chownIoT: Enhancing IoT Privacy by Automated Handling of Ownership Change

Md Sakib Nizam Khan, Samuel Marchal, Sonja Buchegger, N. Asokan

chownIoT: Enhancing IoT Privacy by Automated Handling of Ownership Change

Md Sakib Nizam Khan[1(✉)], Samuel Marchal[2], Sonja Buchegger[1], and N. Asokan[2]

[1] KTH Royal Institute of Technology, Stockholm, Sweden
{msnkhan(✉), buc}@kth.se
[2] Aalto University, Espoo, Finland
{samuel.marchal@aalto.fi, asokan@acm.org}

Abstract. Considering the increasing deployment of smart home IoT devices, their ownership is likely to change during their life-cycle. IoT devices, especially those used in smart home environments, contain privacy-sensitive user data, and any ownership change of such devices can result in privacy leaks. The problem arises when users are either not aware of the need to reset/reformat the device to remove any personal data, or not trained in doing it correctly as it can be unclear what data is kept where. In addition, if the ownership change is due to theft or loss, then there is no opportunity to reset. Although there has been a lot of research on security and privacy of IoT and smart home devices, to the best of our knowledge, there is no prior work specifically on automatically securing ownership changes. We present a system called chownIoT for securely handling ownership change of IoT devices. chownIoT combines authentication (of both users and their smartphone), profile management, data protection by encryption, and automatic inference of ownership change. For the latter, we use a simple technique that leverages the context of a device. Finally, as a proof of concept, we develop a prototype that implements chownIoT inferring ownership change from changes in the WiFi SSID. The performance evaluation of the prototype shows that chownIoT has minimal overhead and is compatible with the dominant IoT boards on the market.

Keywords: Ownership · Privacy · Smart home · IoT.

1 Introduction

Internet of Things (IoT) devices produce and store sensitive information related to their sensing capabilities and contextual awareness. Similarly, they contain information related to configuration settings, credentials for network and user authentication, etc., all of which are privacy sensitive. Security has been one of the major concerns of the IoT paradigm due to a combination of factors, such as a potentially large number of networked devices, unprecedented use cases, resource constraints, and often sensors or other collections of data about user behavior, adding new privacy concerns.

Ownership here refers to the ability to control, manage, and access a particular device. The large growth in deployment of smart home IoT devices has introduced the possibility of device ownership change (due to selling, loss, theft, moving house, lending - handing over to use elsewhere or to a guest at the same location). This change can compromise data or access rights for both the previous and the new owner. For instance, a user forgets to log out from his smart TV box before selling it. In such a scenario, the available data or credentials can easily be misused by the buyer/new owner, ranging from browsing the history of what the previous owner watched to charging downloads to the associated credit card. Therefore, handling ownership change in a secure manner becomes necessary.

Contributions. The major contributions of the paper are the following:

- We present the problem of automatic handling of ownership change of IoT devices, with adversary model and requirements (Section 3).
- chownIoT, the first system capable of protecting owner privacy without user interaction during ownership change of IoT devices (Section 4). We implemented a prototype on Raspberry Pi and evaluated its performance (Section 5). chownIoT has the following features:
  - Automatic detection of ownership change using the context of IoT devices. For the prototype, this is based on the WiFi SSID.
  - A profile management system for authenticating owners.
  - Encryption of data and isolation of owner profiles for owner privacy.
  - A communication protocol between the IoT device and the smartphone device (i.e. used for controlling the IoT device) enabling its implementation.
- We discuss extensions to chownIoT in two directions, more sophisticated context 6 and vendor independence 7.

2 Related Work

Recently, several research works have been done to secure and ensure smooth operation of IoT devices. Our proposed solution is closely related to smart-home device privacy, authentication and access control of IoT devices (especially for ownership change), and context-aware security. We thus divide the related works into these three major categories.

Smart Home Device Privacy. Recently, studies are focusing on mitigating the privacy issues of smart-home devices. Apthorpe et.al. [2], examined different smart home IoT devices and found that even with encrypted traffic, the network-traffic rates of the devices can reveal potentially sensitive user interactions. In another work [1], the same authors proposed mechanisms for preventing network observers from inferring consumers' private in-home behaviors. While the general concern of privacy matches ours, the main difference to our proposed work is that these works only focus on privacy issues of smart-home devices related to passive network observation. The privacy issues related to *ownership* of smart-home IoT devices, the main focus of our work, are not addressed.

Authentication and Access Control of IoT Devices, Ownership Change. Due to their often limited computational capabilities, IoT devices require lightweight yet secure authentication and access control mechanisms. Several research works talked about authentication requirements during ownership change of IoT devices. Tam et.al [21] and Bohn [3] proposed ownership transfer mechanisms for smart devices in their individual works for securely transferring ownership. Similarly, Pradeep et al. [17] also proposed a concept of ownership-authentication transfer for securely handling the ownership transfer of a device to a new owner. The main difference to our proposed system is that in ownership-authentication transfer the seller has to initiate the transfer process, there is no notion of detecting ownership change *automatically*. In addition, the protocol requires a central key server for key management which may not be feasible in the case of smart homes. The protocol does not also mention any data protection mechanism.
Context Aware Security. In recent years, several security solutions [7,24] have started using context to provide better security. Several works have integrated context for providing better automatic access-control techniques [8,9,15,18,25]. Besides access control, Miettinen et al. [14] proposed a new approach for secure zero-interaction pairing intended for IoT and wearable devices, which uses context to identify the pairing devices. Apart from proposing new context-aware security solutions, some studies also focused on finding vulnerabilities in already existing solutions [20]. All of these works leverage context either to take access control, pairing or key agreement decisions to improve security. In our work, we leverage context of a smart-home device for a new purpose: *detecting ownership change*.

3 Models and Requirements

3.1 System and Adversary Model

In our system model, IoT devices are connected to the owner's account with a cloud service through a network connection mediated by an access point. The owner also has a control device (typically her smartphone) to interact with the IoT device directly. Ownership change in our context refers to the IoT device only.

Ownership change of a device involves two parties, the previous and the new owner, that need to be protected from each other, as they are both potential adversaries and targets. We model the adversary as malicious, i.e., assume that they can mount active attacks on security and privacy, with standard assumptions on computational power. We conservatively assume that access to one asset (cloud account, IoT device), unless specifically prevented, implies access to the other. The adversary is interested in access to the other party's data, including credentials and metadata. We now list our specific attacker types and capabilities.

Previous Owner Adversary (POA): access to (and credentials for) the cloud account and the control device associated with the IoT device.

New Owner Adversary (NOA): access to the IoT device.
Advanced New Owner Adversary (ANOA): NOA plus special equipment and dedication to read out from IoT device storage while the device is turned off as well as ability to spoof the AP the device was associated with. To spoof an AP with the correct SSID and MAC, the attacker needs to 1. know these, 2. know the protocol and other authentication parameters used, and 3. participate in the protocol while accepting any credential.

3.2 Requirements

Smart home devices have some special characteristics, such as limited resources, different sensor modalities and application dependencies, which differentiate them from traditional devices. Based on these characteristics, the requirements of an intended solution are:

1. Resource Constraints: Ability to work on resource constrained devices in terms of computation (and thus energy), network, storage, and memory.
2. Security Goal: equivalent to timely reset to factory defaults of the IoT device upon ownership change in terms of confidentiality (privacy), integrity, and availability. Specifically, protect the data on the cloud, the control device, and the IoT device from the respective other owner.
3. Deployability: Adaptable to the largest possible class of devices.
4. Usability: The added functionality may not be outweighed by any burden put on the user. That means minimal user involvement and waiting time as well as minimal consequences for any wrong decisions made automatically or exceptions such as loss of the control device.

4 chownIoT

IoT covers a wide range of heterogeneous devices and diverse scenarios. We therefore first present the algorithmic view of our solution that can be adapted to different environments and protocols. We then explain one concrete instantiation and reasoning for design choices.

4.1 Algorithmic Solution Overview

To protect privacy-sensitive data against adversaries (previous or new owner), chownIoT 1. automatically detects change of ownership, 2. manages owners by maintaining individual profiles for each owner, and 3. verifies the ownership change and protects data based on owner authentication.

The smart home device maintains a profile for each owner, which enables the isolation of one user profile from another. Each *profile* contains owner authentication credentials, cloud credentials, known contexts and user data that is specific to the owner. All data of a particular owner is managed under a profile. The main goal of chownIoT is to protect this data on the device (and any associated data stored in the cloud).

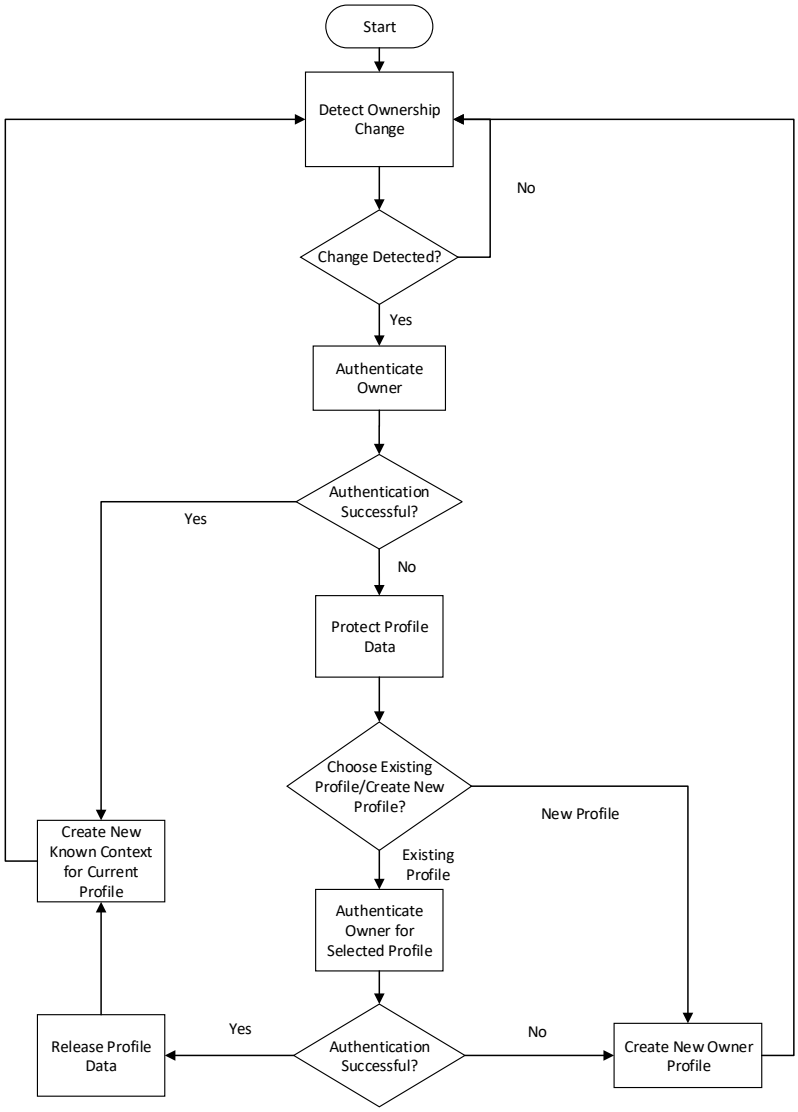

Fig. 1. Flow Diagram of chownIoT

Figure 1 illustrates the flow diagram of chownIoT. The process starts with inferring ownership change based on the change in context of the device. Most smart home devices are either static or semi-static in terms of their mobility. The static devices never move once deployed except if sold (e.g. smart AC, smart fridge) whereas the semi-static devices move rarely after deployment (e.g. baby monitoring camera, smart TV) within a fixed boundary. The deployment context for such devices usually never changes while they are under the same owner.

However, when their ownership changes, the deployment context also changes. Therefore, by identifying the change in the deployment context of a device we can infer ownership change. Thus, if chownIoT detects a context change, it tries to authenticate the owner based on her control device (smartphone) or her own credentials. If the authentication is successful, chownIoT infers that only the context of the device has changed (e.g., the owner has moved the device from the home to the summer cottage) but not the ownership. Thus it creates a new known context for the same owner in the same profile.

In contrast, if the authentication is not successful, chownIoT concludes that the ownership has changed, hence it protects the profile data, using encryption. In chownIoT, only one profile remains active at a time and all others are protected. Once the profile data is protected, the owner can either retrieve an existing profile or create a new profile. If the owner chooses to retrieve an existing profile, chownIoT authenticates the owner for the selected profile. After successful authentication, it releases the profile data and stores the context as a new known context for the selected profile. However, if the authentication is not successful or the user chooses to create a new profile, chownIoT creates a new profile for the owner. With the new profile, the owner gets full control of the device except access to the data of other profiles which remain protected.

This algorithm is executed once the device has been deployed. The life-cycle of an IoT device, however, begins with configuring the device into the deployment network. Currently, most available smart home IoT devices require a smartphone and a vendor provided smartphone application for the initial configuration as well as for later management/control [16]. chownIoT requires some additional steps besides traditional configuration.

chownIoT builds the security mechanism by trusting the control device used during initial configuration of a smart home device. *Control device* here refers to the smartphone used to configure, control and manage the smart home device. During the initial configuration it establishes a security association (rendering the control device a *trusted device*) as well as an owner authentication mechanism for future verification of the owner in case of ownership change. The interactions between the smart home device and the trusted device during initial configuration are depicted in Figure 2.

Apart from the security association, the smart home device also establishes an owner authentication mechanism. With the security association, the authentication mechanism is bound to a particular control device. But realistically, the owner should be able to authenticate with any device.

4.2 Prototype Design and Implementation Choices

To realize chownIoT, we made some design and implementation choices. In this section we discuss and reason about our choices.

Initial Configuration. First, the control device configures the smart home device (1) in Figure 2. The configuration steps includes device specific configuration and some additional steps for chownIoT, namely, turning on the discoverable mode of Bluetooth and creating a server socket which listens for packets.

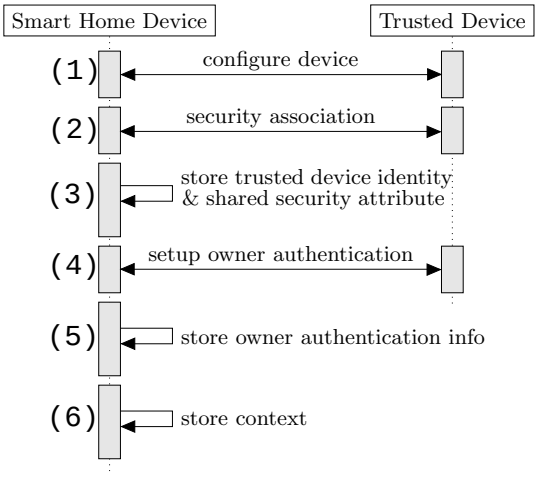

Fig. 2. Sequence Diagram of chownIoT During Initial Configuration

To facilitate the communication between the smart home device and the control device/trusted device we define a simple protocol based on User Datagram Protocol (UDP) (for details see [10]). The *configure device* feature is implemented using Bluetooth pairing. The control device discovers the smart home device and sends a pairing request. The smart home device responds to the pairing request and once paired, the control device performs the necessary configuration.

For the next step, establishing a security association (2) that serves to check whether the context change likely implies an ownership change, there are different techniques available. For instance, the involved parties can have each other's public key and their own private key. Key agreement is another alternative where the involved parties establish a shared secret key between them. The public/private-key mechanism requires larger key sizes than symmetric key to achieve similar security [12]. In addition, they also require more computational processing power [11]. As most smart home IoT devices are resource constrained, we establish a shared secret between the smart home device and the trusted device using Diffie-Hellman key exchange protocol [19]. After establishing a shared secret, chownIoT stores the identity (3) of the trusted device for future verification of ownership change. The identity includes the Bluetooth device name and MAC address of the trusted device and the established shared secret as depicted in Table 1. The trusted device identity is stored in persistent storage.

Trusted Device Identity	Bluetooth Device Name	Bluetooth MAC Address	Shared Secret
Known Context	AP SSID	AP MAC Address	AP Access Credential
Owner Profile	Known Context	Trusted Device Identity	Profile Name

Table 1. Elements Stored During Initial Configuration

In addition to the security association with the trusted device, chownIoT implements an authentication mechanism (4) for the user independent of the control device she is using. There are several candidates for authentication mechanisms, such as password-based authentication, public key based authentication protocol, Authentication and Key Agreement protocol (AKA), and Extensible Authentication Protocol (EAP). Password-based authentication mechanisms are widely used as they are convenient to use and implement [26]. Moreover, using password-based authentication, an owner can be authenticated on any control device very easily. This is due to the fact that by using password-based authentication, we do not need to store any key on the trusted device. Thus, we chose to implement a password-based authentication mechanism. The owner is prompted to choose a profile name and a password during the initial configuration. The smart home device receives the hash of the owner password and a profile name from the trusted device which it stores (5) in the persistent storage. In addition, it stores the triplet <SSID, MAC address of the Access Point (AP), access credential of the AP> as known context (6) in the persistent storage. A *known context* here refers to a context that has been already observed and approved by a particular owner for a particular device. Finally, chownIoT generates a profile which is identified by the profile name provided by the user, and the known contexts and trusted device identity are stored under the profile. The stored elements for both context and owner profile are as depicted in Table 1.

Handling Ownership Change. After the initial configuration, the smart home device starts detecting possible ownership change based on change in context. We chose the Wi-Fi SSID as a simple indicator of potential ownership change. IoT devices mostly achieve Internet connectivity through a Wi-Fi connection with an AP [16]. Devices know the SSID of the wireless network that they are connected to. In a typical smart home scenario, the SSID is the same for the whole house or apartment. Thus, for static and semi-static devices, the connected SSID is unlikely to change while a particular device has the same owner. However, a new owner needs to connect the IoT device to a different network, which also changes the SSID of the particular device.

In chownIoT, the context of a smart home IoT device is linked to the SSID it is currently connected to. The smart home device continuously monitors the SSID of the AP that it is currently connected to and compares it with the stored known context list. If the current SSID is not in the list, it infers a possible ownership change. Once it detects a possible ownership change, at first it triggers a Bluetooth discovery looking for the trusted device based on the identity it stored during initial configuration. If the trusted device is discovered, the smart home device tries to authenticate it using the shared secret established during the initial configuration. The authentication is performed using a challenge-response authentication mechanism. The details of the challenge-response mechanism can be found in [10]. If the trusted device is not available, then password-based authentication is triggered. Although SSID has clear limitations, as it can easily be forged, it has the advantage of being present at every smart home, is independent of which or how many IoT devices there are, and is easy to detect.

However, depending on the setup, other context information can be used and easily be integrated into chownIoT.

Profile Management and Data Protection. During ownership change when the authentication process fails, chownIoT protects the profile data by means of encryption. For data encryption, we use AES-CCM [13] authenticated encryption technique. It is a technique that provides both authentication and encryption at the same time. We use a key derived from the owner-provided password as the encryption key for AES-CCM, with a key length of 128 bits as recommended by NIST [6]. We do not store the encryption key on the device as physical memory access can leak the encryption key. In addition, we also require a profile retrieval mechanism using any control device. Thus, we cannot store the key on the trusted devices only. To fulfill these requirements, we chose to derive the encryption key from the owner password. The main benefit of password-based key derivation is that it can be instantly derived from the owner-provided password without requiring to be stored and it also facilitates profile retrieval using any control device. Password-Based Key Derivation Function 2 (PBKDF2) with 256 bits random salt and 4096 iterations of SHA256 hash algorithm is used to derive the key from the password. The key derivation function is given in Equation (1).

$$Key = H_{\mathrm{SHA256}}(Password_{\mathrm{SHA256}}, Salt_{256\mathrm{bit}}, 4096_{\mathrm{iterations}}) \tag{1}$$

As the smart home device receives the hash of the owner password from the trusted device, it derives the key according to Equation (1). The salt for the key derivation is randomly generated. Once the key derivation is completed, the smart home device only stores the salt and the key in persistent storage, and deletes the password hash. During an ownership change, the smart home device encrypts the profile data except for the salt and the profile name. Once the encryption process is completed, it also deletes the derived key. There are two cases when the profile needs to be encrypted: either the inferred ownership change was a false positive, or it was only temporary. When the owner wants to retrieve the encrypted profile, she is prompted to provide the owner password. Upon receiving the password hash, the smart home device again derives the key using the provided password hash and stored salt value. Once the key is derived, it performs authenticated decryption using AES-CCM and the derived key. If the process is successful, the owner is authenticated and the profile data gets decrypted. chownIoT limits the (configurable) number of failed user authentication attempts, after which the whole profile gets deleted.

5 Evaluation and Discussion

For evaluating chownIoT, we implemented the smart home device features on a Raspberry Pi 3 using C++ and control device features on the Android platform. We evaluate chownIoT based on the requirements identified in Section 3.2.

5.1 Resource Constraints

For **CPU usage**, we measure the performance of the major resource-intensive operations in chownIoT, namely encryption/decryption[3], key derivation, and hashing.

Data Size	CPU Usage (seconds)
10KB	0.010
10MB	5.89
100MB	60.67

Table 2. Encryption CPU Usage

The data to be encrypted includes chownIoT protocol data and user data produced by the specific device. Table 2 lists the CPU usage measured in seconds for different data sizes. 10KB and 10MB data requires only 0.010 second and 5.89 seconds, respectively, for encryption which is not a large computational overhead. By experimenting with a range of IoT devices (e.g. weather stations, smart switches, and different kinds of sensors), we found that most of them produce data between 10KB and 100MB. Thus, encrypting data on such devices is quite feasible. Apart from this, we can also see that 100MB data requires 60.67 seconds. Devices such as surveillance cameras produce this amount of data and such devices do not need to be always active. Thus, spending 60 CPU seconds for encrypting data seems feasible for such devices. For devices that produce larger amounts of data, the corresponding encryption time can be very long. While by default chownIoT encrypts the entirety of the data, depending on the device type not all data are necessarily privacy sensitive. For such cases, it may make sense to allow for device-specific adaptations of the system to identify and protect only more privacy-sensitive data. This opens the possibility of different choices in the trade-off between privacy and performance/usability.

The time required for key derivation and hashing are 0.060 and 0.010 seconds, respectively.

In the current implementation of chownIoT, the smart-home device continuously loops to detect change of SSID for detecting ownership change. This is also resource intensive, as the monitoring process needs to run all time. This can be improved by implementing call backs for when there is any disconnection or state change of the Wi-Fi connection.

Table 3 lists the CPU specifications of three of the most popular IoT boards. Arduino Tian and Intel Edison have less powerful CPUs than Raspberry Pi 3. The resource-hungry operations may need twice the CPU time of Raspberry Pi 3 to execute on these boards. Thus, operations such as encryption or decryption of large amounts data can degrade the usability of the system on these boards.

[3] We only measure encryption, as it yields more conservative results than decryption [4].

Board	CPU	RAM
Raspberry Pi 3	ARM Cortex-A53, 1.2GHz	1GB
Arduino Tian	Atheros AR9342 560MHz	64MB
Intel Edison	Dual-Core Intel Atom 500MHz	1GB

Table 3. Specifications of IoT Boards

We measured the **RAM usage** of each resource-intensive operation, see Table 4. It is constant (approximately 1000KB or 1MB) regardless of data size and operation performed and only uses a small fraction of the RAM available on the boards in Table 3.

Operation	RAM Usage (KB)
Encryption 10KB	1031
Encryption 10MB	1051
Encryption 100MB	1094
Key Derivation	1047
Hashing	1045

Table 4. Memory Usage for Different Operations

The **network overhead** is negligible as all operations, except the initial configuration and authentication, are performed only on the smart-home device itself and do not involve any network communication. In terms of **storage**, chownIoT does not add much overhead either, as encryption using CCM adds minimal message expansion [23].

chownIoT fulfills the requirements from Section 3.2 in terms of computation, network, memory, and storage even for quite resource-constrained devices.

5.2 Security and Privacy

To fulfill the requirements of protecting the cloud account privacy and device data privacy as identified in Section 3.2, chownIoT isolates owner profiles from one another. This isolation is done by encryption with carefully chosen parameters (see Section 4.2) as soon as the absence of the trusted device is detected in case of a suspected ownership change.

The potential entry points for an adversary (NOA, POA, ANOA, as defined in 3.1) to break this isolation are 1. the context used to infer change of ownership (NOA), 2. authentication (both user and device) (NOA, POA), and 3. physical access to the active, unencrypted profile (ANOA).

From an implementation point of view, the known context can be spoofed by replicating the AP SSID and MAC address. They are stored along with access credentials for the AP in the known context information, which is encrypted.

Although chownIoT deletes both the owner password and the derived encryption key on the device once the data gets encrypted, the security of user

authentication is limited to that of password authentication in general, meaning vulnerability to guessing and brute-force attacks. Online guesses for the device are limited and thus protect against a NOA that does not know the password.

Offline attacks are, however, possible for the ANOA, meaning the attacker has physical access to the device and the right equipment to read out the ciphertext and salt.

Regarding device authentication, the challenge-response based authentication is in theory vulnerable to relay attacks [5]. In chownIoT, however, the proximity requirements enforced by Bluetooth communication during device authentication (for inference of ownership change or profile reactivation) makes relay attacks unrealistic and ineffective[4].

In case of a device getting stolen and/or powered off before detecting ownership change, the data of the active profile remains unencrypted and can be read by ANOA. Even if the data were encrypted at all times, the ANOA can do offline password cracking.

In summary, chownIoT withstands the NOA and POA, but in the time window between the device changing hands and being powered on, it is vulnerable to the ANOA, a determined attacker that can spoof the access point, perform an offline brute-force attack on the password by reading from the storage of a powered-off device if the current profile has not yet been encrypted.

5.3 Deployability

chownIoT does not depend on any particular operating system or hardware. The implementation of chownIoT only depends on Wi-Fi communications for ownership-change detection and does not involve any other sensors. The ownership change detection technique can also be adapted for other communication technologies, such as Bluetooth and ZigBee, for instance by analyzing/monitoring the available nearby devices of a smart home device. Thus, it is possible to implement chownIoT on any device with a communication interface. There can, however, be extremely resource-constrained devices that cannot run chownIoT due to the requirements discussed in Section 5.1. Some devices do not store user data deemed sensitive (or any user data at all) and thus do not need chownIoT. We hypothesize that the overlap between these two types of devices is large (e.g. smart light bulbs).

Vendor Dependency The current solution of chownIoT requires vendor cooperation for deploying it on existing and also upcoming IoT devices on the market. According to the present implementation, to deploy chownIoT, a vendor needs to include the smart-home device part of the solution in the firmware of the intended device. In addition, the vendor also needs to include the control device part of the solution in the vendor provided control application. For existing devices, deploying chownIoT would require a device firmware as well as an

[4] Nevertheless, they can easily be mitigated by implementing user consent/notification during the authentication process, at the cost of reduced usability

application update. However, vendor dependency is a major limitation of the current system due to the lack of universally adopted standards, and a workaround to overcome this dependency is needed. Our proposal for reducing/eliminating vendor dependency and the ensuing trade-offs are discussed in Section 7.

5.4 Usability

In chownIoT, besides the regular configuration, we additionally setup an owner-authentication mechanism and a shared secret with the trusted device. While the generation of the shared secret for the security association with the control device is automatic, the owner authentication mechanism requires user participation. In our prototype implementation that means the user sets up a profile name and password. The ownership-change detection is performed automatically, requiring no user interaction. Loss of the trusted control device requires user authentication and a new security association, for the user that only means entering the password again and enabling Bluetooth.

One potential limitation of chownIoT in terms of usability is the possibility of false positives, i.e., detection of ownership change when none occurred. This happens if the change of context and both the device and user authentication fail and yet the inference of ownership change is invalid. While this should be rare, since the change of context with continued ownership most likely involve the owner and/or her control device, it entails user involvement to fix. Once chownIoT assumes an ownership changes, the profile gets encrypted. To retrieve the profile, the owner needs to authenticate herself, in our implementation that means selecting the profile she wants to access and supplying her password. She then has to wait until chownIoT has decrypted her profile data; the time required depends on the size of the profile data and the processing power of the device.

The reverse problem, false negatives, can happen when the ownership and context changes, but the previous owner and her trusted control device are nearby and thus authentication succeeds (or due to a successful relay attack). While this could be remedied by user notifications and consent for each automatic authentication based on the security association, we opted to not include that and err on the side of usability.

6 Advanced Ownership Change Detection

chownIoT uses SSID as a simple indicator of change in the context of an IoT device. Extending chownIoT to overcome the limitations of SSID, Artur Valiev, in his master's thesis [10] proposed a more robust and forge-proof system called *FoundIoT* that uses richer context. The main idea behind FoundIoT is that if a device stays in the same context, it will observe other devices in its vicinity over time. However, if the device goes to a new context, the devices in the vicinity will also change. Thus in FoundIoT, the context change is inferred by monitoring nearby devices over wireless communication channels. Once the data is captured, FoundIoT performs statistical analysis on the data to detect a change in context,

in multiple stages. First, change is detected based on Wireless Stations (STA) which are the other IoT devices in the device vicinity, then APs in vicinity of the device, and finally, Bluetooth-enabled devices. It uses the Jaccard Index and Kullback-Leibler (KL) divergence for finding similarity metrics changes over consecutive scans. If the similarity is low, FoundIoT concludes that there is an ownership change. The author shows that by using such techniques, it is possible to detect ownership change with high accuracy and low false alarms in their system model.

7 iChownIoT

Currently, chownIoT requires vendor cooperation for deployment. One way to eliminate this dependency is to move ownership-change detection to an independent device in the smart-home environment. This device needs to infer ownership change of *other* devices and thus needs context information *about* these other devices instead of *for* the devices themselves, as is the case in chownIoT. This new system, independent chownIoT or iChownIoT, adapts FoundIoT to infer ownership change from the perspective of other devices instead of its own as originally designed.

During the FoundIoT-inspired monitoring process, if iChownIoT notices that a device is missing from the expected devices in the environment, it can notify the user on their smartphone. If the user agrees that indeed there was a change of ownership, then they can take the necessary action to secure their personal data. In such a scenario, even false detection of ownership change will be helpful for the user as a notification of a missing device due to some other technical failures.

In the space between complete vendor independence (iChownIoT) and vendors implementing chownIoT, vendors of IoT devices can choose to use these notifications from the independent monitoring device as a service. For instance, the vendors can open up some web APIs using their cloud services to receive information about changes in the context of their particular device from iChownIoT and take necessary measures to secure the user's personal data. However, in terms of robustness, the basic chownIoT on IoT device is still a better solution as it is able to apply a protection mechanism immediately on the device in case of ownership change rather than depending on a third party such as user or vendor for that.

8 Conclusions

In this work, we present an automatic context-based technique to improve the privacy of smart-home IoT devices during ownership change. While there exists related work for several different aspects of our solution, we are aiming to bridge the research gap for the specific problem. In our evaluation, we found that we can protect owners from each other, unless they have special equipment to read from the IoT device while it is switched off, at low cost of overhead and resource

requirements suitable for most IoT setups. Even if the detected context change is a false positive, the current owner need not be inconvenienced. There are, however, some limitations of chownIoT in its pure form: our solution hinges on the adoption by IoT device manufacturers or service providers, which may be an unrealistic assumption, and a more sophisticated context may improve accuracy. We present a vendor-independent version, iChownIoT. Though limited by the lack of vendor cooperation, such a system can at least alert the user that an ownership change was detected and action is needed. iChownIoT adapts FoundIoT [22], which builds on the first version of chownIoT [10] to make use of richer and forge-proof context.

Acknowledgements. This work is supported by the Academy of Finland under the WiFiUS program (grant 309994), the Wallenberg AI, Autonomous Systems and Software Program (WASP), and the Swedish Foundation for Strategic Research (grant SSF FFL09-0086).

References

1. Apthorpe, N., Reisman, D., Feamster, N.: Closing the blinds: Four strategies for protecting smart home privacy from network observers. arXiv preprint arXiv:1705.06809 (2017)
2. Apthorpe, N., Reisman, D., Feamster, N.: A smart home is no castle: Privacy vulnerabilities of encrypted IoT traffic. arXiv preprint arXiv:1705.06805 (2017)
3. Bohn, J.: Instant personalization and temporary ownership of handheld devices. In: Mobile Computing Systems and Applications, 2004. WMCSA 2004. Sixth IEEE Workshop on, pp. 134–143. IEEE (2004)
4. Ertaul, L., Mudan, A., Sarfaraz, N.: Performance comparison of AES-CCM and AES-GCM authenticated encryption modes. In: Proceedings of the International Conference on Security and Management (SAM), p. 331. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp) (2016)
5. Francis, L., Hancke, G., Mayes, K., Markantonakis, K.: Practical NFC peer-to-peer relay attack using mobile phones. In: International Workshop on Radio Frequency Identification: Security and Privacy Issues, pp. 35–49. Springer (2010)
6. Giry, D.: Keylength - NIST report on cryptographic key length and cryptoperiod (2016). https://www.keylength.com/en/4/ (2017). (Accessed on 05/26/2017)
7. Hu, J., Weaver, A.C.: A dynamic, context-aware security infrastructure for distributed healthcare applications. In: Proceedings of the first workshop on pervasive privacy security, privacy, and trust, pp. 1–8. Citeseer (2004)
8. Jih, W.R., Cheng, S.y., Hsu, J.Y., Tsai, T.M., et al.: Context-aware access control in pervasive healthcare. Computer Science and Information Engineering, National Taiwan University, Taiwan. (2005)
9. Kapsalis, V., Hadellis, L., Karelis, D., Koubias, S.: A dynamic context-aware access control architecture for e-services. computers & security **25**(7), 507–521 (2006)
10. Khan, M.: Enhancing privacy in IoT devices through automated handling of ownership change. Master's thesis, School of Science, Aalto University, Finland (2017-08-28). URL http://urn.fi/URN:NBN:fi:aalto-201709046805

11. Kumar, Y., Munjal, R., Sharma, H.: Comparison of symmetric and asymmetric cryptography with existing vulnerabilities and countermeasures. International Journal of Computer Science and Management Studies **11**(03) (2011)
12. Lenstra, A.K., Verheul, E.R.: Selecting cryptographic key sizes. Journal of cryptology **14**(4), 255–293 (2001)
13. McGrew, D., Bailey, D.: AES-CCM cipher suites for Transport Layer Security (TLS). Tech. rep. (2012)
14. Miettinen, M., Asokan, N., Nguyen, T.D., Sadeghi, A.R., Sobhani, M.: Context-based zero-interaction pairing and key evolution for advanced personal devices. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pp. 880–891. ACM (2014)
15. Miettinen, M., Heuser, S., Kronz, W., Sadeghi, A.R., Asokan, N.: ConXsense: automated context classification for context-aware access control. In: Proceedings of the 9th ACM symposium on information, computer and communications security, pp. 293–304. ACM (2014)
16. Miettinen, M., Marchal, S., Hafeez, I., Asokan, N., Sadeghi, A.R., Tarkoma, S.: IoT Sentinel: Automated Device-Type Identification for Security Enforcement in IoT. arXiv preprint arXiv:1611.04880 (2016)
17. Pradeep, B., Singh, S.: Ownership authentication transfer protocol for ubiquitous computing devices. arXiv preprint arXiv:1208.1712 (2012)
18. Ren, B., Liu, C., Cheng, B., Hong, S., Zhao, S., Chen, J.: Easyguard: enhanced context-aware adaptive access control system for android platform: poster. In: Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking, pp. 458–459. ACM (2016)
19. Rescorla, E.: RFC 2631 - Diffie-Hellman key agreement method. https://tools.ietf.org/html/rfc2631 (1999). (Accessed on 05/04/2017)
20. Shrestha, B., Saxena, N., Truong, H.T.T., Asokan, N.: Contextual proximity detection in the face of context-manipulating adversaries. arXiv preprint arXiv:1511.00905 (2015)
21. Tam, P., Newmarch, J.: Protocol for ownership of physical objects in ubiquitous computing environments. In: IADIS international conference E-Society, vol. 2004, pp. 614–621 (2004)
22. Valiev, A.: Automatic ownership change detection for IoT devices. G2 pro gradu, diplomity (2018-08-20). URL http://urn.fi/URN:NBN:fi:aalto-201809034781
23. Whiting, D., Housley, R., Ferguson, N.: Counter with cbc-mac (ccm). Tech. rep. (2003)
24. Wullems, C., Looi, M., Clark, A.: Towards context-aware security: An authorization architecture for intranet environments. In: Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on, pp. 132–137. IEEE (2004)
25. Zhang, G., Parashar, M.: Context-aware dynamic access control for pervasive applications. In: Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference, pp. 21–30 (2004)
26. Zhang, L., McDowell, W.C.: Am i really at risk? determinants of online users' intentions to use strong passwords. Journal of Internet Commerce **8**(3-4), 180–197 (2009)