the same key is to be avoided. Therefore, on association, a new key is generated. Our preferred mechanism for key update is described in Section 5, providing several desirable cryptographic properties.

To avoid accidental association with a foreign HomeCA, an initial integration of a device must be accepted within a UI presented to the user, i.e. on a smartphone app. Also, an IoT device will only accept a new HomeCA as its owner, if it has been previously released (cf. Section 4.3). If the user actively searches for devices, known and unknown ones are shown. If not, unknown ones are hidden.

On a key update, any sensitive data on the device is also wiped, making them unavailable to the previous owner. This may include access rights to other devices, measurement data, or information received from other devices while with the previous owner[21].

4.7 Ownership Change/Refresh: Rights Management

Optionally, the owner, possibly through the HomeCA app, can assign rights this device should have toward other devices. This is represented as an access control list (ACL), possibly in matrix form, listing allowed operations per target device or device group. An example is shown in Table 1.

4.8 Ownership Change/Refresh: Certificate Creation

The HomeCA signs the public key together with device information, validity period, and the optional ACL. This certificate is passed to the device.

4.9 Refresh: Liveness Verification

Liveness verification is initiated by the IoT device and used to state their availability to HomeCA, to regularly update the certificate with the connection information and optional ACL.

It also allows IoT devices to activate the certificate-based relationship with HomeCA after a long period without interaction, e.g. during holidays that exceeds the time window t.

4.10 Connection Establishment: Service Discovery

Configurationless service discovery for HomeCA is done using DNS Service Discovery (DNS-SD) [22] over multicast DNS (mDNS) [20]; widely known as *Zeroconf* or *Bonjour*. Because this solution does not protect the users' privacy (see e.g. [23]), we include a privacy extension as presented in [24].

4.11 Connection Establishment: (D)TLS connection establishment

An initiator device A presents its certificate using TLS or Datagram Transport Layer Security (DTLS) [25] to a target device B (with optional ACL), claiming authorization to access the device and perform particular operations.

As a result, each connections is secured by (at least three) layers of additional protection:

1. Device A can only be access other HomeCA controlled devices B when presenting a valid certificate.
2. With ACLs in place, the actual operations that A can perform on B are positively enumerated.
3. Device B will only accept a subset of protocols the HomeCA considers secure.
4. The gateway (which could be the HomeCA itself or a dedicated device, cf. Table 1) can translate between incompatible security protocols and provide additional content filters for insecure devices.

If gatewaying/proxying is not provided for by the particular application layer protocol for the device, it can be emulated by TLS Server Name Indication (SNI) as used in Domain Fronting.[26]

5 Secure Key Update

5.1 Trust

As we have seen, ownership change for IoT devices (Section 1) is common, entropy may be hard to come by (Section 4.1), and — as they can work with sensitive data but are hard to control (Section 3.1) — should receive minimal trust.

Table 2. IoT device ownership change: Key knowledge

Entity	Manufacturer key	Old device key	Key delta	New device key
Manufacturer	✓	—	—	—
Previous owner	—	?	—	—
IoT device	✓	✓	✓	✓
New owner	—	—	✓	✓

While open adversities by the manufacturer may be rare, negligence or process errors are not uncommon: In the early days of networking equipment, it started with batches of network adapters with matching hardware addresses. Today, it is identical or low-entropy encryption keys or the stealing of these keys in the facility[17]. So, these initial keys should be minimally trusted, but any entropy in them should be kept.

In the spirit of minimizing any data, as it can be stolen or abused, the HomeCA in turn should not know or be able to calculate the device's private key.

Even though the new private key should only be known to the device, but the HomeCA can be sure it includes any additional entropy from the HomeCA as part of the key update process.

The process below supports all these properties, due to the knowledge splitting shown in Table 2.

5.2 Key Update Protocol

Device d is created by manufacturer M. d is flashed with its software during the manufacturing process. At this time, M provides the following keys that are stored on d: A public key $\{M_{pub}\}$ for the communication with M, and a unique public key infrastructure (PKI) key pair $\{d_{pub}, d_{priv}\}$. M stores $\{d_{pub}\}$ only but must be expected to know the private key, too. HomeCA can verify for d, that update U is actually from M since the update is signed with the certificate of M. Access to d is only granted via HomeCA because device d only accepts access incoming messages encrypted with $\{d_{pub}\}$ which is known to HomeCA only. The protocol steps for the integration of d into the home network with HomeCA protocol is following:

Definition 1. *Device d with a possibly non-unique ECC (Elliptic Curve Cryptography) Key / key, that might also be known to a third party, i.e. Manufacturer M.*

Definition 2. *Manufacturer M that could know the ECC key of d.*

Definition 3. *HomeCA which is a home certification authority, e.g. running within a owners Smart Home system and is a trustworthy entity that shall not know the key.*

Theorem 1. *A new key pair k for device d, that includes the entropy from the previous key and additional entropy from an information exchange between the HomeCA and the device.*

Proof. – HomeCA continuously broadcasts its presence within the private network and requests new devices for authentication.

- d authenticates against HomeCA with its key pub_d .
- Auxiliary condition: Geographically restricted and within a small time window to reduce the attack vector.
- d and HomeCA create a common Diffie-Hellman Key DHk. The result is not predictable by both.
- Entropy check: Verification of the quality of the random number k.
- d creates a new key pub_{d2} and $priv_{d2}$, that are obtained by multiplication of pub_d and $priv_d$ with k.
- HomeCA signs the new key pub_{d2}, that HomeCA can obtain from $k * pub_d$ and provides the certificate to d.

Result:

- $priv_d$ is known to d and potentially also to M.
- k is known to d and HomeCA.
- $priv_{d2} = k * priv_d$ is only known to d
- At the same time, HomeCA knows, that d has integrated material.

6 Conclusions and Future Work

This paper motivates the challenges of the integration of IoT devices into a private network. We present our mostly automated security protocol HomeCA. It focuses mainly on two functions: First, secure integration, which relies on PKI cryptography that prevents attacks from the Internet. Second, scalability to a large number of IoT devices, by the automated integration process without user interaction based on defined protocol steps within the private network.

The steps described ensure that on first purchase and later ownership changes, the keys are updated securely, even when the device lacks a reliable entropy source.

The processes are designed to ensure long-term compatibility and security, even when the devices will not be provided with security updates.

We plan to implement the HomeCA protocol in a real world environment to analyze its viability without requiring changes on existing IoT devices. This prototype will also allow deployment to verify several parameters, including the validity periods and timeouts.

Acknowledgments

The authors would like to thank Adrian Spalka for discussions about the cryptographic properties of the key update algorithm.

References

[1] US-CERT, "Alert (TA16-288A): Heightened DDoS threat posed by Mirai and other botnets," https://www.us-cert.gov/ncas/alerts/TA16-288A, Nov. 2016. 1

[2] M. Waldvogel, "DDoS: What we can do to prevent it," https://netfuture.ch/2016/09/ddos-what-we-can-do-to-prevent-it/, Nov. 2016. 1

[3] T. Zillner, "ZigBee exploited: The good, the bad and the ugly," Cognosec, White paper, Aug. 2015. 1

[4] M. Vanhoef and F. Piessens, "Predicting, decrypting, and abusing WPA2/802.11 group keys," in *25th USENIX Security Symposium*. USENIX, 2016. 1, 2

[5] Wi-Fi Alliance, "Wi-Fi certified Wi-Fi Protected Setup," 2014. 1, 2, 4.5

[6] D. Bongard, "Offline bureforce attack on WiFi Protected Setup," in *HackLu*, 2014. 1, 4.1

[7] Bundesamt für Sicherheit in der Informationstechnologie (BSI), "Cyber-Angriffe durch IoT-Botnetze: BSI fordert Hersteller zu mehr Sicherheitsmaßnahmen auf," https://www.bsi.bund.de/DE/Presse/Pressemitteilungen/Presse2016/Cyber-Angriffe_durch_IoT-Botnetze_25102016.html, Oct. 2016. 1

[8] E. Harmon, "Don't hide DRM in a security update," https://www.eff.org/deeplinks/2016/09/dont-hide-drm-security-update, 2016. 1

[9] Atmel, "AWS zero touch secure provisioning platform," http://www.atmel.com/applications/iot/aws-zero-touch-secure-provisioning-platform/default.aspx, Application note, 2016. 1, 2

[10] H. Suo, J. Wan, C. Zou, and J. Liu, "Security in the internet of things: A review," in *2012 International Conference on Computer Science and Electronics Engineering (ICCSEE)*. IEEE, 2012. 2

[11] M. J. Gander and U. M. Maurer, "On the secret-key rate of binary random variables," in *International Symposium on Information Theory*. IEEE, 1994. 2

[12] P. Svenda, M. Nemec, P. Seka, R. Kvasnovsk, D. Formane, D. Komarek, and V. Matyas, "The million-key question—Investigating the origins of RSA public keys," in *25th USENIX Security Symposium*. USENIX, 2016. 2

[13] J. Liu, Y. Xiao, and C. L. P. Chen, "Authentication and access control in the internet of things," in *32nd International Conference on Distributed Computing Systems Workshops*, 2012. 2

[14] B. Klugah-Brown, J. B. Aristotle, K. Ansuura, and X. Qi, "A signcryption scheme from certificateless to identity-based environment for WSNs into IoT," *International Journal of Computer Applications (0975 – 8887)*, vol. 120, no. 9, 2015. 2

[15] CAPTCHA, "CAPTCHA: Telling humans and computers apart automatically," http://www.captcha.net/. 3.1

[16] R. R. Dube, *Hardware-based Computer Security Techniques to Defeat Hackers: From Biometrics to Quantum Cryptography*. Wiley, 2008. 4.1

[17] J. Scahill and J. Begley, "The great SIM heist," *The Intercept*, Feb. 2015. 4.1, 5.1

[18] IEEE Computer Society, "1149.1-2013 - ieee standard for test access port and boundary-scan architecture," in *Working Group: Boundary Scan Architecture - Standard Test Access and Boundary Scan Architecture WG P1149.1*, 2013. 4.1

[19] F. Stajano and R. Anderson, "The resurrecting duckling: Security issues for ad-hoc wireless networks," in *International Workshop on Security Protocols*, Apr. 1999. 4.3

[20] S. Cheshire and M. Krochmal, "DNS-Based Service Discovery," RFC 6763 (Proposed Standard), Internet Engineering Task Force, Feb. 2013. [Online]. Available: https://tools.ietf.org/html/rfc6763 4.4, 4.10

[21] J. J. Roberts, "Watch out that your rental car doesn't steal your phone data," Sep. 2016. [Online]. Available: https://fortune.com/2016/09/01/rental-cars-data-theft/ 4.6

[22] S. Cheshire and M. Krochmal, "Multicast DNS," RFC 6762 (Proposed Standard), Internet Engineering Task Force, Feb. 2013. [Online]. Available: https://tools.ietf.org/html/rfc6762 4.10

[23] D. Kaiser and M. Waldvogel, "Adding privacy to multicast DNS service discovery," in *Trust, Security and Privacy in Computing and Communications (TrustCom), 2014 IEEE 13th International Conference on.* IEEE, 2014, pp. 809–816. 4.10

[24] ——, "Efficient privacy preserving multicast DNS service discovery," in *High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC, CSS, ICESS), 2014 IEEE Intl Conf on.* IEEE, 2014, pp. 1229–1236. 4.10

[25] E. Rescorla and N. Modadugu, "Datagram Transport Layer Security Version 1.2," RFC 6347 (Proposed Standard), Internet Engineering Task Force, Jan. 2012, updated by RFCs 7507, 7905. [Online]. Available: https://tools.ietf.org/html/rfc6347 4.11

[26] D. Fifield, C. Lan, R. Hynes, P. Wegmann, and V. Paxson, "Blocking-resistant communication through domain fronting," in *Proceedings on Privacy Enhancing Technologies*, no. 2, Jun. 2015, pp. 46–64. 4.11