ON SECURITY BEST PRACTICES, SYSTEMATIC ANALYSIS OF SECURITY ADVICE, AND INTERNET OF THINGS DEVICES

by

Christopher Bellman

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

Carleton University
Ottawa, Ontario

August, 2022

Table of Contents

List of Tables

List of Figures

Abstract

While Internet of Things (IoT) security best practices have recently attracted considerable attention from industry and governments, academic research has highlighted the failure of many IoT product manufacturers to follow accepted practices. We begin by investigating a surprising lack of consensus, and void in the literature, on what (generically) *best practice* means, and provide a technical examination of related terminology. We use iterative inducting coding to design an analysis methodology for categorizing security advice and measuring its *actionability*. We use this methodology to analyze three datasets: a set of 1013 IoT security best practices, recommendations, and guidelines, and two formally recommended IoT security advice documents. We find all three sets to be largely non-actionable. Through design and use of this methodology, we identify the characteristics of actionable security advice. We also analyze recent work on IoT device identification based on three identification objectives (distinguish device instances, distinguish device classes, and authenticate device identity), and the technical approaches by which they are reached: device *fingerprinting*, *classification*, and *authentication*. We differentiate the role of these objectives and approaches in IoT security, and develop a model relating them.

Acknowledgements

First and foremost, I would like to thank my supervisor Dr. Paul C. Van Oorschot for his invaluable guidance and support (both academic and financial) throughout my time at Carleton as a PhD student. I attribute the significant improvements (typically after amazing detailed feedback on my many drafts) of our research papers and this thesis to Paul's incredible attention to detail, deep knowledge of computer security, and his ability to convey his thoughts to his students.

I offer a personal anecdote highlighting an important formative moment in my PhD timeline, for which I am grateful. After the initial few years of my PhD studies, I was struggling to find a research direction that was unique and worthy of in-depth study, but also an appropriate fit for my background. One day during one of our weekly meetings, seeing that I was struggling, Paul explained his idea for a general research direction based on a line from a paper that we had both read, but I had overlooked at the time. I found the research direction—IoT security advice—interesting, unique, and it seemed appropriate for my background. I am ever grateful for Paul's suggestion, not only as it eventually led to the unique line of research culminating in this thesis, but that he first allowed me to struggle on my own (with guidance, of course) before intervening with a strong nudge in the right direction. I believe this, and all the other lessons he has taught me, has made me a better researcher.

Additionally, I would like to thank the members of the Carleton Security Research Labs (CCSL and CISL) for their support and guidance over the years. In particular, I'd like to thank my close colleague Hemant Gupta, who has been both supportive and helpful since I first joined the research lab; and Dr. David Barrera for assisting with portions of the research in this thesis, and general guidance in navigating the challenges of a PhD program.

I would like to thank the committee members for reading and providing valuable feedback for this thesis. Members (listed alphabetically) include: Dr. AbdelRahman Abdou (Carleton University), Dr. Carlisle Adams (University of Ottawa), Dr.

Ashraf Matrawy (Carleton University), and Dr. Mohammad Zulkernine (Queen's University).

Finally, a special thank-you goes to my family and friends for all the support over the past five years—your encouragement is deeply appreciated, and I could not have done this without your support.

Christopher Bellman
August, 2022.

Chapter 1

Introduction

The Internet of Things (IoT) is commonly described as adding computation and network communication capabilities to traditionally non-networked items or "things" [219]. It surrounds us with a variety of network-connected devices such as smart light bulbs, door locks, web cameras, audio speakers, thermostats, and objects not traditionally associated with Internet communication like fridges, traffic lights, or sensors and controllers built into critical infrastructure systems. These devices, while offering convenience or new functionality, have acquired a reputation [14] of poor security and misconfiguration, leading to huge numbers of network-accessible devices being vulnerable to a variety of attacks and exposed to new threats (e.g., [21, 177, 192]). As IoT devices may be more isolated or resource-constrained (e.g., battery power, processors, memory) than their Internet of Computers (IoC)[1] counterparts, or lacking in software update support, their security issues are often hard to address. Their numbers are expected to exceed 50-billion by 2025 [108]. Given the scale of the Mirai botnet attack [125], the widespread realization of potential IoT-related damage has made researchers aware of the threats that IoT devices pose.

1.1 Thesis Scope

Unlike IoC, which can only indirectly affect the physical world, IoT has implications for not only computer security, but also safety. One of the over-arching questions underlying our research is: *How is IoT security different from IoC security?* We describe how this scoping question motivates our research in the next section. Our work herein has primary focus on consumer IoT devices (in particular, smart home or wearable devices; discussed further in Chapter 2). These are commonly recognized as being poorly secured, or having little investment in security [14, 21, 138]. Our

[1] Pre-IoT devices such as mobile phones, laptop/desktop computers, and servers [191].

scope does not include internet-connected or autonomous vehicles. Since consumer IoT devices are heavily marketed and sold to everyday end-users, users rely on these devices within their daily lives, and most users do not have the knowledge or ability to mitigate known vulnerabilities, the user population is heavily reliant on a strong security implementation from manufacturers. In contrast, other IoT segments such as critical infrastructure, industrial IoT, or smart cities, may reasonably be expected to have individuals dedicated to the ongoing maintenance and protection of such devices. For completeness, in Chapter 2 we offer an overview of the consumer IoT landscape, including how IoT differs from IoC and the new challenges introduced.

We scope our work to two domains. In the first, we focus our research questions on IoT security advice for consumer-grade devices, including security *best practices*, desired security *outcomes*, and security *principles* (among other related terms; we will give definitions for these terms in Chapter 3) as they relate to the development of secure IoT devices and their communication. We focus on the advice itself as an approach for improving IoT device security; investigating conformance with advice or advice recipient motivation to follow advice is not within this thesis' scope. When we generally mention *security advice*, we mean guidance for how to secure a computer-related component (e.g., reduce vulnerabilities or prevent attacks of an IoT device, a computer, a network) as proposed by someone typically of authority (discussed further in Chapter 3). This includes security of hardware supporting devices, but primarily the software run by devices (e.g., OSes, software packages, APIs, communication protocols). This further includes security advice for how devices communicate with network services (e.g., in local networks such as a hub device, or cloud services), but excludes focus of securing cloud services themselves. While we briefly discuss surrounding government policy and IoT security-focused legislation, our primary focus is on IoT security advice itself, versus mechanisms by which to enforce its use.

In the second domain, we focus our research questions in the area of identification of consumer IoT devices (*device identification* defined in Chapter 7). This includes determining which devices (and groups of devices) exist within an administrative domain, e.g., end-user home or enterprise offices. While not physical "things", we

include what are called *virtual devices*, e.g., abstract interfaces to devices that enable a user to remotely interact with their physical counterparts; these are common interfaces within IoT's remote-access paradigm, allowing remote interaction with devices via the Internet.

1.2 Motivation

As mentioned in our scope (preceding section), a general motivation for our work is determining how IoT security differs from IoC security, and how those differences are exhibited in consumer-grade IoT deployments. We focus on two areas where noting differences in IoT and IoC have led us to open research questions.

Our first focus is IoT security advice. Unlike typical IoC deployments, the cyber-physical nature of IoT—interfacing with physical world objects—results in threats to our physical world as well as to networks and other internet hosts [125]. This has resulted in considerable attention (e.g., [14, 61, 68, 151]) to security advice for IoT security. IoT security best practices are often stated to be widely available, but existing advice, as we have observed through our preliminary investigation, commonly equates best practices with desired security outcomes (the security goals to reach). The focus on outcomes (versus how to reach them) leaves the steps to reach security goals unspecified or ambiguous. An underlying problem that seems largely unrecognized to date is the lack of general consensus or awareness of the difference between a desired outcome and an *actionable practice*[2] (in the context of security). These observations motivate our research questions and investigation into the current state of IoT security advice, where we build a methodology that we argue allows security advice givers and relevant stakeholders to analyze security advice provided and determine if it is actionable for the target audience, and identify the characteristics of advice that lead to actionable advice.

For reference, Table 1.1 briefly outlines four primary documents whose contents are analyzed in Chapters 3–6. Each document relates to the others in specific ways, as summarized here. While these descriptions may not yet be useful to the reader,

[2]By *actionable* we mean a practice that focuses on steps or actions to follow rather than outcomes to reach. Chapter 3 defines this more carefully.

Table 1.1: Summary of primary IoT security advice documents used in this thesis. Document Name is the label we use to refer to each document. Formal names are provided where documents are described.

Document Name	Description
1. DCMS 1013-item dataset [59]	A large set of IoT security advice items. Our pre-processing of the set results in 1013 items for use in our analyses. Primarily used in the analysis of Chapter 4. Dataset advice referenced in Documents 2, 3, and 4.
2. DCMS 13 guidelines [62]	Contains 13 guidelines for securing IoT devices. Each guideline used as a category for advice in the large dataset (see Document 3). Analyzed and compared with the ETSI provisions document (Document 4) in Chapter 6 (§6.1).
3. DCMS mapping document [63]	Maps each item from the large DCMS dataset (Document 1) to one of the 13 DCMS guidelines (Document 2). Primarily used in the discussion of Chapter 6.
4. ETSI provisions [68]	A set of security advice items from ETSI. References the security advice items from Document 1. Analyzed and compared with the DCMS 13 guidelines (Document 2) in Chapter 6 (§6.1).

they are intended as a convenient overall reference.

A second topic we argue is different between IoC and IoT is identification. In IoC, it is common for the users to be the subject of identification (i.e., the users are identified, and their devices are tools for the user to interact with services); in IoT there is commonly no obviously identifiable user (discussed further in Chapter 2). One characteristic of IoT devices that differs from IoC is large deployment numbers. In IoC, particularly in a consumer environment (e.g., a home), devices are typically few in number and interacted with explicitly (e.g., using a computer or tablet's interface to interact with the tablet). In IoT, where devices tend to be indirectly interacted with (e.g., turning smart light bulbs on or off as regular bulbs; interacting with hub devices instead of smart devices themselves), it is more difficult to keep track of the status of each device (e.g., software they run, update status), and unauthorized access may go unnoticed. As such, determining which devices are within a domain has attracted research attention, yet *IoT device identification* appears to be (like IoT best practices) another area plagued by inconsistent terminology regarding approaches and goals. We use this as a starting observation to motivate our investigation into IoT device identification, and first explore what (generally) it means to identify an IoT device.

Investigating IoT device authentication more deeply (versus user authentication in IoC), we first consider: if IoC authentication techniques can be adapted for use in IoT, it may not be necessary to develop entirely new, untested techniques specifically for IoT. However, due to unique characteristics of IoT (Chapter 2), adapting IoC approaches for IoT may not be as simple as directly reusing known techniques, and the extensive history of flaws in authentication of constrained devices (e.g., Bluetooth [87,199,218]) suggests this is a difficult problem likely without a single solution for all use cases. As such, identifying why IoC approaches cannot be directly used may light the path to where progress can be made. This line of thought motivates our investigation into IoT device authentication approaches.

Based on these observations and motivations, the general goals of our research include: (i) to determine the security implications of how IoT differs from IoC, (ii) to understand the current state of consumer IoT security advice and determine the nature of advice being offered to consumer-focused IoT security stakeholders, and (iii) to better understand consumer IoT device identification, its goals, and by which approaches they are reached. These general goals motivate the research questions we pursue in this thesis.

1.3 Research Questions

The main research questions we address in this thesis are as follows.

RQ1. What are IoT security *best practices*, and how do they relate to security *design principles* and other commonly-used terms; how are these terms used to describe or characterize IoT security advice they are applied to?

RQ2. Can we design a methodology for objectively characterizing security advice?

- Here, by *characterizing* we mean to determine what category of advice an advice item is (i.e., in the sense of specific actions to take, outcomes to reach, or general principles to follow).

RQ3. How *actionable* is the current state of IoT security advice; does it primarily consist of security objectives to reach, or more specifically ways to reach those

objectives (i.e., actionable practices)?

RQ4. How can the coding tree methodology (developed in answering RQ2) be used to compare the actionability of different sets of IoT security advice, or generally characterize any improvement in subsequent versions of a given set?

RQ5. In what ways do researchers use the term *IoT device identification*, what are the most common goals of IoT device identification, and what approaches are being used to reach them?

- This allows us to explore the overlap between IoT and IoC authentication approaches, and how authentication goals are being met in recent IoT authentication proposals.
- Here, authentication goals may include, e.g., establishing initial keying material, carrying out entity authentication, establishing session keys.

1.4 Contributions

The main contributions of this thesis are as follows.

1. We provide what we believe is the first in-depth technical analysis of apparently intended meanings of the term *security best practice*, and related terms. We categorize commonly-used terms related to best practices, and suggest use-cases where terms in each category are best used.

   - This provides uniform, consistent terminology for use in the creation and presentation of security advice.
   - Among other commonly-used terms, we distinguish and define (*actionable*) security *practices* distinct from *desired security outcomes* and security *principles* (definitions given in Chapter 3).

2. We provide a novel analysis methodology (Section 4.1's security advice coding method, i.e., the *SAcoding method*; informally the *coding tree*) for characterizing security advice based on the terminology refined herein.

- We identify specific characteristics of security advice that contribute to the advice being actionable, and offer guidance on using these to specify actionable practices.

3. We apply the SAcoding method, first on the DCMS 1013-item dataset (which we believe represents current IoT security advice; see Table 1.1), enabling a novel analysis and characterization of the current state of IoT security advice.

   - This methodology allows a novel analysis of how actionable (defined in Chapter 3) several large sets of current advice are, specifically the DCMS 1013-item dataset and those mentioned in item 4.
   - We cross-check and critique the new methodology itself by exploring the reproducibility of DCMS 1013 dataset analysis results by a second coder.

4. We use the SAcoding method to compare the actionability of advice items from the DCMS 13 guidelines and ETSI provisions (Documents 2 and 4 in Table 1.1).

   - This analysis highlights the methodology's value for comparing and characterizing improvement in sets of security advice.

5. We offer a model relating the concepts and approaches involved in IoT device identification to typical identification goals (Fig. 7.2). We use the model to analyze a selection of recent IoT device identification proposals and extract the most common end-goals, and the approaches commonly used to reach them.

   - As a byproduct, some ambiguity in common IoT identification terminology is resolved.

1.5 Outline

Chapter 2 discusses background related to IoT in general, including the differences between IoT and IoC, and new security challenges involved with IoT. Chapter 3 examines terminology used in the documentation and discussion of security advice;

we disambiguate commonly-used terms (notably, *best practice*) and provide concrete definitions for terms that we use to analyze existing security advice. Chapter 4 develops an analysis methodology for analyzing security advice, and we conduct an empirical study of the DCMS 1013-item dataset to determine (among other analyses) how actionable the advice is that government, industry, and academic sources are providing. Chapter 5 is a critique of the coding tree methodology, and considers the degree to which the results from Chapter 4 are matched by a second coder, and offers insights derived from a comparison of two coders' tagging results. Chapter 6 compares and critiques the DCMS 13 guidelines and the ETSI provisions, and applies the coding tree methodology to both documents' advice items to illustrate its utility for comparing the actionability of different sets of advice and characterizing how or if one set improves on another. Chapter 7 investigates IoT device identification, including categorizations of the desired goals of IoT identification, and the approaches used to reach them. Chapter 8 concludes with a discussion of remaining challenges in the creation of IoT security advice, identification of IoT devices, and directions for future research.

1.6 List of Publications

Chapter 2 content is available in the proceedings of a peer-reviewed conference [36]:

> Christopher Bellman and Paul C. van Oorschot. Analysis, Implications, and Challenges of an Evolving Consumer IoT Security Landscape. In proceedings of the International Conference on Privacy, Security and Trust (PST), 2019.

Content from Chapters 3 and 4 has been submitted for journal publication and is currently undergoing minor revisions [32]. An earlier version of part of this work was filed as a technical report:

> Christopher Bellman and Paul C. van Oorschot. Best Practices for IoT Security: What Does That Even Mean? Apr 2020. Technical report available at: `https://arxiv.org/abs/2004.12179`

Content from Chapter 5 has been submitted for journal publication [33]. Parts of research involved in Chapters 4 and 5 were done in collaboration with Dr. David Barrera, who contributed through discussion and as a test coder in methodology test trials, to the development of our coding tree methodology, and was one coder for the main tagging exercise of the 1013-item IoT security advice dataset (described in Chapters 4 and 5). As such, his tagging data is included in the analyses of Chapter 5.

Content from Chapter 6 has been submitted for journal publication and is currently undergoing minor revisions [37].

Chapter 2

Internet of Things vs. Internet of Computers

In this chapter, we discuss general background for consumer-grade IoT architecture and devices, and how IoT is different from the existing Internet of Computers (IoC). In particular, we identify five distinguishing characteristics of consumer IoT devices, discuss the implications each has for security, and highlight how these result in new or different challenges than those of IoC. These characteristics, implications for security, and challenges provide context and background for the problems we address throughout this thesis.

IoT—in both its definitions and the security community's perceptions of it—has evolved considerably over the past decade. As such, there is need for a revised understanding of the landscape of IoT security. Fig. 2.1 gives a partial taxonomy reflecting our view of major categories of IoT. Surveys prior to the Mirai botnet (e.g., [189, 196, 230]; others also noted in the next section) lack technical details as work done in more specific areas of IoT security was yet to be carried out in depth. Technical progress over the past decade allows a new understanding of the challenges and opportunities in IoT security. We take a fresh look at the characteristics that distinguish IoT from IoC (Section 2.3), and consider the implications of each for security as they impact, e.g., IoT network architecture, security and networking protocols, and cryptographic implementations. We relate these implications to current security challenges, and those that are expected to appear as IoT continues to grow, permeating our environments.

2.1 Brief IoT Security Literature Review

As with IoT itself, literature surrounding IoT is growing. Here we first mention related work in the form of selected security-focused publications and surveys of IoT security. Smith [191] discusses emerging threats in IoT and differences between IoT

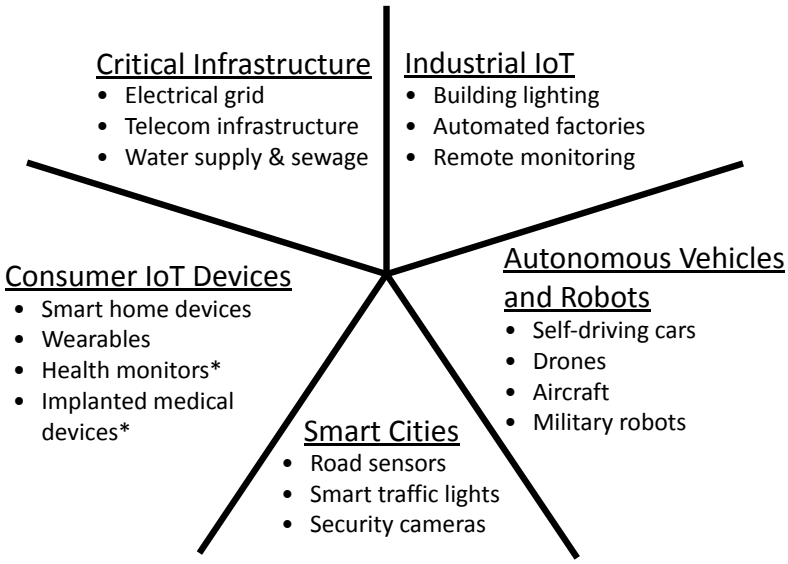


Figure 2.1: Partial IoT taxonomy and examples of sub-areas. Our focus herein is on consumer IoT devices. We exclude medical devices (e.g., implanted devices or those provided by a medical service such as a hospital) from our scope (denoted by $*$) as they appear to be more safety-critical and may involve more regulation; however, some consumer-grade wearable devices included in our scope (e.g., off-the-shelf devices) may offer health-focused features such as pulse or blood-pressure monitors, but are not necessarily designed for clinical use.

IoT security space for non-technical audiences.

Alrawi et al. [13] highlight components of the malware lifecycle (e.g., infection vector, malware payload, capabilities) and compare how each affects desktop, mobile, and IoT platforms. Notably, their Table 1 shows how similar the malware lifecycle is for all three platforms, with the most major differences appearing among how devices are infected (the infection vector). Beyond how devices are infected, the three platforms have many similarities with how malware is designed for each; notably, the desktop and IoT platforms share the majority of components, highlighting that IoT and IoC (particularly desktop platforms) are still similar in many ways. From a malware perspective, IoT devices may be more similar to some components of IoC such as desktop platforms than others like mobile platforms [13].

Ferrag et al. [77] conduct a review of authentication proposals for specific IoT subdomains (e.g., "Internet of Vehicles", "Internet of Sensors"), and compare the proposals' threat models, security technologies (e.g., hashing, data encryption, and

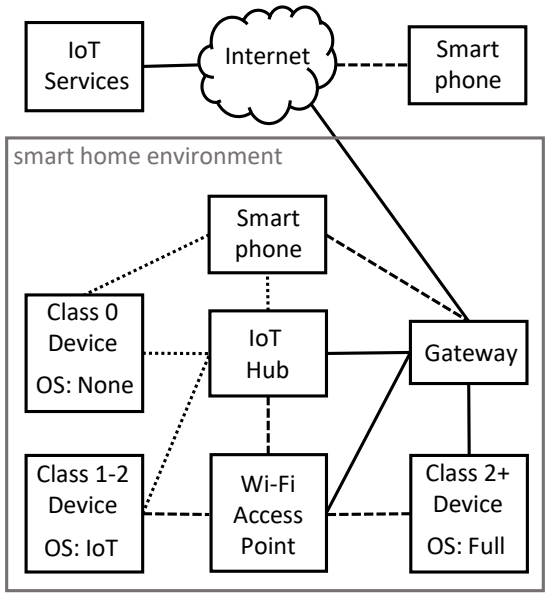


Figure 2.2: Generic architecture of smart home IoT deployment. Solid lines denote wired connections, dashed lines represent Wi-Fi, dotted lines represent low-power wireless, e.g., Zigbee, Bluetooth Low-Energy. "IoT" OSs are specifically for IoT devices (Section 2.3.1). "Full" refers to traditional OSs, e.g., Linux.

session-key establishment techniques), and method for formal security validation. They identify challenges and open issues within each subdomain. Their IoT subdomains map on to our IoT sub-areas from Fig. 2.1, e.g., among others, their "Internet of Energy" roughly maps to our *Critical Infrastructure* sub-area, their "Internet of Sensors" roughly maps to our *Smart Cities*.

2.2 Generic Architecture of Consumer-Grade IoT Devices

The architecture of IoT is defined by the mechanisms and physical structure by which each device in the network communicates with others. Fig. 2.2 depicts a simplified view of the network architecture for a smart home. *IoT services* include interoperability and trigger-action programming functions (e.g., IFTTT [109]) and management platforms (e.g., Amazon AWS IoT Core).

Low-end devices make use of lightweight communication protocols and standards. IoT-friendly (lower resource consumption) upper-layer protocols are commonly used for communicating with other devices or services [97]. Hub devices (e.g., Phillips

Table 2.1: IoT device characteristics and their implications for security.

1. Low-Cost (Section 2.3.1)
   - Constrained resources
   - Smaller/no OS
   - Need for more efficient protocols
   - Need for lightweight cryptographic algorithms and protocols
   - Over-provisioned functionality (low-cost component re-use)
   - Manufacturer security inexperience (IoT sub-component)
2. Non-Standard Interfaces (Section 2.3.2)
   - New attack surfaces
   - Greater physical access to devices (by attackers)
   - Device management, configuration, and updates complicated; exacerbated by scale
3. Cyberphysical Interaction (Section 2.3.3)
   - Successful network attack may affect physical world
   - Implied trust in manufacturer (devices can cause physical damage to real-world)
4. Expectation of Long-Lived Devices (Section 2.3.4)
   - Lack of support for software updates may leave vulnerabilities unpatched
   - Forgotten devices remain attractive targets
   - Device outliving manufacturer impacts software updates
   - Cryptographic algorithms and protocols must be future-proofed
5. "Many-User" Devices with Unclear Authority (2.3.5)
   - Home guests may be denied functionality of critical services
   - Rogue guests may retain remote access
   - Difficult to differentiate authorized and unauthorized users

Hue Bridge, Samsung SmartThings Hub) are used to manage local devices and bridge communications to other hosts; these are typically less resource constrained. Devices that require an intermediate node for communication connect to a hub via low-power wireless to forward messages. Higher-end devices connect directly to a gateway or Wi-Fi access point. From the gateway, traffic can be routed as normal on the Internet. Alternatively, smart devices such as wearables connect directly to a smartphone via low-energy wireless and/or to the Internet via cellular signal or Wi-Fi. Remote devices can be used to access IoT cloud services and smart home devices.

2.3 Distinguishing Characteristics of IoT

IoT has characteristics that distinguish it from IoC. Here we discuss select characteristics and their security implications (Table 2.1).

In IoT, an individual user may have a variety of devices associated with them

in addition to standard IoC devices (e.g., laptop, smartphone, desktop computer). Combined with non-personal devices associated instead with environments, the number of IoT devices is expected to dwarf the number of IoC devices [108]. This scale impacts essentially all characteristics as the number of devices in a domain commonly exacerbates existing issues.

2.3.1 Low-Cost

An IoT device might simply be a standard device with a small built-in computer/ communications component. When an IoT device is referred to as "low-cost", we often mean its IoT component. Manufacturers typically minimize the cost of an IoT component, favoring market presence over security [154].

Device resource constraints are typical consequences of low costs. Precisely defining what it means to be *resource constrained* is difficult, as definitions based on hardware quickly become outdated (e.g., 32-/64-bit processors are becoming more commonly used in IoT devices [23]); terminology used to describe existing devices has blurred with time (e.g., home entertainment set-top boxes formerly described as "embedded systems" are now described as an IoT device or "embedded IoT" [21] due to their communication capabilities); and resource requirements for different functions may vary considerably, making one set of resources sufficient for one application, but not for another.

Some resource constraints in IoT are the absence of input/output (e.g., screen, keypad), memory sizes, processor speeds, and battery size. RFC 7228 [46] defines three classes of resource constrained devices (Table 2.2). Class 0 devices are generally too constrained to communicate directly with hosts on the Internet securely, relying on an intermediate node to communicate via low-power protocol such as Bluetooth Low-Energy (BLE), Zigbee, or 6LoWPAN. They typically use single-purpose specialized microcontrollers [99]. Class 1 devices commonly struggle to communicate over the Internet using standard upper-layer communication protocols (e.g., HTTP, TLS), instead using lighter-weight protocols through intermediate nodes; and may use a basic IoT-focused OS. Class 2 devices still leverage protocols and features designed for resource constrained devices, but may (depending on hardware and software) also

Table 2.2: Resource-constrained device classes, memory limitations [46], operating systems (if any), and communication methods [99].

Class	Volatile memory	Non-volatile memory	OS & Communication
0	<<10 KiB	<<100 KiB	Function-specific hardware, few IoT OSs. Basic health indicator and keep-alive messages, requires intermediate node.
1	∼10 KiB	∼100 KiB	IoT-specific OS. Lightweight wireless (e.g., BLE)/wired, UDP-based protocols.
2	∼50 KiB	∼250 KiB	IoT-specific OS. Lightweight wireless/wired, UDP-based protocols, commonly-used upper-layer protocols.
2+	>50 KiB	>250 KiB	IoT-specific, or full OS. Commonly-used upper-layer protocols.

be capable of running standard protocols to communicate on the Internet.

An *unconstrained* IoT device is often mainline powered, has one or more 32 or 64-bit processors [23], potentially gigabytes of memory, and possibly a screen. In the latter case, we may not even consider it to be an IoT device, as it has roughly equivalent specifications to an IoC device.

In this thesis, when we discuss resource-constrained devices, we mean devices that: may be battery powered (versus mainline power), have relatively limited memory (e.g., potentially unable to support common cryptographic libraries or store large cryptographic keys), are not equipped to communicate over wired or Wi-Fi networks (instead using, e.g., ZigBee and Bluetooth), or have less powerful processors than typical IoC devices (e.g., 8/16-bit processors) [46]. The resources available to a device may restrict its functionality or limit its capabilities—security features feasible for an unconstrained device may be infeasible for a constrained device (e.g., due to missing standard interfaces, see Section 2.3.2; battery power preservation, limiting communication or computation; missing hardware features such as PUFs, see Chapter 7). As such, developing security best practices for IoT devices (Chapter 3) must take the target devices' constraints into account when selecting appropriate security mechanisms, and identification and authentication approaches (Chapter 7) must be appropriate for the targeted devices.

Implications for Security. Seeking cost reductions, manufacturers may use open-source software or generic hardware in their devices, choosing components that

provide the required functionality. Use of over-provisioned components increases risks. For example, unused modules and features, often not properly disabled, provide additional attack surface. A common example is using Linux for the OS of a device (and not disabling functions or services that are unused). This consequence is related to manufacturer inexperience as new manufacturers who do not fully understand their technical or functional needs may choose generic, potentially over-provisioned solutions; standard (over-provisioned) Linux may also be deployed simply due to cost-cutting, as disabling unused functionality may require additional time-cost.

Class 0 devices are highly resource constrained—they are typically specialized microcontrollers that have static and highly specific functions [99]. Those above Class 2 may be capable of running a full operating system like Linux. Devices in between (classes 1 through 2) can use a variety of open- and closed-source OSs tailored for resource constrained devices (e.g., *FreeRTOS*, *TinyOS*, *Contiki* [99], *Tock* [133]).

A wide variety of low-power protocols (e.g., BLE, 6LoWPAN, Zigbee) are typically used to communicate with other physically-near devices. Depending on the hardware, less resource constrained devices are capable of running Wi-Fi and common upper-layer protocols for Internet communication. IETF work currently underway is developing new or adapted suites of protocols designed for constrained devices (e.g., CoAP [187]).

Both communication and cryptography functions require processing and memory, so lightweight cryptographic algorithms and wireless protocols need to be used, especially for Class 0 devices. Devices must be able to run common cryptographic algorithms at acceptable speeds to meet secure communication requirements. Generating and storing sufficiently-long asymmetric keys (e.g., $\geq$2048 bits for RSA [31]) in IoT is more challenging than in IoC. Further, best practices for key sizes will grow over time (e.g., 3072 bits recommended for RSA by 2031 [30]), which is increasingly problematic for IoT. In comparison, elliptic curve (EC) cryptography boasts faster computation and smaller key sizes (than RSA) [38, 136].[1] This (along with better computational efficiency) explains the adoption of EC cryptographic algorithms in IoT environments.

[1]For example, a 224-bit ECDSA key has comparable strength to a 2048-bit RSA key [30].

Now that IoT has become attractive for manufacturers, the "IoT" label on a device may be used as a marketing feature; however, the addition of IoT functionality is often not accompanied by security expertise (for IoT subcomponents). Manufacturer inexperience amplifies safety issues as any weaknesses with a device adds to the potential impact of attacks and problems related to non-standard interfaces.

New Problems/What is Different. Class 1 and 2 devices are capable of running lightweight OSs designed specifically for IoT. Common constraints include limited memory (≈10–50 KiB volatile, ≈100–250 KiB non-volatile memory, per Table 2.2), low power usage (must operate for months without battery replacement), low-delay processing (commonly requires real-time responses), and built-in security mechanisms (cryptographic/security protocols, access control) [99].

A common characteristic of IoT OSs is their development in the C language [99]. C and C++ have historically been the choice for IoC operating systems and tools; however, they bring with them a number of vulnerabilities such as memory safety errors (e.g., buffer overflows), integer-based vulnerabilities, and race conditions. Many past security issues experienced in IoC OS design are at risk of reappearing in IoT as C/C++ are used in IoT-specific OSs [99].

For wide-spread adoption of secure communication, toolkits will need to support both expert and non-expert developers. Many such toolkits exist for IoC and higher-end IoT devices (e.g., *OpenSSL*, *NSS*, *wolfCrypt*—commonly written in C). Libraries such as *micro-ecc*, *TinyECC*, and *μNaCl* bring limited cryptographic functions to heavily resource constrained 8-bit microcontrollers. Some IoT development boards use dedicated processors (e.g., Microchip ATECC608A) for cryptographic algorithms and key storage.

Prevention and/or mitigation of malicious action is required to address device compromise. The recently-proposed Manufacturer Usage Descriptions (MUDs) are manufacturer-provided descriptions of how their devices are designed to behave [102]. MUDs obtained directly from the manufacturer are intended to simplify misbehavior detection and may simplify detection of anomalous activity. In the absence of MUDs, automatic generation of communication policies at the network level may be pursued [34, 152].

2.3.2 Non-Standard Interfaces

Device interfaces vary significantly between IoC and IoT. For usability, the challenge in IoT is often greater in the configuration of a device rather than in its standard function. Interaction design is related to the ways a user interacts with a device. In IoC, this is almost exclusively using a keyboard and monitor, or touch screen. IoT devices commonly require some alternative method for device setup or configuration (e.g., smartphone app, cloud management service). This leads to a number of challenges for users to manage device updates, configuration, and decommissioning.

IoT is still fairly new, and device diversity is high. Diversity is amplified by the wide range of what we define as an IoT device and makes it difficult to standardize interaction and management methods. This exacerbates problems such as secure device configuration or communication between devices. Coupled with hardware differences, in highly-constrained devices, software is specialized for a specific task making it difficult to produce software for, update, and manage a wide variety of devices. This is ameliorated in devices with IoT-focused OSs providing common code bases [99].

Implications for Security. New interaction interfaces introduce new attack surfaces. Voice commands have proven useful in smart home devices, taking any sound in the environment as a potential command. Sensor inputs (temperature, noise) can be abused to provide falsified data (e.g., manipulating sensor readings). Cloud services present additional attack surface, with the core cloud services inheriting IoC challenges. Finally, physical access to IoT devices is an additional attack surface; it is likely easier for a guest or intruder to physically access or steal a small IoT device than a laptop or desktop computer due to the devices' placement and ubiquity. Once stolen, the attacker could potentially access the home network using this device or recover sensitive data from its storage.

Problems with the usability of a device's interfaces are exacerbated by the scale of devices to be maintained. It is one thing to configure a small handful of devices, but scaling the quantity up may cause users to become frustrated and potentially leave devices configured with default (and possible insecure) settings (e.g., default passwords, as exploited by the Mirai botnet [21, 125]). IoT devices—particularly

consumer devices—should be simple to set up and maintain for the average user. This makes designs involving safe default settings important. If a device functions correctly (from the user's perspective) without a secure configuration, users may choose to not configure it. Enforcing device configuration before allowing it to function would solve this issue [14], but it would impact usability and frustrate users, suggesting an important aspect of secure device onboarding is usability.

New Problems/What is Different. For each new interaction mode, new protection methods may be needed to mitigate attacks conducted via their use. For voice inputs, unauthorized users might send inputs to a device via audio commands. For example, "Hidden voice commands" can trigger functions on voice-activated devices without being understandable voice commands to the human ear [50], leading to attacks that users cannot identify. Sensors should ideally determine if a reading has been falsified or manipulated. Cross-checking of device readings may be viable in some cases (e.g., comparing a thermometer's reading to a reading from across the room) to determine consistency, but requires communication either between (possibly heterogeneous) sensors or with a central hub device. Remote management requires that each communication hop between a device and its remote access point (Fig. 2.2) be secure. IoT resource constraints require new solutions for secure end-device communication. IoT cloud services are a more usable approach to interacting with devices compared to individually connecting to each one, but the security of such platforms is beyond users' control, and they must trust the service provider to maintain security of service infrastructure and communications.

If a device lacks standard interfaces, users may find the device difficult to update or configure, and this may motivate users to interact with it as little as possible or use default configurations. A non-standard interface is a general nuisance for users (if not highly usable) that adds a barrier to the adoption of some security mechanisms such as authentication approaches, where a user may be required to update or configure keying material. This is discussed in more depth in Chapter 7.

As smaller, more pervasive devices are easier to physically access or steal, each individual device would ideally protect itself from physical or digital attacks. One approach is to minimize the sensitive data stored (such as user data or sensitive

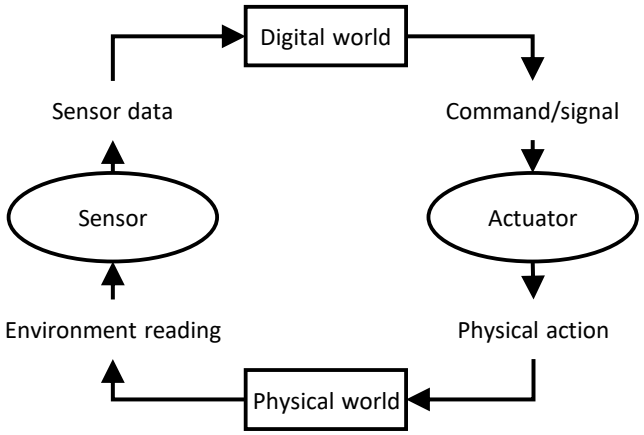


Figure 2.3: Relationship between sensors and actuators, and their role in IoT.

communication information), under the assumption that the device will be stolen, and the data then recovered by offline attack. Thus, an additional security design challenge for IoT devices is that they should store only the minimum information required to successfully function.

2.3.3 Cyberphysical Interaction

Over time, the terms "cyberphysical system" and "IoT" have merged, and are now commonly used interchangeably [90].[2] Both share the characteristic of linking the physical environment to the digital space, so we refer to a cyberphysical device as a device that interacts with its environment. Fig. 2.3 depicts two common classes of cyberphysical IoT components—sensors and actuators—and their relationship to one another and the world. Sensors convert environmental readings into data, and actuators convert digital commands into signals that trigger physical manipulation of an environment.

Implications for Security. In IoC, attacking a system means interacting with data, but in IoT there is a potential impact on the physical world. The new risk in IoT is that network attacks can affect physical environments through actuation. Further, with devices surrounding us, it is possible to link data from various sources.

[2]Cyberphysical systems tend to be viewed as being more about controlling mechanical systems (in a more industrial sense), whereas IoT is viewed more as the integration of digital information and communication with the physical world [90], intuitively suggesting that an IoT system involves less actuation than a cyberphysical system.

Recorded data from our environment can reveal richer knowledge about us and our environments. Purchasing and using an IoT device implicitly demands trust in the manufacturer and service providers, in the case that data is stored and/or processed by them, and also in the case that devices can cause harm to a user (e.g., a thermostat could be disabled, causing a home to freeze during winter; a thermostat could be set to maximum, increasing heating costs).

New Problems/What is Different. In IoT, triggering an actuator in a device alters the physical environment, thus security vulnerabilities may result in greater damage (or personal harm) and threat models need to consider this [217].

Currently, environmental monitoring across a variety of devices is difficult due to the heterogeneity inherent in IoT [215]; however, with greater standardization and interoperability between devices and ecosystems, tracking user behavior (for beneficial or malicious use) will become easier as data sources become available to more devices in an environment. Even if the data itself is unusable (encrypted, anonymized), metadata could be linked to extract personal information about the user or their environment [147, 229]. However, cross-checking data from multiple sources may also provide a defense against misbehavior (e.g., manipulation of sensors, malicious alteration of data; unauthorized devices added to the network, discussed in Chapter 7) [215]. Multiple devices can use contextual data or metadata provided by its surrounding devices to sanity-check other input data.

2.3.4 Expectation of Long-Lived Devices

Consumer IoT devices may operate for a long time (e.g., 5–10 years). This requires that devices remain functional and secure over this period. Many types of device are designed to minimize interaction, as in "set-and-forget". Once set, they may continue functioning without frequent (or any) maintenance. Devices such as smart light bulbs should function properly as light bulbs, but not require the user to check for software updates or perform maintenance. The IoT device lifecycle is further discussed in Chapter 3.

Implications for Security. The need for software updates is a significant threat to IoT devices as lack of updates means potentially unpatched vulnerabilities. It

may aid usability if manufacturers can take responsibility for the update process, however users may need to be made aware of the existence of vulnerabilities and updates, should automatic, manufacturer-controlled updates be unavailable. Further complicating the issue of update strategy, a device could outlive its manufacturer—software updates may not be available throughout a product's lifetime. Will third-party developers be able to fill the gap as is common in the IoC? With (relatively) few major commodity operating systems in IoC (Windows, macOS, Linux), software development toolkits are commonly accessible. In IoT where low-end (e.g., Class 0 from Table 2.2) devices make use of specialized hardware and commonly run no OSs at all, building third-party software is more difficult (no accessible toolkits/developer APIs) and may be not commercially viable.

Devices may continue to be used, after an initial IoT-supported configuration, without further end-user reliance on IoT functionality. For example, smart light bulbs may be set once to a standard white color and high brightness, then never changed again—thereafter turned on and off like a normal light bulb. In cases where devices remain network connected but no longer relying on network connectivity (for their IoT functionality), the network connectivity may remain as an attack surface for attackers to exploit. The combination of network connectivity, lack of monitoring, and potentially little or no manufacturer update strategy leads to devices remaining vulnerable for significant periods.

Another potential issue is quantum computing and its impact on currently infeasible computational problems. A risk is that long-promised quantum computing advances may result in the failure of public-key algorithms built on RSA, Diffie-Hellman, and corresponding EC analogues [96]. Quantum-safe (also known as post-quantum) cryptographic algorithms and protocols are applicable to IoC as well. This poses a serious problem for IoT devices that are constrained and not easily updated.

New Problems/What is Different. Software updates must be received with verifiable integrity (i.e., authenticity). A method for determining which devices need updates will be required to point users to out-of-date devices and to enforce security policies [152].[3] In the case of a bad update (whether by accidental corruption or a

[3]Device identification is discussed in Chapter 7.

malicious image), a device may operate with vulnerabilities or attacker access, or may not function at all. The ability to roll back an update may help; however, this function might be unknown to users or fail if the update was malicious or disabled the device. Given the set-and-forget nature of IoT, updates presented to the user (if any) may well be ignored or declined. Push-based automatic updates may be more appealing in the scale of IoT (as opposed to users manually pulling or approving updates). This is, however, less easily accomplished than in IoC where near constant TCP/IP Internet connections are expected.

What happens if a vulnerability is found, but a user can not patch it (unavailability or no knowledge of patch existence)? If a manufacturer is unable to provide updates, the responsibility could be transferred to another entity. Regardless of the OS a device is using or the status of a manufacturer, a software update policy set by the manufacturer could be used to enforce how and when updates are applied, and who is responsible for them [154]. This is highly related to the lifespan of a device as short-lived devices would be seen as less important to spend resources developing software for. Regardless, a software update strategy for long-lived devices is needed, one option being deactivation of devices past their manufacturer-supported lifetimes. Inactive devices (devices connected to a network but no longer used by end-users) should have their network access disabled in order to protect the integrity of the network.

2.3.5 "Many-User" Devices with Unclear Authority

In IoC, systems are commonly labeled as "multi-user" or "single-user" based on their architecture and usage. In these systems a user is identified (by, e.g., a username or user ID) and they interact with a system within the context of being an identified user. In IoT, devices commonly belong to an environment and are generally not linked to an individual user or group of users, thus being a "many-user" device (e.g., sensors, voice assistants, lights). While this is not the case for all IoT devices, it is a characteristic that is common.

Implications for Security. Device-to-device access control is a matter of configuration by the owner or trusted user, which could be (and commonly is) solved by

standard role-based or discretionary access control [145]. IoT devices may now belong to an environment rather than a user. This makes it difficult for systems to differentiate between authorized and unauthorized users. Modern smart home hubs provide different approaches (based on device or device ecosystem) to access control such as differing levels of control (privileged versus unprivileged user), guest accounts, and time-/location-based policies [145]; however, these are for hub devices—individual "things" in an environment require different approaches as standard role-based or discretionary access controls are not applicable when an individual user can not be uniquely identified.

A homeowner who deploys devices within a house has full access to the devices they own. The owner of a home who provides it to a rental service (e.g., Airbnb, VRBO) requires administrator access to critical devices (e.g., lighting, heating) but guests should be granted basic user access to at least operate the devices; however, different IoT ecosystems provide guest access in different ways, which presents a usability issue for how to manage guests [145]. Different hubs and more advanced IoT devices may have interfaces that employ user accounts for access control [145]. For devices incapable of this or ones that are used frequently, conveniently usable methods for access control appear to be necessary as it becomes tedious to manage users or authenticate frequently (discussed below). To highlight one access control issue, rogue guests could continue to access devices after leaving and returning to an environment and, for example, unlock the front door of a rental unit if their credentials were not disabled.

New Problems/What is Different. A single solution for access control that covers all permissions and devices within a system may be feasible if all devices belong to the same ecosystem and are designed to be interoperable, but may be impossible for devices of different ecosystems that are incompatible with a single shared access control system. Some ecosystems' hub devices may limit access to configuration settings to authorized users, but providing a user with access to use a light bulb and thermostat of two different brands becomes more difficult as an interoperability layer may be required, or two separate means for authorization. Further, access control may be done differently in different devices or ecosystems, making access control

more open to misconfiguration in environments with device heterogeneity.

Once user access is granted, it may need to be revoked. Not all devices provide access control mechanisms that separate privileged and unprivileged actions [14]. Administrators should ideally have the ability to configure revocation strategies; however, some revocation methods require identification of individual users. Alternatively, solutions such as time-limited access tokens provide automatic revocation, but require careful configuration of user access windows to balance user access and exposure window (time after legitimate token use but before token revocation). For devices accessed via smartphone, this is not a problem as identification becomes linked to the user's phone.

2.4 Discussion and Concluding Remarks

It is generally acknowledged [14, 21] that many consumer-grade IoT devices have easily exploited security vulnerabilities, but existing toolkits (for critical software tools and cryptographic protocols, Section 2.3.1) may not adequately support the needs of constrained devices. Given IoT hardware capabilities, cryptographic libraries require re-engineering to meet the constrained capabilities, and commonly relied upon algorithms in IoC need to be re-evaluated, e.g., as in now using ECC over RSA [121, 136] to meet performance challenges. Research over the past two decades has explored lightweight cryptography in a number of areas including wireless sensor networks [134, 136, 205, 224] and smart grid applications [142]. Findings in these areas can be applied in IoT.

A common theme that appears in all characteristics and problems discussed in this chapter (Table 2.1 on page 13) is the scale of IoT. Scale exacerbates all security and usability challenges. Both manufacturing and deploying properly secured devices is already difficult in IoC for a home environment; IoT brings comparably many more devices, complicating access control (fine-grained access control becomes infeasible with scale), secure configuration (of a larger number of devices), and device software updates (secure protocol design, acquisition, integrity, authorization, and installation).

While scale is an important characteristic of IoT that differs from IoC, as discussed throughout this chapter, other characteristics of IoT security (Table 2.1 on page 13) differ from IoC. To follow our general Chapter 1 guiding theme of how IoT security differs from IoC security, throughout this thesis we consider how these characteristics are exhibited in consumer-grade IoT deployments as they (the characteristics) impact the topics we discuss.

Chapter 3

Disambiguation of Security Advice Terminology

In this chapter, we examine the terminology used in the documentation and discussion of security advice, and disambiguate commonly-used terms (notably, *best practice*). Comparing and contrasting different perspectives, we provide concrete definitions for terms that we use to analyze existing security advice, and our analysis from this chapter motivates portions of subsequent analyses in Chapters 4–6.

The term *best practice* is commonly assumed to be intuitively understood, yet academic work in this area (as noted in Section 3.2) lacks consensus on informal definitions for the term, and closer inspection suggests a clear explicit definition is needed. We argue that this assumption results in ambiguity and contributes to security problems, and that intuitive understandings are at best foggy and differ considerably across even experts.

Large collections of documents from industrial, government, and academic sources also conflate best practice with common terms such as *recommendation* and *guideline* [59]. How do best practices, good practices, and standard practices differ? Or guidelines, recommendations, and requirements? If something is not *actionable*, does it make sense to recommend it as a best practice?

We provide what we believe is the first in-depth technical examination of intended meanings of the term *security best practice*, and the surrounding related terms noted above. We argue that confusion and ambiguity result from the lack of a common understanding and precise definition of these terms, and that this confusion permeates official best practice recommendations (as discussed in Chapters 4 and 6). We support this argument by first investigating current use of terms related to best practices, and explain how meanings of each term differ qualitatively (Section 3.2). We classify these descriptive terms into three categories and separately define (actionable) security *practices* distinct from *desired security outcomes* and security *principles*.

3.1 Background and Overview of Established IoT Security Advice

In this section we discuss key areas of IoT and their role in the adoption of security best practices. These areas include: the IoT device lifecycle (and when in a device's lifecycle security advice is applicable), which stakeholders have the most significant impact on the security of a device, and existing security advice, which we analyze in Section 3.1.2, Chapter 4, and Chapter 6.

3.1.1 Lifecycle of IoT Devices

The lifecycle of a consumer IoT device includes phases it goes through from early design to the time it is discarded (possibly re-used, or never used again) [82]. We model the full lifecycle of a device, as decisions made within one part of the lifecycle (particularly the pre-deployment stages) may affect later phases.

Once IoT products have left the hands of manufacturers, it becomes more challenging to address vulnerabilities. We believe it is important to understand the stages within each phase, as security advice to be followed relates directly to the processes carried out within specific stages. Fig. 3.1 presents our model of a typical lifecycle of an IoT device based on existing work [82], modified to incorporate what we believe are the most relevant phases (and stages within them). Our model highlights our interpretation of the four major phases where IoT security advice is generally applicable. Our analysis (Section 4.2) includes a discussion on the impact of security advice followed at each lifecycle stage.

The Creation phase takes place under the authority of the manufacturer, where a device is designed, developed, and pre-configured. The Creation phase happens *pre-deployment*, i.e., before the device is sold to an end-user. This excludes when a user receives the device from another user.

In the Installation phase, the user has received the device and readies it for normal use. This is the first post-deployment phase, and contains *onboarding* or *bootstrapping* (often used interchangeably or meaning slightly different things, depending on who uses it) [184] which includes technical details such as key management, registration and identification of devices, establishing trust relationships, and other device configuration.

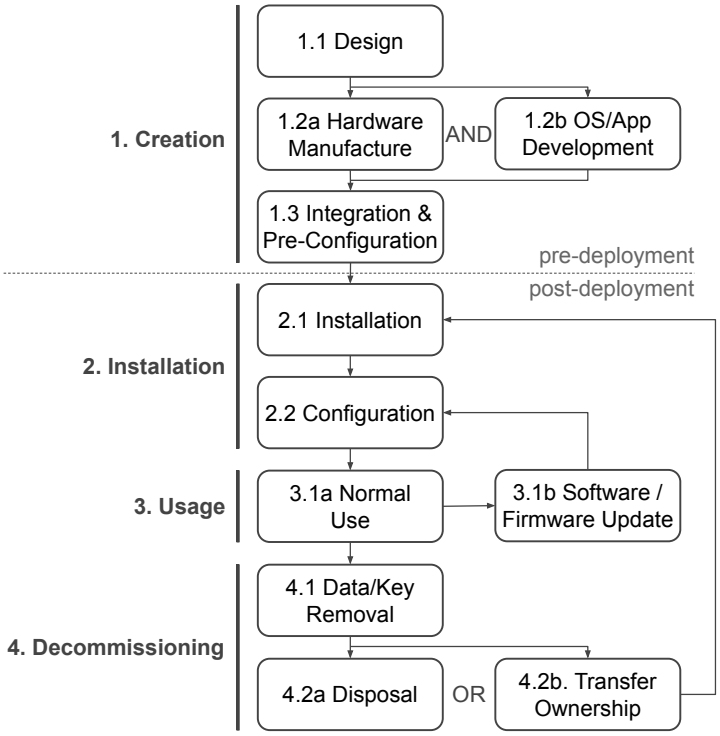


Figure 3.1: Typical IoT device lifecycle (initial design to end-of-life). Broad *phases* (1–4) encapsulate multiple *stages* (e.g., 2.1, 2.2).

The Usage phase involves using the device as intended (e.g., a light bulb provides light, a smart thermostat controls temperature, a home security camera provides live camera access). While containing only two stages, the device is expected to spend most of its life in this phase. Software/firmware updates take place in this phase.

The Decommissioning phase is where a device ends its life with respect to a single user or organization. The device is readied for removal from its environment (data/key removal from the device), physically removed, and leaves the end-user's ownership (either via disposal or transfer of ownership to another user). If device ownership is transferred to another user, the device returns to the Installation phase where the post-deployment ownership phases begin again.

Table 3.1: Assignment of UK DCMS guidelines to lifecycle phases. Detailed guideline descriptions are given in the DCMS 13 guidelines document [62].

Guideline Title	Example Sources	Lifecycle Phase (Fig. 3.1)
UK-1 *No default passwords*	[57, 61, 71]	1.3
UK-2 *Implement a vulnerability disclosure policy*	[48, 71, 135]	1.1 3.1a
UK-3 *Keep software updated*	[10, 48, 49]	1.1 1.2b 3.1b
UK-4 *Securely store credentials and security-sensitive data*	[57, 58, 71]	1.2a 1.2b 1.3
UK-5 *Communicate securely*	[48, 58, 71]	1.2b 1.3
UK-6 *Minimise exposed attack surfaces*	[11, 26, 54]	1.2a 1.2b 1.3
UK-7 *Ensure software integrity*	[71, 92, 200]	1.2b
UK-8 *Ensure that personal data is protected*	[9, 11, 53]	1.2a 1.2b 1.3
UK-9 *Make systems resilient to outages*	[48, 49, 71]	1.1 1.2b
UK-10 *Monitor system telemetry data*	[54, 57, 71]	1.1 1.2b 3.1a
UK-11 *Make it easy for consumers to delete personal data*	[70, 164, 193]	1.1 1.2b
UK-12 *Make installation and maintenance of devices easy*	[135, 193, 200]	1.1 1.2a 1.2b
UK-13 *Validate input data*	[71, 165, 193]	1.2b

3.1.2 Established IoT Security Advice

In this thesis we conduct analyses of IoT security advice contained in one dataset, one UK DCMS document, and one ETSI document.[1] We describe these advice sources here for context, and they are analyzed throughout Chapters 4–6.

DCMS 13 guidelines. The DCMS *Code of Practice for Consumer IoT Security* [62], proposes 13 guidelines.[2] Each includes a brief summary title and a more detailed guideline. This set of guidelines is analyzed in detail in Chapter 6. Table 3.1 lists all 13 guideline titles and our assignment of the IoT lifecycle stages in which these guidelines are applicable, and examples of informal (unrefereed) primary source documents suggesting such advice.

The DCMS 13 guidelines are intended to provide stakeholders practical advice for securing IoT devices. The guidelines target four stakeholders: device manufacturers, IoT service providers, mobile application developers, and retailers [62]. We note here that end-users are not mentioned as a targeted stakeholder.

DCMS 1013-item dataset. This dataset (the Version 3 dataset from [59])

[1] These documents were introduced in Table 1.1 on page 4 and are described in greater detail early in Chapter 6.

[2] The DCMS 13 guidelines are also used in Australia [27].

contains individual IoT security advice items extracted from existing academic, industry, and government documents for manufacturers of IoT products [63]. Starting with 1052 advice items, we manually filtered the dataset to remove duplicate items (i.e., if one advice item was word-for-word identical to another, only one of the two were kept). After removing duplicates, 1013 items remained. The security advice (henceforth *items* or *advice items*) originated from 69 mostly informal documents, from 49 different organizations, and is positioned as security guidance rather than extremely detailed specification-level advice [63] (Section 3.2 elaborates on this). Organizations represented in the collection include major organizations such as the IoT Security Foundation [111], the European Union Agency for Network and Information Security (ENISA) [71], and the GSM Association (GSMA) [92]. Industrial security-focused organizations comprise the most highly-referenced sources in the collection; however, a number of government entities and businesses are included such as the US Senate [1], the US National Telecommunications and Information Administration (NTIA) [208], and Microsoft [151]. The preceding examples represent the breadth of organizations that have contributed advice to the IoT security space. We chose this dataset for our analysis as it represents, to our knowledge, the most comprehensive publicly available collection of IoT security advice.

DCMS mapping document. A second DCMS document, *Mapping of IoT Security Recommendations, Guidance and Standards to the UK's Code of Practice for Consumer IoT Security* [63] maps each advice item in the 1013-item dataset (1052 items before our pre-processing) to one of the 13 guidelines from the DCMS 13 guidelines document. While itself not analyzed, we mention this document here, as it is referenced in both the DCMS 13 guidelines document and the ETSI provisions document (next), and throughout this thesis.

ETSI provisions. The European Telecommunications Standards Institute (ETSI) has also published a document of "baseline requirements" for IoT security [68] that appears to be an evolution of the DCMS 13 guidelines document. It includes all 13 categories as major headers (most often with slightly modified wording), but elaborates on each with a set of requirements that fit the theme of that category, much like what is done in the DCMS mapping document. For example, Provision 5.1-2 from

the section titled *5.1 No universal default passwords* (comparable to the DCMS' *No default passwords* guideline header) states [68]:

> *Where pre-installed unique per device passwords are used, these shall be generated with a mechanism that reduces the risk of automated attacks against a class or type of device.*
>
> *EXAMPLE: Pre-installed passwords are sufficiently randomized.*
>
> *As a counter-example, passwords with incremental counters (such as "password1", "password2" and so on) are easily guessable. Further, using a password that is related in an obvious way to public information (sent over the air or within a network), such as MAC address or Wi-Fi SSID, can allow for password retrieval using automated means.*

The majority of the advice items' topics contained therein are represented in the large DCMS dataset we analyze in Chapter 4. We mention the ETSI document here for context, and analyze the advice items therein further in Chapter 6.

While this chapter deals primarily with terminology, we introduce these documents as examples of security advice that we also analyze in later chapters.

3.2 Defining 'Best Practice'

In this section we consider definitions for *best practice*, including our own definition taking into account the concepts of outcomes, actions, and *actionable* practices, and discuss related terms commonly appearing in the literature. Through this, we provide a refined, self-consistent vocabulary for security best practices and also disambiguate a wide variety of qualifying terms into three semantic categories.

3.2.1 Definition and Analysis

The definition of best practice is largely taken for granted. Few documents that use it make any effort to explicitly define it. Of note, even RFC 1818/BCP 1 [169], the first of the IETF RFCs specifying what a Best Current Practice document is, fails to define best practice. Thus, the term (and concept of) best practice is, at least in

security, almost always used casually, versus scientifically—the implicit assumption being that everyone understands what it means well enough to not require an explicit definition.

Some examples where best practices are used casually, and to express different meaning, include the following. In considering Cloud Security Providers (CSPs), Huang et al. [106] refer to: "*security mechanisms that have been implemented across a large portion of the CSP industry [are thus] considered standardized into a 'best-practice'.*" Here, best practice appears to mean *widely implemented*. In their evaluation of home-based IoT devices, Alrawi et al. [14] note numerous violations of security design principles, and assert "*Best practices and guidelines for the IoT components are readily available*", but offer neither citations for best practices among 108 references, nor their own definition. In a recent Canadian national news article [118] on banks disclaiming liability for customer losses from e-transfer fraud, and one-sided online banking agreements, a defensive bank representative is quoted: "*We regularly review our policies and procedures to ensure they align with best practices.*" This quote appears to be not about security, but rather legal best practices in the sense of *our agreements are no worse than our competitors'*.

A negative consequence of this is that different experts also implicitly redefine *best practice* to suit their own needs or context (examples are given above). This leads to ambiguity, where certain uses of best practice have different meanings and connotations, while elsewhere different phrases may imply the same concept. To address this, we first propose a definition for *practice* (separate from *best* practice) that we build on below:

A *practice* is a specific means intended to achieve a given desired outcome.

A practice specifies actions (explained on page 37) to reach an outcome, but does not necessarily imply any level of quality or security with respect to the means or mechanism used, or the outcome (discussed further in Section 3.2.5). Building on this definition, to reduce ambiguity and provide more precision to analyze security best practices, we propose the following practical definition for *best practice*:

> A *best practice* is a specific means intended to achieve a given desired outcome, and that is considered to be better than, or "at least as good", as the best of other widely-considered means to achieve that same outcome.[3]

Note that by our definition, a best practice is something that can be done (an action), not something that is desired to be achieved (an outcome).[4] A community in which a best practice is developed may have their own measure for quality, and quality requirements may vary based on the community, environment, and context. Further, as by our definition, best practices are specific to the outcome they aim to achieve, there is generally no "silver-bullet" best practice for use across all applications—practices typically must be tailored for the context [188, Chapter 2] (e.g., a surgeon has different hand-washing best practices than an individual preparing a family meal). Best practices may be intended for manufacturers, but for the benefit of end-users (other stakeholders are often involved). Stakeholders that benefit may or may not be involved in a best practice's implementation. For example, a user of a product might not care how a manufacturer implements a best practice, despite relying on it for security.

For comparison, we now review four notable definitions of best practice and compare them to our definition above. While focused more on human aspects of best practices, of specific note is King's discussion of security best practices, where they are defined as [124]:

> [Best practices are] *practices that have proven effective when used by one or more organizations and which, therefore, promise to be effective if adapted by other organizations.*

King's discussion covers several important concepts, including that effectiveness is based on evidence of multiple instances (implying some degree of consensus), that a practice must be applicable to real situations (not theoretical), and that it may exist among a set of practices of equal quality to carry out a security process [124].

[3] While one view of "best" might imply being above all known others, another is that "best" is a category that may have more than one member. It is thus reasonable to allow (by definition) multiple best practices for a given desired outcome (consistent with King [124]).

[4] We discuss outcomes and actions in greater depth on page 37.

McGraw [149] gives a view that best practices (in the context of development activities aiming to improve software security) are:

> *[...] usually described as those practices expounded by experts and adopted by practitioners.*

His view describes how best practices are created by a group of experts and often intended for use by non-experts. This reference also regularly refers to *touchpoints*, which are described as "*a set of software security best practices*" [149] (implying touchpoints *are* best practices). However, the description throughout that book indicates that these touchpoints are general categories of recommended processes and generic activities (e.g., code review or penetration testing [149]) rather than specific procedures that our definitions would recognize as actionable practices.

Garfinkel et al. [84] describe best practices (in the context of operating system and internet security) as:

> *[...] a series of recommendations, procedures, and policies that are generally accepted within the community of security practitioners to give organizations a reasonable level of overall security and risk mitigation at a reasonable cost.*

This definition emphasizes general agreement within a community about the quality of a practice (however difficult that agreement may be to attain) as a defining feature of a best practice. Notably, their description suggests that following a best practice is often a trade-off between cost and quality, meeting somewhere in the middle where both values are acceptable for an organization (we discuss the *feasibility* of a practice in Chapter 4).

If we consider the definitions by McGraw [149] and Garfinkel et al. [84] on a spectrum of how specific they envision best practices to be (the scope of practices themselves), McGraw's description is at one extreme (the coarse or general end), while Garfinkel et al.'s is at the opposite end (very specific). McGraw views best practices as being comprised of general categories of activities to follow at different stages of the software development lifecycle; Garfinkel views them as fine-grained, e.g., at the level of command-line arguments and exact details specific to a given OS version or specific system configuration commands.

Shostack and Stewart [188, pp.36–38] describe best practices as:

> *[...] activities that are supposed to represent collective wisdom within a field* [and] *designed to be vague enough to apply in the general case.*

Our definition agrees regarding "*collective wisdom within a field*", but we call for best practices to be more specific than vague. While not as general as McGraw [149], nor as specific as Garfinkel et al. [84], Shostack and Stewart's perspective of best practices fits somewhere between these two on the above spectrum.

While we use our own definition for best practice, we avoid saying that others are necessarily "wrong"—our definition is based on what we find useful to explain the concept of best practice, is informed by a general consensus of the goals of best practices (ultimately to improve security), and serves as a concrete, explicit reference definition throughout this thesis. Our definition, compared to the above four, builds on our explicit definition of what a practice is (i.e., a specific set of steps to reach a desired outcome) rather than a more general view of "good things to do" or an outcome to reach. Our definition builds on a common theme of the others: the acceptance of a practice (and resulting outcome) as being high-quality by experts of a community. We specifically require that a best practice is not a goal (outcome), but a method for achieving a goal.

It is often infeasible to specify a *universal* best practice, i.e., one that is widely applicable for every application. A determination of a universal best practice would require knowledge of not only every practice and some way to measure their relative quality and cost, but also a one-size-fits-all practice that addresses all circumstances. While it may be difficult to find such universal best practices, best practices are in our view (for typically narrower ranges of applications) born of a consensus, e.g., of the experts in a given field or community.

One can also consider the implications of best practices from legal, technical, and social angles. From a legal perspective, following a best practice may be used as an argument to escape or limit liability, as in "following the crowd" or consensus as surely being reasonable. For example, financial institutions citing "industry best practices" to disclaim liability, per our example on page 33 [118]. Technically, a best

practice is often the best way known to technical experts or researchers for achieving an outcome (supported by some form of consensus), or as a way to limit risk [84]. Less formally, best practice often implies the most common (if not necessarily best) way to do something. At one level, one might argue that each of these are similar, but at a semantic level, they are different uses of the same term.

3.2.2 Outcomes vs. Actions

We now define what we will mean by the terms *outcome* and *action*.

> An *outcome* is a desired end goal that a stakeholder aims to reach.

> An *action* is an activity involving one or more steps or specific methods carried out by a person or computer, typically (in our context) to achieve a desired outcome.

For example, an outcome may be having created *a strong password*, and an action to (partially) achieve this outcome may be to *enforce a minimum password length of 8 characters* [89].

In practice, outcomes or goals that are vague or broad may not give stakeholders a clear idea of any concrete set of actions that can be taken to achieve the goal. A desired outcome of "strong security", for example, is nebulous and cannot be mapped to specific actions to achieve the goal. Defining tightly-scoped outcomes or specifying an objective to withstand specific attacks allows for successful mapping to corresponding actions.

A given practice may be viewed as *actionable* if it can be carried out without guesswork by an advice target. We argue that being actionable is the crucial characteristic, the key point being to formulate advice such that the steps to be executed are explicit or well understood by targeted advice recipients. Our view of actionability as a desired advice characteristic is not based on consensus—we use it in this thesis as a concrete position to provide a baseline for discussing its potential benefit in establishing security advice, but acknowledge that this position lacks external confirmation. This leads to our next definition.

> By *actionable practice* we mean a practice that involves a known, unambiguous sequence of steps, whose means of execution are understood (by the target advice recipients).

While we use *actionable practice* here to emphasize that a practice must be one that a target subject can actually carry out, describing a practice as actionable is redundant, as all practices are necessarily actionable by our definition of a practice (page 33, i.e., a *specific means* to achieve a desired outcome). An outcome (alone) cannot be a best practice (or even a practice, and therefore not actionable), as an outcome alone does not typically or necessarily specify a specific means to an end. In what follows, when we use the term *practice*, we generally mean a practice that is actionable.

It follows that a recommendation specifying an outcome, but the path to which is an open research problem, cannot (and in our view should not) be considered a practice. Specifying "advice" that implies use of techniques that are experimental or unproven introduces ambiguity in how to carry out the advice and may result in inconsistent execution of the advice (which is not, by our definition, a practice). We argue that it is important for the security community—whether by academic, industrial, or government efforts—to identify and agree on practices with concrete desired outcomes for use by those targeted by the advice. Best practices adopted for specific use-cases will ideally lead to more reliable (correct) execution of the practices and thereby improve security.

Our heavy focus on (actionable) practices arises from our belief that, if stated clearly, they may be a promising, direct way to help pre-deployment stakeholders improve security. Improving security advice is separate from the issue of motivating advice recipients to actually follow the advice—actionable security advice does not improve the security of a device if it is not followed. Lack of actionable security advice is not necessarily the root cause of IoT device insecurity (or advice not being followed) [24, 127, 156, 174], but we find in this thesis that existing advice is largely non-actionable. In this way, we aim to contribute to improving security advice; motivating its use is beyond the scope of this thesis.

Using the page 38 discussion of *actionable*, upon examining the 13 DCMS guidelines (Table 3.1) we find only one has a detailed description in the original document [62] meeting our sense of the term actionable. It states [62]:

> *UK-1* ("*No default passwords*"): *All IoT device passwords shall be unique and not resettable to any universal factory default value.*

As a consequence, we expect that most are unlikely to be reliably implemented from this advice alone.

As another example, consider [62]:

> *UK-5* ("*Communicate securely*"): *Security-sensitive data, including any remote management and control, should be encrypted in transit, appropriate to the properties of the technology and usage. All keys should be managed securely. The use of open, peer-reviewed internet standards is strongly encouraged.*

This somewhat vague guideline is not, by our definition, actionable, as it is non-specific about which actions to take to follow it, and is unfocused on a single security topic (discussed further in Section 4.2.3).

Guidelines may result in the inference of implementation details based on the experience of the implementer, but this is not, by our reading, how these 13 guidelines are positioned. The associated DCMS mapping document [63][5] is intended to provide additional details and context for how the guidelines should be followed, but as we later find (Section 4.2), the advice in the mapping document is largely non-actionable. The 13 guidelines are further analyzed in Chapter 6.

Target audience of advice. Advice authors must understand who the intended target audience is (i.e., understand their level of knowledge and limitations) in order to create suitable advice. A sequence of steps described as *generally understood* (from our definition of *actionable* on page 38) implies that the target audience has the appropriate level of knowledge to follow the advice (independent of having sufficient resources, discussed below). Advice that is not understood or not specific enough to be implementable by the target audience becomes non-actionable to that

[5]Recall Table 1.1 on page 4.

audience, even if actionable to others. This means wording and outcomes must be understood from both a (semantic) language perspective and a technical perspective. Appropriate audience targeting is discussed in Chapter 6.

A practice need not necessarily specify a full sequence of low-level, specific, detailed steps; it may be sufficient to state high-level steps, provided they are actionable. For example, a practice involving the use of AES does not necessarily require a line-by-line implementation as specified by NIST [158]; similarly, it may suffice for a practice to state a library or function to use, how it should be used, and specify any desired configuration details. To use a non-security example, a car mechanic does not need to build a car's alternator from scratch, but they are expected to be able to follow a guide to install and configure a pre-assembled one. This again highlights the importance of an appropriate target audience selection—depending on the context, specification-level details are important for, e.g., those building libraries and toolkits (to use the AES example), while others may require only the details needed to properly use available libraries. Both situations can have practices developed for them, while reaching desired outcomes and being appropriate for their respective audiences.

Advice that simply mentions a general technique by name (without details) is non-actionable, by our definition. However, pointers to "next-level" implementation details (discussed further in Chapter 6) may meet our requirement of unambiguous steps (for actionability), e.g., with details in an external reference. In this way, advice may direct advice recipients to non-prescriptive techniques or approaches (e.g., key management techniques as in the UK-5 example above), but then link to further sources for specific details. This allows specifying how to carry out an advice item generally (how to approach a problem), while avoiding fine detail and lengthy descriptions, yet remaining actionable via links to detailed unambiguous steps. This avoids advice items that dictate an exhaustive number of individual steps, and avoids needing advice updates due to, e.g., parameter changes or algorithm upgrades; low-level details can be more frequently updated in external sources, without need to re-issue higher-level best practice advice (as it remains valid for longer periods).

While our definition of an actionable practice is designed to match what we expect

is practically followable, clearly indicating what advice recipients must do, in some cases recipients can infer how to execute advice even if it lacks details. For example, depending on a target's experience, what a "standard algorithm" is may be understood. While we retain our definition of a practice being actionable, we acknowledge that in some cases, some advice recipients have sufficient experience to infer actionable detail from otherwise non-actionable advice—making explicit step-by-step instructions unnecessary. Nonetheless, because security experts are not always the audience responsible for executing security advice (e.g., at an IoT device manufacturer), we encourage development of actionable practices (or providing actionable next-level details, as described in the previous paragraph) for specific targeted audiences.

Infeasible advice. We separate the concepts of a practice being actionable, and a target audience having the resources to execute the practice. A target audience must have the resources (technical, financial, personnel) available to carry out some practices, but we hereby clarify that the intention of our definition of a practice is that availability of resources does not affect the inherent actionability of a practice. (In other words: though a practice is actionable in general, that does not guarantee that a given party has the resources to carry out the practice.) A practice that involves a significant cost may be ruled out as a best practice by a recommending group, governing body, or peer community. Similarly, while still actionable by our definition, a practice that has (for example) 300 well-defined, unambiguous steps and takes 14 years to complete would rarely be considered a best practice. Such a practice would be considered *infeasible*, which we define as follows:

> An *infeasible practice* is a practice that remains actionable, but viewed by a non-negligible subset of advice recipients as impractically inefficient or excessively costly.

Note this is illustrated by our continuum of the actionability of practices (Fig. 4.2 on page 55). As will be defined in Chapter 4, an *Infeasible Practice* (*P3*) is actionable, but by fewer parties (as indicated by its placement toward the *actionable by fewer parties* labelled end of the continuum) than a practice requiring a *Security Expert*

(*P4*)—as practically speaking, high costs may reduce the number of parties able to implement a practice.

3.2.3 Imperative and Declarative Advice vs. Actions and Outcomes

By our definition (page 33), the statement of a best practice includes specifying a means to reach a desired outcome. We briefly consider now the utility of advice items that do not specify any such means or specific set of actions. As a particular case, consider an advice item that specifies an outcome whose attainment can be verified (but leaving it to an advice recipient to determine a specific means). Some advice recipients may still be able to attain the outcome, and auditors could verify attainment. Would such advice—which we call *declarative* advice, next paragraph—be equivalent to a best practice? Not by our best practice definition, which requires a specific means; by our definition, an outcome and a best practice are categorically different. Nonetheless, if the means used to reach the outcome is of secondary importance to an advice giver or authority, and their primary interest is attaining the outcome, then advice items in the form of (verifiable) declarative outcomes may be useful alternatives to (actionable) best practices—for advice recipients who can independently determine a means to reach the outcome. Having made this observation,[6] we proceed herein to use our (Section 3.2.1, page 33) definition of best practice.

As supporting context, we note that actions and outcomes can respectively be mapped to *imperative advice* (advice that includes specific steps or actions to reach an outcome), and *declarative advice* (advice that specifies an end result or outcome to reach, but not any specific method by which to reach it) [44, 75].

Depending on their nature, some outcomes may be verifiable, e.g., through a test that yields a yes/no answer to whether the goal was reached, or a measure used against a pass-fail threshold. For example, consider the advice item [193]:

> *Where a device or devices are capable of having their ownership transferred to a different owner, all the previous owner's Personal Information shall be removed from the device(s) and registered services.*

[6] We thank an anonymous referee for raising this question.

This could be verified, for example, by checking that any memory region designated for storing user personal information has zeros in every byte.

In contrast, an example of a non-verifiable advice item is *use a randomly generated salt with a minimum length of 32 bits for hashing with passwords* [89]. If we assume a verifier is only presented with the fixed-length output from a hashing function (i.e., h from $h = H(p, s)$, where p is a password and s is a salt value), this practice is not verifiable, as the output provides no indication of the salt's length or method of generation. For example, a password of "$password123$" and a salt of "$ABCD$" produces a SHA3-256 hash of "$e0aa...beef$" (truncated), which does not reveal any characteristics of the salt. While we use this as an example, in practice, salts are typically stored with the hash output—if a verifier were to recover one, they would then have access to the other.

3.2.4 Commonly-Used Qualifying Terms

A number of what we call *qualifying terms* are widely used as an adjective before the word practice (e.g., *common*, *good*, *best*) but without definition of the qualifying term itself. Being widely used might suggest that readers know (and are in universal agreement on) what authors mean when they use these terms. Like *best practice*, while security community members are apparently expected to have a general intuitive understanding of the meanings of these terms, consensus has not been reached on the meanings of these terms either.

For example, IETF BCP draft *Best Current Practices for Securing Internet of Things Devices* [153] contains advice that is arguably positioned in three different ways:

- as advice within a *best current practices* document (containing advice considered to be the best current practices);
- as *recommendations* (suggesting that use of the advice items is endorsed); and
- as *minimum requirements* (their use is a minimum expectation).

In an effort to both highlight existing terminology and move toward more consistent use of terminology, we associate these highlighted terms (among others) with one of

Table 3.2: Categories of commonly used qualifying terms related to best practices.

Category Focus		Qualifying Terms (examples)	Suggested Use
Quality	Über	"state-of-the-art" "gold standard"	For practices considered superior to all others, even if not widely adopted. These terms imply elite quality, possibly at high cost or complexity.
	Best	"best current practice" "best practice"	For practices *widely-considered* to be high quality (plus widely adopted, ideally).
	Good	"recommended practice" "suggested practice" "good practice"	For practices that are beneficial (e.g., to improve security), without implying that better practices do not exist. Here, "recommended" and "suggested" do not imply a formal endorsement.
Commonality		"minimum expectation" "baseline practice" "accepted practice" "common practice" "standard practice"	For practices not necessarily implying quality, but reflecting wide use. Alternatively, these may be de facto practices or functionality, informally recognized by experts as generally expected.
Stipulation		"regulation" "mandatory practice/requirement" "formal standard" "code of practice" "recommendation" "guideline"	For practices endorsed (formally) or mandated in some capacity by an organization or individual. Includes practices that may be, in some way, enforced by an entity such that there implies a negative consequence if the advice is not followed.

three distinct categories of qualifying terms, summarized in Table 3.2. These three categories can be used to characterize a given advice item: *quality* (page 45), *commonality* (page 46), and *stipulation* (page 47). Table 3.2 also suggests where/when each qualifying term should be used and gives examples.

While we primarily categorize terms by what we view as each term's dominant goal (i.e., identifying the quality of an advice item, how commonly an advice item is used, and acknowledging a governing authority's stipulation of the advice item), an advice item can share the characteristics of more than one category. For example, an advice item that is considered to be a *good practice* (quality category) can also be a *standard practice* (commonality category) through wide use, and a *best practice* (quality) can be included in a formal standard (stipulation).

Table 3.2 does not explicitly define the commonly-used qualifying terms contained therein; rather, it describes how we suggest each term (belonging to a category) be

most appropriately used. For example, here our Suggested Use for *best practice* expresses its relationship to being widely considered of high quality (albeit a higher quality tier exists), while our definition (page 32) explicitly notes that best practices are better than (or at least as good as) other high quality practices with wide consideration. In what follows, we discuss these terms in greater detail.

3.2.5 Category 1: Quality-based Terms

Quality-based terms provide a natural basis on which to differentiate practices. Conceptually, we order *über*, *best*, and *good* practices along a quality continuum. We note that terms used to describe practices of low quality (i.e., below good) receive less attention in literature as documents promoting security advice focus more on good than bad practices. Our definition of a good practice (the lowest quality we formally recognize) implies that anything lower does not improve security.

Über practices. The sub-category or group *Über* suggests practices that are in some way superior to best practices, or beyond what would be considered already high quality. *State-of-the-art* or *gold standard* implies something of elite technical quality, but perhaps not yet widely adopted. Consider as a practical example: in luxury cars, a heated steering wheel. While more comfortable on a cold winter day, best practice would likely be to ensure correct function and adequate steering grip to reduce the likelihood of accidents. A heating function may be the "gold standard" or "state-of-the-art" (typically at higher cost).

Best practices. The group *Best* suggests practices widely considered to be high quality, and often, widely adopted. While technically better practices may exist, best practices are widely accepted within a community to be high quality.

Good practices. The group *Good* suggests practices that improve security but are not necessarily the best practices available. They generally are not lauded for high quality per se. A good practice often either does not have wide acceptance as being the best, or is perhaps not widely practiced or not considered essential even if easy and beneficial (e.g., a good practice is to apply the emergency brake when parking facing down a hill, while a best practice is to both apply the emergency brake *and* turn the wheels to the curb). Further context may prove useful for understanding

their use. For example, access control to a low-value free online newspaper account may not require a best practice authentication method (per our definition); a good practice may suffice [84]. In other words [124]: "*sometimes the good is good enough*".

3.2.6 Category 2: Commonality-based Terms

Commonality-based terms also often include the word *practice* (*accepted practice*, *common practice*), but their unifying trait is frequency of use rather than quality.

Baseline practice/minimum expectation. These terms suggest a minimum level to be reached. We assign these to the commonality category, as it is expected that the minimum acceptable level of advice is commonly followed.

Common/standard/accepted practices. These terms reflect broad usage. For example, it may (unfortunately) be common to store passwords in plaintext within a database (thus being a common practice), but that is not best practice (or even a good practice).

We repeat that commonality does not necessarily imply quality. Terms in this category are less clearly ordered than in the quality category, and some terms are used interchangeably (e.g., *common/standard/accepted*). As *baseline* and *minimum expectation* both imply a lowest reasonable threshold to start from, these may be considered more of a priority to be followed, thus we order them higher in the group than the *common/standard/accepted* practices.

Correlated with commonality is the *maturity* of advice, typically reflecting the length of time that advice has been, or continues to be, given or known. To follow an earlier example, while not considered even a good practice, storing passwords in plaintext has become a mature practice [168]. Ideally, a best practice would be mature as well as widely considered to be high quality (Section 3.2.5), but greater maturity of an advice item does not always imply higher quality (e.g., DES is a mature cipher, but no longer best practice).

Security design principles. Security design principles are a known set of guiding rules which aim to improve security [182]. These principles are generally based on experience, suggesting their maturity. Security design principles are also generally expected (by experts) to be followed, and are complementary to the existing

categories, but we intentionally omit them from Table 3.2, and discuss them further in Section 4.1.3.

3.2.7 Category 3: Stipulation-based Terms

Distinct from quality and commonality, some terms related to best practices have more to do with the endorsement by an authority, the authority's jurisdiction, and whether the advice is mandatory (i.e., a firm requirement). Note that the entity creating advice is not necessarily the authority mandating its use. Our *stipulation* category contains qualifying terms describing advice that is mandated or endorsed by an entity in some way. These too can be ordered along a continuum. On the strict end are terms that imply a negative consequence for not following the advice (e.g., *mandatory practice*, *requirement*, *regulation*). On the looser end are terms that are stipulated, but not necessarily enforced (e.g., *guideline/guidance*, *recommendation*). As with a best practice, stipulations should, in our view, ideally be accompanied by an explanation of the intended outcome.

Regulation. We use *regulation* to mean a directive from an authority stating specific advice that must be followed to be allowed to operate within a *jurisdiction*. Here, a jurisdiction refers to the legal or authoritative domain, or the context of the deployment environment or use cases (e.g., home IoT may require different practices than IoT devices for government; physical locations, e.g., to meet certain requirements to be allowed to be sold in a country; or scope of technology, e.g., certain practices may be more appropriate for IoT devices rather than desktop computers).

Mandatory practices/requirements. Hereafter just "*requirements*", these do not necessarily imply the quality of a given practice, but rather that it is stipulated by some governing body or regulation, suggesting official endorsement. These may be considered "enough" for some purposes (e.g., *enough to not be sued* or *enough to pass inspection*). Practices across a range of qualities may be requirements depending on the governing body or motivation, although a practice established as high quality is more likely to become a requirement.

Formal standard. We take *formal standard* to mean a formally documented

(endorsed by some authority) specification. This typically (but not necessarily) implies acceptable quality; the main point is to officially specify details and recognize, e.g., a particular method or measurement. The purpose of a standard may be interoperability—e.g., standards for the gauge of rail tracks or pipes. In this context, formal standard differs from *standard practice* (i.e., common practice, above) and is not related to frequency of use, e.g., it is *standard* (practice) to eat at 12 noon. Standards are typically sufficiently detailed such that conformance or compliance can be judged by, e.g., an auditor, or interoperability tests [84].

Code of practice. We take *code of practice* to mean a set of guidelines designed to help inform others (traditionally within a profession) of expectations. They often pertain to ethical or safety issues. Codes of practice are often stipulated (within an organization or industry), but may be viewed as voluntary in that, e.g., failure to follow them typically does not result in major penalties unlike stricter terms (mandatory practices/requirements). In our use, a code of practice is distinct from formal regulations such as an "electrical code" or "building code".

Recommendations and guidelines. A *recommendation* is an endorsement of, e.g., a practice by an individual or organization as their suggested way to do something. Recommendations (depending on the recommending entity) may be subject to bias or be self-serving, and do not necessarily reflect expertise or universal consensus. Some recommendations, depending on their sources, are, in essence, requirements. Recommendations commonly suggest following a standard [63]. Similarly, a *guideline* or *guidance* is often given to promote a suggested way to achieve a goal (or as described by Garfinkel et al. [84], something that *should* be done). A guideline may be used in the spirit of a recommendation—offered as help, versus imposing rules.

3.3 Concluding Remarks

The basic concept of best practices is familiar to experts and non-experts alike. We discussed what we argue is (and return to in later chapters) an important characteristic for security advice: whether it is actionable. We offered uniform, consistent

terminology (Section 3.2) that characterizes and separates concepts. This allows systematic exploration (Chapter 4) that begins here with generic discussion and classification grounded through specific focus on consumer IoT devices. We argue that actionability of security advice should be a primary goal of advice-givers, in order that advice be executable by the target audience. Actionability plays a significant role in the following chapters.

As final thoughts on best practices and related terminology, we noted inconsistency in the use of common terms (Section 3.2) and the situations in which they are used, with the same terms used with different meanings in different situations. We argue that supporting consistent use of terms, and separating the concepts of quality, commonality, and stipulation provides a better foundation to discuss and analyze security advice within the community.

Chapter 4

Coding Tree and Analysis of 1013 Security Advice Items

In this chapter, we develop a methodology that we call the *security advice coding method* (hereafter *SAcoding method*, or informally the *coding tree*), for analyzing security advice, and we conduct an empirical study of the 1013-item DCMS collection of IoT security advice to determine how *actionable* (Chapter 3) the advice is.[1] The dataset reflects advice from government, industry, and academic sources. Application of the coding tree methodology to an advice item dataset (expressed in natural language, e.g., English) results in the assignment of labels to the advice items, characterizing the advice. The labels allow a measure of how actionable each item is for target audiences to follow. We also consider where in the IoT lifecycle each advice item would be most effectively applied; this exercise confirms the critical role predeployment IoT stakeholders can play in securing consumer IoT devices, and follows from IoT device manufacturers being the primary target of the advice datasets that we analyze.

We begin our analysis of the 1013-item IoT security dataset by using *iterative inductive coding* [148,202] to create codes by which to categorize the dataset's advice items. In an effort to improve reproducibility of coding results (i.e., independent coders reaching the same code for a given advice item), we create a novel coding tree for coding of security advice items. This tree is used to guide coders toward tags in an objective manner rather than manually applying codes directly to advice items. The inductive coding process is used to establish and iteratively refine item codes, and we iteratively refine the structure of the coding tree which uses the codes as its leaves (discussed in Section 4.1.1). We then used a methodology based on the coding tree to categorize the advice in the 1013-item dataset.

[1] Recall our position from page 37 that actionability is a key characteristic of security advice.

4.1 Security Advice Coding Tree Methodology and Development

Our methodology for the analysis of existing IoT security advice begins with iterative inductive coding. Our specific goal is to categorize (*code*) 1013 IoT security advice items[2] for further analysis and thereby to characterize the current state of IoT security advice. Such categorization is commonly done on, e.g., qualitative data from interview responses, to allow further analysis. The iterative inductive coding process begins with reviewing and understanding the dataset and developing codes (collectively, a *codebook*) to assign to dataset items, and iteratively refining the codes as new themes, relationships, and insights emerge from further review [202]. Inductive coding techniques are commonly used in computer science and security research in the analysis of qualitative data (e.g., [120, 127, 156]).

4.1.1 Establishing Analysis Tools

Establishing an initial codebook

To begin development of a codebook for inductive coding, a first *coder*[3] initially reviewed the 1013-item IoT security advice dataset (Section 3.1.2) to extract unrefined categories (*codes*) that characterize advice items. Discussion and preliminary test codings based on the extracted categories by the first and third coder (cf. footnote 3) resulted in the following coarse codebook (set of codes):

- *Practice*
- *Incompletely specified practice*
- *Outcome*
- *Security design principle*
- *Too vague to tell*

[2] This is not to be confused with the DCMS 13 guidelines introduced on page 4 and detailed on page 30.

[3] A *coder* is a researcher that conducts iterative inductive coding as described in this chapter. The first coder (C1) is the thesis author, the second coder (C2) is an additional coder, the third coder (C3) is the thesis author's research supervisor.

- *Out of scope*

After establishing this early codebook (with associated definitions[4]), test sets consisting of ten new, mechanically selected items from the dataset (to test across topics within the dataset; e.g., item numbers 100, 200, [...], 1000) were coded to determine agreement between the two coders. This consisted of each coder reading, interpreting, and assigning each item in the test set to a code (informally called a *tag*). This process was done a second time on a distinct test set of 10 items. From this process we refined the codebook by creating new codes for items that did not fit well into existing codes. Assigning an item directly to a code was subjective, resulting in low inter-coder agreement of between 30–40% for the first two test sets. This motivated the development of the coding tree (described next).

Establishing the coding tree

To reduce subjectivity, what we call a *coding tree* was built to more objectively guide coders towards codes based on a sequence of *yes/no* questions (final version is Fig. 4.1 on page 54; the final codebook is in Fig. 4.4 on page 56).[5] For a given advice item, starting at the top of the tree, each question progressively directs coders to a next question via branches down the tree, finally arriving at a leaf node (containing the resulting code). An additional coder[6] was used to assist in test codings and further refinement of the coding tree.

This method was iteratively refined through five test coding sets, where coders refined the codebook (the codes), questions, and organization of the tree. Improvement was measured based on inter-coder agreement after modification (i.e., addition/modification of codes and/or questions) of the tree. Over several iterations, coders discussed the results to resolve ambiguities and gaps in code definitions or questions in the coding tree, refining questions difficult to reliably answer and coming to an agreement on the codes [105], improving inter-coder agreement through a

[4]The final codebook definitions are given in Fig. 4.4 on page 56.

[5]To our knowledge, we are the first to build a coding *tree* that uses questions to guide coders toward tags for qualitative data; most coding is done using a codebook and iterating on the codes therein (e.g., [105, 120, 127, 156]).

[6]For consistency, we call this coder *C2* in the remainder of this chapter and Chapter 5.

relatively concise decision path. To code items that required more reflection, coders could optionally consult a further detailed annotation (see Fig. 4.5), which was created by the first coder during the refinement phase and used by coders during the full coding exercise.

Many advice items positioned as practices in the DCMS 1013-item dataset were explicit about technique or technical method to address security, but lacked actionable detail. For example, Item #50 in the 1013-item dataset [200] states:[7]

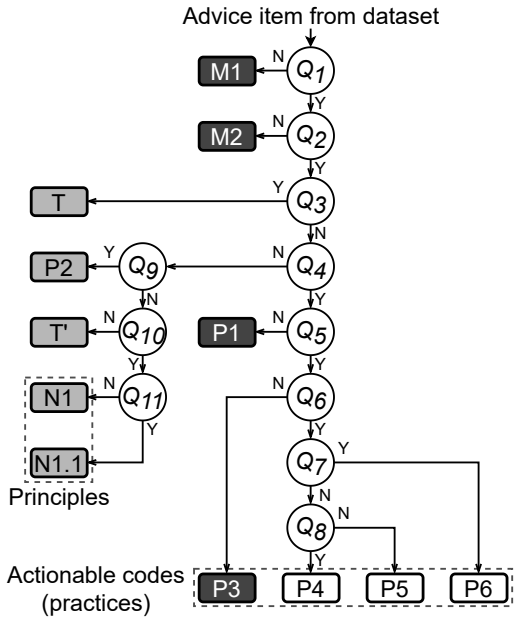
Endpoints must always use standard cryptographic algorithms.

This led us to develop the continuum of Fig. 4.2. On its left side, practices are less widely actionable (an incompletely specified practice, a general practice/policy). These often specify vague technical directions to take or methods to use ("*standard cryptographic algorithms*" in the example), but not explicit actions as typically needed to allow successful execution. On the right side are practices that even end-users would be able to carry out (*P6*), suggesting that if an end-user would be able to carry out the practice, so would a more experienced implementer. Moving left requires more in-depth knowledge and experience to understand (or infer a direction from) a practice, and implementation details become more ambiguous or unclear (even to a security expert). Coders did not use this continuum for coding, but we use it to visually represent where each category of practice may exist in relation to each other as a companion to the coding tree.

The binary *yes/no* decisions made by coders (through use of the coding tree) resulted in codes being assigned to advice items; as a side effect, by reaching one of the codes also in Fig. 4.2, advice items were indirectly placed onto the continuum. In contrast, where to directly (manually) place an advice item on the continuum (*P1–P6* in Fig. 4.2) may be less clear or might result in some point between two codes. For example, Item #90 states [193]:

Communications protocols should be latest versions with no publicly known vulnerabilities and/or appropriate for the product

[7]Numbered dataset is available in Appendix A.1.



Q_1. Is the item conveyed in unambiguous language, and relatively focused?
Q_2. Is it arguably helpful for security?
Q_3. Is it focused on a desired outcome more than how to achieve it?
Q_4. Does it suggest a security technique, mechanism, software tool, or specific rule?
Q_5. Does it describe or imply steps or explicit actions to take?
Q_6. Is it viable to accomplish with reasonable resources?
Q_7. Is it intended that the end-user carry out this item?
Q_8. Is it intended that a security expert carry out this item?
Q_9. Is it a general policy, general practice, or general procedure?
Q_{10}. Is it a broad approach or security property?
Q_{11}. Does it relate to a principle in the design stage?

Figure 4.1: Decision tree for assigning codes to advice items (coding tree). Leaf node codes explained in Fig. 4.4. Black shading (*M1*, *M2*, *P1*, *P3*) represents codes we consider not beneficial to include in advice (for lack of actionability or feasibility); white codes (*P4*, *P5*, *P6*) are desirable actionable practices (excluding infeasible *P3*); grey codes (T, T', *N1*, *N1.1*, *P2*) are considered useful for context, but not actionable.

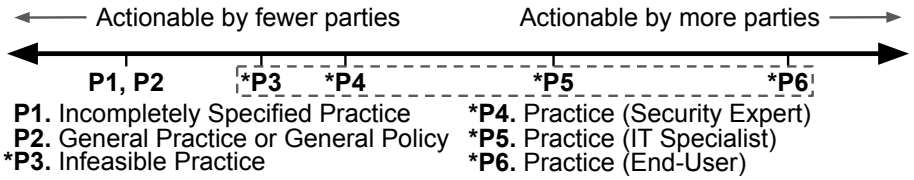


Figure 4.2: Practice categories: Actionability continuum. Terms defined in Fig. 4.4. An asterisk (*) indicates practice categories that we define as actionable.

This advice item might be (manually, i.e., directly) coded as either *P4* or *P5* depending on knowledge of the coder (a limitation, discussed in Chapter 5).

Making a second code available to coders

To further address reproducibility in use of the coding tree, a second code (for a given advice item) is optionally available to a coder. If the coder reaches a question that, based on their interpretation of the item, could be answered both as *yes* and *no*, this option allows (one time only, per advice item) both the *yes* and *no* edges in the coding tree to be followed to their respective leaf node code. As a result, two codes (*first* and *second* codes) would be assigned to the advice item. For example, Item #977 states [170]:

> *The RTOS makes use of secure storage to protect sensitive application data and secrets and additionally binds the data to a specific device instance.*

A coder may answer *no* to Question 5 if they believe this advice does not describe

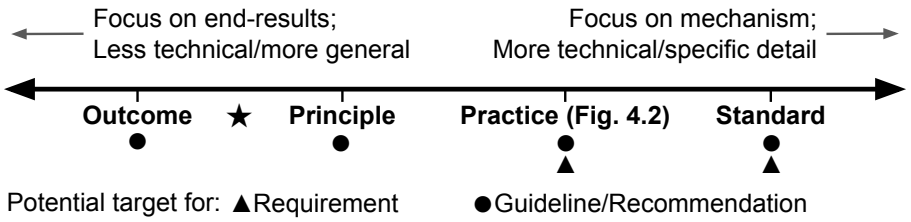


Figure 4.3: Relationship between terms based on inferred focus of advice's intent. The ⋆ denotes where we suggest policies [83] fit on this continuum; arguably, they might alternatively be located in parallel or just to the right of principles. We note that practices and policies may not always align—practices may be, e.g., delayed by policies through need for organization approval, or rejected due to incompatibility with policy goals.

M1. Not Useful (too vague/unclear or multiple items): Advice that does not make sense from a language perspective (e.g., not full sentence, unclear grammar), or is not focused on a specific task/action to complete.

M2. Beyond Scope of Security: Advice that is not clearly an item that would be executed for the benefit of security.

N1. Security Principle: Advice that suggests a generic (as in applying to many situations) rule to follow that has shown through experience to improve security outcomes or reduce security exposures. Discussed in Section 4.1.3.

N1.1. Security Design Principle: Advice that suggests a *Security Principle*, but specifically for the Design phase of the lifecycle (therefore a subset of *security principle*).

T, T′. Desired Outcome: Advice that suggests a generic, high-level end goal that a stakeholder would like to attain (as opposed to a means by which to reach a goal).

P1. Incompletely Specified Practice: Advice that suggests a technical direction of a practice (e.g., a technical method/technique, software tool, specific rule), but lacks clear indication of any steps to be taken, and fails to meet our definition of actionable.

P2. General Practice or General Policy: Advice that is not explicit about any techniques or tools, but is considered a general approach to improving security. This may also be policy-related advice. These are not considered actionable (despite being labeled as a practice) due to their general, less specific nature.

***P3. Infeasible Practice:** A practice, but one whose execution would require an unreasonable amount of resources (e.g., time, financial, human), or cost vastly more than what benefit would be gained.

***P4. Specific Practice—Security Expert:** A practice requiring an expert in security to execute. These may require in-depth knowledge and experience of security topics, and often rely on the advice recipient to infer steps that are not clearly defined in the advice.

***P5. Specific Practice—IT Specialist:** A practice that IT specialists (dedicated IT and development employees) developing or maintaining a product would be able to execute. These practices do not require the advice recipient to be a *security* expert, but assumes basic knowledge of computer security such as that gained through coursework in formal or informal education.

***P6. Specific Practice—End-User:** A practice an end-user would be able to execute. These are actionable by that audience, and typically executed by the user via direct interaction with the device, using a mobile app, or cloud service.

Figure 4.4: Codes and descriptions for coding tree of Fig. 4.1. As discussed on page 67, coders who reach *M1* for an advice item may also opt to chose a sub-code to denote the item as *Unfocused*. An asterisk (*) indicates practice categories that we define as actionable (matching Fig. 4.2).

Q_1. Does the advice make sense from a language perspective (e.g., it is a sentence that you can read and makes sense), unambiguous (i.e., you can tell what they are trying to convey from a language perspective, not technical), and not multiple items grouped into one piece of advice? Is the advice focused on one topic, whether it is a step to take, an outcome to achieve, or security principle? If the advice seems to have multiple topics being discussed or has multiple outcomes it wants an implementer to reach, this would be considered unfocused.

Q_2. Is the advice arguably useful for pursuing security in some way? Does it seem like it will help improve security outcomes rather than processes unrelated to security?

Q_3. Is the advice a high-level outcome rather than some method (or meta-outcome) for how to achieve an outcome? E.g., *data is secured in transit* would be an outcome because it is a desired goal or state, whereas *encrypt data in transit* is not because it explains a method for achieving that outcome (in this case, encryption). *Encryption* may be considered a meta-outcome, as it is not meaningful to the end-user's ultimate goal of protected data.

Q_4. Is the item a method used in achieving/following the advice? E.g., *encryption* or *replacing a password with black dots* are techniques/mechanisms, but *secure data* or *making the password unreadable* are not. An example of a specific rule: *no hard-coded credentials*—this is a rule that is fairly specific as to its goal and would be followed like a practice, but not necessarily with actionable steps.

Q_5. Does the advice suggest actionable technical steps (one or more) that suffice to follow the advice? It has sufficient detail to suggest a step/action to take.
Actionable: Involving a known, unambiguous sequence of steps, whose means of execution is generally understood.

Q_6. Could the advice item be followed with an acceptable cost?. E.g., the advice would not take years to follow, or have cost out of line with the anticipated benefit.

Q_7. Does the item suggest that the end-user will be responsible for carrying out this practice? Note that end-users first interact with devices after the Creation phase.

Q_8. Does following this advice require an expert understanding of security and security implementation in order to properly follow the advice? Someone following this advice item would have to be an expert in security to be able to understand it and successfully follow it, or be capable of extracting actionable steps from an otherwise non-actionable item based on their experience.

Q_9. Is the item a security policy (general rule) to improve security, but is not explicit about what technical means is used? These are less actionable (akin to incompletely specified practices—see definition in Q_5), and are not technically explicit. A general policy often has more emphasis on what is (dis)allowed (or may be a general rule closely related to a desired outcome), rather than on how to achieve it.

Q_{10}. Is the item a general way or general strategy, or a property that would improve security? A security property is a characteristic or attribute of a system related to security. E.g., an *open design*.

Q_{11}. Some principles relate to the core design phase of the product/system rather than later lifecycle phases.

Figure 4.5: Detailed annotation for the coding tree questions. Annotations were available to coders if they needed further details to answer a particular question regarding an advice item.

or imply actions to take (thus resulting in a code of *Incompletely Specified Practice*, *P1*), but if they could argue it does, they could answer *yes* to Question 5, leading them toward codes *P3–P6*. To encourage coders to be decisive, and to avoid introducing unnecessary complexity into analysis, only one additional tag was allowed, and permitted in the coding interface; as it turned out, neither coder tagging the full 1013-item dataset (C1 from this chapter, C2 from Chapter 5) expressed the need for more than two tags, so this design decision was not revisited.

As an argument could be made by a coder that a question could be answered *yes and no*, we made the decision to consider both codes as equals in the analyses of this chapter (and Chapter 5), i.e., neither code is considered more or less important than the other; both codes assigned to an advice item are counted in the results (Section 4.2). For calculating agreement in trial codings, in cases where two coders availed themselves to a second coding on the same item, if coders agreed on at least one code, it was counted as an agreement for that item.

Determining test set inter-coder agreement

For the development of the coding tree, i.e., during test codings, as opposed to the two-coder full coding of the 1013-item dataset described in Chapter 5, by convention, we considered agreement between two test coders if:

- their coding resulted in agreement on at least one code (one coder's *first* or *second* code matches either the *first* or *second* code of the other); OR
- both coders' decisions resulted in an item coded into any code category from *P1* to *P6* per Fig. 4.2; AND those two codes were within plus-or-minus one code distance away on the Fig. 4.2 continuum.

For example, if one coding was *Infeasible Practice* (*P3*) and the other *Specific Practice—Security Expert* (*P4*, one position right of *P3*), we declared this an agreement on the basis that their proximity on the continuum implies equivalence, taking into account the subjective nature of coders answering the decision questions. We refer to this as the "plus-or-minus one rule" ("±1"). Fig. 4.3 relates practices in our

continuum (Fig. 4.2) with other concepts.[8]

Final test set coding and inter-coder agreement

A final test coding was done with a set of 20 items. Based on first/second codes (page 55) and ±1 rule (page 58), the mean agreement rate between the three coders was 73%. The following are the proportion of the 20 items that were agreements between each coder (C1–C3),[9] and their Cohen's Kappa (κ) measure of inter-coder reliability [148]:

1. C1 and C2: 75% agreement, $\kappa = 0.67$
2. C1 and C3: 80%, $\kappa = 0.74$
3. C2 and C3: 65%, $\kappa = 0.59$

Particularly between C1 and C3 (item 1 above), we view 80% as a good level of agreement, and the inter-coder reliability score of $\kappa = 0.74$ is "substantial", according to a scale by Landis and Koch [131].[10] Between C1 and C2, we consider 75% agreement to be high; however, the reliability score drops significantly to $\kappa = 0.67$. Between C2 and C3, we are disappointed with the level of agreement, and the reliability score $\kappa = 0.59$ is also disappointing ("moderate" on Landis and Koch's Kappa statistic scale [131]). Chapter 5 investigates reasons for coder non-agreement in detail.

After the final test coding by the three coders, a detailed technical analysis and full coding of the 1013 item dataset was done by the first coder using the coding tree of Fig. 4.1. *The coding exercise was facilitated by a software tool*[11] that we developed to ease coding and record results (including first and second code, where present, and to also collect other notes such as the lifecycle stage, page 61; and coder comments/notes to later revisit). As the advice items in the dataset are grouped by the 13 guidelines in the DCMS mapping document [63], all advice items in the

[8] This is discussed further in Section 4.1.3.

[9] Recall from page 51 that Coder 1 (C1) is the thesis author, C2 is the additional coder, and C3 is the thesis author's research supervisor.

[10] Landis and Koch [131] regard their scale as "clearly arbitrary". Nonetheless, as it is commonly used by others, we use it here as a benchmark for comparing our coder agreements.

[11] Information about the coding tree interface tool's availability is discussed in App. A.1.

coding of the full 1013 item set were randomly ordered to avoid bias from reading similar advice in repetition.

In contrast to typical inductive coding exercises where a code is manually assigned to an item by a coder, when we say an advice item is "coded" by a coder, we mean they used the coding tree, and the associated methodology assigned the resulting code. The coding interface tool that we built displayed the coding tree (Fig. 4.1), code definitions (Fig. 4.4), detailed annotations for each question (Fig. 4.5), and two drop-down boxes where coders were asked to input the codes delivered through use of the coding tree.

Coding tree methodology summary

In summary, an iterative inductive coding methodology was used both to derive codes and build the coding tree, and the coding tree was used to code the 1013 advice items (Section 4.2). While the coders were asked to follow the coding tree down to the leaf nodes and then enter the code delivered into the drop-down boxes, in our software implementation there were no restrictions to stop coders from immediately selecting a code based on their first reading of the advice item. This limitation is discussed in Chapter 5.

We note that many of the terms discussed in Chapter 3 (e.g., categories from Section 3.2; the quality, commonality, and stipulation categories) are not represented in our codes. Where no extra context is provided about an individual advice item, and we use only the text of the advice (as was our case with analysis herein), it is difficult to know if an item belongs to any of these categories. For example, determining the quality of an advice item would require knowledge of how a community rates a practice, to know its commonality among practitioners requires knowledge of how frequently that advice is used, and to know if it is stipulated requires knowledge of how that practice is mandated in possibly widely varying real world environments. As such, terms like *best practice*, *common practice*, or *regulation* are not used in our codes (Fig. 4.4). We have instead used codes that can be applied to advice items without requiring (unavailable) contextual information.

A decision was made to use a single coder for the 1013-item coding exercise described in this chapter,[12] which was based on all test coders reaching a consensus on the final set of codes and questions in the coding tree (consistent with the methodology of, e.g., Huaman et al. [105]), the acceptably high level of agreement during test codings, and the work effort required to manually code (via the coding tree) 1013 items. Using a single coder is noted in Chapter 5 as a limitation of this chapter's work; however, Chapter 5 extends this chapter's analysis to include a second coder, aligning more closely with the coding methodologies of, e.g., Krombholz et al. [127], Kang et al. [120], and Naiakshina et al. [156],[13] and pursues detailed explanations of the larger deviations found between the coding results of coder C1 and coder C2.

4.1.2 Advice Categorization by Lifecycle Phase

Separate from the inductive coding of Section 4.1.1, the first coder assigned each actionable item (*P3–P6*) to a stage in the IoT lifecycle (Fig. 3.1 on page 29) where the item could be best carried out (in the subjective opinion of the coder), independent of the codes defined and used in the coding tree. This was done by determining which stakeholder would be in a position (in our view) to carry out the item, and matching where this would appear to best occur in the lifecycle. This determination was based on which stakeholders *could reasonably* execute a practice (within reason—an end-user given an API would not be likely to implement a best practice or fix a vulnerability), not necessarily the single stakeholder in the *best/most effective position* to implement it, thus allowing for items to be associated with multiple stages. For example, Item #191 [92] states:

> *When a product is being developed it is often enabled with debugging and testing technologies to facilitate the engineering process. This is entirely normal. However, when a device is ready for production deployment, these technologies should be stripped from the production environment prior to the definition of the Approved Configuration.*

[12] At the time we conducted this chapter's research, the second coder (C2 from Section 4.1.1) had not yet completed the tagging of the full 1013-item dataset. Chapter 5 reports on the second coder's full coding results.

[13] The coding methodologies of these studies are compared with ours in Chapter 5's Section 5.4.

This item could either be executed in the OS/App Development stage (1.2b) where code is stripped from software before completion, or during the Integration & Pre-Configuration (1.3) stage where features may be disabled or left out of device integration. We considered only practices (being implicitly actionable) for this categorization, as without actionability, it is difficult to determine what steps would need to be taken and when (in the lifecycle) they would be executed.

While an indication of where the advice item would be carried out in the lifecycle was included by many items in the advice statement itself (e.g., do not hard-code secret access codes for testing/debugging in software [113]), others required subjective judgement for placement (e.g., "*keep software updated*" [67] could be targeting the Creation phase or Usage phase depending on which stakeholder it implies should maintain software). If the item did not have an obvious or implied associated lifecycle phase, we categorized it as *Unclear* (see Fig. 4.7 on page 69).

4.1.3 Relationship to Security Principles

We observed that many security advice items were rephrasings of established security principles. In our context of computer security, we define a *principle* to be a generic (applying to many situations) rule shown through experience to improve security outcomes or reduce exposures, and a *design principle* to be a subset specifically guiding the *design* of a system. Other subsets may relate to other lifecycle phases. For context, note Saltzer and Schroeder [182] define eight "*[...] useful principles that can guide the design and contribute to an implementation without security flaws*". NIST [197] notes "*the primary focus of these principles is the implementation of technical controls*",[14] suggesting that security principles are appropriate targets to be implemented via practices. In our coding tree, both security principles and (more specifically) security design principles are leaf nodes.

Fig. 4.3 on page 55 conveys our view of the relationship between security principles, practices, outcomes, and other terms. On one extreme (left) are concepts more focused on end-result (outcomes); on the other are the most actionable items that

[14]Examples of these principles (from the quote) are "*protect against all likely classes of 'attacks'*", "*use unique identities to ensure accountability*", and "*limit or contain vulnerabilities*" [197].

focus on mechanisms to reach outcomes, often specified in fine detail (standards). As Fig. 4.3 indicates, some items on this continuum may serve as a guideline or requirement (Table 3.2 on page 44). Ideally, in our view, what is imposed as requirements by governing bodies should be practices or standards (versus principles or outcomes), as requirements should be actionable so those subject to them have a clear understanding of how to follow them (cf. Section 3.2.2 for *actionable*). This is represented on this continuum by the labeling of *Practice* and *Standard* as potential targets for requirements (denoted by the triangle).

4.1.4 Actual Use of Security Advice Coding Tree Methodology

We envision the methodology described in Section 4.1 to be used primarily in two ways. The first is for measuring the actionability of existing advice as a means to establish a general view of the current state of IoT security advice, and to determine where advice fails to meet the needs of security practitioners. This is the primary focus of the coding exercise described in Section 4.1, and analyzed in Section 4.2.

The methodology can be used in a second way for the analysis of new IoT security advice, as a tool to assist advice authors in creating actionable advice. If advice authors themselves use the coding tree on their own advice items, they can differentiate actionable from non-actionable advice (among other more fine-grained characteristics of advice). After using the coding tree, security advice items that analysis tags with an undesirable code can then be revisited by advice authors to revise, reword, and clarify the explanation of the advice.

For advice dataset creators to gain maximum benefit from the coding tree methodology, they would ideally have developed their datasets based on a predefined set of target categories/codes, as the coding tree reveals categories of advice within a set, but cannot itself determine which codes are intended or desired by an advice dataset creator. We view the creation of actionable practices as the preferred objective for advice authors; however, creating practices might not be the intention of all advice authors—some may instead intentionally craft non-actionable guidance in the form of Outcomes (*T*/*T′*), General Practices or General Policies (*P2*), or Security Principles (*N1*/*N1.1*)—Fig. 4.4 gives descriptions of these forms of advice. Advice authors

may use the coding tree to steer their advice toward non-actionable codes, but we recommend it be used in the pursuit of (feasible) actionable codes (*P4–P6*).

Once an advice item is revised, the questions in the coding tree may yield different answers, giving advice authors feedback about whether their changes have had a positive impact on the actionability of their advice, or if it follows a path down the tree to a more desirable code. If the coding tree outputs an undesirable code (e.g., non-actionable), those giving the advice may be able to observe (from the coding tree) at which question the advice diverged from a path to a desired code.

For example: "*encrypt stored passwords*" gives vague advice and is ambiguous on how to achieve encryption. Question 5 (from the coding tree, page 54) sends this advice to *Incompletely Specified Practice* (*P1*), as it lacks actions (explicit or implicit) to take. The advice item could be reworded to specify a particular encryption algorithm and mode. An accompanying document or note could also provide explicit references to aid implementation, thus now passing *Q5* as actionable.

4.2 Empirical Analysis of IoT Security Advice Dataset

In this section we carry out our (single coder) analysis of the 1013-item IoT security advice dataset, which for context, is the dataset of security advice from which the DCMS created their 13 guidelines [62]. We coded the items in this collection using the methodology of Section 4.1. The primary goal of assigning each item to codes (and associated definitions) is to provide a general sense of how well existing advice dataset lists and literature specify practices (as opposed to advice *positioned* as practices, but failing to be actionable, as required by our definition). Identifying where practices are carried out throughout the IoT lifecycle allows us to see which stakeholders are in the best position, or have the greatest number of items to address regarding contributing to overall device security.

4.2.1 Results of Coding

Fig. 4.6 summarizes the distribution of codes given to all advice items in the DCMS 1013-item dataset, as coded by the first coder (C1). For distinguishing actionable versus non-actionable advice (bottom of figure), if an item had two codes (first and

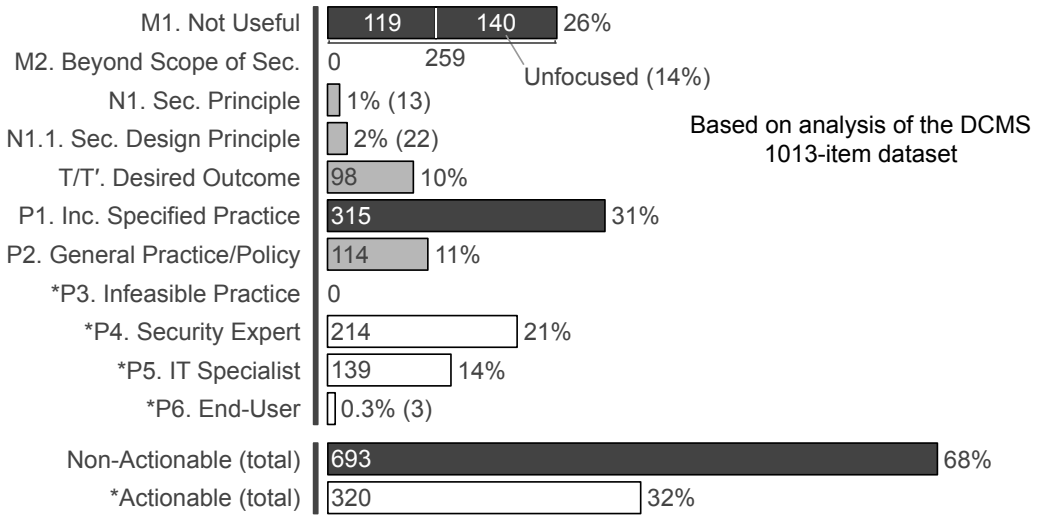


Figure 4.6: Main summary of advice coding. Sum in top portion of graph exceeds 1013 as items may be assigned two codes (first/second—recall Section 4.1.1). Shading intensity follows the same scheme as Fig. 4.1 on p. 54. See p. 67 for discussion of left portion of *M1* bar.

second) and at least one was an actionable code (i.e., *P3–P6*), we declared the item actionable based on the reasoning that an argument could be made for its actionability. For example, if an item was coded both as an *Incompletely Specified Practice* (*P1*) and *Specific Practice—Security Expert* (*P4*), we declared the item actionable as *P4* is defined to be actionable. As such, the sum of actionable and non-actionable items in Fig. 4.6 adds to 1013.

The coding tree's software interface allowed a coder to designate whether an item was specific to IoT. None of the items in the dataset were designated in this way. This suggests our methodology is broadly applicable. Following this finding, we intentionally left the resulting coding tree generic (non IoT-specific). While herein, we used the coding tree to explore IoT security, we believe that its design and resulting structure apply more widely to analysis of security advice in general.

4.2.2 Proportion of Non-Actionable Advice

Our analysis shows that organizations—often highly credible ones—are producing recommendations for manufacturers that are not, by our definitions and analysis, actionable, thus we believe at greater risk of being improperly (or not at all) executed

or implemented. This low proportion of actionable practices (of the 1013-item set) is, in our view, a signal that the security community must significantly improve how we capture and state "best practices" if manufacturers are expected to follow recommendations.

The methodology used declares any code after the *yes* branch of Question 5 in the coding tree (*P3–P6*, see Fig. 4.1) to meet the definition of actionable (Section 3.2.2). In total, 32% (320/1013) of advice items were found to be actionable (at least one actionable code as explained in Section 4.1.1; see Fig. 4.6):[15]

- 21% (214/1013) of items were coded as *Specific Practice—Security Expert* (*P4*);
- 14% (139/1013) as *Specific Practice—IT Specialist* (*P5*); and
- <1% (3/1013) as *Specific Practice—End-User* (*P6*).

The *Infeasible Practice* (*P3*) code went unused; this was encouraging, suggesting that advice providers have an understanding of what sorts of practices are feasible (in both resources and knowledge) for their target audience. Similarly, the code *Beyond the Scope of Security* (*M2*) was also unused; however, this is arguably due to source documents being generally targeted at computer security.

As Fig. 4.6 notes, 68% of the advice was declared to be non-actionable. We would expect that this significant majority of advice items will either be poorly implemented despite the advice recipients' best effort to understand the advice, or not at all; both cases representing a failure. This is not to say that we believe non-actionable advice is useless—outcomes, principles, and general practices (Fig. 4.4) still specify desirable end-results (outcomes) and generic goals. Actionability may not be essential in all use cases (cf. page 63); however, our underlying assumption is that advice givers (for the advice datasets under discussion) intend to be offering advice positioned as best practices. We argue that actionability should be considered a high (if not the highest) priority among the characteristics of such security advice (see Section 3.2.2).

[15]Percentages in this list sum to more than 32% (proportion of actionable advice) due to the use of a second code in some cases.

4.2.3 'Not Useful' Advice

During the iterative development of our coding tree methodology, we observed many advice items in the DCMS 1013-item dataset (described in Section 3.1.2) tended to not be actions to take, but descriptions of security techniques (e.g., a hardware security module, public-key encryption) or threats to a system (e.g., unused but accessible network ports), and offered no suggestion for any action to take or execute. Item #387 [200] provides an example of this:

> *Network firewalls are message-oriented filtering gateways used extensively to segment IIoT [industrial IoT] systems. Most firewalls are Layer 2, 3 or 4 IP routers/message forwarders with sophisticated message filters. Firewalls may be deployed as either physical or virtual network devices. A firewall's filtering function examines every message received by the firewall. If the filter determines that the message agrees with the firewall's configured traffic policy, the message is passed to the firewall's router component to be forwarded.*

One could make the argument that the description of a technique implies that the advice giver wants a follower to use the technique, but the italic text block above reads quite different from "do this" security advice and is lacking in actionable detail. As such, we consider advice of this nature to be not sufficient for a stakeholder to execute. 26% (259/1013) of items were coded as *Not Useful* (*M1*).[16] Note that *M1* is also used for advice items that are judged to "not make sense" from a grammar or language perspective.

Similarly, individual advice "sub-items" are commonly given in rapid succession within a single advice item (which may take the form of several sentences or a paragraph). As a sub-category of the *Not Useful* (*M1*) code, we added an *Unfocused* supplementary code for coders to use when they find multiple sub-items within one item (represented as the left sub-bar of the *Not Useful* code in Fig. 4.6 on page 65). For example, Item #84 [48] is in our view, an example of this:

> *IoT Devices Should Follow Security & Cryptography Best Practices. [1] BITAG*

[16] See Fig. 4.6 and *M1* description on page 56 (Fig. 4.4).

recommends that IoT device manufacturers secure communications using Transport Layer Security (TLS) or Lightweight Cryptography (LWC). Some devices can perform symmetric key encryption in near-real time. In addition, Lightweight Cryptography (LWC) provides additional options for securing traffic to and from resource constrained devices. [2] If devices rely on a public key infrastructure (PKI), then an authorized entity must be able to revoke certificates when they become compromised, as web browsers and PC operating systems do. Cloud services can strengthen the integrity of certificates issued by certificate authorities through, for example, participating in Certificate Transparency. [3] Finally, manufacturers should take care to avoid encryption methods, protocols, and key sizes with known weaknesses. [4] Vendors who rely on cloud-hosted support for IoT devices should configure their servers to follow best practices, such as configuring the TLS implementation to only accept the latest TLS protocol versions.

This advice item jumps across four topics (we inserted the numbers for exposition): (1) the use of TLS or lightweight cryptography, (2) certificate revocation, (3) avoiding weak or vulnerable key sizes, and (4) avoiding outdated TLS versions. Trying to successfully code advice such as this (i.e., as a coder) was a challenge, as different sub-items could be coded differently.

We found that advice items with a longer word length were often unfocused in this way. In total, 54% (140/259) of the items that were tagged with *Not Useful* (*M1*) codes, or 13.8% (140/1013) of all items were coded as *Unfocused* implying that in the judgement of the coder (thesis author) they contained multiple distinct topics within the advice item (similar to the above example). Had we extracted each sub-item from the original dataset (making them more narrowly focused, but as a result, potentially removing them from surrounding context), these may (most likely would; Chapter 6 examines this issue) have been coded differently. This is a limitation of this work (discussed in Chapter 5).

4.2.4 Associating Advice Items with IoT Lifecycle Stages

Fig. 4.7 shows the results of the thesis author's manual association of each of the 320 actionable practices (Fig. 4.6 on page 65) with one or more of the lifecycle stages.

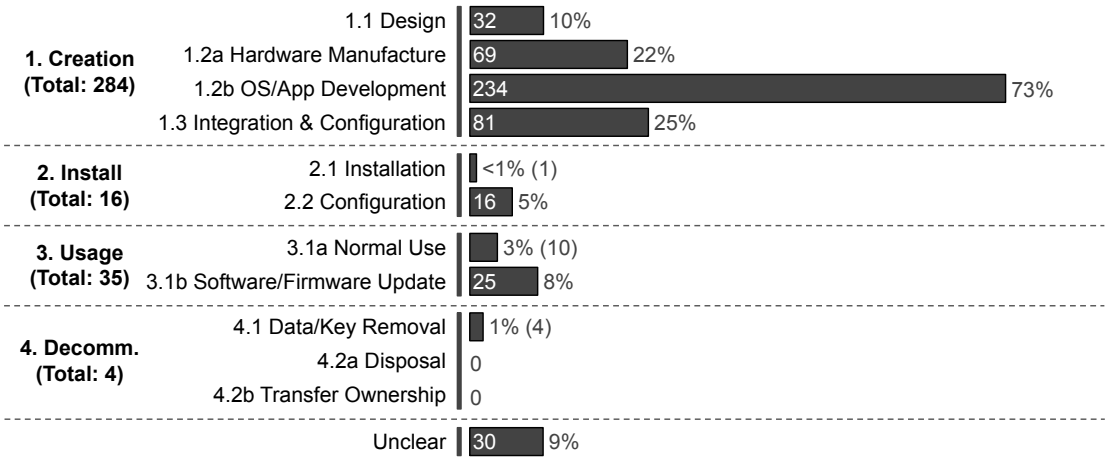


Figure 4.7: Number of actionable practices that we declared as suitable to implement at each lifecycle phase (Fig. 3.1). Total of all numbers and percentages exceed actionable practice total (320), phase totals, and 100% as practices may be suitable to implement in multiple stages. Percentages are proportion of 320 actionable practices.

All lifecycle phases associated with each practice (Section 4.1.2) were combined to yield the count shown for each bar. For example, if an item was coded as taking place in both the OS/App Development (1.2b) and Design (1.1) stages, each of these counts were increased by one (1).

The practice distribution among phases reveals important information about the overall execution of best practices: 89% (284/320) of practices that were deemed actionable could be implemented in at least one lifecycle phase within the manufacturer's control (i.e., the Creation phase), i.e., the designers and manufacturers are in a position to implement them. As a subset of this, the OS/app developers alone (Phase 1.2b, Fig. 3.1) are in a position to implement 73% (234/320) of practices. This follows from many advice items being software-related, thus suitable for implementation by one or more of several stakeholders involved in software development, before the product is in end-user hands.

While this finding may seem self-evident, it draws focus to the importance of attention to security during the product (device) Creation phase, and in particular, the importance of IoT security advice being implementable (and understandable) by IoT device manufacturers and their software development partners.

4.3 Related Work

Explicit formal definitions for the term *best practice* are rare in the security literature. Literature about the nature and definition of security best practices (as opposed to examples of best practices) is discussed in Section 3.2. Beyond this mention here of defining *best practice*, in this section we discuss related work on establishing security practices for IoT (as opposed to the characterization of what makes advice useful, as done in Chapter 6).

Tschofenig and Baccelli [204] discuss efforts by The European Union Agency for Cybersecurity (ENISA) and the IETF to provide recommendations and specifications on IoT security. They categorize technical and organizational areas to be considered for the secure development and use of IoT devices. Moore et al. [153] pursue specific best practices for IoT, specifically regarding network-based attacks. Based on our definitions herein, most of their advice items are not actionable (thereby not what we consider to be practices). Dingman et al. [66] examined six sets of IoT security advice and looked to determine whether three large-scale security events may have been averted if their collected security advice was followed; our work in this thesis focuses on the categories of advice (as captured by the codes, and how they relate to actionability) rather than the technical content contained in advice.

A number of government and industrial agencies provide security advice for IoT, for both manufacturers and groups looking to acquire IoT devices for their organizations. ENISA [71] published an expansive document about IoT security. This includes a substantial set of security recommendations, but also useful contextual and informative sections including (to single out a select few) the document's target audience (cf. Section 3.2.2), an overview of what IoT is and the relevant components, threat and risk analyses, and technical measures for executing the advice (these measures appear to be positioned as technical steps to complement other security advice).

ETSI [68] (cf. Section 3.1.2) provides a series of baseline requirements for IoT security. These requirements use the 13 DCMS guidelines [62] as general topic headers (adding a new one of their own), but provide more detail about technical steps to be taken. These baseline requirements are discussed in depth in Chapter 6. To supplement the baseline requirements document, ETSI provides a document describing how

to confirm conformance with the advice therein, noting that the advice in the support document is independent of an assurance scheme [69]. Assurance is historically associated with products for governmental use [43, Chapters 18–21] [88, Chapter 13], but is typically considered too expensive or otherwise unsuitable to the consumer space. For DCMS documents [59, 62, 63] used in this thesis, see Section 3.1.2.

NIST published three documents surrounding the interaction between US federal government agencies and IoT manufacturers [159]. Two of these NIST documents [72, 73] aim to assist IoT manufacturers to produce secure devices specifically for use in the US federal government by offering technical and non-technical baseline guidance. One of the NIST documents [74] is intended to help government agencies learn what features or characteristics they should seek when procuring IoT devices.

RFC 8576 [82] proposes a generic lifecycle model of an IoT device, presented as a simplified model. Other descriptions of lifecycles may include the key functional components that describe its primary function (versus the entirety of its life), e.g., Alrawi et al.'s IoT malware lifecycle components [13]. NIST's SP 800-27 [197] outlines five major general computer and IT system lifecycle phases and suggests 33 IT security principles. While developed independently, our lifecycle of Section 3.1.1 unsurprisingly has similarities, e.g., design/development, primary usage, and disposal/end-of-life phases. The NIST SP suggests that many individual principles are vital to positive security outcomes across multiple phases, implying there are important principles for phases other than the design phase. NIST SP 800-160 [178] outlines a taxonomy of 32 security design principles covering three areas of systems security: *security architecture and design*, *security capability and intrinsic behaviors*, and *lifecycle security*; the latter two are not specific to the design phase.

Morgner et al. [154] explore the relationship between efforts in formal IoT technical standards and the (unfortunate) reality of the economics of IoT security and its implications for the general security of manufacturers.

4.4 Concluding Remarks

The main contribution of this chapter is the development of the coding tree methodology and its use to analyze a large collection of IoT security advice. In particular,

we examined how actionable current advice[17] is, and what advice characteristics (i.e., corresponding to the codes of Fig. 4.4) emerge from this dataset. Our main focus has been the DCMS 1013-item IoT security advice dataset, which itself originates from other organizations.

For our analysis, iterative inductive coding was used to create a codebook that represents the characteristics of security advice (e.g., whether they are objectives to reach or practices to follow). To more objectively assign a code to each item in the dataset, we designed a coding tree. We suggest that IoT security advice-giving organizations consider using the coding tree and methodology of Section 4.1 to measure whether potential advice is actionable, and take steps to improve the advice's actionability, unless their explicit goal is to target, e.g., security principles or outcomes to reach (Fig. 4.4).

From our analysis of 1013 advice items from industrial, governmental, and academic sources (Section 4.2), we were surprised to find that the majority of advice items are not actionable practices that can be followed, but rather, what we deem to be non-actionable advice. Among the practices we identified as actionable, 73% are suitable to implement in the OS/App Development lifecycle phase of an IoT device (Section 4.2.4), thus by the product manufacturer and its software development partners. It is generally recognized that poor security practices early in the lifecycle accrue what we might call a *security debt*, with negative consequences in later phases (analogous to *tech debt* where technical shortcuts during development incur later costs [128]); from this, our results in this chapter highlight the fundamental role of pre-deployment stakeholders to underpin security for aspects that they alone are in a position to control.

As the Internet of Computers has grown into the Internet of Things, an old problem remains: how to ensure that security best practices are followed. An open question is whether the research community can find ways to help advice-givers (including governments) to compile more effective guidance, and have manufacturers embrace and execute advice given. Our work argues that currently, even in the security and technology communities (not to mention the general public), ambiguity

[17] We use the DCMS 1013-item dataset as representative of current IoT security advice.

surrounds the language of technical *best practices*—such that arguably, the term does as much harm to the security community as good. One hypothetical path forward (to provoke thoughts, more than as a practical suggestion) is to seek agreement within the technical security community that the term itself is vague and nebulous, and its use should be boycotted. Another path forward is to work towards consensus on definitions (as we pursue in Chapter 3).

We suggest that organizations proposing and endorsing "best practice" advice have a clear idea of whether they are recommending practices, specifying baseline security requirements, or simply offering advice about good principles to think about. If the goal is that relevant stakeholders adopt and implement specific practices aiming to reduce security exposures, we believe it is imperative that (actionable) best practices be identified and clearly stated, versus vague outcomes—lest the target stakeholders be unable to map advice to a concrete practice, even if so motivated. In summary: if security experts do not find guidelines clearly actionable, we should not expect (security non-expert) manufacturers to magically find a way to adopt and implement the advice. The economic motivation of manufacturers [154] (keeping in mind markets for lemons [5]), their poor track record in IoT security, and lack of accountability for vulnerabilities, point to a worrisome future. We hope that our work is a step towards improving the efficacy of advice on best practices.

Chapter 5

Critique of Coding Tree Methodology

In this chapter, we perform a critique of the SAcoding method to determine if the results from Chapter 4 can be reproduced by a second coder. To do this, using the SAcoding method, a second coder conducted the tagging exercise on the 1013-item DCMS advice dataset (the same as the first coder from Chapter 4), and we compared their results with those of the first coder. Through an analysis of coder agreement on tags and answers to individual questions, we provide a critique of the coding tree methodology itself. Our analysis highlights areas where the coding tree (its structure, instructions, and question descriptions) could be improved to enhance coder tag agreement consistency, and provides other insights into the coding tree's utility, supporting our arguments on why it is useful. As we would like others to have confidence to use our coding tree methodology to analyze different sets of security advice (in both IoT and other security areas), we focus on the analysis of our results and coder comparison.

Ideally, the coding tree would produce zero nonagreements (i.e., both coders agree on all tags given to advice items). Despite our best efforts to create coding tree questions that are as objective as possible (i.e., each question has clear criteria for when to answer *yes* or *no* for an advice item), coders had numerous nonagreements throughout the DCMS 1013-item dataset and in this chapter we look to identify questions in our coding tree at which a large (or small) number of nonagreements occur, and consider potential reasons underlying nonagreements. In this chapter, we primarily focus on the coding tree's structure and its instructions as vectors for possible improvement. Chapter 6 focuses on criticism and improvement of security advice.

Note that in the context of this chapter, an *agreement* (defined in Section 5.1.3) takes place between two coders tagging advice using the coding tree; in contrast,

Chapter 4's use of agreement was in the context of the trial codings with test sets to develop the coding tree (pages 51–61), not its use on full datasets. We use the findings and lessons learned from our analyses herein to inform Chapter 6's analyses.

5.1 Methodology and Results

In this section, we discuss the methodology used in this chapter to gather and analyze tagging data.[1] To establish the data for these analyses (in the following sections), a second coder (*C2*; the coder in the original analysis is denoted as *C1*) underwent the same coding exercise as the first coder. Coding procedures (i.e., the coding tree, instructions given to the coder, coding interface) for C2 remained the same as for C1 in Chapter 4. After C2 completed tagging the full DCMS 1013-item dataset, there were two distinct sets of tagging results: C1's results produced during the coding of Chapter 4, and C2's results as just described.

5.1.1 Extracting Tags Used by Coders

We analyze coder tag distribution and frequency as data for the analysis of coder agreement. To determine how C2's results compare to C1's, we first extract the tags resulting from each coder applying the coding tree methodology to all advice items. To do this, for each advice item in the dataset of 1013 items (Section 3.1.2), using all tags from each coder (a first, and optionally a second), the number of times each tag occurred was counted separately for each coder. For example, for a given advice item, if C1's first and second tags were *P5* and *P4*, C1's count for each of P4 and P5 would be incremented; this would not affect C2's count for P4 or P5. As these counts include both first and (when optionally selected) second tags, the sum of all tag counts for each coder exceeds the number of the advice items in the dataset (1013). Fig. 5.1 summarizes the counts; Table 5.1 shows each coder's tag distribution partitioned into the 13 groups specified in the DCMS guidelines document.

[1] Our methodology is compared and contrasted with similar research in Section 5.4.

Table 5.1: Distribution of two coders' (*C1* and *C2*) advice item codes across the 13 DCMS guidelines, based on categorization of advice items using the DCMS mapping document ([63]; §3.1.2 herein). Cell values reflect the percentage of each guideline that were tagged with column 1 codes, for each coder. Column sums for a coder may exceed 100% as coders could select up to two tags per advice item. Table identifies the codes that advice topics (each UK guideline) most commonly received. The heatmap allows for comparison of the coders' code distributions. Results given in this table are interpreted in Section 5.2 on page 88.

		UK Guideline												
	Coder	UK-1	UK-2	UK-3	UK-4	UK-5	UK-6	UK-7	UK-8	UK-9	UK-10	UK-11	UK-12	UK-13
Items in category (/1013)		81	63	145	84	165	161	65	98	39	51	22	18	21
M1. Not Useful	C1	17%	33%	30%	20%	24%	17%	28%	21%	41%	49%	9%	22%	10%
	C2	5%	14%	13%	11%	16%	12%	12%	14%	21%	29%	5%	11%	10%
M2. Beyond Scope of Security	C1	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	C2	0%	5%	1%	0%	1%	0%	0%	2%	0%	0%	0%	6%	0%
N1. Security Principle	C1	0%	2%	2%	0%	0%	3%	2%	2%	3%	0%	0%	0%	0%
	C2	0%	0%	0%	0%	4%	10%	3%	1%	3%	0%	0%	0%	5%
N1.1. Security Design Principle	C1	0%	0%	0%	1%	1%	7%	3%	4%	5%	0%	0%	6%	0%
	C2	0%	0%	1%	2%	1%	6%	2%	2%	5%	0%	0%	11%	10%
T. Desired Outcome	C1	6%	2%	10%	2%	7%	12%	15%	12%	8%	10%	27%	6%	0%
	C2	9%	14%	35%	14%	17%	22%	25%	28%	26%	28%	32%	44%	10%
T′. Desired Outcome	C1	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	6%	0%
	C2	3%	0%	1%	1%	4%	1%	0%	1%	0%	0%	9%	0%	0%
P1. Incompletely Specified Practice	C1	14%	11%	23%	32%	33%	27%	25%	18%	31%	33%	32%	17%	33%
	C2	20%	0%	13%	19%	22%	16%	22%	11%	10%	20%	9%	0%	10%
P2. General Practice or General Policy	C1	1%	43%	17%	0%	1%	2%	0%	35%	5%	4%	18%	28%	0%
	C2	1%	57%	15%	2%	3%	3%	3%	27%	5%	12%	14%	17%	5%
P3. Infeasible Practice	C1	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	C2	0%	0%	0%	0%	0%	1%	0%	1%	0%	0%	0%	0%	0%
P4. Specific Practice - Security Expert	C1	17%	2%	12%	37%	27%	15%	23%	1%	5%	4%	5%	6%	29%
	C2	31%	3%	14%	44%	24%	14%	25%	4%	8%	6%	5%	0%	33%
P5. Specific Practice - Other Individual	C1	43%	8%	5%	7%	7%	17%	5%	5%	3%	0%	9%	11%	29%
	C2	32%	6%	8%	6%	7%	15%	9%	8%	23%	6%	27%	11%	19%
P6. Specific Practice - End-User	C1	1%	0%	0%	0%	0%	0%	0%	1%	0%	0%	0%	0%	0%
	C2	0%	0%	1%	0%	0%	0%	0%	1%	0%	0%	0%	0%	0%
% tagged as actionable	C1	62%	10%	17%	44%	34%	32%	28%	7%	8%	4%	14%	17%	57%
	C2	63%	10%	22%	50%	32%	29%	34%	14%	31%	12%	32%	11%	52%
Coder actionable difference		1%	0%	5%	6%	2%	3%	6%	7%	23%	8%	18%	6%	5%

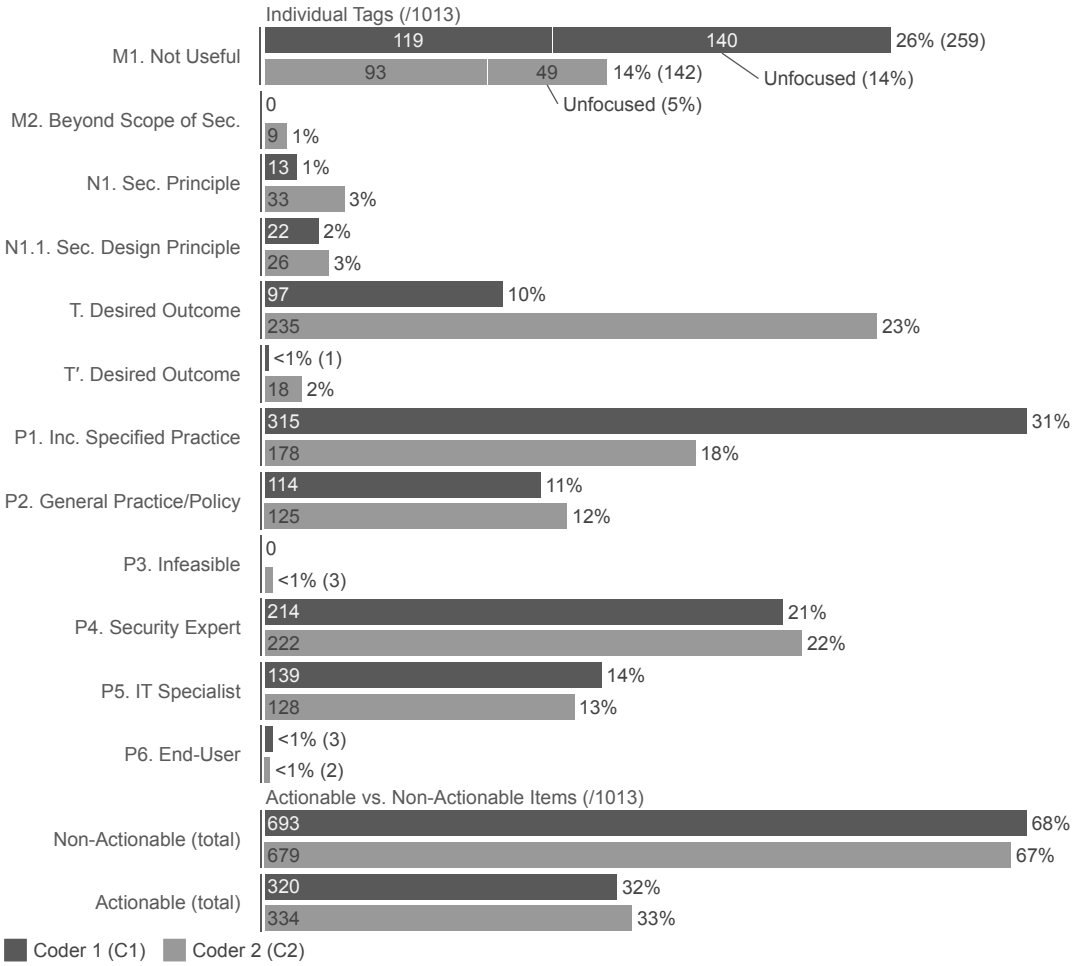


Figure 5.1: Tag distribution of coding the full 1013-item DCMS dataset. Sum for each coder (excluding actionable and non-actionable bars) exceeds 1013 as items may be assigned two codes. C1's results duplicated from Chapter 4 to aid visual comparison. Actionable vs. Non-Actionable bars not drawn to scale. Note: While the graph shows the number of tags of a given type for each coder, an identical number of tags of, e.g., *P2* would not imply that exactly the same individual advice items were tagged *P2* by both coders.

5.1.2 Proportion of Non-Actionable Advice

To determine whether an advice item is considered actionable or not, for each coder, if either the first *or* (if present) second tag given to an item is actionable (*P3–P6*), that advice item is considered actionable (as the coder could make an argument for its actionability, as noted in Section 4.1.1). If either tag is actionable, this will increment the count in the Actionable bar of Fig. 5.1 for that coder, otherwise the

count is incremented for the Non-Actionable bar. As this occurs exactly once per advice item (per coder), the Actionable and Non-Actionable bars of Fig. 5.1 sum to 1013 (total number of advice items) for each coder. See Section 4.1.1 for the explanation of actionable tags.

From Fig. 5.1, we see C2 tagged 33% of items with an actionable tag. This agrees with our finding on how actionable the DCMS 1013-item dataset is (C1's 32% from Chapter 4). Further, in more than 80% of advice items, both coders agreed on whether any given advice item was actionable (discussed in detail in Section 5.2.4). This suggests that the wording and selection of the coding tree's questions and their relative positions or locations in the tree allow for effective classification of actionable items (versus non-actionable) using the coding tree.

Of the actionable tags, coders had similar proportions of use for the tags *P5* (*IT Specialist*—14% and 13%) and *P4* (*Security Expert*—21% and 22%). The other two actionable tags (*P6*, *End-User*; and *P3*, *Infeasible)* were used a total of only 8 times (out of the total 2026 items tagged by both coders), suggesting the advice items in the dataset are, in general, appropriately targeted for their desired audience.[2] Only 5 items were tagged as targeting end-users (*P6*), who are beyond the primary scope of the advice; and 3 were considered infeasible (*P3*), suggesting that when advice in this dataset is actionable (all tags beyond Q_5's *yes* branch), it is usually feasible to follow.

Given that these two results align with the scope of the tagged advice (i.e., that the advice is primarily for pre-deployment stakeholders), we believe the low use of tags *P3* and *P6* indicates that this advice dataset meets its objective of providing feasible (if actionable) advice for pre-deployment stakeholders [62, 68].[3]

While both coders tagged the 1013-item dataset with similar numbers of actionable tags overall, some tags were used more frequently by one coder or the other:

- *M1* (*Not Useful*): 26% vs. 14% for C1 and C2, respectively
- *T*, *T′* (*Desired Outcome*): 25% vs. 10%
- *P1* (*Incompletely Specified Practice*): 31% versus 18%

[2]The targeted audience is very broad and non-specific, as discussed in Chapter 6.
[3]Section 5.2.2 discusses the low use of the *P3* and *P6* tags further.

Fig. 5.1 also shows that codes *N1* (Security Principle) and *N1.1* (Security Design Principle) were used around twice as often by C2 than by C1 ($33 + 26 = 59$ vs. $13 + 22 = 35$). For the other 6 code categories, coders had similar tag distributions in that the number of items assigned to each tag differed between coders by at most 1 per cent of the number of items in the dataset. A similar code distribution does not, however, imply coders agreed on the same codes for individual advice items. Initially, a similar tag distribution appears promising, but overall it is premature to claim that the SAcoding method can reliably estimate the proportion of a dataset's advice items in (most) categories; further evidence and perhaps methodology changes (discussed in Section 5.3.3) would be needed to support a claim that tag distribution is reproducible. The relationship between coder tag distribution and tag nonagreements is discussed further in Section 5.1.7.

5.1.3 Coder Nonagreements

After extracting the tags used by each coder, we evaluated at which questions coders had *agreements* and *nonagreements* for each advice item. We use agreements and nonagreements in two contexts. In the context of *tags*, an *agreement* (hereafter a *T-agreement*) is when at least one of the tags given to an advice item by a coder (through use of the coding tree) is the same as one of the tags given by the other coder; otherwise, it is a *T-nonagreement* (i.e., the coders have no tags in common among their first and possibly second tags for an advice item).

In the context of *a question*, an agreement (hereafter a *Q-agreement*) is when both coders provide the same answer to a question in the tree (e.g., both coders answer *yes*, or both answer *no*). A *Q-nonagreement* is when coders answer differently to a question in the tree (e.g., one *yes*, the other *no*).

A T-agreement is determined based on tags the coding tree assigns to advice items (for each coder), a Q-agreement is determined based on individual questions in the coding tree that two coders answer. Thus, Q-agreements are related to T-agreements, as each coder's path to a code through the coding tree dictates which questions Q-agreements take place on.

How agreements and nonagreements (for both tags and questions) are determined

is described more thoroughly below. We describe non-matching tags and question answers as nonagreements instead of disagreements, based on our view that the term *disagreement* implies two coders explicitly disagreed on something (e.g., the tag to be applied, or the answer to a question), rather than coders independently selecting different sequences through the coding tree and the outcomes differing. Understanding where coders had nonagreements most frequently (e.g., which questions in the coding tree they diverge on) allows us to see where the coding tree might be improved to more reliably converge coder answers toward *yes* or *no*, and highlight advice items that are vague or open to subjective interpretation by a coder.

A *coding sequence* (hereafter *sequence*) is the sequence of question nodes resulting from use of the coding tree on an advice item; the nodes are joined by edges, as determined by *yes*/*no* question answers.[4] The number of nodes in the sequence (n) is the number of questions a coder answers before reaching a tag. For example, the tag *P1* is reached by answering Questions 1 (*yes*), 2 (*yes*), 3 (*no*), 4 (*yes*), and 5 (*no*), resulting in the sequence $(Q_1, Q_2, Q_3, Q_4, Q_5)$ of length $n = 5$. As a sequence describes the path taken through the coding tree to reach a single code, a coder determines one sequence for each tag assigned to an advice item. Sequences are not explicitly created by coders, but result from answering a question at each node until reaching a tag.

For a given advice item, a *diverging question* is the tree question at which two coders give different answers, thus taking a different exit path from that node, eventually yielding different tags. When two coders yield a different tag on a given advice item, we determine the diverging question by first finding where the two coders' sequences overlap. Starting from Q_1 in one coder's sequence, we compare it with the first node of the other coder's sequence. If these match (Q_1 always matches), the node (question number) is appended to the overlapping sequence. The remaining nodes of both sequences are compared in this way (see example below). As the overlapping sequence only contains nodes common to both coders' sequences, question order is preserved. The last node in the overlapping sequence is the diverging question, as thereafter the paths differ. Consider the following example (underlined node

[4]See Fig. 4.1's Y and N labeled edges from Q_1 to a leaf node (code).

indicates the diverging question):

$C1$: $(Q_1, Q_2, Q_3, \underline{Q_4}, Q_5, Q_6, Q_7, Q_8) \Rightarrow$ P5

$C2$: $(Q_1, Q_2, Q_3, \underline{Q_4}, Q_9) \Rightarrow$ P2

Overlap: $(Q_1, Q_2, Q_3, \underline{Q_4}) \Rightarrow Q_4$ Q-nonagreement occurs at this question

We use this method to determine at which question the coders have Q-nonagreements.

As the number of tags given to an advice item by two coders can vary from 2 to 4, we consider three *types* of advice tag comparison (based on T-agreements). Each comparison is within the scope of a single advice item, i.e., by looking at the tags each coder (C1 and C2) assigned to that item.

As we consider first (subscript "1", e.g., $C1_1$) and second (subscript "2", e.g., $C1_2$) tags to be of equal importance (Section 4.1.1), these types are based on the *number* of tags assigned by coders, not the tags' status as either being the first or second assigned to an item. Among all three comparison types, a *match* means that two tags (one from each coder) are identical (e.g., $C1_1 = C2_1$)—a match never takes place between two tags of the same coder.

5.1.4 Type A Tag Comparisons

In a *Type A* comparison, each coder gave only one tag (a first tag) to the advice item. In this type, a T-agreement occurs if both coders' first tag matched; a T-nonagreement occurs otherwise. Consider the following example of a Type A T-agreement and T-nonagreement:

T-agreement ($C1 = C2$):

$C1$: $(Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7, Q_8) \Rightarrow$ P4

$C2$: $(Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7, Q_8) \Rightarrow$ P4

T-nonagreement ($C1 \neq C2$):

$C1$: $(Q_1, Q_2, Q_3, \underline{Q_4}, Q_5, Q_6, Q_7, Q_8) \Rightarrow$ P4

$C2$: $(Q_1, Q_2, Q_3, \underline{Q_4}, Q_9) \Rightarrow$ P2

T-nonagreement overlap:

$C1, C2$: $(Q_1, Q_2, Q_3, \underline{Q_4}) \Rightarrow Q_4$ Q-nonagreement occurs at this question

Note that there are only two sequences to compare in a Type A comparison $(C1_1, C2_1)$.

5.1.5 Type B Tag Comparisons

In a *Type B* comparison, one coder gave one tag (first), and the other opted for two tags (first and second), for a total of three tags to one advice item. In this type, a T-agreement occurs if the single-tag coder's first tag matched either the double-tag coder's first *or* second tag (again, as we consider both the first and second tags to be of equal importance, a T-agreement is declared if either tag matches); a T-nonagreement means the single-tag coder's tag was not identical to either of the double-tag coder's tags. Consider the following example of a Type B T-agreement and T-nonagreement (single and double underline indicate first and second overlap T-nonagreement questions, respectively):

T-agreement ($C1 = C2_2$):

$C1$: $(Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7, Q_8) \Rightarrow$ P4

$C2_1$: $(Q_1, Q_2, Q_3, Q_4, Q_5) \Rightarrow$ P1

$C2_2$: $(Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7, Q_8) \Rightarrow$ P4

T-nonagreement ($C1_1 \neq C2$ and $C1_2 \neq C2$):

$C1_1$: $(Q_1, Q_2, Q_3, \underline{Q_4}, Q_5, Q_6, Q_7, Q_8) \Rightarrow$ P4

$C1_2$: $(\underline{\underline{Q_1}}) \Rightarrow$ M1

$C2$: $(\underline{\underline{Q_1}}, Q_2, Q_3, \underline{Q_4}, Q_9) \Rightarrow$ P2

T-nonagreement overlaps:

1. $C1_1, C2$: $(Q_1, Q_2, Q_3, \underline{Q_4}) \Rightarrow$ length: 4 (diverges at Q_4)
2. $C1_2, C2$: $(\underline{\underline{Q_1}}) \Rightarrow$ length: 1 (diverges at Q_1)

In Type B comparisons, to determine a single question where coders diverged (a Q-nonagreement), we use the longest T-nonagreement overlapping sequence ($\{C1_1, C2\} = 4 > \{C1_2, C2\} = 1$ from above), and declare the final question in that sequence to be

where coders had the Q-nonagreement. If we used the shortest of the two overlapping sequences, in every instance that a coder tagged an item as *M1* (Q_1's *no* answer), the diverging question would be Q_1, as it is the final question in the overlapping sequence.[5] As in a T-nonagreement at most one coder could yield an *M1* tag (it would be a T-agreement if both did), the other sequence of the two-tag coder is necessarily longer, providing a longer overlapping sequence to analyze. We recorded no cases where one advice item had two overlapping sequences of the same length, but containing different nodes. The above T-nonagreement example illustrates overlapping sequences and how the T-nonagreement is determined for a Type B comparison.

5.1.6 Type C Tag Comparisons

In a *Type C* comparison, both coders opt for two tags (first and second), resulting in a total of four tags assigned to an advice item. In this type, we declare a T-agreement if one coder's first or second tag is identical to either tag of the other coder; a T-nonagreement occurs if neither of the two tags from the first coder is identical to either from the second coder. Consider the following example of a Type C T-agreement and T-nonagreement:

T-agreement ($C1_1 = C2_2$):

$C1_1$: $(Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7, Q_8) \Rightarrow$ P4 (matches $C2_2$)

$C1_2$: $(Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7, Q_8) \Rightarrow$ P5

$C2_1$: $(Q_1, Q_2, Q_3, Q_4, Q_5) \Rightarrow$ P1

$C2_2$: $(Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7, Q_8) \Rightarrow$ P4 (matches $C1_1$)

T-nonagreement ($C1_1 \neq C2_1$, $C1_1 \neq C2_2$, $C1_2 \neq C2_1$, $C1_2 \neq C2_2$):

$C1_1$: $(Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7, Q_8) \Rightarrow$ P4

$C1_2$: $(Q_1, Q_2, Q_3, Q_4, Q_5) \Rightarrow$ P1

$C2_1$: $(Q_1, Q_2, Q_3, Q_4, Q_9) \Rightarrow$ P2

$C2_2$: $(Q_1, Q_2, Q_3, Q_4, Q_9, Q_{10}, Q_{11}) \Rightarrow$ N1

[5] This example is discussed further in Section 5.2.

Table 5.2: The number of advice items considered in each tag comparison type (A–C) and the number of T-agreements within each, and T-agreements where both coders reached tags that were actionable, or both were non-actionable. T-nonagreement percentages can be calculated as 100% minus agreement % stated in table. Regarding denominators, note that $760 + 234 + 19 = 1013$.

Type	Num of advice items	T-agreements (§5.1.3)	Actionable *or* non-actionable (§5.2.4)
Type A	760/1013 (75%)	308/760 (41%)	608/760 (80%)
Type B	234/1013 (23%)	130/234 (56%)	204/234 (87%)
Type C	19/1013 (2%)	17/19 (89%)	18/19 (95%)

As there are so few instances where both coders used two tags (as discussed shortly), Type C is excluded from most of our subsequent analysis.

5.1.7 T-agreements Summary and Results

Using the above methodology for calculating T-agreements, 760 advice items received one code from each coder (Type A), 234 had one code from one coder and two from the other (Type B), and 19 had two codes from each coder (Type C).[6] Table 5.2 summarizes the results of calculating T-agreements and T-nonagreements for Types A, B, and C. Table 5.3 summarizes the T-agreements of the Regular calculation type (column 3 from Table 5.2, page 84) and the number of times each coder's code selection order (*first* and *second* codes) resulted in a match with the other coder's codes across Types A, B, and C.

In Table 5.2, the high percentage of T-agreements among Type C advice items for

[6]The number of instances from each comparison type are the denominator in the last two columns of Table 5.2.

Table 5.3: The number of times each *first* or *second* codes from each coder resulted in a T-agreement (column 3 from Table 5.2) for the three comparison types. Each numerator is the number of T-agreements within a code comparison pair; denominator is total number of T-agreements in a comparison type.

	T-agreement case (code pair order)		
Type	First-First	First-Second	Second-Second
Type A (308 T-agreements)	308/308 (100%)	–	–
Type B (130 T-agreements)	78/130 (60%)	52/130 (40%)	–
Type C (17 T-agreements)	5/17 (29%)	7/17 (42%)	5/17 (29%)

the Regular calculation type may be partially explained by the additional involvement of the second tag from both coders. As shown in Table 5.3, for the Type C comparison, a combined 71% of T-agreements were First-Second or Second-Second, i.e., occurred between either one coder's first tag and the other's second tag, or both coders' second tags. While there were few Type C T-agreements on first codes (29%, much lower than even Type A's 41% T-agreements from Table 5.2), they tended to agree more often (proportionally) when the second code was involved (as exemplified in Table 5.2's column 3 where T-agreements increase in Types B and C).

Overall, coders agreed on codes in 45% of cases ((308 + 130 + 17)/1013 from Table 5.2).[7] Comparing coder tags assigned to an individual advice item [33], for a T-nonagreement, we can determine which tags were selected by each coder, i.e., *when one coder reached a tag, what non-agreeing tag did the other coder reach?*[8] This provides additional information about tags that were reached across the entire dataset with similar frequencies by both coders (e.g., *P4*'s 1 percentage point difference), but for which there often failed to be a T-agreement on individual items. This provides further details related to the assertion that a similar coder tag distribution does not imply tag agreements for individual items (mentioned in Section 5.1.2).

Type A and Type B comparisons account for 98% of two-coder tag comparisons in column 2 of Table 5.2. As so few advice items were tagged twice by both coders (19/1013), we do not analyze or discuss Type C agreements (of tags or questions) further (beyond Tables 5.2 and 5.3).

5.1.8 Proportion of Q-nonagreements Within Each Question

Fig. 5.2 summarizes the distribution of Q-nonagreements at each question for comparison types A and B. To understand how commonly coders had Q-nonagreements

[7]Coder agreement rates here (45%) are notably lower than the test set agreement rate of Chapter 4 (73%). We attribute this in-part to Chapter 4's use of the ±1 rule (Section 4.1.1 on page 58), which is not used in this chapter's analysis. Nonetheless, these rates were disappointing and below our expectation.

[8]This analysis—that we call the 'tag-vs-tag table' analysis—was conducted in joint work with co-authors [33] done after the original submission of the thesis, but before the oral defense and subsequent minor revision. The details do not appear in this thesis, but the finding is mentioned here to clarify the relationship between coder tag distribution and agreement on individual advice items.

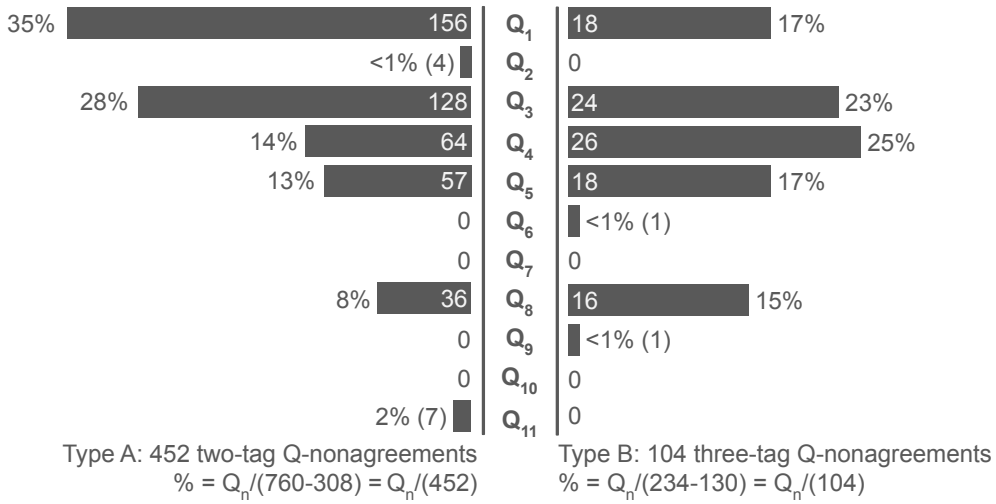


Figure 5.2: Distribution of Q-nonagreements across diverging questions between two coders' tags for Type A and Type B (Q-nonagreements and diverging questions as defined in §5.1.3). Compare with Figs. 5.3 and 5.4. Q_n denotes question n from the coding tree (page 54). Note Table 5.2 has 308 Type A T-agreements, and $760 - 308 = 452$ (number of Type A Q-nonagreements here).

at specific questions (versus T-nonagreements of tags for an advice item as above), we considered the number of Q-nonagreements at each question relative to the number of times each question was encountered by both coders.

To calculate this, we used the longest sequence of overlapping nodes from each coder's node sequences (as explained in subsection 5.1.3). For each question in the coding tree we counted the number of times coders had a Q-agreement (the same answer, either both *yes* or both *no*), and the number of times coders had a Q-nonagreement. If the coders agreed on the final tag for an advice item, the Q-agreement count of each node in the sequence (including the final Q node) is incremented. If the coders did not agree on the final tag for an advice item, the Q-agreement count was incremented for each node in the overlapping sequence *except* the diverging node. For the final Q node in the overlapping sequence for a T-nonagreement (Q_n), the Q-nonagreement count was incremented, as on this question the coders answered differently (the diverging question).

Since Q_1 is the first question, always asked regardless of any other nodes visited by each coder, the number of times Q_1 is asked will be the total number of advice items in each comparison type (760 for Type A, 234 for Type B, and 19 for Type C;

see Table 5.2).

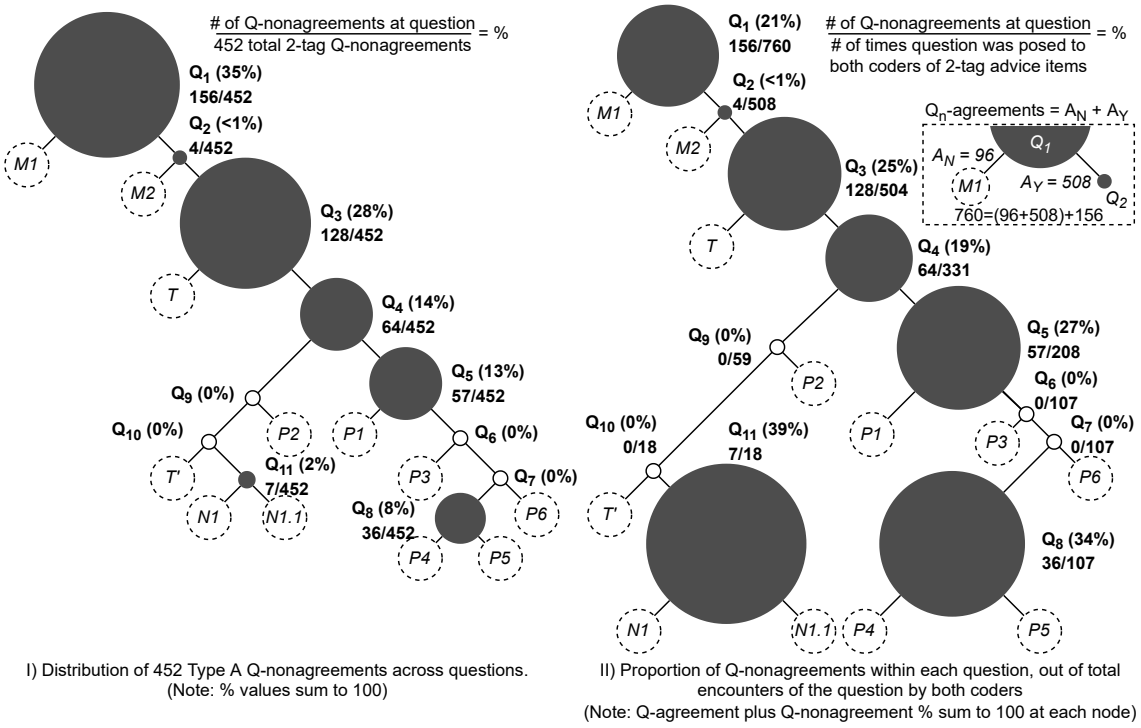


Figure 5.3: Q-nonagreement results for Type A comparison (one tag from each coder; 760 instances, cf. Table 5.2). (I) Duplicating data from Fig. 5.2's Q-nonagreement results. Shows proportion of Q-nonagreements (relative to all other nodes) occurring at each question. (II) Within each question, ratio of Q-nonagreements to how often that node was visited by both coders, including Q-agreements and Q-nonagreements; circle size represents proportion of Q-nonagreements at that question relative to Q-agreements plus Q-nonagreements at that question. Sum of Q-agreements at each question include Q-agreements on both *yes* and *no* answers (see dashed box). Questions labelled 0% imply no Q-nonagreements occurred.

Figs 5.3 and 5.4 interpret Fig. 5.2's data and show the distribution of Q-nonagreements across all questions (subfigure I), and what percentage of joint encounters of each question resulted in Q-nonagreements (subfigure II).

5.2 Interpretation of Nonagreement Results

In this section, we interpret our nonagreement results (for tags and questions) and provide observations and insights on the utility of the coding tree; and we focus

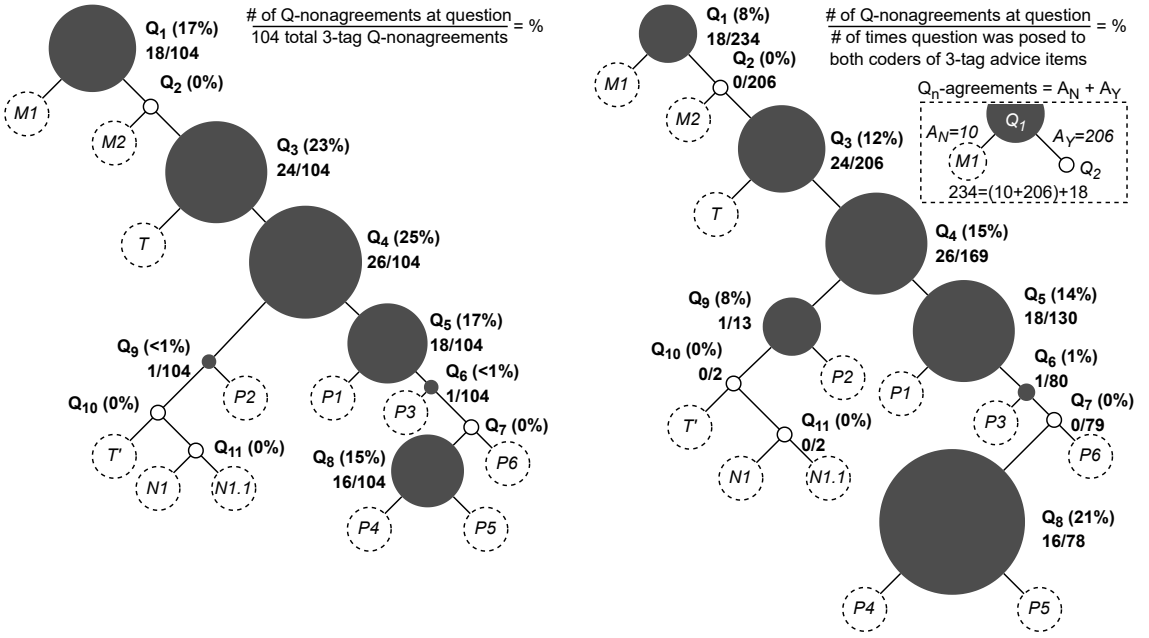


Figure 5.4: Q-nonagreement results for Type B comparison (one tag from each coder; 234 instances, cf. Table 5.2). (I) Duplicating data from Fig. 5.2's Q-nonagreement results. Shows proportion of Q-nonagreements (relative to all other nodes) occurring at each question. (II) Within each question, ratio of Q-nonagreements to how often that node was visited by both coders, including Q-agreements and Q-nonagreements; circle size represents proportion of Q-nonagreements at that question relative to Q-agreements plus Q-nonagreements at that question. Sum of Q-agreements at each question include Q-agreements on both *yes* and *no* answers (see dashed box). Questions labelled 0% imply no nonagreements occurred.

on specific areas of the coding tree associated with a large or small number of Q-nonagreements in order to indicate potential areas for improvement of the coding tree, or highlight tree areas where coding of advice items differed most between coders. In some cases we can provide concrete explanations for why Q-nonagreements take place more or less at certain questions; in cases where we cannot (e.g., due to the subjective nature of the coding tree), we offer conjectures.

Beyond the full-set tag distribution of Fig. 5.1, Table 5.1 on page 76 shows the distribution of the two coders' tags across each of the 13 UK guidelines.[9] From

[9]Recall that the guideline document is paired with a "mapping" document, which maps each of the 1013 advice items to one of 13 guidelines; see Section 3.1.2.

Fig. 5.1, we see large differences in the frequency of occurrence between coders for some tags. Comparing the differences in Fig. 5.1 to Table 5.1, for example, C1 tagged 49% of the advice mapped to the *UK-10* guideline as *Not Useful* (*M1*), whereas for C2, *M1* resulted for 29% of items. Proportions of other tags within *UK-10* are different. For example, for *Desired Outcome* (*T*), the cells for C1 and C2 are respectively 10% and 28%, suggesting coders disagreed significantly on (as it turns out, in several categories of) tags for items in *UK-10*. In the *UK-10* column, *M1* occurs more frequently for C1 (than C2), and *T* more frequently for C2 (than C1); however, we have no evidence of any causal relationship in this pairing of *T* and *M1*. As the coding tree tool did not explicitly prevent coders from assigning tags based on their first reading and the tag definition (bypassing the tool; see limitation discussion of Section 5.3.2), these differences in tag distribution (and Q-nonagreements) may be partially explained by unintended coder direct use of *tag* descriptions, versus *question* details (as intended by our design) whereby answers to each question determine the tag assigned.

5.2.1 High Numbers of Q-nonagreements

As evident from Fig. 5.2, in both Type A and B comparison types, nearly all Q-nonagreements (98% of both) come from Questions 1, 3, 4, 5, and 8. We examine the three questions with the most Q-nonagreements, and conjecture why these dominate the Q-nonagreements.

Question 1 (35% and 17% Q-nonagreements for Type A and B, respectively) asks if advice is unambiguous and focused. Q_1 has the greatest difference in proportion of Q-nonagreement between Type A and Type B (35% versus 17%), and also the greatest combined sum of Q-nonagreements for both Type A and B. Here we conjecture why Type A's Q_1 has significantly more Q-nonagreements than Type B's.

A Type A comparison has 4 possibilities for both coders' (C1, C2) Q_1 answers:

Case 1. *no*, *no* (Q-agreement on *no*)

Case 2. *no*, *yes* (Q-nonagreement)

Case 3. *yes*, *no* (Q-nonagreement)

Case 4. *yes*, *yes* (Q-agreement on *yes*)

In a Type A Q-nonagreement (Cases 2 and 3; 50% of cases), if either coder answers *no* to Q_1 (resulting in a tag of *M1*), the diverging question is always Q_1 regardless of how many more questions the other coder has in their sequence.

A Type B comparison has 4 possibilities for both coders' Q_1 answers ($C1$, $\{C2_1, C2_2\}$):

Case 1. *no*, {*yes*, *no*} (Q-agreement on *no*)

Case 2. *no*, {*yes*, *yes*} (Q-nonagreement)

Case 3. *yes*, {*yes*, *no*} (Q-agreement on *yes*)

Case 4. *yes*, {*yes*, *yes*} (Q-agreement on *yes*)

Note that we depict a Type B comparison as having 4 possibilities instead of 8 because the order of C2's codes do not matter for Q-agreement calculation (if either code is identical to C1's, it is a Q-agreement), and the two-tag coder cannot select *no* twice, else they would be tagging an advice item as *M1* twice. These two considerations result in a Type B comparison only having 4 possibilities.[10]

In a Type B nonagreement (Case 2; 25% of cases), as the one-tag coder answered *no*, regardless of how the two-tag coder answers questions beyond Q_1, the only node that both coders' sequences will share is Q_1. Thus, Case 2 may exemplify why there are fewer Q-nonagreements in the Type B comparison—if the one-tag coder answers *no* to Q_1, the diverging question is always Q_1 because it is the only node shared by both coders. Note that a Q-nonagreement is impossible at Q_1 if the one-tag coder answers *yes* to Q_1. In Type A, either coder answering *no* to Q_1 (two of the four cases, see above) forces the diverging question to be Q_1. In Type B, only one of the four cases result in a Q-nonagreement at Q_1, which explains in part why there are fewer Q-nonagreements at Q_1 for Type B (35% of Type A versus 17% of Type B), and why there are more Type B Q-nonagreements in the questions following Q_1 (e.g., Q_3, Q_4, Q_5) than for Type B's Q_1 (right side of Fig. 5.2 on page 86).

[10]The 4 excluded cases are: (*no*, {*no*, *no*}), (*no*, {*no*, *yes*}), (*yes*, {*no*, *no*}), (*yes*, {*no*, *yes*}).

Question 3 (28% and 23% Q-nonagreements for Type A and B, respectively) asks if an item is an outcome, versus an action to take. For reference, we repeat Q_3, the definition of *Outcome* (page 37), and the extended annotation of Q_3 (Fig. 4.5 on page 57) here.

> Q_3: Is it [the advice item] focused on a desired *outcome* more than how to achieve it?
>
> Outcome: An *outcome* is a desired end goal that a stakeholder aims to reach.
>
> Q_3's annotation details: Is the advice a high-level outcome rather than some method (or meta-outcome) for how to achieve an outcome? E.g., *data is secured in transit* would be an outcome because it is a desired goal or state, whereas *encrypt data in transit* is not because it explains a method for achieving that outcome (in this case, encryption). *Encryption* may be considered a meta-outcome, as it is not meaningful to the end-user's ultimate goal of protected data.

Our definition of *outcome* may not be clear enough, or perhaps difficult to relate to the items in the dataset, leading to coder confusion about whether advice is an outcome. As C2 selected T 2.4 times as often as C1 (235 versus 97), C2 often answered Q_3 resulting in T while C1's answer to Q_3 resulted in a tag deeper in the tree, leading to frequent Q-nonagreements when the advice item was interpreted as an outcome by C2. In cases where C2 produced T and C1 another tag, C1 produced a tag that was below T (i.e., all tags below Q_4 in the tree) 106/142 times (75%) for Type A, and 39/40 times (98%) for Type B. This suggests that the description of an *outcome* (above) is unclear or may be interpreted differently by coders with differing background or experience; thus the definition of *outcome*, the coding tree's question about it (Q_3), or its extended annotation (Fig. 4.5 on page 57) might be clarified or improved in the coding tree materials that we provide to coders.

For example, our definition as written (above) relies on coders sharing a mutual understanding of what a "desired end goal" is. If their understanding differs, one may be more or less likely to select it for a given item than the other. A desired end goal is briefly described in the annotation details using an example (above), but

for more complex cases in the 1013-item dataset, it may become less clear to coders whether a given item fits our definition of *outcome*.

Question 4 (14% and 25% Q-nonagreements for Type A and B, respectively) asks if an item suggests any of: a security technique, mechanism, software tool, or specific rule. This relies heavily on the definitions of these terms, which are not defined in the question itself (but are given in part in the annotations provided to the coders, along with examples, Fig. 4.5 on page 57). The advice item may not align well with the examples in the annotation, resulting in coders using their own subjective definitions. Q_4's annotation might possibly offer more explicit and specific definitions for each term in the question.

Finally, we note that Q_4 is the main branch point where an advice item will either head toward actionable tags (or *P1*), or toward more general advice such as principles. Q-nonagreements at Q_4 create a major divergence in coder sequences, in that the resulting tags will differ even in terms of actionable versus non-actionable (see Table 5.2).

5.2.2 Low Numbers of Q-nonagreements

As Fig. 5.2 on page 86 suggests, Questions 2, 6, and 7 rarely have Q-nonagreements, but as these questions are related to the scope of the advice, how they are answered (and whether they invoke Q-nonagreements) may change based on the advice set the coding tree is applied to. We examine these questions and conjecture why these have so few (if any) Q-nonagreements.

Question 2 (<1% and 0% Q-nonagreements for Type A and B, respectively) asks if the advice item is helpful for security. Given the scope and target audience of the 1013-item dataset (see Section 3.1.2 on page 30), and as the second question was asked of coders 67% (508/760, Fig. 5.3.II) and 88% (206/234, Fig. 5.4.II) of the time (for tag comparisons of Type A and B, respectively), but only answered as *no* 9 times, suggests the advice in the DCMS 1013-item dataset was largely perceived by both coders to be generally related to security issues. Thus, the few Q-nonagreements at this question combined with its high frequency of use suggests it is an appropriate question for filtering out out-of-scope advice. Our understanding is also that this

particular dataset was deliberately crafted to fall into the *yes* case of this question. If our coding tree methodology is applied to a different dataset of advice, a larger number of Q-nonagreements might occur at Q_2.

Question 6 (0% and <1% Q-nonagreements for Type A and B, respectively) asks if the advice is viable to accomplish with reasonable resources. This question was rarely answered *no* to (in total 3 times, only one of which was a nonagreement). Like Q_2, this suggests nearly all actionable advice items in the 1013-item dataset are reasonable to implement (per Q_6's description of reasonable, and our description of infeasible advice in Section 3.2.2 on page 41).

Question 7 (0% Q-nonagreements for both types) asks if it is expected that the end-users of a product would be expected to carry out the advice item. As the target audience for the DCMS guidelines document (which makes use of the 1013-item dataset; see Section 3.1.2 on page 30 for its scope) does not include end-users, it makes sense that few *Specific Practice—End-User* (*P6*) tags would be used (*yes* to Q_7), thus no Q-nonagreements.

5.2.3 Observations of Q-nonagreement Distributions

We expect the overall distribution for Q-nonagreements (Fig. 5.3.I) to be weighted toward the top, with the proportion of Q-nonagreements at subsequent nodes decreasing, as Q-nonagreements higher in the coding tree preclude Q-nonagreements lower in the coding tree for a given advice item. We do not expect this trend when considering Q-nonagreements within each question (Fig. 5.3.II), as here a Q-nonagreement higher in the tree has no impact on the proportion of Q-nonagreements within each question lower in the tree (beyond a reduced number of visits).

In general, Figs. 5.3 and 5.4 match our expectations for how Q-nonagreements are distributed, with some exceptions. We note some cases where specific questions have a small overall share of the Q-nonagreements (Figs. 5.3.I and 5.4.I), but a large proportion of within-question Q-nonagreements (i.e., a high percentage of times a question was posed to both coders, there was a Q-nonagreement; Figs. 5.3.II and 5.4.II). Two such cases are Q_8 and Q_{11}, which we discuss here.

Question 11 accounts for only 2% of the total Q-nonagreements. It asks whether

the advice item relates to the design phase of the product lifecycle. While it was rarely reached by both coders, there was Q-nonagreement 39% of the times it was reached in Type A (zero Q-nonagreements for Type B). As Q_{11} asks coders to answer based on their interpretation of where in the device lifecycle the advice would be followed (i.e., the design phase, or after), this may suggest a knowledge gap (among coders) on mapping advice to lifecycle phase, or differing views of what comprises the *design* phase of a product. The Q-nonagreements here at Q_{11} appear to be an issue with the coding tree instructions rather than the input advice. As Q_{11} appears at the bottom of the tree and leads to two end codes which are similar, Q-nonagreements here do not impact the ending tag of an advice item as significantly as other questions—regardless of whether coders had a Q-nonagreement at Q_{11}, the ending tag is either a security principle (*N1*) or security design principle (*N1.1*), with the latter viewed as a sub-set of the former. While ideally Q-nonagreements here would be avoided, we do not consider this a major problem for the tree, and its removal (e.g., removing Q_{11} and having Q_{10}'s *yes* go directly to security principle, *N1*) would only minorly impact the overall coding results.

Question 8 is similar to Q_{11}, in that it is a node that leads to two leaf nodes. The Type A difference between overall Q-nonagreement proportion (Fig. 5.3.I, 8%) and within-question Q-nonagreement proportion (Fig. 5.3.II, 34%) indicates that when coders both reached Q_8, it was common for them to non-agree. Q_8 asks about the level of security experience advice-recipients are expected to have. A coder with more security experience may answer differently than a less experienced coder, as they apply their personal security knowledge to determine which groups (security expert *P4*, or IT specialist *P5*). This may help explain why a large proportion of Q_8 answers were Q-nonagreements, as in our study the two coders' security experience differed significantly (a limitation, see Section 5.3).

Unlike Q_{11}, Q-nonagreements at Q_8 cause concern, as we expect IoT device manufacturers to typically have IT specialists developing devices (*P5*), but fewer to employ security experts (*P4*). As such, determining which of the two audiences an item of advice is intended for is important, as an advice item that targets a security expert might not be useful to (not executable by) a non-expert. If our tool is able to

identify advice items that do not match target recipients, this offers an opportunity to improve advice.

5.2.4 Comparing Actionable and Non-Actionable Agreements

To determine how well the coding tree can determine the actionability of an advice dataset,[11] we further calculated agreement within actionable tags, and within non-actionable tags (i.e., *P3–P6*; versus all others) for comparison Types A and B. The method follows that for determining agreements and nonagreements from Section 5.1.3. This data is included in Table 5.2 (column 5) on page 84.

For Type A, we determine the single tag of each of the two coders for a given advice item, and if both *or* neither are an actionable tag, that counts as an agreement; else, a nonagreement. For Type A, coders agreed on 608/760 (80%) advice items (both actionable or both non-actionable). For Type B, we compare the single-tag coder's tag to each of the double-tag coder's tags. If in either pair ($\{C1, C2_1\}$ or $\{C1, C2_2\}$) both tags are actionable or both are non-actionable, it counts as agreement; else, a nonagreement. For Type B, coders agreed on 204/234 (87%) advice items regarding actionability. Unsurprisingly, this percentage is greater compared to Type A because two pairs are being considered, and either pair may yield an agreement. We note this as a limitation and give more weight here to the results from Type A.

This analysis indicates that, using the coding tree, coders are able to distinguish a practice (that is actionable by our definition; Section 3.2.2) from a non-practice in most cases—it appears actionable and non-actionable advice items are easily distinguishable. This does not, however, imply the coders will not have T-nonagreements about *which* practice code an item is (e.g., see Q_8 in Figs. 5.3.II and 5.4.II). Particularly for Type A comparisons, where the chance of T-agreement is lower (i.e., only one pair can match, versus the two of Type B), this suggests the coding tree is useful for estimating how actionable an overall advice dataset is (in this case, the DCMS 1013-item dataset).

[11] Recall that we prioritize actionability as a characteristic of useful security advice.

5.3 Coding Tree Utility and Limitations

Despite the iterative refinement and improvement of the coding tree (see Chapter 4), coders still had a significant number of T-nonagreements on tags for advice items. Overall, there is room for improvement of the coding tree methodology. While improving the coding tree is the subject of future work (and we encourage others to pursue this also), here we summarize areas where we believe the coding tree has been demonstrated to be useful, and also limitations of the coding tree and its design methodology.

5.3.1 Utility of the Coding Tree Methodology

The following are what we believe the coding tree methodology is useful for, and the benefits it offers to the IoT security field (and as noted earlier, we believe to the wider security community also).

Summarizing characteristics of security advice. Applying the method to analyze the 1013-item advice dataset (believed to broadly represent IoT security advice) allowed a characterization of the dataset. This included, for the advice items offered, the comparative number of items in designated categories (outcomes, principles, practices, policies, and those unclassifiable). The tagging by two separate coders resulted in similar proportions of tags for many, but far from all, tag categories; however our confidence in this result is weak, as the T-agreement rates in Table 5.2, and more detailed analysis [33] revealed that the coders often did not assign the same tag to a given individual item. We thus find it premature to make a broad claim that the method is useful for identifying advice that belongs to the high-level categories represented by the tags, or for accurate estimates of the overall proportion of advice that falls into each category.

These categories are used in Chapter 6 to compare different datasets, whereas this chapter is focused on categorizing the advice in the DCMS 1013-item dataset specifically.

Identifying non-actionability of advice. Applying the method to the 1013-item advice dataset identified a significant proportion of current IoT security advice to be non-actionable. The method thus appears to be useful to identify non-actionable advice in a dataset. While its use in this thesis is scoped to IoT advice datasets, we believe that both advice givers and advice recipients in other security areas can also benefit from the method.[12]

Measuring and characterizing actionability. The SAcoding method helps measure not only what proportion of advice in a dataset is actionable, but in characterizing *why* an individual advice item is (or is not) actionable. As the tree directs coders toward tags that characterize a given advice item, the questions at each node along the path contribute to a characterization or description of the assigned tag. For tags that are actionable (all tags after Q_6), the answers to questions leading to those tags characterize each practice tag and what makes them actionable. These characteristics include whether the advice provides a technique or mechanism to use (Q_4), how technically detailed the advice is (Q_5), and which target audience the advice is scoped to (Q_7 and Q_8). In this chapter, these properties are used to characterize actionable advice in the DCMS 1013-item dataset; in Chapter 6 these are used to compare actionability of the DCMS 13 guidelines and ETSI provisions, which are smaller datasets.

Cross-checking target recipients and target codes. The method would appear to be useful for advice givers (dataset creators) to cross-check the following, for advice datasets under creation:

1. that intended target audiences match coding results regarding targets implied by codes P_4–P_6;

2. that advice item wording delivers the higher-level specific types of advice intended by advice givers, with respect to advice categories of actionable practices, principles, approaches/policies, or outcomes; and

3. that items falling into code M_1 (unclear or unfocused) are either clarified or split

[12]Recall from Section 4.2.1 on page 65 that while the coding interface allowed ticking a checkbox to denote items as IoT-specific, neither C1 nor C2 selected this for any of 1013 items.

into finer-grained items, as appropriate;[13] those falling into M_2 (not security related) are removed; and those falling into P_1 (incompletely specified) and P_3 (infeasible) are appropriately revised. These codes are the dark-shaded items in Fig. 4.1.

5.3.2 Limitations of the Coding Tree Methodology

The following are acknowledged limitations of our coding tree methodology.

Findings based on DCMS 1013-item dataset. While we expect the coding tree to be useful on other datasets (both non-IoT security areas and other IoT security datasets), the findings herein (and in Chapter 4) are based on the DCMS 1013-item dataset (Section 3.1.2). Our coding tree questions were designed to be generic and applicable beyond IoT security advice. The upper-tree questions are broad filters to determine if advice generally makes sense (from a language perspective, and if it is focused),[14] lower-tree questions differentiate tags at a more fine-grained level (e.g., Q_{11} differentiates security *design* principles from security principles, Q_8 differentiates practices intended for security experts from those intended for IT specialists), and Q_6 asks whether a practice is viable with reasonable resources, filtering out practices that are not considered feasible. However, while we argue that the coding tree questions are not specific to IoT security advice, we have tested our methodology only on IoT security advice datasets to date.

Limited number of coders. Our full dataset coding exercise involved two coders. The tagging results from the second coder discussed in Section 5.1 show promising reproducibility for determining actionability of an advice dataset, but extending the analysis with a greater number of coders would provide stronger evidence of reproducibility across coders.[15] Our intuition is that if a third coder was to tag

[13]Of advice items that both coders agreed were M_1, in about 55% of cases, both coders did not select the *Unfocused* sub-label, suggesting that most of these items were tagged M_1 due to being not *unambiguous*, rather than not *relatively focused* (cf. Q_1's terminology, Fig. 4.1 on page 54) [33].

[14]Recall that page 65 describes a checkbox on the coding interface, allowing coders to select whether they believed the advice item was specific to IoT. This option was selected for 0 advice items.

[15]Following ACM definitions [25], *reproducibility* would mean to obtain similar results with a different team (coders) using the same experimental setup (SAcoding method and dataset), *repeatability* is for the same team and same experimental setup, and *replicability* is for a different team and different experimental setup.

the 1013-item dataset using the coding tree as it exists currently (i.e., without improvements as suggested in the next subsection), we would see a similar number of agreements on whether items are actionable (Section 5.2.4), but we expect the number of T-agreements (cf. Table 5.2 on page 84) would be similarly low between the new coder and the first two.

Introducing additional coders is difficult for large datasets, as it takes significant manual time for each coder (e.g., to read each advice item and answer a sequence of coding tree questions until reaching a tag for each). One approach to alleviate the coder time burden on individual coders might be to recruit numerous coders and have each tag a randomly-sampled subset of advice items from a full dataset.

Coders of different security experience. We consider the two coders described herein (C1 and C2) to be security experts; however, one has significantly more experience in security. While difficult to measure and not something we had considered in the design and analysis of the coding tree, the difference in experience may have led to advice items and coding tree questions being interpreted/answered differently, leading to different tags and more Q-nonagreements. Examples of where in the coding tree this may cause Q-nonagreements are discussed in Section 5.2.2.

Coding tree methodology not strictly enforced. As mentioned in Section 4.1.1, our software interface tool that coders used to tag advice items did not ensure that coders followed the tree, question by question, allowing the possibility to "short-cut" their answers by selecting a final tag. While coding reported in Chapters 4 and 5 generally avoided use of such short-cut coding, in retrospect, in a preferred implementation the coding tree software interface would force coders to select *yes*/*no* answers until a code is automatically assigned to an advice item.

Advice sub-items not extracted before tagging. As mentioned in Section 5.1.2, a significant number of advice items were tagged with the supplementary code *Unfocused* (an option of *M1*; 14% and 5% for C1 and C2, respectively), meaning they contained numerous sub-topics. Prior to tagging an advice item, if its sub-topics were extracted and tagged individually (instead of together as a composite item), each may have received a different tag than *M1*, altering the results of the full-set tagging. Extracting and tagging sub-topics is conducted on a different dataset in

5.3.3 Avenues for Coding Tree Methodology Improvement

In this subsection, as a summary of the above limitations and discussion throughout this chapter, we briefly describe areas where the coding tree methodology might be improved. In particular, we focus on its questions, tree structure, and instructions provided to coders, and how they might be enhanced to improve a coder's consistency (i.e., ensuring they tag each advice item using the same process each time) or simplify the tree.

Tree structure. Some coding tree questions have more of an impact on the tag given to an advice item than others, and therefore the characterization of an advice item. For example, Q_4 and Q_5 are vital in determining whether an advice item receives an actionable versus non-actionable tag. In contrast, Q_{11} was used infrequently (asked 94 times for both coders combined, Fig. 5.1) and its role is to distinguish two tags that are very similar: a security principle (*N1*) and security design principle (*N1.1*). Arguably, Q_{11} could be removed without largely impacting the coding results, illustrating an area where the tree could be simplified.

Simplified coding tree interface. The coding tree software interface used in the coding exercise of Chapters 4 and 5 was functional, but may have led to inconsistent use of the coding tree by the coders through, e.g., "short-cut" coding (see Section 5.3.2 limitations above). In the case of short-cut coding, the coding tree could be improved to enforce that coders select *yes* or *no* answers for each question, with tags strictly assigned based on the answers, as was our intent.

Additional iteration on tags and questions. The test codings of Chapter 4 involved three coders, and were concluded when acceptably high inter-coder agreement was attained (Section 4.1.1). In Section 5.2 we identified questions where coders had more Q-nonagreements than other questions, suggesting these questions might be further refined to increase answer agreements across coders.

5.4 Related Work

In this section, we discuss work related to qualitative dataset coding [60] (used in our coding tree methodology), to give context on how other security researchers have used this technique.

McDonald et al. [148] note a common lack in research papers of detailed coding process descriptions; in contrast, we explicitly describe our methodology in Chapter 4 and in this chapter, including: number of coders (1 for the initial coding exercise and 2 here in Chapter 5), codebook establishment (page 52), inter-rater reliability of test sets (page 59), and coder agreement calculation (page 79).

The following examples are used for comparison with our work and represent only a small subset of qualitative coding papers in security-related research. Huaman et al. [105] coded user feedback of password managers. After appearing to formally agree on a final codebook (versus describing agreement or calculating inter-rater reliability), they split their dataset among 3 coders who independently coded one third of the dataset each using their final codebook. Kang et al. [120] coded verbalized participant thoughts during drawing tasks about Internet-related tasks. They appear to use 1 coder for coding their dataset and include a second coder for 15% of the dataset to calculate inter-coder agreement. Their process appears to use inter-coder agreement as a cross-check for their coding process, versus for establishing their codebook. Naiakshina et al. [156] coded interview responses about how participants used secure password storage mechanisms. They used 2 coders to independently create codebooks for the full dataset and compared their codebooks using Cohen's kappa (it is unclear how this was done). Krombholz et al. [127] coded verbalized participant thoughts during a system configuration task. They used 2 coders to independently establish codes, agree on a final codebook, code their dataset, and calculate inter-coder reliability (it is unclear if the calculation is on test sets or the final full coding). Ukrop et al. [207] coded interview responses from participants that had evaluated certificate validation status outputs (e.g., warnings, errors). They used 2 coders to independently establish initial codebooks, which were merged through discussion, and the responses were re-coded using the final codebook. A third coder coded half the responses and inter-rater reliability was calculated between the initial

2 coders and the third coder.

While we do not code qualitative data produced by users, the 1013-item dataset used in Chapters 4 and 5 is comprised of qualitative data (security advice items), and we apply coding methods similar to the above examples. We used 3 coders on multiple test sets to iterate on a codebook, where a high final coder agreement (73% mean agreement between 3 coders) and "substantial" [131] inter-rater reliability is achieved ($\kappa = 0.69$ mean; not included in, e.g., [105, 120]). Instead of splitting our dataset across multiple coders (as in [105, 120]), our two coders in this chapter independently tagged the full 1013-item dataset, allowing comparison of full results.

5.5 Concluding Remarks

We conducted an analysis of two coders' results (beyond just the single-coder analysis of Chapter 4) to determine the degree to which the coding tree results are reproducible across two coders; and as coders did not produce identical results, to explore where the coders found challenges in reproducibly tagging a large advice dataset. Notably, coders had high agreement (80%–87%) on which advice items are actionable and which are not actionable (independent of the proportion of actionable tags assigned by each coder; Section 5.2.4), but we were disappointed in the overall coder T-agreement rate ($(308 + 130 + 17)/1013 = 45\%$, from Table 5.2). We also acknowledged several limitations of the coding tree methodology and areas where improvement could be made.

Overall, we believe the results here in Chapter 5, combined with the primary results from Chapter 4, allow insights regarding IoT security advice that were not visible without use of the coding tree. We hope that future work (possibly by others) may extend this work to gain a better understanding of the coding tree's usefulness in measuring and analyzing security advice. We believe our results are useful for practical analysis of security advice beyond the DCMS 1013-item dataset. We apply our methodology to two sets of IoT security advice from the DCMS (distinct from the 1013-item set) and ETSI in the next chapter.

Chapter 6

Comparing Three IoT Advice Datasets Using SAcoding

In this chapter, we first provide an informal comparison and critique of the DCMS 13 guidelines document (distinct from the 1013-item dataset used in Chapters 4 and 5) and the European Telecommunications Standards Institute (ETSI) document of "baseline requirements" for IoT security [68],[1] and identify areas where the ETSI document has improved over the DCMS 13 document. We then apply the security advice coding (SAcoding) method to the advice items in each document (two significantly smaller, coarse-grained sets of IoT security advice) and find that the ETSI provisions document has a greater proportion of actionable advice. This demonstrates the utility of the SAcoding method (versus its reproducibility, as investigated in Chapter 5), including how it can be used on a wide variety of advice datasets.

Combining the comparison and critique, analysis of actionability, and the insights gained through the analyses of Chapters 4 and 5, we describe the characteristics of advice that contribute to actionable security advice for the target audience, and how the coding tree methodology can be used to characterize improvement in different sets of security advice.

The analyses of Chapters 4 and 5 focus on the 1013-item dataset—a fine-grained set of advice, consisting of a variety of advice categories and specific advice items—and critique of our coding tree. Here, we focus on two coarser (higher level) advice datasets, analyze and compare them, and consider ways that advice datasets in general might be improved. Our hope is that understanding both what makes useful security advice and the challenges involved, advice-givers can produce more well-targeted, actionable advice for advice recipients to rely upon.

[1]This is the *ETSI Provisions* document from Table 1.1 on page 4, which appears to be an evolution of the DCMS 13 guidelines document. This is discussed in Section 6.2.

6.1 DCMS and ETSI Document Summaries

In this section we describe the relevant documents for this chapter's analysis and discussion.[2] While not all parts of each document are necessarily relevant to our discussion, each potentially impacts the overall effectiveness of security advice. As we aim to provide an analysis and comparison of security advice datasets from two specific documents (Documents 1 and 2 below), we specifically focus on their primary advice content (i.e., not preamble or non-advice appendices). For context, we first identify the main documents and their relationship to a few others, then give more details on the two main documents in the subsections that follow.

Document 1: DCMS 13 Guidelines [62]. The DCMS "Code of Practice" (October 2018) consists of 13 IoT security guidelines.[3] These guidelines are re-used in-part as the major headings for security topics in the ETSI Baseline Requirements document (next).

Document 2: ETSI Provisions [68]. The ETSI "Baseline Requirements" (June 2020) uses each of the high-level DCMS 13 guidelines (Document 1) as category headings (with minor changes), under which it provides a larger, finer-grained set of "provisions" for advice recipients to follow. This document is positioned as an evolution of the DCMS guidelines, as discussed below.

DCMS 1013-item dataset [59]. This includes the DCMS 1013-item dataset of IoT security advice used in Chapters 4 and 5 as a proxy representative of current IoT security advice to determine the current state of IoT security advice. We mention it here only because it is used in a fourth document, the **DCMS mapping document** [63], which maps each advice item thematically to one of the DCMS 13 guidelines. The mapping document is positioned as a "reference and tool for users of the Code of Practice [guidelines document]" [63], suggesting those looking to follow the guidelines would consult this mapping document for technical details supporting each guideline. We note that the ETSI document suggests the use of the DCMS 1013-item dataset for technical details, albeit without explaining how readers are expected to use it.

[2] Table 1.1 on page 4 also gives a summary of the documents used in this chapter.
[3] Recall that we discussed this DCMS 13 guidelines document in Chapter 3.

6.1.1 Document 1: DCMS 13 Guidelines Document

The DCMS 13 guidelines document (Document 1 above) defines 13 guidelines (Table 3.1 on page 30 gives their titles), designed for pre-deployment stakeholders to use for improving the security of their products and services [62]. These guidelines are positioned as "practical steps", and also as "outcome-focused" (cf. D2 on page 108, and related discussion there), giving stakeholders the flexibility to follow each guideline on their own terms rather than offering specific means to execute them [62].[4] As such, these are high-level guidelines (i.e., lacking in technical detail) that cover a wide variety of IoT security topics. The first three guidelines are intentionally ordered first to signal their priority—these are the guidelines that the DCMS suggests will have the greatest and immediate security impact for a stakeholder [62].

Each of the 13 guidelines follow a basic structure consisting of four parts: (1) a shortened title for the guideline (itemized in Table 3.1), attempting to be capable of conveying the general direction of the guideline and fitting onto a single line; (2) a highlighted box containing the core guideline text itself; (3) text that further explains guideline rationale; and (4) a brief footnotes section (not found on many guidelines) indicating where additional external information can be found. We now elaborate on these parts.

(1) Guideline Title. The guideline title describes the primary topic of the guideline. It seems intended to be short to convey this information for easy comprehension and fit on a single line. The title is related to the core guideline itself, but not necessarily representative of the full recommendation as outlined in the guideline description (Section 6.1.1). We refer to the guideline's title as the *guideline category*, as it is generally representative of the guideline's theme.

(2) Guideline Description. The guideline descriptions seem to be the most important part of each guideline (or are the guideline itself), as they express what is intended for the advice follower to reach, and the text is presented in a distinct pink box. As such, this is the text that we primarily focus on for our analysis. For

[4] We argue this is a contradiction, discussed in Section 6.2.1.

example, the fifth guideline (DCMS-5) has this description [62]:

> *Security-sensitive data, including any remote management and control, should be encrypted in transit, appropriate to the properties of the technology and usage. All keys should be managed securely.*

The guideline description does not seem to aim to express *how* a guideline should be followed, but the goal to reach. This is directly contradicted by their description of being "*practical steps*" (suggesting to us actionability), but then contradicted again with another description of the guidelines being "*outcome-focused, rather than prescriptive*" [62] (suggesting non-actionability—cf. Section 3.2.2).

(3) Further Description/Explanation. Each guideline is accompanied by additional supporting text that provides a non-technical explanation of why the guideline is important, or how it solves an existing problem. While this is its primary use, in a few cases this text includes additional avenues to solving a general problem. In some cases it expands on the guideline description (2); in others it goes off in independent directions.

(4) Guideline Footnote. Two of the 13 guidelines have a footnotes section giving a short list of references to external sources of additional information, or brief clarification on the guideline (e.g., what "competent industry bodies" means [62], then referencing the GSMA [94] and the IoT Security Foundation [112]).

The end of the document gives additional explanation for 6 of the guidelines intending to answer frequent questions. This is offered as "additional explanatory notes" [62], suggesting these are not necessary to be able to carry out the main advice. These sometimes include additional advice or context about why the guideline is important to be followed.

6.1.2 Document 2: ETSI Provisions

An ETSI document titled *Cyber Security for Consumer Internet of Things: Baseline Requirements* [68][5] extends the 13 guideline categories established in the DCMS work.

[5]This is the *ETSI Provisions* document from page 104.

This document appears to be endorsed by the DCMS as an extension of their work, as they list it within the *Secure by Design* project [64], and contribute a portion of its content (particularly the introductory material and guideline titles).

While a portion of each document's description is similar, the DCMS 13 guidelines and ETSI provisions seem to differ in their overall scope. The DCMS guidelines seem to be more about high-level advice that should be followed at the discretion of a manufacturer [62], while the ETSI document is positioned as "technical controls" with the intent to be measurable for compliance when combined with technical details such as those from the DCMS 1013 dataset [59], or advice from ENISA [71], the IoT Security Foundation [193], and the GSMA [93].

The core of the ETSI provisions document revolves around the 13 guideline categories shared (with minor editorial changes) with the DCMS' guidelines (i.e., the guideline title from Section 6.1.1), plus two additional sections for (1) how to report on the implementation of the provided advice, and (2) guidance on how to protect user data processed by a device. As (1) is a meta-guideline (i.e., how to report on the use of the advice, not about security practice itself), we do not consider it in our analysis; however, as (2) is a security topic with implementation details, we chose to include it in our analysis.

Each of the 13 categories has, in this document, an optional category description (a few sentences describing the overall goals of the category) and a series of "provisions". The number of provisions in a category ranges from 1 to 16. Each provision has the following structure: (1) a one-sentence description of what is required by the advice follower or implementer; (2) additional description of the provision; (3) one or more examples of where or how the provision would be applied, as context; and (4) one or more notes which tend to briefly provide context on where a provision may apply. While all four components exist in many cases, a number of provisions only have the basic description text.

6.2 Informal Comparison and Critique of DCMS and ETSI Documents

In this section, we informally compare and critique the DCMS 13 guidelines and ETSI provisions documents. As the ETSI document appears to be designed as an

evolution of the DCMS guidelines document, beyond criticism we note areas where improvements have been made. Each subsection describes and compares select components of the documents. This discussion of improvements helps us characterize what makes useful security advice, and is used in our analysis throughout this chapter.

6.2.1 Positioning of DCMS and ETSI Documents

We first distill our view of the purpose and positioning of each document. Part of our criticism, however, is that the positioning of both documents is unclear, and (we believe) contradictory in places.

DCMS guidelines positioning

The DCMS guidelines document presents four statements describing its positioning [62]:

> *(D1) This Code of Practice sets out practical steps for IoT manufacturers and other industry stakeholders to improve the security of consumer IoT products and associated services.*
>
> *(D2) The guidelines bring together what is widely considered good practice in IoT security. They are outcome-focused, rather than prescriptive, giving organisations the flexibility to innovate and implement security solutions appropriate for their products.*
>
> *(D3) A number of industry bodies and international fora are developing security recommendations and standards for IoT. This Code of Practice is designed to be complementary to and supportive of those efforts and relevant published cyber security standards.*
>
> *(D4) The Code of Practice is supported by a mapping document and an open data JSON file that link each of the Code's guidelines against the main industry standards, recommendations and guidance. This mapping gives additional context to the Code's thirteen guidelines and helps industry to implement them.*

The guidelines are described as practical steps (something to do, actions; D1), but also as high-level outcomes (something to reach; D2) to be achieved by following the supporting set of security advice.[6] By our definitions (Section 3.2.2 on page 37), actions and outcomes are quite different in function. The guidelines are intended to allow advice targets to select methods (to follow the guidelines) appropriate for their devices (D2), suggesting the document should be used as a high-level set of outcomes to try to reach rather than steps to follow. Readers of the guidelines may be misled about whether the guidelines alone are sufficient to reach security goals, or just high-level explanations of what should be done and supported by further external details (which seems to be the intent of D3–D4).

ETSI baseline requirements positioning

The ETSI baseline requirements document provides the following positioning statements [68]:

> *(E1) The present document brings together widely considered good practice in security for Internet-connected consumer devices in a set of high-level outcome-focused provisions. The objective of the present document is to support all parties involved in the development and manufacturing of consumer IoT with guidance on securing their products.*
>
> *(E2) The provisions are primarily outcome-focused, rather than prescriptive, giving organizations the flexibility to innovate and implement security solutions appropriate for their products.*
>
> *(E3) [...] the focus is on the technical controls and organizational policies that matter most in addressing the most significant and widespread security shortcomings.*
>
> *(E4) Overall, a baseline level of security is considered; this is intended to protect against elementary attacks on fundamental design weaknesses (such as the use of easily guessable passwords).*

[6] D2 mentions "good practice"; however, it is unclear whether their use of good practice implies *practices* (as we use the term, cf. Table 3.2 on page 44), or just generally "good things to do".

(E5) The present document provides a set of baseline provisions applicable to all consumer IoT devices. It is intended to be complemented by other standards defining more specific provisions and fully testable and/or verifiable requirements for specific devices which, together with the present document, will facilitate the development of assurance schemes.

E1 and E2 suggest the advice is intended to be high-level and focused on outcomes, but in E1, also good practice.[7] E3 mentions that the advice should be technical controls and organizational policies. The former (technical controls) suggests a greater level of detail when compared to the DCMS guidelines, but not as specific as to suggest technical specification, which they again defer to more detailed documentation elsewhere (E5). E4 describes the advice as protecting against "elementary" attacks, suggesting their advice targets generic versus targeted or niche threats (or those specific to a use case), e.g., threats that are widely applicable to IoT devices. Recall that the DCMS 13 guidelines are ordered to identify advice deemed the most impactful.

The term *provision* is not explicitly defined in the ETSI provisions document, but based on context in the document and our understanding of how it is used therein, we infer that a provision is *intended* to be what we call a practice,[8] as it appears something that advice recipients should do to improve security. Note that we believe provisions are *intended* to be practices, but in many cases those specified in the document lack "specific means" (as required by our definition of *practice*) to reach a goal (the provisions are analyzed in Section 6.3.2).

By our reading, both the DCMS and ETSI documents are positioned similarly as high-level advice to be supported by more detailed advice, but the ETSI document is clearer about how advice can be carried out. Both are useful for understanding the security landscape: the DCMS guidelines provides a high-level understanding of the general direction for security protections, while the ETSI document provides advice closer to (our definition of) practices for how to reach security goals.

[7] As mentioned in footnote 6 on page 109, it is unclear what is meant by "good practice".

[8] From page 33, "a *practice* is a specific means intended to achieve a given desired outcome".

6.2.2 Reference to External Advice

In our view, both documents largely fail to provide direct reference to complimentary security advice when providing high-level guidance, despite mentioning that their advice should be paired with complementary advice from external sources. We now give support for this view.

The DCMS 13 guidelines document suggests (in statements D2, D3, and D4 above) that stakeholders should use advice in the DCMS 1013-item dataset (mapped to each guideline in the mapping document) for technical details to follow the guidelines. The 13 guidelines document, however, does not provide a clear indication of *which* advice items to use within the 1013-item dataset, and apparently expects stakeholders to select on their own. As discussed in Chapter 4, few items in the 1013-item dataset are actionable per our definition, so the document implicitly expects the advice followers to be knowledgeable enough to select appropriate advice from the 1013-item set to follow a guideline.

Similarly, the ETSI provisions document does not provide specific implementation details for many provisions. The document does, however, provide a references section that points to external documents, which are occasionally referenced throughout the provisions. In most cases, these are references to relevant industry organizations, or entire documents (instead of specific lines or sections within a document) from which the reader would be expected to independently locate and extract relevant information. Like the 13 guidelines document, the ETSI provisions document mentions it uses the 1013-item advice dataset [59] (among others) to provide next-level advice. This carries the same question of specifically which next-level advice to use within the 1013-item dataset.

As both documents suggest their guidelines and categories deliver *good practice*, e.g, from the 1013-item advice dataset, it is unclear what exactly is expected of advice recipients when the documents don't actually specify (actionable) practices. Can an IT employee at an IoT manufacturer reliably select a small number of relevant items from the 1013-item dataset and properly execute the advice? Advice recipients may have to select advice from the next-level guidance document that they deem most appropriate to reach their desired security goals.

6.2.3 Target Audience

The DCMS 13 guidelines document targets device manufacturers, IoT service providers, mobile application developers, and retailers [62]. As this is a broad audience, it is unclear what expertise those following the advice are expected to have. ETSI describes their audience as the "*organizations involved in the development and manufacturing of consumer IoT*", which is even less specific than the DCMS audience. While the ETSI document appears to be an evolution of the DCMS document, in this sense it does not improve.

The DCMS 13 guidelines document does explicitly label each guideline with one of the above four target audiences (e.g., the first DCMS guideline "*no default passwords*" is labelled as applying to device manufacturers), so not all guidelines apply to all four targets, but specific groups within each target (e.g., departments) are not described. While here we are critical of the wording used by both documents to describe their target audience, we expect following security advice would be the responsibility of those in an appropriate development or security position within organizations following the advice.

Within technically proficient target groups, do these documents assume it is a security expert (with extensive knowledge and experience in security) that will be following the advice; or a more typical developer or IT specialist, familiar with basic security, but not themselves a security expert? This is important, as both documents expect (stated explicitly for the DCMS 13 guidelines; implicitly for the ETSI provisions) the reader to consult external sources for implementation details, which may or may not be understandable based on one's security expertise. As we believe that it is beneficial for security advice to be actionable for the target audience (Section 3.2.2), we argue that crafting advice specifically for the target audience is important as it contributes to its actionability.

6.2.4 Distinct Advice Topics

Related to our Chapter 4 analysis of *Not useful* advice,[9] within a single DCMS guideline, numerous sub-topics may be offered. For example, DCMS-3 "Keep software updated" states [62]:

> *[1] Software components in internet-connected devices should be securely updateable. [2] Updates shall be timely and [3] should not impact on the functioning of the device. [4] An end-of-life policy shall be published for end-point devices which explicitly states the minimum length of time for which a device will receive software updates and the reasons for the length of the support period. The need for each update should be made clear to consumers and an update should be easy to implement. [5] For constrained devices that cannot physically be updated, the product should be isolatable and replaceable.*

In the above guideline we can see five distinct sub-topics being discussed (we inserted the numbers for exposition): (1) secure updates, (2) timely updates, (3) updates should not interrupt device function, (4) publish policy about update status, and (5) the ability to isolate and replace a device if it can not be updated. Each of these sub-topics could have distinct practices associated with them, but here are combined in a single guideline. Using our coding tree methodology, this guideline would be tagged *Not Useful* (*M1*), with the *Unfocused* supplementary tag given.

In contrast, the ETSI document offers individual provisions that each have a distinct topic. For example, Provision 5.3-1 under the same DCMS-3 heading states [68]:

> "*All software components in consumer IoT devices should be securely updateable*"
>
> [Notes and examples omitted]

This provision is explicitly about secure updates, corresponding to item [1] in the above DCMS guideline. Other sub-topics (2–5) are handled through separate provisions (in the ETSI document) for achieving their own goals.

[9] The *Not Useful* (*M1*) tag and *Unfocused* supplementary tag are described in Section 4.2.3 on page 67.

As a side note, given the 28 sub-topics in the DCMS 13 guidelines (the number of all sub-topics that we extract from the 13 guidelines in Section 6.3.1), the ETSI document, if aiming to cover those same topics, might be expected to have about 28 provisions. However, the number of ETSI provisions is more than twice this, at 67. The ETSI document uses IoT security advice documents found in the 1013-item IoT security advice dataset and additional reputable sources (mentioned in the ETSI provisions document's introduction). It is unclear whether the increased number of advice items is the result of new topics extracted from additional sources, or if similar sub-topics to those found in the DCMS guidelines were sliced into finer-grained topics. In either case, the ETSI document refined source datasets into more specific, fine-grained items. Finally, the observation that the major categories (the titles of each DCMS guideline) were largely retained while incorporating additional advice from further sources suggests that the original categories are largely representative of the general topics that IoT security advice-givers find important to address.

We believe it is reasonable to expect that advice recipients are able to extract sub-topics as we do above. Our analysis in Section 6.3 takes this into account by independently analyzing the DCMS guidelines from two perspectives: each guideline as presented in the source document, and with each sub-topic extracted and used as individual advice items (akin to ETSI's provisions).

6.2.5 Technical Content

As discussed in Section 6.2.1 (page 108), we view the positioning of the DCMS guidelines [62] and ETSI provisions [68] as self-contradictory, claiming to offer "practical steps" or "technical controls" (D1 and E3), but also describing their advice as "outcome-focused" (D2 and E2).

If the intention is to offer practical steps or technical controls, we expect that there would be technical details included to provide advice targets clear instruction on how to execute the advice. If an advice item is vague about techniques expected to be used, it is unlikely to be categorized (by our coding tree) as an actionable practice, and we expect that some advice recipients would be unclear about how to execute it without a further external source for technical details. The DCMS guidelines (and

the sub-topics therein) often briefly mention a technical approach (the topic of Q_4 in our coding tree), but offer no explicit or obvious implicit steps or actions to take (Q_5 in the coding tree).

As an example of a guideline lacking (in our view) sufficient technical details to enable reliable execution, DCMS-4 states [62]:

> *Any credentials shall be stored securely within services and on devices. Hard-coded credentials in device software are not acceptable.*

Not using hard-coded credentials would generally be regarded as a clear request not requiring further technical detail. In contrast, the first sentence requests credentials be "stored securely" without explaining that phrase (nor specific steps on how to achieve that, if the guideline aimed to be actionable by our definition). As technical details are largely absent from the DCMS guidelines document, it appears the document, despite being positioned ambiguously,[10] is intended as high-level advice with the apparent expectation that external references be sought for technical detail of how to execute advice (indicated in D4).

ETSI provisions commonly suggest a technique or tool to use for achieving a goal, but often do not describe further execution detail (this corresponds to the *Incompletely Specified Practice* code in the coding tree). They often also include an example, which provides additional context or describes a real-world scenario where application of the provision would be beneficial for security.[11] While such examples do not appear intended to replace technical detail, they offer additional context that may help advice followers understand how to follow the advice. Like the DCMS guidelines, it appears the intent is that the advice is primarily to be used with external sources for technical details.

Summary of informal comparison. From our informal analysis above, our view is that the DCMS 13 guidelines and ETSI provisions are positioned differently. As we are interested in IoT security stakeholders being able to execute the provided

[10] Section 6.2.1 describes the DCMS 13 guidelines document's ambiguous positioning.

[11] An example of a provision with an example is included in Provision 5.2-1 on page 32 of Chapter 3.

advice, we note that the ETSI document appears to contain more technical content and details than the DCMS guidelines (improving actionability).[12] We briefly summarize each above subsection here.

- **Positioning.** Both documents propose what are described as "outcome-focused" advice for consumer IoT devices, suggesting they want to give high-level advice, but contradict this (in our view) by also positioning their advice as "practical steps" (DCMS) and "technical controls" (ETSI), setting up an expectation to deliver details enabling advice recipients to execute the advice.

- **Reference to external advice.** Both documents rely on often vague references to external sources for more specific details of how to execute advice or reach security goals, leaving it unclear about which sources to follow to reach their goals.

- **Target audience.** Particularly in the ETSI provisions document, the target audience is vague, making unclear the required knowledge level of the audience expected to be executing the advice. The DCMS 13 guidelines document details which of 4 general audiences (described in Section 6.2.3) each guideline is intended for.

- **Distinct advice topics.** Both documents categorize their advice in similar ways using high-level categories. However, the ETSI provisions document separates individual sub-items within each category into stand-alone provisions, whereas the DCMS 13 guidelines are contained within a single block of text.

- **Technical content.** The DCMS 13 guidelines provide little technical detail for how to follow the advice, limiting the advice's actionability by omitting technical details that might help advice recipients reach security goals. The ETSI provisions generally provide more technical detail, but it appears the intent of both documents is for advice recipients to reference external advice for further execution detail (see above).

[12] Recall our view (Section 3.2.2 on page 37) that actionability is an important characteristic of security advice that allows it to be reliably executed.

6.3 Analysis of Actionability Using the Coding Tree

In this section we apply our coding tree tool (introduced in Chapter 4) to the DCMS 13 guidelines and the ETSI provisions to determine what proportion of the provided advice is actionable by our definition. This allows a baseline measurement of the actionability of the DCMS guidelines, and for a comparison to the ETSI provisions. From this, we proceed to make observations of whether improvements have been made in the ETSI provisions over the DCMS guidelines. We aim to support more formally, via the coding tree methodology, the preliminary view from the informal analysis of Section 6.2 that the ETSI provisions are an improvement over the DCMS guidelines.

6.3.1 Analysis Methodology

We apply our coding tree methodology to three datasets: 2 based on the DCMS guidelines, and 1 based on the ETSI provisions. We describe how these are established here.

From the DCMS 13 guidelines document, we build two sets of advice. For the first, we extract each guideline as a whole, considering the entire guideline as one advice item. This is done to assess the actionability of the guidelines (as a set) while considering the document's advice as presented. In total there are 13 items in this set (one for each full guideline). We call this set the DCMS *Full* guideline dataset.

For the second, we extract all sub-topics from each DCMS 13 guideline (sub-topic extraction is described in Section 6.2.4). This is done to assess each sub-topic as independent advice items (versus how actionable the DCMS *Full* dataset is). In total there are 28 items in this set. We call this the DCMS *Sub-Topics* guidelines dataset.

From the ETSI provisions document we extracted each provision (including their associated examples and notes) from each of the 13 major categories and the one new category (on how to protect user data used by the device, Section 6.1.2). This yielded 67 advice items (provisions). As each ETSI provision is largely self-contained and typically focuses on a single topic (like each of the DCMS sub-topics), ETSI provisions are not subdivided. We call this set the ETSI *Provisions* dataset.

In summary, the above extractions resulted in three datasets that we compare.

- DCMS *Full* guidelines (13 items)
- DCMS *Sub-Topics* guidelines (28 items)
- ETSI *Provisions* (67 items)

For each dataset, the coding tree methodology and interface tool is applied to each advice item in the set to determine its code; recall that each question in the coding tree is answered about the advice item until a code is assigned. As discussed on page 55 (Fig. 4.2), each code is pre-classified as being either actionable or non-actionable, allowing a count of how many actionable advice items there are in each dataset (thus the proportion of each set that is actionable). This was done for all three sets by the coder (thesis author) that performed the Chapter 4 tagging of the 1013-item dataset (Coder C1 from Chapter 5's analysis).[13] The methodology used was identical to that in Section 4.2.

6.3.2 Results

Recall that *DCMS-i* and *ETSI-i* indicate the same 13 category headers (page 104). The headers convey the same or very similar information (e.g., DCMS' "*Monitor system telemetry data*" [62] versus ETSI's "*Examine system telemetry data*" [68]), but the content of the DCMS guidelines differs from the content of ETSI provisions. We analyze the content.

Table 6.1 shows the results for the tagging of both the DCMS Sub-Topics set and the DCMS Full guideline set. Note that only the tags represented in the analysis results are shown in the table; unused tags are omitted. For the Full guideline tagging, only 1 of the 13 guidelines (DCMS-1) was tagged as actionable (specifically, *P5*). For sub-topic tagging, 7 of 28 sub-topic items (25%) were tagged *P5* (the only actionable tag that appeared in the results). The remaining 75% are non-actionable tags.

[13]This chapter's analysis focuses on the utility of the SAcoding method and how it can be applied to different datasets, versus investigating reproducibility across coders (Chapter 5).

Table 6.1: Results of DCMS sub-topic and full guideline coding. *n* indicates number of sub-topics manually extracted. *Sub-Topics* headers are leaf node codes. *Full* column shows codes assigned to each individual guideline (i.e., DCMS-i). ∗ denotes actionable codes, see Fig. 6.1. Page 121 explains absence of codes P3, P4, P6, and M2. Relative code frequency illustrated in Fig. 6.1.

		Frequency of code assigned to Sub-Topic					Full
Guideline	n	P1	P2	*P5	N1.1	T	
DCMS-1	1			1			*P5
DCMS-2	2	1		1			M1
DCMS-3	5	1		1		3	M1
DCMS-4	2	1		1			M1
DCMS-5	2	1				1	M1
DCMS-6	1				1		N1.1
DCMS-7	2	1		1			M1
DCMS-8	3	1		2			M1
DCMS-9	4	1				3	M1
DCMS-10	1	1					P1
DCMS-11	2	1	1				M1
DCMS-12	2	1	1				M1
DCMS-13	1	1					P1
Total (28)		11	2	7	1	7	
Proportion of Total		39.3%	7.1%	25.0%	3.6%	25.0%	

Note that only 5 of the 11 tags available (from Fig. 4.4 on page 56) appear in the Table 6.1 Sub-Topics results, and 4 of the 11 tags appear in the Full results (6 distinct tags in total).

Table 6.2 shows the results of tagging the ETSI provisions. Across the 67 provisions, 7 different tags appear in Table 6.2 (the same tags as in Table 6.1, plus *P4*). A combined 43% of provisions were tagged with actionable tags (in this case, including both *P4* and *P5*).

Figure 6.1 summarizes (based on data from Tables 6.1 and 6.2) the proportion of the advice from the three analyses (DCMS *Full* and *Sub-Topics*, and the ETSI *Provisions*) that was tagged with each code. The next subsection gives our interpretation of these results.

6.3.3 Interpretation of Results and Comparative Analysis

Based on our methodology (Section 6.3.1), the ETSI provisions are an improvement over the DCMS 13 guidelines in terms of proportion of actionable advice.

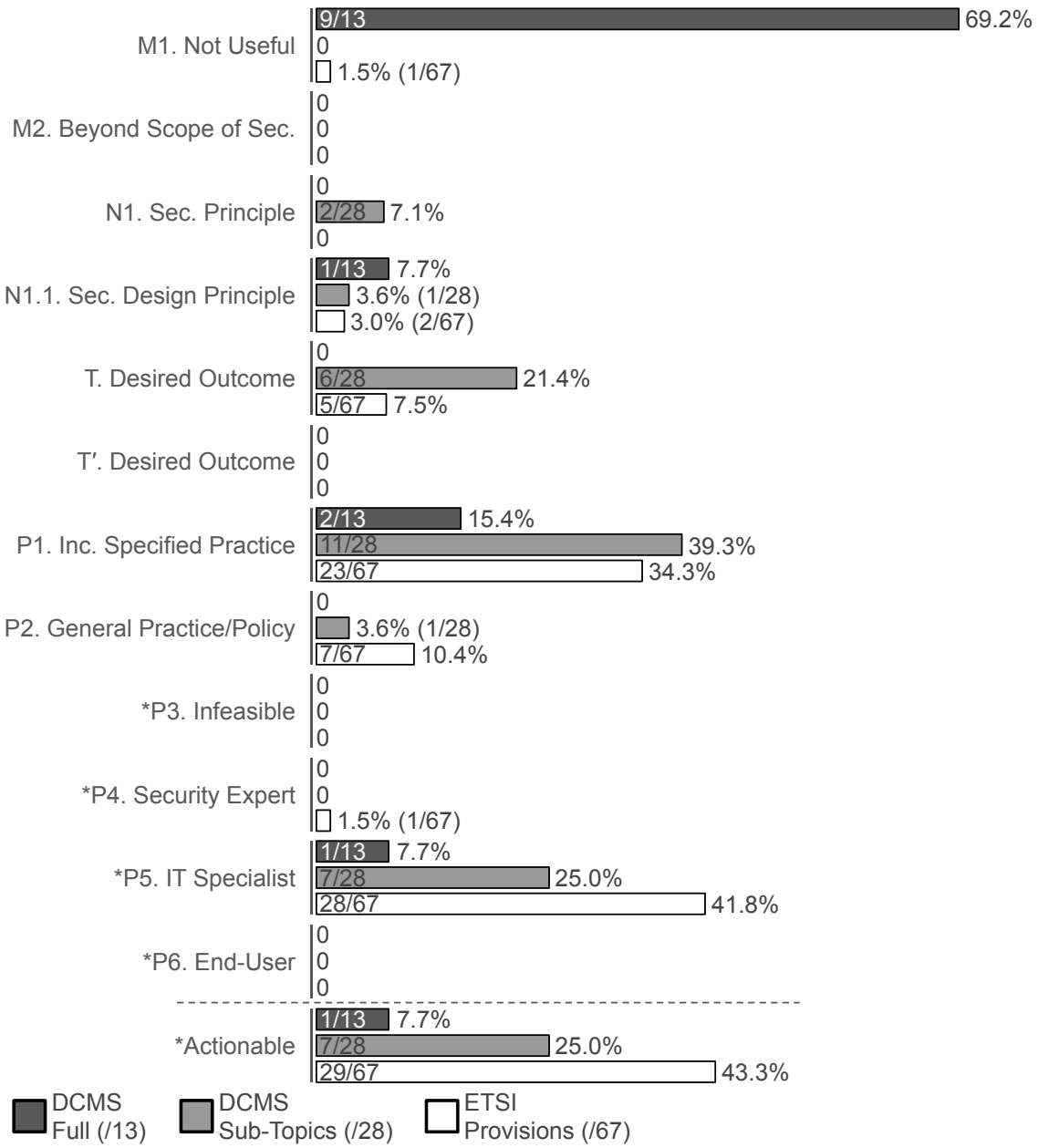


Figure 6.1: Tag distribution across advice sets and actionability (data from Tables 6.1 and 6.2). Graph depicts the proportion of advice items in each dataset tagged with each code. *Actionable* bars depict proportion of each set tagged with actionable codes. Tagging based on the coding tree methodology of Chapter 4. An asterisk (*) indicates codes we define as actionable. See Fig. 4.4 on page 56 for code descriptions.

Table 6.2: Results of ETSI provisions coding. Parenthesized numbers in column 1 are the number of provisions. Other column headings correspond to tags of coding tree leaf nodes. $*$ denotes actionable tags, see Fig. 6.1. Page 121 explains absence of codes P3, P6, and M2. Code frequency compared in Fig. 6.1.

	n	P1	P2	*P4	*P5	T	N1.1	M1
ETSI-1	5	2			3			
ETSI-2	3		1		2			
ETSI-3	16	5	4		5	1		1
ETSI-4	4			1	3			
ETSI-5	8	5	1		1		1	
ETSI-6	9	2	1		3	2	1	
ETSI-7	2	1			1			
ETSI-8	3	2			1			
ETSI-9	3				1	2		
ETSI-10	1	1						
ETSI-11	4	2			2			
ETSI-12	3	1			2			
ETSI-13	1	1						
ETSI-DP	5	1			4			
Total (67)		23	7	1	28	5	2	1
Proportion of Total		34.3%	10.4%	1.5%	41.8%	7.5%	3.0%	1.5%

As a first observation, the *Incompletely Specified Practice* (*P1*) code was used 15.4%, 39.3%, and 34.3% of the time for the DCMS Full, DCMS Sub-Topics, and ETSI Provisions sets, respectively (Fig. 6.1). The frequency with which this code appeared (through use of the coding tree) signals to us a heavy reliance on external advice—without technical detail from the advice item itself or in a reference to an external resource for additional detail, these items are not actionable (by our definition). Note that the DCMS Full results appear to have significantly fewer *P1* codes than the DCMS Sub-Topics and ETSI Provisions results (which, intuitively, appears desirable), but this is perhaps misleading, as in 9/13 cases (Table 6.1) the Full guidelines did not have an opportunity to reach *P1* in the coding tree due to being immediately assigned *M1*.[14]

Like the 1013-item dataset analyzed in Chapters 4 and 5, the *Beyond the Scope of Security* (*M2*) and *Infeasible Practice* (*P3*) tags were not assigned to any items of the three sets used in this chapter (Fig. 6.1). This is likely for the same reason as

[14]Recall from Fig. 4.1 on page 54 that Q_1 asks whether an item is relatively focused, and many of the Full guidelines result in a *no* answer to Q_1, being assigned *M1*.

explained in Chapter 5.[15] *P4* (*Security expert*) was used 1 time in the ETSI Provisions set and *P6* (*End-User*) was used 0 times, suggesting both advice documents matched their asserted target audience (manufacturers and other pre-deployment stakeholders, Section 6.2.3).

As Fig. 6.1 highlights, the ETSI Provisions dataset substantially improves the proportion of actionable advice within the dataset (43.3% versus 7.7% and 25.0% for DCMS Full and Sub-Topics guidelines sets, respectively). As a side note, while we expected that all three datasets would be less actionable than the next-level detail they reference (32–33% of the 1013-item dataset, Fig. 5.1),[16] our analysis here finds that one of these, the ETSI Provisions, is more actionable (cf. Fig. 6.1). We attribute this improvement to the ETSI Provisions having been refined into a clearer, focused set of 67 items.

Below we discuss the results of each set tagging, and how the ETSI provision set improves over the two DCMS sets.

DCMS 13 guidelines (tagging results and actionability)

It is unsurprising that the DCMS full guidelines containing more than one topic are so frequently tagged *M1* (*Not Useful—too vague/unclear or multiple items*) when considered in their entirety. This suggests that our coding tree methodology is useful for identifying vague and unclear items (signalling these as candidates for clarification); from above, we see that it can provide more insightful analysis results if such unfocused advice items are manually preprocessed and separated into finer-grained advice items. If producing actionable security advice is the goal of an advice giver, the advice items in source documents themselves likewise should be refined into narrower individual items confined within a single topic rather than in groups spanning several topics. This is supported by Table 6.1.

Our tagging of the DCMS guideline sub-topics provided a more granular look at the advice. While our view is that the guidelines were intended to be digested as

[15] Page 78 explains that when advice in the 1013-item dataset is actionable, it is almost always feasible to carry out (Q_6); and that advice intended to be security advice is rarely assigned *M2* at Q_2.

[16] Recall that both the DCMS 13 guidelines document and ETSI provisions document suggest use of the 1013-item dataset for next-level detail.

a whole,[17] our results show that it appears to be more useful (i.e., the sub-topics individually are more actionable, as measured by our methodology, than together, Fig. 6.1) to consider each sub-topic as a distinct piece of advice if actionability is important to the advice follower.

Table 6.1 shows *M1* as the most common (by far) code in the Full dataset, but did not appear in the results from the Sub-Topics set—where sub-topics were deliberately extracted and apparently all items were understandable and contain only one topic, corresponding to a *yes* answer to Q_1. All *M1* codes in the Full set were due to the *Unfocused* supplementary topic being selected by the coder.[18]

Considering the sub-topics individually, 25% of the sub-topics were categorized as actionable (Table 6.1). This drops to 8% if we consider the Full guidelines (1 of 13 guidelines was tagged as actionable). As a side point, in both cases, some items were actionable by IT specialists (corresponding to code *P5*, occurring proportionally as a ratio 1/13 and 7/28 for Full and Sub-Topics sets, respectively), but none were tagged as *P4* (security experts). This suggests that the sub-topics that *are* actionable were appropriate for IT specialists, but not requiring the further expertise of security experts. While few in number, we view the occurrence of *P5* (instead of *P4*) positively, as we expect IoT device manufacturers to typically have IT specialists developing devices. Thus, the subset of Sub-Topics advice that is actionable matches what we believe to be the target audience.

ETSI provisions (tagging results and actionability)

For the ETSI tagging, 42% of advice (Fig. 6.1) was tagged as *Specific Practice—IT Specialist* (*P5*). One practice (1/67), in ETSI-4 in Table 6.2, was tagged as requiring a Security Expert (*P4*). Of actionable codes, 28/29 are *P5*, matching what we believe is the target audience (IT specialists rather than security experts). A greater proportion of the ETSI Provisions set were tagged as *P5* than the DCMS Sub-Topics set (41.8% vs. 25%), implying the ETSI provisions are more appropriate than the DCMS sub-topics for what we believe is the ideal target audience for the

[17] This is based on the guidelines commonly being presented as a paragraph and within a highlighted section; see Section 6.1.1 on page 105.

[18] *Unfocused* code described in detail in Section 4.2.3 on page 67.

advice. Combined, actionable codes make up 43.3% of ETSI provisions.

Comparing the ETSI Provisions advice set to the DCMS Full and Sub-Topics sets, the ETSI Provisions set is significantly more actionable (see the *Actionable* bars of Fig. 6.1).

Comparative improvement in ETSI baseline requirements

We now use our results from this section so far (Section 6.3) to compare our two main advice set documents in three selected areas, noting several aspects from the informal analysis in Section 6.2 that improve the actionability of ETSI's provisions.

(1) Actionability of advice. The ETSI provisions are considerably more actionable than the DCMS sub-topics and single guidelines. The actionability of the DCMS sub-topics, at 25%, more than triples the corresponding 7.7% for the single guidelines, but is itself almost doubled by ETSI provisions (43% actionable). We attribute this largely to the improvement in technical detail among ETSI provisions (next item).

(2) Technical detail included in advice. For the DCMS Sub-Topics set, a combined 61% (17/28) evoked a *yes* at Q_4 (Fig. 4.1 on page 54), suggesting they described a security technique, mechanism, software tool, or specific rule. This is the first technical question toward an actionable code. 11 of these 17 items exited the path toward an actionable code at Q_5 (asking: does the advice item describe or imply steps to take), moving instead to *Incompletely Specified Practice* (*P1*). In the case of the DCMS Sub-Topics set, this was typically because of a lack of technical detail in the advice. In contrast, for the ETSI provisions, 78% of the 67 items progressed to an actionability path at Q_4 and 43% likewise at Q_5. We interpret the proportion of a dataset that is actionable as a signal of how technically detailed the dataset is—an advice item reaching Q_6 (beyond which all tags are actionable) has enough technical detail to be considered actionable. Thus, we observe a significant improvement in the level of technical detail in the ETSI Provisions set (43.3% of ETSI provisions answered *yes* to Q_5 versus 25% of DCMS sub-topics).

(3) Fewer incomplete practices. For the DCMS Sub-Topics set, 39.3% were tagged as *Incompletely Specified Practice* (*P1*). This drops to 34.3% for the ETSI

document, suggesting that when ETSI provisions are able to make it to Q_5 about technical steps, they continue on the actionable path by specifying technical steps more frequently than the DCMS sub-topics.[19] While only 15.4% of the guidelines from the DCMS Full set were tagged as *P1* (significantly fewer incomplete practices than the other two sets), as discussed on page 121, this is explained by the early assignment of the *M1* tag rather than a specific improvement in the DCMS Full set advice.

In summary, using the coding tree methodology, we analyzed two security advice documents from authoritative sources, and identified that the ETSI provisions improve on the DCMS guidelines in proportion of actionable advice, increased technical detail, and a reduction in the number of incomplete practices.

6.4 DCMS Guidelines and ETSI Provisions Coding Output

Tables 6.3 and 6.5 respectively show our coding for both DCMS datasets (Full and Sub-Topics) and the ETSI Provisions from this chapter.

[19] A *yes* to Q_5 branches to actionable codes (*P3–P6*), while answering *no* yields the *P1* leaf.

Table 6.3: Our coding of the DCMS Full guidelines (13 items) and Sub-Topics (28 items), using the SAcoding method. DCMS Full guidelines coded using guideline description (versus title, as included here). Table 6.1 on page 119 is derived from this data.

Guideline	Full Code	Sub-Topic Code	Sub-Topic Text (from [62])
DCMS-1	P5		• DCMS-1 title: **No default passwords**
1.1		P5	All IoT device passwords shall be unique and not resettable to any universal factory default value.
DCMS-2	M1		• DCMS-2 title: **Implement a vulnerability disclosure policy**
2.1		P5	All companies that provide internet-connected devices and services shall provide a public point of contact as part of a vulnerability disclosure policy in order that security researchers and others are able to report issues.
2.2		P1	Disclosed vulnerabilities should be acted on in a timely manner.
DCMS-3	M1		• DCMS-3 title: **Keep software updated**
3.1		T	Software components in internet-connected devices should be securely updateable.
3.2		P1	Updates shall be timely and [...]
3.3		T	[Updates] should not impact on the functioning of the device.
3.4		P5	An end-of-life policy shall be published for end-point devices which explicitly states the minimum length of time for which a device will receive software updates and the reasons for the length of the support period. The need for each update should be made clear to consumers and an update should be easy to implement.
3.5		T	For constrained devices that cannot physically be updated, the product should be isolatable and replaceable.
DCMS-4	M1		• DCMS-4 title: **Securely store credentials and security-sensitive data**
4.1		P1	Any credentials shall be stored securely within services and on devices.
4.2		P5	Hard-coded credentials in device software are not acceptable.
DCMS-5	M1		• DCMS-5 title: **Communicate securely**
5.1		P1	Security-sensitive data, including any remote management and control, should be encrypted in transit, appropriate to the properties of the technology and usage.
5.2		T	All keys should be managed securely.
DCMS-6	N1.1		• DCMS-6 title: **Minimise exposed attack surfaces**
6.1		N1.1	All devices and services should operate on the 'principle of least privilege'; unused ports should be closed, hardware should not unnecessarily expose access, services should not be available if they are not used and code should be minimized to the functionality necessary for the service to operate. Software should run with appropriate privileges, taking account of both security and functionality
DCMS-7	M1		• DCMS-7 title: **Ensure software integrity**
7.1		P1	Software on IoT devices should be verified using secure boot mechanisms.
7.2		P5	If an unauthorised change is detected, the device should alert the consumer/administrator to an issue and should not connect to wider networks than those necessary to perform the alerting function.

Table 6.4: Table 6.3 continued.

Guideline	Full Code	Sub-Topic Code	Sub-Topic Text (from [62])
DCMS-8	M1		• DCMS-8 title: **Ensure that personal data is protected**
8.1		P5	Where devices and/or services process personal data, they shall do so in accordance with applicable data protection law, such as the General Data Protection Regulation (GDPR) and the Data Protection Act 2018.
8.2		P5	Device manufacturers and IoT service providers shall provide consumers with clear and transparent information about how their data is being used, by whom, and for what purposes, for each device and service. This also applies to any third parties that may be involved (including advertisers).
8.3		P1	Where personal data is processed on the basis of consumers' consent, this shall be validly and lawfully obtained, with those consumers being given the opportunity to withdraw it at any time.
DCMS-9	M1		• DCMS-9 title: **Make systems resilient to outages**
9.1		T	Resilience should be built in to IoT devices and services where required by their usage or by other relying systems, taking into account the possibility of outages of data networks and power.
9.2		T	As far as reasonably possible, IoT services should remain operating and locally functional in the case of a loss of network and [...]
9.3		T	[Devices] should recover cleanly in the case of restoration of a loss of power.
9.4		P1	Devices should be able to return to a network in a sensible state and in an orderly fashion, rather than in a massive scale reconnect.
DCMS-10	P1		• DCMS-10 title: **Monitor system telemetry data**
10.1		P1	If telemetry data is collected from IoT devices and services, such as usage and measurement data, it should be monitored for security anomalies.
DCMS-11	M1		• DCMS-11 title: **Make it easy for consumers to delete personal data**
11.1		P1	Devices and services should be configured such that personal data can easily be removed from them when there is a transfer of ownership, when the consumer wishes to delete it and/or when the consumer wishes to dispose of the device.
11.2		P2	Consumers should be given clear instructions on how to delete their personal data.
DCMS-12	M1		• DCMS-12 title: **Make installation and maintenance of devices easy**
12.1		P1	Installation and maintenance of IoT devices should employ minimal steps and should follow security best practice on usability.
12.2		P2	Consumers should also be provided with guidance on how to securely set up their device.
DCMS-13	P1		• DCMS-13 title: **Validate input data**
13.1		P1	Data input via user interfaces and transferred via application programming interfaces (APIs) or between networks in services and devices shall be validated.

Table 6.5: Our coding of ETSI Provisions dataset (67 items), using the SAcoding method. Table 6.2 on page 121 uses this data.

Provision	Code
1-1	P5
1-2	P1
1-3	P1
1-4	P5
1-5	P5
2-1	P5
2-2	P2
2-3	P5
3-1	T
3-2	P1
3-3	P1
3-4	M1
3-5	P1
3-6	P5
3-7	P1
3-8	P2
3-9	P1
3-10	P5
3-11	P5
3-12	P5
3-13	P2
3-14	P2
3-15	P2
3-16	P5

Provision	Code
4-1	P5
4-2	P4
4-3	P5
4-4	P5
5-1	P1
5-2	N1.1
5-3	P1
5-4	P5
5-5	P2
5-6	P1
5-7	P1
5-8	P1
6-1	P1
6-2	T
6-3	T
6-4	P5
6-5	P5
6-6	P5
6-7	N1.1
6-8	P1
6-9	P2
7-1	P1
7-2	P5

Provision	Code
8-1	P1
8-2	P1
8-3	P5
9-1	T
9-2	T
9-3	P5
10-1	P1
11-1	P1
11-2	P1
11-3	P5
11-4	P5
12-1	P1
12-2	P5
12-3	P5
13-1	P1
DP-1	P5
DP-2	P5
DP-3	P5
DP-4	P1
DP-5	P5

6.5 Related Work

Discussion about the development of security advice often focuses on the *usability* of advice (e.g., [3, 104, 173] below). While we focus on pre-deployment IoT security stakeholders (i.e., before an end-user receives a device), others discuss the impact of the usability of general security advice on end-users of IoC devices (e.g., [104, 174] below). Related work on IoT security advice often focuses on the technical content of the practices themselves (see Chapter 4's related work section). In this section, we discuss related work on characterizing security advice and what makes advice useful (or not) for followers (versus the technical content of IoT security advice in Chapter 4, or coding methodologies in Chapter 5).

Redmiles et al. [173] measure the readability of security advice from both expert and non-expert online sources. In our work, readability is a crucial aspect in actionability (you cannot follow advice if you cannot understand it). Our coding tree methodology (Section 6.3.1) handles readability by disqualifying advice at the first branch of the coding tree if it is vague or unfocused (Q_1 of Fig. 4.1 on page 54). In another paper, Redmiles et al. [174] collected and analyzed a large set of end-user security advice and found, through a user-study, most advice to be perceived as actionable, but users are unclear about which advice is most important to execute. Herley [104] studied the costs and benefits of following security advice for end-users, and suggests that users reject common security advice because on average it costs them (effort required to comply) more to follow the advice than it benefits them.

Acar et al. [3] note that software developers often find security advice lacking (e.g., out of date, missing concrete examples, missing important topics), and find technical detail or pointers to external advice to be frequently missing from advice given by major organizations (e.g., Microsoft, Mozilla, OWASP). Renaud [175] notes in small and medium-sized enterprises (SMEs), a large amount of security advice leads to uncertainty and confusion, and suggests that advice-giving organizations need to agree on a reduced number of simple, clear sets of security advice. Assal and Chiasson [24] note that available informational resources for software development security practices vary in their level of technical detail, and report frequent non-compliance with common software development security advice, e.g., circumventing annoying or frustrating security features that hinder development speed. RFC 2119 [47] (a "best current practice" document), defines specific words such as "MUST", "SHALL", "MUST NOT" (among others) that describe to readers how advice in RFCs should be interpreted. While not directly about security advice, using well-defined (and well-adopted) terms such as these help clarify what is expected of the advice follower.

Beyond the above papers on usability and usefulness of general security advice (whether for end-users [81, 104, 174] or software developers [3, 24, 175]), we are not aware of any related work that focuses on crafting actionable security advice specifically for pre-deployment IoT stakeholders. This said, as IoT security advice overlaps

heavily with IoC advice,[20] it is expected that findings about developing actionable security advice for IoC (including the research mentioned above) would be directly applicable to IoT.

6.6 Concluding Remarks

Given IoT's rapid growth and popularity over the past two decades, many organizations have offered security advice to IoT security stakeholders. While the advice appears to be well-intentioned, it is unclear how actionable the advice is for different IoT security stakeholders (recall that our main focus is manufacturers and software/hardware developers). Actionable advice can then lead to adoption of strong security development practices, which have unfortunately been found to be lacking in consumer IoT [14].

Our coding tree methodology enabled us to explore the improvement in actionability of IoT security advice from the UK government's DCMS 13 guidelines [62] to the advice from ETSI's provisions [68]. We suggest that use of the coding tree and analysis of these documents may serve as a basis from which to begin a measurable process of improving security advice (particularly its actionability; and the components of security advice documents such as its positioning, references to external advice, and target audience). However, from the results of our analysis of the DCMS guidelines and ETSI provisions documents, constructing sets of actionable security advice may not be as simple as the wide breadth of existing advice would make it seem, and what is positioned to be widely "appropriate" advice may be ineffective for mismatched audiences (e.g., expert level security advice used by non-experts).

Despite noting improvement, our analysis found both the DCMS 13 guidelines and ETSI provisions advice sets leave room for improvement, suggesting more work is needed not only in generating actionable advice, but also on means to evaluate security advice itself. The design and use of the coding tree methodology has focused on characteristics of security advice that we believe are important. Our analysis and

[20]Neither coder from Chapter 5 selected the option in the coding tree interface that denotes an advice item as IoT-specific, suggesting IoT and IoC advice are similar in nature. This, in part, motivated our decision to design the coding tree methodology for security advice in general rather than specifically for IoT advice. See page 65 explanation.

discussion in Section 6.3 suggests the following characteristics be considered when constructing actionable security advice.

Explicit declaration of target audience. An explicit declaration (and characterization) of the target audience of security advice is essential in our view. This plays directly into whether advice is actionable or not, as advice is often tailored specifically for an audience, and to their knowledge. Without this declaration, unintended audiences who use the advice may struggle to understand what is expected, or lack details to successfully (and reliably) execute advice.

Appropriate level of technical details. A level of detail appropriate to the target audience and matching the positioning of the document should be included. This follows our definition of an actionable practice and target audience is a concept built into our coding tree methodology. In our view, the level of detail should be enough to provide the target recipient—implicitly or explicitly—unambiguous and clear steps for how to reach a desired outcome. This may be accomplished by advice that contains details itself (enough so an advice recipient can understand, e.g., the technical mechanisms required), or reference external sources for further details (see next item).

Pointers to next-level detail. Related to the previous item, if advice itself lacks sufficient technical detail to be directly followed by the target audience, it should provide specific references to resources containing the necessary details.[21] Finally, while a perhaps obvious point, we suggest that ideally, a referenced document not itself send the advice recipient down a long chain of further references: an important detail three levels further may overwhelm, and later documents may be written for a different target audience.

Fine-grained advice items. It appears beneficial to package advice as fine-grained items addressing narrower or single topics. Our systematic analysis comparing the DCMS Full and DCMS Sub-Topics datasets supports this—when blocks of advice text from the Full set were split into extracted sub-items for the Sub-Topics

[21] This is not specific to IoT security advice—another specific example of the usefulness of pointing to additional technical information is for technical documentation regarding certificate validation errors [207].

set, all *Not Useful* (*M1*) codes were replaced by codes conveying more meaningful information reflecting sub-item content. Analysis of the ETSI Provisions, which are finer-grained than the DCMS Full set, also supported this view; the provisions attracted in total just one *Not Useful* code among 67 items.

In addition to these characteristics, we believe that security advice documents should have a clear indication of whether the advice provided is intended to convey, e.g., practices, guidelines, or requirements, to avoid contradictions such as in the DCMS 13 guidelines where advice was suggested as being both actionable and outcomes to reach [62]. Ambiguously positioned advice may cause confusion about whether it is expected to be followed as presented to reach security goals, or if additional details are needed. As we discuss in Chapter 3, the terminology used to convey security advice is important, and often signals how it is designed, or expected to be used. A document containing practices (possibly called *best practices*), by our definition (Section 3.2.2 on page 38), is more actionable than one guiding towards only high-level outcomes.

We believe that these recommendations provide useful guidance to those who themselves provide guidance, i.e., advice-givers as they craft security advice intended to clearly and coherently guide advice recipients.

Chapter 7

Explication of IoT Device Identification

In this chapter, we revisit the question: *How is IoT security different from IoC security?* In Chapter 2 we examined the general differences; in this chapter we view the differences between IoT and IoC through a focus on *device identification*, and analyze how IoT devices are differentiated from other devices. Our focus in this chapter is the identification of consumer IoT devices, and we exclude, e.g., autonomous vehicles and smart cities/infrastructure from our scope (see Chapter 1).

In the context of IoT, from our background reading we began from the view that *device identification* is an overloaded term (i.e., used in many, or ambiguous ways) and often not explicitly defined, requiring a reader to assume their own implicit understanding of this term. One popular subtopic is authentication of IoT devices, which (as we will explain) has many similarities with authentication in IoC. Authentication of resource-constrained devices is well-studied (as exemplified through recent proposals we examine herein, and historical examples such as improvements to Bluetooth spanning more than 20 years [218]), but has lingering usability and key management problems such as secure device pairing for devices with interfaces that differ significantly from IoC devices [129, 138, 194, 199, 209] and long-term keying material management (e.g., device key pairs and certificate replacement [12, 216] [191, Chapter 5], passwords [125, 177]; discussed in Section 7.4). While imperfect, if existing approaches from IoC can be employed for many IoT authentication use cases, authentication objectives can be met through known means allowing us to apply past authentication experience to IoT instead of developing entirely new and untested approaches.

We first clarify what we believe to be the general objectives of IoT device identification (based on our review of research literature), and the approaches used to

reach them. Toward this goal, we carry out a literature review to identify three approaches commonly appearing in recent proposals that pursue IoT identification. We then conduct several categorizations of such proposals to extract the identification objectives, and the technical approaches used to reach them. Based on our literature review, the major contribution from this chapter is a model unifying IoT device identification approaches and objectives, and a comparison of the supplementary data used in their fulfillment. Selecting IoT device authentication for further analysis (the other two identification approaches we discuss are *device fingerprinting* and *device classification*), we extract the authentication approaches used in recent IoT device authentication proposals and categorize them. Through this categorization, we recognize the overlap between IoT and IoC authentication approaches, and how device authentication goals are being met in recent IoT device identification proposals.

7.1 Unwrapping "IoT Device Identification" (Background and Models)

Each IoT device identification proposal paper analyzed in this chapter uses one or more approaches: device fingerprinting, classification, and authentication.[1] In this section, as background, we first give an overview of each approach and the identification objectives they achieve. Other operations involved in identification and the relationship between these approaches are then unified in our main model (Fig. 7.2), after a first model of device authentication (our focus in Section 7.3).

Throughout this chapter, we use the following definitions.

Device instance. A device *instance* refers to a single physical IoT device that is self-contained, has computing power, and communicates. For example, two devices of the same smart light bulb model are separate device instances. A single instance could have multiple communication interfaces (e.g., 802.11 radios, Bluetooth, ZigBee, wired network interfaces) and/or run many services.

Device type. A device *type* refers to a set of functional characteristics that defines or implies a device's core functionality (e.g., a smart speaker, smart light bulb, smart lock).

Device model. A device *model* refers to a manufacturer-specific product of a

[1]These approaches are described in the three subsections below.

given device type. All devices of a given model are expected to be functionally the same, while different models may have different functions or features, physical form, or cost. For example, different models of Amazon Echo Dot devices (smart speakers) differ in these aspects, but are the same in their core functionality of a smart speaker with voice assistant.

Device class. A device *class* refers to a collection of devices that is defined by one or more characteristics or attributes. A class could be a particular device model, a device type, or defined by a feature. For example, if an administrator wanted to record (and later retrieve) all the IP addresses of a building's IP cameras, they might define an "IP camera" class. Device *classification* (informally, the act of associating a device with a class) is discussed in Section 7.1.2.

Identifier. An *identifier* (colloquially an "ID") is an explicit label assigned to a device instance that is used to distinguish the device from others within its scope/domain.

7.1.1 Device Fingerprinting

For our purposes within this chapter, we use the following definition.

> By *device fingerprinting* we mean using available information to uniquely distinguish a device instance from a set of devices (or approaches that attempt to do this).[2]

Fingerprinting involves a one-to-many test, aiming to match a device's measured *features* with a stored profile. For example, looking for Bob in a crowded room, Alice might try to determine if the features of each person (e.g., hair color, height, facial structure) matches her memory (of stored profiles) of what Bob looks like.

Similar to biometric authentication [115,116], device fingerprinting starts with an enrollment phase, where a sample is taken from a device, features are extracted and measured, and a profile is created and stored. Features are characteristics, commonly the set of attributes and behaviors that a device exhibits such as the hardware

[2]Note that consistent with the later Fig. 7.2 on page 139, we associate this term with a matching phase, not a profile-building phase.

platform (and version), OS and applications (and their version), and network traffic patterns. Ideally the features vary sufficiently across device instances such that the probability is negligible that two devices produce indistinguishable samples.

In a later (second) phase, a measurement of a fresh sample from a device is matched against stored profiles. If a similarity score in testing against a given profile exceeds a threshold, the identity of the measured device is assumed to be that associated with the profile. Note that in fingerprinting, the device itself asserts no explicit identity. The level of assurance of identity depends on the number and granularity of measured features, and the matching threshold.

7.1.2 Device Classification

For our purposes, we use the following definition.

> By *device classification* we mean associating a device with an explicitly-defined class based on attributes or features.

Classification is useful when a use case calls for a method to logically group devices, and then apply rules or functions to them, e.g., to apply network policies to devices based on their types [152, 227], detect device types or models with known vulnerabilities [144], or to detect abnormal traffic by comparing existing traffic with known baseline traffic for a device model [181].

7.1.3 Device Authentication

For our purposes, we use the following definition.

> By *device authentication* we mean proving or corroborating an asserted identity. Typically this is done using some form of secret or cryptographic keying material.

By this definition, a device authenticates itself to another party by offering some evidence that "proves" (provides some degree of assurance) that it is the asserted device. In contrast with device fingerprinting, authentication is a one-to-one test, as the offered evidence is verified relative to evidence expected from the asserted identity.

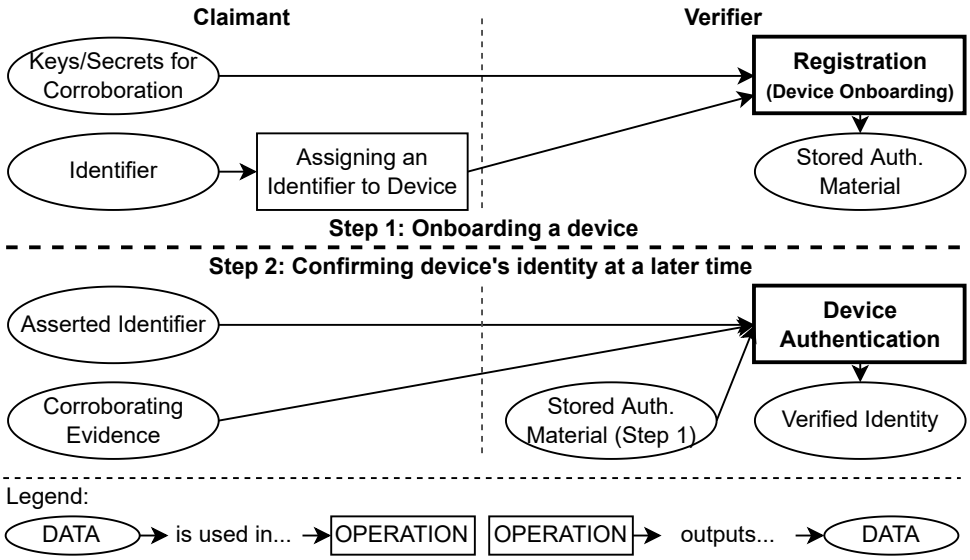


Figure 7.1: Two-step setup-verification model for device authentication. The first step onboards a device into a trust domain, and the second later verifies the device's identity to confirm it is the same device that was registered (onboarded).

Authentication uses an asserted identifier previously associated with some device, and corroborating evidence produced from long-term keying material established during onboarding (i.e., when a device is first added to a domain; discussed in Section 7.1.3). Identifiers are associated with a device during onboarding (Step 1 of Fig. 7.1), and later asserted during authentication with evidence to be compared with its stored authentication material (Step 2 of Fig. 7.1). Corroborating evidence is the information that a device uses to corroborate its identity. Examples of this include demonstrated knowledge of a secret shared between two devices, or demonstrated knowledge of a private key used for digital signatures. When we say that a device is *asserting an identity*, we mean that a device is providing an identifier (for whatever purpose), thus implicitly asserting that it (the identifier) is associated with the device (which has characteristics, behaviors, etc.).

Secure Communication Phases. It is a common requirement for devices to communicate with data confidentiality and integrity. This is accomplished through (1) an onboarding phase, and a communication session with two phases: (2) authenticating a communicating device's identity, and (3) establishing parameters (typically cryptographic session keys) for a secure channel by which to communicate.

Phase (1) onboarding installs long-term keying material into a verifier (e.g., sharing a password, installing a certificate), often when a device is added to a network for the first time (Step 1 of Fig. 7.1). Onboarding (in the case of IoT devices) typically involves a human installing the device (place at a location, connect to power/network, provide initial configuration beyond default), and perhaps providing initial keying material. Phase (2) uses initial keying material to authenticate the device (Step 2 of Fig. 7.1) and the verifier (for mutual authentication). Phase (3) uses an authenticated key establishment protocol (involving Phase 1 long-term material, e.g., passwords [2,211] or digital signature keys [65]) to create and authenticate fresh communication session keys. As a common reason for authentication is to communicate securely, authenticating a device (Phase 2) and establishing an authenticated session key (Phase 3) is often done using an integrated protocol.

7.1.4 Model Relating IoT Identification Approaches and Objectives

The model of Fig. 7.2 provides a foundation for our analysis and classification of IoT device identification proposals in this chapter. The model relates identification objectives (Section 7.2.1) and approaches used to reach them (Section 7.2.2). This includes the three previously noted identification approaches (device fingerprinting, classification, and authentication), along with supporting operations (that transform or use data, e.g., profile generation, device onboarding).

Our model is based on the three categorizations we provide in this chapter (summarized in Table 7.1), but itself also informed the categorizations. As the categorizations progressed, the model was refined to better align data and approaches to objectives, which in-turn led to further refinements of both the category definitions and assignment of proposals to categories as our understanding of IoT identification (and approaches to it) increased.

Each of the three main identification approaches (Sections 7.1.1–7.1.3) are supported by other processes and data, and belong to Phase Two. Phase One is a setup and registration phase, where data is collected, provisioned, or configured for use in Phase Two; this includes gathering data (e.g., device attributes and behaviors, identifiers, authentication material; Fig. 7.2) and processing it. Phase Two aims to reach

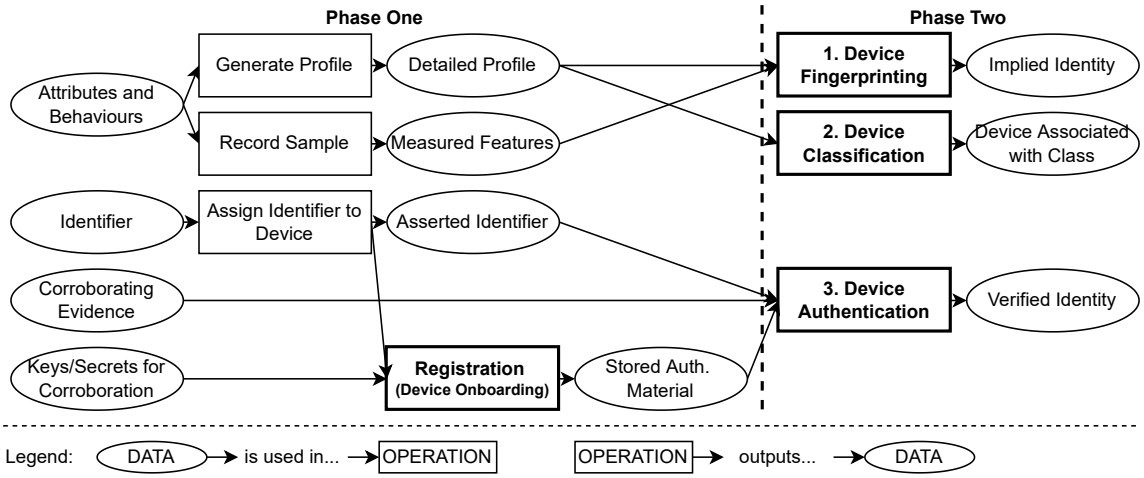


Figure 7.2: Device identification model and relationships between operations and approaches. Some use cases may employ only a subset of components. The three main identification approaches are labelled 1, 2, 3 to aid discussion; the Registration phase (also in bold) is the main component in Phase One.

an identification objective through an identification approach. By an *identification use case* we mean a specific scenario where identification is used.

Each of the identification approaches address different identification objectives as described in Section 7.2.1. Regarding the desired objectives of each approach and the processes and data involved, note from Fig. 7.2 that there is no overlap between device authentication and device fingerprinting data or processes, i.e., the data and processes used to achieve authentication and fingerprinting are distinct.[3]

Our main model of Fig. 7.2 provides a clear differentiation and mapping of the goals each approach addresses, to the data and processes each uses. This model is used as a roadmap for the rest of this chapter, and for unifying context.

7.2 Categorizing IoT Device Identification Proposals

In this section we extract from recent IoT identification literature the objectives that researchers pursue through IoT device identification, and the approaches employed. Our sources include: papers from 2019–2020 of the top four security conferences (IEEE Symp. S&P, ACM CCS, Usenix Security, NDSS); a selection of papers from

[3]Section 7.2.3 discusses possible misuse of fingerprinting for device authentication.

2019–2022 of Google Scholar Alerts on the search queries "IoT identification", "IoT classification", and "IoT authentication"; and related papers referenced by these. In the following two subsections, we provide two categorizations of this literature: based on the identification objectives being pursued, and on the approaches used. Both categorizations, and citations to the literature set, are summarized in Table 7.1.

7.2.1 Categorization 1: Identification Objectives

We use the term *identification objective* for (our interpretation of) the end-purpose pursued by IoT identification proposals, i.e., the purpose for which they want to distinguish devices. To determine these objectives, we manually extracted the research problems and IoT identification use cases from the selected papers. We find identification objectives largely fall into three categories:

1) determine device instances within a domain;

2) determine device classes; and

3) authenticate devices.

We now discuss these three objectives individually.

1. Determine device instances within a domain. The goal here is to determine all device instances (whether authorized or not) that are connected to a service point within a domain (e.g., a network, a home, connected to a hub). The approaches used to reach this objective typically fall short of providing strong assurances of the identity of a device (as explained in Section 7.2.3), but rather aim to enumerate device instances within a domain. In other words, there are relatively weak (if any) assurances regarding device identity.

2. Determine device classes. The goal here is to group devices into existing classes. The goal is not to enumerate device instances; such an enumeration is an assumed given. Establishing such classes may be a preliminary exercise. We note that, beyond the device identification process (our focus) per se, the resulting classification of device instances is often used in subsequent management or processing tasks as observed in our literature review (e.g., applying policies to specific classes of devices).

Table 7.1: Categorization of identification objectives and approaches. Items ordered to highlight patterns. For authentication-focused papers, final column denotes category of authentication approach used.

Prior Work	Determine device instances	Determine device groups	Authenticate device	Device Fingerprinting (§7.1.1)	Device Classification (§7.1.2)	Device Authentication (§7.1.3)	Authentication category (§7.3.1)	
	Objectives (§7.2.1)			Approaches (§7.2.2)				
Bauer [35]	●			✓				
Fischer [79]	●			✓				
Noguchi [161]	●			✓				
Perdisci [167]	●			✓				
Ray [171]	●		●	✓				AA3
Yousefnezhad [225]			●	✓				AA3
Aneja [19]			●	✓				AA3
Chowdhury [52]		●	●	✓				AA3
Hamad [100]		●	●	✓	✓			AA3
Anantharaman [18]	●	●		✓	✓			
Kotak [126]	●	●		✓	✓			
Ammar [15]		●			✓			
Ammar [16]		●			✓			
Santos [183]		●			✓			
Meidan [150]		●			✓			
Singla [190]		●			✓			
Marchal [144]		●			✓			
Bao [28]		●			✓			
Thomsen [203]		●			✓			
Miettinen [152]		●			✓			
Yu [227]		●			✓			
Bezawada [40]		●			✓			
Salman [181]		●			✓			
Marchal [144]		●			✓			
Aksoy [6]		●			✓			
Kumar [130]			●			✓		AA4
Anantharaman [17]			●			✓		AA1, AA4
Mukhandi [155]			●			✓		AA1
Bin-Rabiah [41]			●			✓		AA1
Gritti [91]			●			✓		AA1
Kalra [119]			●			✓		AA1
Malche [143]			●			✓		AA1
Nashwan [157]			●			✓		AA1
Kim [122]			●			✓		AA2
Kim [123]			●			✓		AA1, AA2
Wang [214]			●			✓		AA2
Niya [160]			●			✓		AA2

3. Authenticate devices. The goal here is to verify that a device instance matches the identity it is asserting. This allows, e.g., network infrastructure to have confidence that devices connected to the network are who they claim to be (i.e., entity authentication, Section 7.1.3). This is commonly desired (in reviewed papers) for three purposes: to provide data origin authentication (i.e., to indicate the source of a message/data), to prevent unauthorized devices from masquerading as legitimate devices within the network, and to prevent unauthorized devices from accessing network resources.

Our categorization of papers by identification objectives is summarized in Table 7.1. These objectives, once reached, can be leveraged for subsequent functional objectives. For example, once devices are classified by type (e.g., camera, light bulb, thermometer), administrators might apply specific security policies to certain classes.

7.2.2 Categorization 2: Identification Approaches

We use the term *identification approach* to mean a general approach by which devices are identified, i.e., *how* an identification objective is reached. Here we categorize the same selection of papers (Table 7.1) into three approach-based categories used to reach one of the three main identification objectives (from Categorization 1): fingerprinting, classification, and authentication.

To assign the main proposals of each paper to the identification approach categories, we manually extracted (our interpretation of) the general device identification approaches each uses. (So, to state the obvious: papers that use fingerprinting sampled devices and compared them with stored profiles, classification papers grouped devices based on profiles, and authentication papers confirmed the identity of devices using corroborating evidence.) In most cases, papers used a single approach to reach a single objective; however, three papers used multiple approaches to reach multiple objectives, as discussed in Section 7.2.3.

7.2.3 Objective and Approach Categorization Insights

Table 7.1 summarizes the identification objectives of each selected paper, and the approaches used. Comparing approaches with objectives, we see a natural pattern

(highlighted by the ordering of papers in Table 7.1): device fingerprinting is typically used to determine device instances in a network, device classification is used when determining device groups in a network, and device authentication to provide corroboration of a claimed identity. It is evident from Table 7.1 that in almost all cases reviewed, there is a one-to-one mapping of objective to approach. In cases where a paper looks to reach two objectives (e.g., [17, 100, 126]), typically one approach is used for each objective.

Our categorization suggests that, even when identification is not explicitly defined, researchers tend to have a fairly clear understanding of what *they* intend when they use the term, and follow an expected mapping of approach to identification objective.

We note, however, one exception which comprises five proposals in Table 7.1: some proposals (mis)use fingerprinting approaches for authentication. We single this out as a misuse because IoT devices typically have a small set of primary functions—a thermometer records temperature, a smart lock locks and unlocks, a smart light bulb provides light. Sampling the attributes and behaviors of such devices may be enough to classify devices by type; however, within a given device class or type, device instances are often largely indistinguishable using fingerprinting among very similar (or the same) device models [35, 100]. Also, since fingerprinting does not provide strong assurance of device identity (as previously noted in Section 7.1.1), it is thus not suitable for device authentication, except in a non-adversarial environment.

7.3 Further Analysis of One Identification Approach: Authentication

In this section we categorize recent IoT device authentication proposals. We focus on device authentication (the objective and approach), as it has a long history in IoC, implying mature methods; and is described in the literature in technical detail more precisely than other high-level use cases. We look to determine if the objectives of each authentication phase from Section 7.1.3 (establishing initial keying material, authenticating an entity, establishing session keys) can be met with existing IoC approaches rather than inventing new ones specifically for IoT.

7.3.1 Categorization 3: Authentication Approaches

From a variety of IoT device authentication proposals as listed in Table 7.1, we manually extracted the authentication approaches. By an *authentication approach* (in this section) we mean the general method for authenticating a device, largely based on the technology used but above the level of specific protocols (including, e.g., symmetric secrets, public key-based). Here authentication approach should not be confused with the third of our three identification approaches of Section 7.1, i.e., fingerprinting, classification, and authentication.

From the papers in Table 7.1, we manually extract the approaches used for device authentication to provide a third categorization. Each category employs an authentication approach in specific IoT use cases. We extracted the following four categories of *authentication approach*.

AA1: Well-known cryptographic primitives. This authentication approach category involves well-known cryptographic primitives. These include use of symmetric (including passwords) and asymmetric secrets (public-key pairs) and functions, developed into techniques for specific authentication use cases. The specific techniques are essentially well-known and commonly used for authentication in IoC.

Experiments already in 2004 and 2005 showed elliptic curve-based (EC) operations (e.g., ECDH, ECDSA) on an 8-bit processor are possible, but times increase significantly for future key sizes [29,98,213]. EC operations have become more widely used in constrained devices [38, 218], and lower power consumption also makes EC appropriate for battery-operated devices [213]. Thus, for all but the very heavily constrained devices (see Chapter 2), EC public-key techniques are feasible in IoT.

AA1 approaches include the following proposals (summarized in Table 7.1). Bin-Rabiah et al. [41] use a pair of symmetric keys established during the onboarding phase for subsequent authentication of a device. Nashwan [157] uses symmetric and public-/private-key pairs to authenticate nodes in smart irrigation systems. Gritti et al. [91] use identity-based cryptography (a private key is associated with a device's identifier, which serves as the corresponding public key) and aggregate signatures [45] within a local network. Mukhandi et al. [155] store digitally signed device identities on a blockchain, and use IoT device node consensus to provide integrity of the

stored identity. Anantharaman et al. [17] suggest using a certificate trust infrastructure where smart appliances are given a private key and a signed (by the appliance manufacturer) certificate. Kalra and Sood [119] use EC operations to authenticate a constrained IoT device. Malche et al. [143] use public-key signatures to authenticate devices to servers. Kim et al. [123] use PUFs (discussed next) to generate key pairs for digital signatures to authenticate devices to cloud services and other devices.

AA2: Physically unclonable functions (PUFs). This authentication approach category is populated by proposals that apply PUFs to authentication use cases. PUFs leverage variations in physical objects (e.g., power-up state of SRAM, crystal structure, paper structure) to produce unique outputs [22, 140, 141]. *Strong PUFs* are appropriately suited to produce challenge-response pairs (CR-pairs) for authentication (e.g., [228]), as they are characterized by their large number of responses to unique inputs [140]. *Weak PUFs* are capable of comparatively (to strong PUFs) few or just one output, and are commonly used to produce long-term cryptographic keys [51, 85, 160] or as a random number generator (e.g., by using a pseudo-random number as input and post-processing the PUF output to remove 0/1 bias [163]).

PUF challenge-response (CR) authentication involves a device receiving a challenge from a verifier, applying it as input to a device's PUF, returning the response to the verifier, and the verifier confirming the response matches the stored response. Since weak PUFs can generate only a limited number of CR-pairs, they are most appropriate for uses that do not directly expose PUF responses outside the device (e.g., generating internal random numbers, cryptographic keys).

Given challenges of using PUFs in IoT (Section 7.4) and lack of wide-spread, time-tested applications, the practical value of using PUFs in commodity IoT devices (both in their hardware and software interfaces) remains unclear; however, use of PUFs for authentication and key establishment is appearing now frequently in specialized security-focused hardware products [110, 220].

Reviewed AA2 proposals include the following. Kim et al. [122], aiming to reduce the impact of CR-pair exposures and the CR-pair storage requirement on verifiers, store only one CR-pair in a verifier at a time, storing a fresh CR-pair upon successful authentication with the verifier. Kim et al. [123] also add PUF-based authentication

to the OCF IoTivity framework [114] by using PUFs to authenticate devices with CR-pairs and public/private key pairs. Wang et al. [214], with the goal of reducing device resource use, use CR-pairs from PUFs to authenticate resource-constrained devices and generate session keys. Niya et al. [160] use smart contracts on the Ethereum blockchain to store and verify device identifiers and bind them to end-users, and uses a CR-pair stored during manufacturing for authenticating a device to the manufacturer's server.

AA3: Device fingerprinting. This authentication approach category includes proposals that aim to authenticate devices using fingerprinting (based on attributes and behaviors). These proposals are positioned as providing authentication, but we call this device fingerprinting because in our view, our label is less misleading and removes ambiguity (distinguished in Fig. 7.2).

Reviewed AA3 proposals include the following. Yousefnezhad et al. [225] use individual device certificates and HTTPS to authenticate devices, and use device fingerprinting in an attempt to distinguish device instances. Similarly, Chowdhury et al. [52], Hamad et al. [100], Aneja et al. [19], and Ray et al. [171] use fingerprinting of communication behavioral patterns as a method for distinguishing device instances.

AA4: Exceptions to above categories. We use this authentication approach category for two proposals that fall outside the above three. Approaches in this category apply known techniques (shared secrets, public-key cryptography), but in novel ways through supporting infrastructure to manage and update keying material.

Reviewed AA4 proposals include the following. Anantharaman et al. [17] propose employing macaroons [42] for authenticating smart home appliances, which use chained MACs for integrity checking, with the secret input (known by both the hub device and a new smart appliance) to validate public *caveats* such as a validity period. Kumar et al. [130] propose JEDI, which is an encryption protocol for IoT that uses a hierarchical namespace and identity-based encryption to provide lightweight end-to-end encryption and key delegation; however, at the cost of requiring infrastructure to delegate keys, manage topic subscriptions (URIs are used as topics to subscribe to, similar to other publish-subscribe protocols such as, e.g., MQTT [162], and only specific users are granted keys to decrypt messages published in them), and perform

Table 7.2: Authentication approaches used in the three phases of device authentication (Section 7.1.3) based on authentication papers from Table 7.1 on page 141. *Category* column indicates authentication category from that table.

Authentication Papers	Category	Phase 1: Symmetric secret (1-1)	Phase 1: CA-signed certificate (1-2)	Phase 1: Manually configured public key (1-3)	Phase 1: PUF CR-pair (1-4)	Phase 2: Symmetric-key (2-1)	Phase 2: Public-key (2-2)	Phase 2: PUF CR (Challenge-Response) (2-3)	Phase 3: Diffie-Hellman key agreement (3-1)	Phase 3: Hash/KDF of shared nonces & keys (3-2)	Phase 3: Hybrid encryption (random symm. key) (3-3)	Phase 3: PUF response (3-4)
Anantharaman [17]	AA1, AA4	●	●			✓	✓					
Nashwan [157]	AA1	●				✓			▲			
Bin-Rabiah [41]	AA1	●				✓				▲		
Kalra [119]	AA1			●			✓			▲		
Gritti [91]	AA1			●			✓					
Malche [143]	AA1			●			✓					
Mukhandi [155]	AA1			●			✓					
Kumar [130]	AA4		●	●			✓				▲	
Kim [123]	AA1, AA2			●	●		✓					
Wang [214]	AA2				●			✓		▲		
Kim [122]	AA2				●			✓				▲
Niya [160]	AA2				●			✓				

revocation of key pairs (a similar challenge as in certificate-based approaches, Section 7.4). The primitives used to facilitate authentication and encryption in JEDI and authorization with macaroons may be lightweight, but the infrastructure and management requirements are non-trivial.

In an extension of the above categorization, in Table 7.2 we now extract the approaches (or keying material, in the case of Phase 1) proposals use to facilitate each of the authentication Secure Communication Phases noted on page 137: (1) onboarding and establishing long-term keying material, (2) using Phase 1 material to authenticate a device, and (3) establishing and authenticating fresh session keys. Phase (1) long-term keying material types extracted from the reviewed proposals include:

1-1. symmetric secrets (e.g., a password or shared symmetric key);

1-2. CA-signed certificates (more generally, a centralized trust anchor) and corresponding private key stored by the device;

1-3. manually configured public keys (manually exchanged and configured in a verifier); and

1-4. PUF CR-pair (a challenge value that will be input into a PUF, and the response value that the PUF returns).

Phase (2) approaches extracted from the reviewed proposals include:

2-1. symmetric-key authentication, involving corroborating a device's identity using a symmetric secret (e.g., using a MAC or challenge-response with a hash function);

2-2. public-key authentication, involving a device and verifier using public-/private-key pairs for authentication; and

2-3. PUF challenge-response authentication (see Category 2 from Section 7.3.1).

Phase (3) approaches extracted from the reviewed proposals include:

3-1. DH-based authenticated key agreement;

3-2. using the output of a hash or key derivation function of shared nonces or keys;

3-3. using hybrid encryption (encrypting a random session key with a communicating partner's public key); and

3-4. using a PUF response as a session key (e.g., hashing a PUF challenge and response [122, 198], using a response as a seed for key generation algorithms [198]).

For reasons discussed in Section 7.2.3, we exclude authentication proposals that employ fingerprinting from Table 7.2.

As expected, for a given proposal, the authentication approaches it uses to facilitate each of the authentication phases align with the proposal's category (from above), e.g., AA1 proposals use symmetric secrets or public-key pairs for Phases 1 and 2, AA2 proposals use PUFs. Additionally, as the three authentication phases are closely related (i.e., Phase 1 material typically influences the approaches used for Phases 2 and 3), it is unsurprising that Phase 2 authentication uses the same general authentication approach as for establishing initial keying material in Phase 1, e.g., in Kalra [119] (cf. Table 7.2 on page 147), public-key authentication in Phase 2 (2-2) uses public-key material from Phase 1 (1-3).

We note (with some disappointment) that few papers in our selection describe how session keys are established. This reflects that most authentication proposals in Table 7.2 focus on entity authentication (i.e., confirming the identity of an IoT device) and stop short of describing how authenticated session keys are established, suggesting only the authentication of devices (Fig. 7.1) is their primary focus (versus establishing an authenticated communication channel). We view this as a weakness in these proposals (given that it is generally accepted among cryptographers that session key establishment should be tied to entity authentication) [210, p.94], [65].

Summary discussion. The last column of Table 7.1 (page 141) shows our assignment of these 4 categories to each authentication proposal. AA1 approaches (symmetric and asymmetric techniques) are well-known, and are in use in both IoT and IoC. PUFs (AA2) have seen a great deal of attention, particularly in IoT constrained devices that require lightweight authentication mechanisms and tamper-resistant key generation/storage (weak PUFs have been described as being essentially a key storage mechanism [180]). While not yet enjoying massive widespread use in IoC or IoT, PUFs have seen use for authentication and key establishment in specialized security-focused hardware products [110, 220], suggesting promise of broad future use in IoT devices. While AA3 proposals seek to use it to authenticate devices, as noted in Section 7.2.3, fingerprinting appears ill-suited for authentication. Except for AA4 (which itself still uses known primitives, as noted in Section 7.5), all of our authentication approach categories use techniques that are already well-known (research on PUFs dating back to 2002 [85]). This underscores that authors

of new IoT device authentication proposals for specific use cases appear to be relying heavily on known security primitives, rather than inventing novel primitives for IoT. One might alternatively say that there is a catalog of existing security primitives for identification that have shown to be applicable to IoT use cases. Nonetheless, challenges with these existing techniques are not necessarily trivial (discussed next in Section 7.4), and niche use cases may benefit from additional research into, e.g., new IoT-specific primitives or infrastructure, to fit specific needs (which we do not address in this chapter); as a side effect, research into new IoT-specific techniques may be beneficial to non-niche IoT use cases using existing primitives.

7.4 Challenges Adapting IoC Authentication Approaches for IoT

In support of Section 7.3's categorization of IoT device authentication approaches, in this section we briefly identify notable challenges in adapting each category's approaches for IoT based on the initial keying material used by each (Phase 1 from Table 7.2 on page 147). The characteristics of IoT that distinguish it from IoC [36][4] are reflected in many of the challenges discussed below; how these characteristics impact IoT security is discussed in Chapter 2.

Symmetric secrets. Symmetric secrets require a bootstrapping mechanism. This could be, for example, a user securely pairing two devices [129,194,209] (relying on the user to verify both devices' identities and establish long-term keying material; complicated by non-standard interfaces), or a pre-established global secret that all devices of a specific type or manufacturer know (generally discouraged, and often disastrous [177]).

While onboarding becomes tedious even with a small number of devices, adding new devices incrementally (a plausible situation for consumer devices) distributes this burden; however, the usability of onboarding mechanisms, i.e., how users establish initial keying material, remains an open usability problem. Frustrations with device onboarding highlight an important usability aspect of device authentication [129, 194,209], as a user is often involved with bootstrapping the authentication material.

[4]Relevant IoT device characteristics include: low-cost, non-standard interfaces, and expectation of long-lived devices; Table 2.1 on page 13.

Alternatively, onboarding techniques such as the *resurrecting duckling* model exist, where a device trusts the keying material from the first device to form a security association with it [195], alleviating some of the burden in use cases involving a private domain where access to devices is controlled by a single or small number of verified users.

CA-signed certificates. While certificates are well-understood and feasible from a technical perspective, it is a technically heavier-weight approach for consumer IoT device authentication than shared secrets or manual public-key exchange.

Challenges and recurring issues surrounding IoC certificate revocation and replacement are well-known for, e.g., the browser trust model [137] [230] [4, Chapter 7]. Of the papers cited in Table 7.1, we are not aware of any that specifically look to solve the problem of certificate revocation in IoT (however, some acknowledge revocation in their proposals, e.g., the JEDI proposal [130]). The apparent reason is that this remains, in practice, a largely unsolved problem in IoC and appears to be made more difficult in IoT [12, 191]. Further, any device aiming to verify another's certificate must maintain a store of trust anchors and be able to process any associated certificate chains, adding an additional storage and communication requirement, posing an issue to highly constrained devices where storage is often minimal and communication (message size and frequency) is kept to a minimum.

If a hub device (e.g., acting as a verifier for the local network) wanted to verify devices from another manufacturer, it must have and trust as a trust anchor the other manufacturer's public key (a noted challenge in federated systems such as email [56,146]), requiring the various manufacturers to either rely on one (or multiple) third-party root CAs, or agree on storing each other's anchors within their devices. The CA/browser model [55] (for example) has been built slowly over many years, and relies on a small number of well-established entities that manage trust stores through their software (e.g., Google, Apple, Microsoft, Mozilla). It seems unlikely that IoT, with many more manufacturers, will be able to coordinate this securely in practice.

Manual public-key configuration. With manually-configured public keys, distributing public keys shares many of the same challenges of shared secrets (above),

including the need for a secure onboarding where public keys are registered, facilitated by a user. This does not require signing certificates (and managing trust anchors), but relies on a bootstrapping phase where the two devices' public keys can be manually distributed (public key trust-on-first-use, which involves the user rather than a central party; or third-party monitoring and cross-checking [216]), sharing some of the same problems with the shared secrets approach. In examples such as SSH [210, Chapter 10], removing the user from the loop has not been solved in IoC, so it is unlikely that a solution will be found for this in IoT where typically the goal is even less user interaction. As the manufacturer is not directly involved with the keying material for this approach, key management challenges are often passed on to the user.

Physically unclonable functions (PUFs). PUFs provide a number of benefits. They act as a form of secure key storage separate from non-volatile memory, not requiring additional space for authentication information; are difficult to extract through physical attacks [85, 180], and can support other well-known authentication approaches (i.e., shared secrets and public-key material, but then also sharing their challenges). However, devices must be provisioned with specific PUF hardware and software tools to interface with them.

Intrinsic PUFs are PUFs based on hardware that already exists within a device as a result of a manufacturing process (as opposed to deliberately being added) [22, 95], e.g., SRAM PUFs (a memory-based type), which use the initial state of a device's SRAM to produce a unique output [51, 95]. Intrinsic PUFs also include the mechanism to be measured (e.g., reading memory) [22]. Other types may require specific hardware for the embedding of the PUF or its measurement (e.g., arbiter PUFs [201], coating PUFs [206], optical PUFs [101]).

For authentication using challenge-response pairs (CR-pairs), a bootstrapping phase is still required to install initial CR-pairs in a verifier and in general they must not be re-used (to avoid replay attacks) [228], but subsequent CR-pair updates can be done upon successful authentication. Weak PUFs may be limited in how many CR-pairs they can generate [85, 179], and also require bootstrapping if used to generate long-term keying material.

Virtual devices (i.e., logical copies of end devices that enable a user to remotely interact with their physical counterparts) or services that run in the cloud are unable to use PUFs for authentication, as multiple instances of virtual devices may share the same physical hardware.

Section 7.3.1 identified approaches that have been used in recent IoT authentication proposals; however, each approach has its own challenges and problems.

7.5 Related Work

While we have discussed related work throughout this chapter (including Table 7.1), here we discuss additional work on IoT identification including work on the following topics: conflation of "classification" and "identification" (a motivating factor for our model in Section 7.1), device authentication in related subject areas (e.g., wireless sensor networks, embedded systems, Bluetooth devices), device onboarding proposals, and identifier design in IoT and networks.

In IoT literature, "classification" often means to differentiate devices by class instead of a device instance, matching our observation (Table 7.1) that many use cases seek to recognize or authenticate not a single device, but a group of devices to then be managed. We note examples here. Miettinen et al. [152] determine device types, suggesting the differentiation of types of devices on a given network. Singla and Bose [190] classify unknown devices based on usage patterns. Yu et al. [226] use banner grabbing over multiple protocols to determine brand and model of devices, and define device identification as a classification function (i.e., grouping devices into classes) versus labelling a unique device instance [227]. Perdisci et al. [167] classify IoT devices based on DNS protocol fingerprints and compare individual device types.

Wireless sensor networks (WSNs [134,205]) and embedded systems [103] are characterized by some of the same properties as some sub-areas of IoT (e.g., constrained IoT consumer devices, industrial IoT, and smart cities; Fig. 2.1 from Chapter 2): resource constraints, network communication, and cyberphysical interaction [224].

WSN authentication has received considerable attention, shares many of the challenges of IoT device authentication, and has historically used many techniques discussed in Sections 7.3.1 and 7.4. Xue et al. [221] and Li et al. [134] use pre-shared passwords (a shared secret) for mutual authentication between sensor devices, gateway devices, and a user. Yang et al. [222] use CR-pairs from PUFs (Section 7.3.1) to authenticate mobile sensor devices. Selimis et al. [185] use PUFs to generate cryptographic material and identifiers.

Onboarding, authentication, and establishment of initial keying material for related areas we might now describe as IoT (e.g., WSNs, embedded systems; mobile ad hoc networks, i.e., MANETs [107]) are historically well-studied areas. One example of a prevalent technology is Bluetooth, which has faced many of the same problems that IoT is currently struggling with, including constrained device pairing and authentication. Its extensive history is marked with specification updates to address many security issues [87, 117, 138, 199, 218]. While many vulnerabilities have been addressed, new ones appear regularly [166]. Device onboarding and authentication remains a challenge for heterogeneous, communicating, long-lived, and constrained devices—characteristics which define large subsets of IoT.

The FIDO Alliance [78] defined an IoT platform onboarding specification that involves installing onboarding credentials into devices during manufacturing, then tracking the changes of ownership through the supply chain. It may be feasible for manufacturers that have strong control over their supply chain to track each ownership transfer; however, we expect that for smaller manufacturers or those that cannot rely on each transfer to be secure and recorded, this mechanism for onboarding is infeasible, i.e., this onboarding method may be appropriate for, e.g., manufacturers that control their supply chain, but not for small manufacturers or those lacking such control.

The Alliance for Internet of Things Innovation (AIOTI) [8] highlights identifier requirements for IoT use cases, noting that the format for identifiers varies widely depending on the use case, suggesting no single scheme suits all IoT use cases. Citing scale concerns and issues with existing internet naming infrastructure, Venkataramani et al. [212] propose the design of a new name service for Internet-communicating

devices (part of the MobilityFirst architecture [172]). Shang et al. [186] discuss how Named Data Networking (NDN) can be applied to IoT. As an example of privacy implications of including identifiers in devices, Fischer et al. [80] describe the identifiers built into the Pentium III processor.

Manufacturer Usage Descriptions (MUDs) [132] are a method for devices to specify the network access they require based on descriptions from their manufacturer. They are used by network infrastructure to monitor a device's network behavior and enforce access policies on the device. Hamza et al. [102] generate MUD profiles based on observed device network traffic.

Roman et al. [176] analyzed then-current (as of 2011) key management systems for WSNs and IoT, and suggested public-key approaches are viable in use cases where client devices need to infrequently communicate with servers, and symmetric secrets are viable in small-scale use cases (few devices) and for constrained client nodes. They discuss challenges of each approach (we discuss authentication approach challenges in Section 7.4). Gehrmann et al. [86] describe techniques for wireless device authentication using manual exchange of authentication information. They acknowledge, while per-device trusted CA-signed certificates would help simplify authentication, the required effort to install unique certificates to each device during manufacturing may be unreasonable for low-cost devices. Suomalainen et al. [199] analyze and classify protocols for key establishment in wireless and wireline personal networking technologies (including some commonly used in IoT such as Bluetooth and Wi-Fi, and Wireless Universal Serial Bus and HomePlugAV), and provide a taxonomy of key establishment methods, highlighting approaches and techniques for related protocol classes.

Mimicry-resistance is a term used to describe something about a user or device (e.g., behavior, data) that is resistant to being impersonated by an attacker [7]. Ray et al. [171] use behavioral fingerprinting as mimicry-resistant identities for devices.

7.6 Concluding Remarks

The IoT device identification proposals included in the analyses of this chapter tend to align with one or more of three approaches associated with identification: device

fingerprinting, classification, and authentication. Each approach is used to reach a distinct IoT device identification objective. We provide a model of the relationship between these approaches and objectives in Section 7.1.4 (Fig. 7.2 on page 139).

In order to further investigate the overlap between IoT and IoC authentication (in particular, the goals and approaches of IoT device authentication and how they compare to those of IoC), we narrowed our focus to IoT device authentication. From our analysis of recent literature, we found that IoT device authentication research primarily falls into three categories: modifying existing authentication approaches that use well-known cryptographic primitives for use in specific IoT use cases, applying PUFs to IoT for authentication, and using fingerprinting to uniquely distinguish and authenticate devices.[5]

Almost all IoT device authentication approaches from our literature analysis are using existing, well-understood mechanisms from the pre-IoT world (with the exception of PUFs, which are understood but not widely deployed in either IoC or IoT; AA2 on page 145), rather than fundamentally new approaches. Niche IoT device authentication use cases may benefit from further research into novel approaches and primitives (not covered in the scope of our work), which may in turn be of interest and provide benefit to broader IoT use cases. This said, existing authentication approaches often have their own challenges which complicate their use in IoT (discussed in Section 7.4).

To return to our general scoping question of Chapter 1 (*how is IoT security different from IoC security?*), upon investigation of IoT device authentication, we find IoT and IoC authentication are similar in many respects, and share similar goals (Section 7.3) and approaches (e.g., the three authentication phases and approaches; Table 7.2), suggesting techniques already in common use may largely suffice to address many of the emerging IoT use cases. Another consideration (unexplored in this thesis) is whether the parties conducting the authentication in IoT (i.e., the verifiers, Fig. 7.1 on page 137) differ from those in IoC—verifier capabilities and their relationship to claimants may affect or constrain the subset of appropriate IoT authentication approaches for a given use case.

[5]We believe fingerprinting is inappropriate for authentication, as noted in Section 7.2.3 and Fig. 7.2.

Chapter 8

Conclusion and Future Work

In this thesis, we sought to answer five primary research questions relating to IoT security advice and device identification. Here, we return to these research questions (from Chapter 1), discuss how we answered them, and propose potential areas for future research to expand on the work done in this thesis.

At the outset of this thesis, we did not envision our research would end up producing the security advice coding tree method. After early success in these ideas, we dedicated more energy to this line of questioning (i.e., characterizing security advice) and were able to extend it significantly in Chapters 4–6 (cf. Section 1.6). Conversely, Chapter 7's work regarding IoT device identification yielded fewer results than we had initially expected. As such, it forms a smaller portion of this thesis than we had originally envisioned.

8.1 Answering Research Questions

In this section we describe how we answered our research questions through the work done in this thesis.

RQ1: What are IoT security *best practices*,[1] and how do they relate to security *design principles* and other commonly-used terms; how are these terms used to describe or characterize IoT security advice they are applied to?

In Chapter 3 we analyzed and categorized terminology commonly used in the discussion and specification of security advice. We created a categorization for these terms (which we call *qualifying terms*), and in doing so, defined and compared *best practice* (in the *quality-based* category, where "best" implies a practice that is widely-considered to be high quality) with others. We view *security design principles* and

[1]Note that we discuss the *term* "best practice", not specifying technical practices.

practices—based on our definitions from Chapter 3 and the terms' positions in the coding tree from Chapter 4—as categorically different (i.e., they focus on independent concepts). Later, in Chapter 4 (page 62) we compare practices with principles, and determine principles to be more focused on end-results than practices (which are more focused on mechanisms to implement). Of course, if one takes the view that it is "best practice" to follow a certain design principle, then our view of this categorical difference is broken.

The relationship between practices and security design principles was further described by their relative positions and questions that lead to them in the security advice coding tree of Chapter 4 (discussed next).

Here we emphasize the relationship between best practices and design principles, as our early research put focus on this relationship. As our research progressed, through broad discussion of these and other related terms, emphasis was moved away from design principles as it (the term) became a part of the coding tree.

RQ2: Can we design a methodology for objectively characterizing security advice?

In Chapter 4 we constructed a security advice coding tree that allows for effective categorization and characterization of security advice based on a subset of the terminology refined in Chapter 3. We began with a basic codebook of unrefined codes reflecting concepts in the DCMS 1013-item IoT security advice dataset, then used iterative inductive coding to refine and augment the codebook, and iterated on the coding tree's structure, questions, and coder instructions to develop our coding tree. As part of this coding tree, a subset of the specific tags for advice were identified as being *actionable* (used in RQ3), meaning the advice they are applied to contains a clear sequence of steps whose means of execution are understood by target advice recipients.

We intended the design of the coding tree to produce reproducible and repeatable results because its questions, structure, and instructions to coders are designed to reduce subjectivity as much as possible,[2] allowing new coders to reliably achieve

[2]Recall from Section 4.1.1 on page 52 that a major motivation for creating the coding tree methodology was to reduce subjectivity.

similar results to ours, and for coders to repeat their own results on a dataset. Chapter 5 compares two coders' 1013-item tagging results as a first detailed analysis exploring how reproducible Chapter 4's results are by one additional coder. While coder agreement on whether advice items were actionable was promising, as rates of agreement on specific tags (for individual advice items) were below our expectation, it appears premature to claim the method is useful for accurately estimating the proportion of advice in a dataset that falls into each category.

Our research into security best practices and IoT security advice eventually led to the use of the SAcoding tree as a method to analyze security advice. This was not what we originally intended to do, but our findings from using the SAcoding tree prompted further investigation (RQ4 below). Chapter 5 is a cross-check of the SAcoding method (from Chapter 4); we did not anticipate this early in our research, but it was conducted in the process of refining our research questions.

RQ3: How *actionable* is the current state of IoT security advice; does it primarily consist of security objectives to reach, or more specifically ways to reach those objectives (i.e., actionable practices)?

In Chapter 4 (after construction of coding tree methodology), we characterized the advice currently being recommended by advice-giving organizations for (typically pre-deployment) IoT security stakeholders. One coder applied our security advice coding tree to the DCMS 1013-item security advice dataset,[3] and in Chapter 5, to allow a cross-check and further analysis of the SAcoding method itself, a second coder tagged the full DCMS 1013-item dataset.

Through our analysis, we found that the majority of advice in the dataset is not actionable, and largely comprised of security objectives to reach rather than practices to follow, advice positioned as practices but lacking technical detail, or advice that was deemed (by the coding tree) to be not useful. This suggests to us that there is room for IoT security advice to be improved to become more actionable for IoT device manufacturers (and other pre-deployment stakeholders).

RQ4: How can the coding tree methodology (developed in answering RQ2) be

[3]We use the DCMS 1013-item dataset as a proxy representative of current IoT security advice.

used to compare the actionability of different sets of IoT security advice, or generally characterize any improvement in subsequent versions of a given set?

In Chapter 6 we used our coding tree methodology on two sets of formally-recommended IoT security advice: a set of guidelines from the UK government's DCMS (the DCMS 13 guidelines document from Table 1.1 on page 4), and a set of provisions from ETSI (the ETSI provisions document). As the ETSI provisions document appears to be positioned as an evolution of the DCMS 13 guidelines document, we began by conducting an informal comparison and analysis of the two sets to determine how the ETSI set improves (if indeed it does) over the DCMS set. We then applied our SAcoding method to each set to characterize them (i.e., determine which advice codes they are comprised of), and determine how actionable their advice is.

Following the characterization of how the ETSI advice improves over the DCMS advice and how our SAcoding method can be used to compare sets of security advice (both characterizing a set of advice and determining how actionable it is), we used the characteristics of actionable advice (questions at each node along the path to a final tag which ask about, e.g., technical detail, target audience, and advice granularity) to recommend aspects of security advice that we suggest advice-givers should include if looking to provide actionable security practices.

RQ5: In what ways do researchers use the term *IoT device identification*, what are the most common goals of IoT device identification, and what approaches are being used to reach them?

In Chapter 7, we analyzed recent academic papers that involved IoT device identification. From these papers, we aimed (1) to determine what objectives identification proposals were looking to achieve through identification, and (2) to determine the approaches the proposals used to reach their objectives. In doing this, we determined that *device identification* is commonly approached in three ways: *fingerprinting*, *classifying*, and *authenticating* devices. To explicate the operations and approaches involved in IoT device identification, we developed a model (Fig. 7.2 on page 139) that outlines distinct operations involved in reaching identification goals (through the three approaches from above).

As authentication is a popular research subject, we chose to investigate IoT device authentication further to explore how authentication in IoT compares with authentication in IoC. Extracting the authentication approaches from recent IoT device authentication research papers, we found that many approaches being used or proposed to authenticate devices are well-known approaches from IoC, suggesting the goals of numerous IoT authentication use cases can be met using similar methods to IoC.

8.2 Future Research Directions

In this section we describe avenues to investigate for future work. The discussion of future work is based on extensions to the findings we discuss throughout this thesis, or the challenges or new problems we encountered while conducting our research.

The security advice coding tree (of Chapter 4) is one of the primary contributions and deliverables of this thesis. While in this thesis we focus on the coding tree's application to IoT security advice, we intentionally left the coding tree as broadly applicable to computer security advice in general rather than focus its questions, tags, or coder instructions specifically on IoT security (as discussed in Section 4.2.1). It is our hope that the coding tree will find use (by the security community) outside of IoT, and that its adoption will not only help validate its design, but become valuable as a method for analyzing and improving security advice across broad computer security audiences. To our knowledge, our SAcoding method is the first systematic method for analyzing security advice.

We briefly outline several potential directions for future research here.

Coding tree improvements. Most of the discussion about future work regarding coding tree improvements (e.g., to improve coder consistency and simplify the coding tree and instructions) appears in Chapter 5. Repeated here for convenience, future work might further investigate the coding tree's design and structure, and make improvements (beginning with the suggestions in Chapter 5). Aside from improving the SAcoding method, an alternative is to develop other methods for systematically analyzing security advice.

Quantifying coding tree utility. In Section 8.1 we summarized how we intended for the coding tree to be useful for advice givers. These two general goals were intended in the design of the coding tree; however, verifying whether these goals were met is not conducted within this thesis (the second goal—reproducibility—was partially investigated through the addition of a second coder in Chapter 5). As future work, we suggest a comparison with other inductive coding methodologies to determine how the utility of the SAcoding method compares to them (possibly by, e.g., measure of the time it takes to code items, or subjective opinion gathered through further analyses involving experts using the coding process), and pursuing further refinements and improvements of the method (discussed in Chapter 5).

Measuring practice quality. In Chapter 3 we outline what we call *quality-based* qualifying terms, and in Chapter 4's coding tree we imply some tags are 'better" than others (e.g., advice assigned an actionable tag is "better" than those without). In this thesis we do not investigate the quality of a practice itself, i.e., how "good" are the outcomes associated with carrying out a practice (including what makes a practice better than another, and how to measure such a trait). Future work could further categorize actionable practices (those with *P3–P6* tags from Fig. 4.1 on page 54) by new measures of practice quality.

Verifiable outcomes. Section 3.2.3 in Chapter 3 discussed imperative and declarative advice,[4] questioning that if an end result could be verified (i.e., to confirm that a desired goal was achieved; how this could be done might be a research question itself), would it be equivalent to a practice? If declarative advice can be verified, what role does it have in determining best practices. Instead of working from the view that creating actionable advice should be given the highest priority (as we do in this thesis), future research could instead analyze security advice and explore if stated outcomes (possibly those that we categorize in the DCMS 1013-item dataset in Chapter 4) could be verified, thereby allowing advice followers to use any practices they desire to reach the end-goal.

[4]Recall from Chapter 3 that declarative advice is specifying an end result rather than a method to reach it [44, 75].

Bibliography

[1] Bill—S.1691 - Internet of Things (IoT) Cybersecurity Improvement Act of 2017 (Bill), 2017. United States Senate https://www.congress.gov/bill/115th-congress/senate-bill/1691/text?format=txt.

[2] 1Password. 1Password Security Design. https://1passwordstatic.com/files/security/1password-white-paper.pdf, 2021.

[3] Yasemin Acar, Christian Stransky, Dominik Wermke, Charles Weir, Michelle L. Mazurek, and Sascha Fahl. Developers Need Support, Too: A Survey of Security Advice for Software Developers. In *IEEE Cybersecurity Development (SecDev)*, 2017.

[4] Carlisle Adams and Steve Lloyd. *Understanding PKI: Concepts, Standards, and Deployment Considerations (2nd edition)*. Addison-Wesley Professional, 2002.

[5] George A. Akerlof. The Market for "Lemons": Quality Uncertainty and the Market Mechanism. *The Quarterly Journal of Economics*, 84(3):488–500, 1970.

[6] Ahmet Aksoy and Mehmet Hadi Gunes. Automated IoT Device Identification using Network Traffic. In *International Conference on Communications*, 2019.

[7] Furkan Alaca, AbdelRahman Abdou, and Paul C. van Oorschot. Comparative Analysis and Framework Evaluating Mimicry-Resistant and Invisible Web Authentication Schemes. *IEEE TDSC*, 18(2):534–549, 2021.

[8] Alliance for Internet of Things Innovation. Identifiers in Internet of Things (IoT). https://aioti.eu/wp-content/uploads/2018/03/AIOTI-Identifiers_in_IoT-1_0.pdf.pdf, 2018.

[9] Alliance for Internet of Things Innovation (AIOTI). Report: Working Group 4—Policy. https://aioti.eu/wp-content/uploads/2017/03/AIOTIWG04Report2015-Policy-Issues.pdf, 2015.

[10] Alliance for Internet of Things Innovation (AIOTI). AIOTI Digitisation of Industry Policy Recommendations. https://aioti.eu/wp-content/uploads/2017/03/AIOTI-Digitisation-of-Ind-policy-doc-Nov-2016.pdf, 2016.

[11] Alliance for Internet of Things Innovation (AIOTI). Report on Workshop on Security and Privacy in the Hyper connected World. https://aioti-space.org/wp-content/uploads/2017/03/AIOTI-Workshop-on-Security-and-P

rivacy-in-the-Hyper-connected-World-Report-20160616_vFinal.pdf, 2016.

[12] Arwa Alrawais, Abdulrahman Alhothaily, Chunqiang Hu, and Xiuzhen Cheng. Fog computing for the internet of things: Security and privacy issues. *IEEE Internet Computing*, 21(2):34–42, 2017.

[13] Omar Alrawi, Charles Lever, Kevin Valakuzhy, Ryan Court, Kevin Snow, Fabian Monrose, and Manos Antonakakis. The Circle Of Life: A Large-Scale Study of The IoT Malware Lifecycle. In *USENIX Security Symp.*, 2021.

[14] Omar Alrawi, Chaz Lever, Manos Antonakakis, and Fabian Monrose. SoK: Security Evaluation of Home-Based IoT Deployments. In *IEEE Symp. Security and Privacy*, 2019.

[15] Nesrine Ammar, Ludovic Noirie, and Sebastien Tixeuil. Autonomous IoT Device Identification Prototype. In *Network Traffic Measurement and Analysis Conference (TMA)*, 2019.

[16] Nesrine Ammar, Ludovic Noirie, and Sebastien Tixeuil. Network-Protocol-Based IoT Device Identification. In *International Conference on Fog and Mobile Edge Computing (FMEC)*, 2019.

[17] Prashant Anantharaman, Kartik Palani, David Nicol, and Sean W. Smith. I Am Joe's Fridge: Scalable Identity in the Internet of Things. In *International Conf. on Internet of Things (iThings)*, pages 129–135. IEEE, 2016.

[18] Prashant Anantharaman, Liwei Song, Ioannis Agadakos, Gabriela Ciocarlie, Bogdan Copos, Ulf Lindqvist, and Michael E. Locasto. IoTHound: Environment-agnostic Device Identification and Monitoring. In *International Conference on the Internet of Things*, pages 1–9, 2019.

[19] Sandhya Aneja, Nagender Aneja, and Md Shohidul Islam. IoT Device Fingerprint using Deep Learning. In *IEEE International Conference on Internet of Things and Intelligence System (IOTAIS)*, pages 174–179, 2018.

[20] Kishore Angrishi. Turning Internet of Things (IoT) into Internet of Vulnerabilities (IoV): IoT botnets. *preprint arXiv:1702.03681*, 2017.

[21] Manos Antonakakis, Tim April, Michael Bailey, Elie Bursztein, Jaime Cochran, Zakir Durumeric, Alex Halderman, J, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou. Understanding the Mirai botnet. In *USENIX Security Symp.*, 2017.

[22] Frederik Armknecht, Roel Maes, Ahmad-Reza Sadeghi, François-Xavier Standaert, and Christian Wachsmann. A Formalization of the Security Features of Physical Functions. In *IEEE Symposium on Security and Privacy*, 2011.

[23] AspenCore. 2019 Embedded Markets Study—Integrating IoT and Advanced Technology Designs, Application Development & Processing Environments, 2019. https://www.embedded.com/wp-content/uploads/2019/11/EETimes_-Embedded_2019_Embedded_Markets_Study.pdf.

[24] Hala Assal and Sonia Chiasson. Security in the software development lifecycle. In *Symp. on Usable Privacy and Security (SOUPS)*, pages 281–296. USENIX, August 2018.

[25] Association for Computing Machinery (ACM). Artifact Review and Badging Version 1.1. `https://www.acm.org/publications/policies/artifact-review-and-badging-current`, 2020.

[26] AT&T. The CEO's Guide to Securing the Internet of Things. `https://www.business.att.com/cybersecurity/docs/exploringiotsecurity.pdf`, 2016.

[27] Australian Department of Home Affairs and Australian Cyber Security Centre. Code of Practice—Securing the Internet of Things for Consumers. `https://www.homeaffairs.gov.au/reports-and-pubs/files/code-of-practice.pdf`, 2020.

[28] Jiaqi Bao, Bechir Hamdaoui, and Weng-keen Wong. IoT Device Type Identification Using Hybrid Deep Learning Approach for Increased IoT Security. In *International Wireless Communications and Mobile Computing (IWCMC)*, 2020.

[29] Elaine Barker. SP 800-57 Part 1—Recommendation for Key Management: Part 1–General, 2020. NIST.

[30] Elaine B. Barker. Recommendation for key management. NIST SP (Special Publication) 800-57 Part 1, January 2016.

[31] Elaine B. Barker and Allen L. Roginsky. Transitioning the use of cryptographic algorithms and key lengths. NIST SP (Special Publication) 800-131A, March 2019.

[32] David Barrera, Christopher Bellman, and Paul C. van Oorschot. Security Best Practices: A Critical Analysis Using IoT as a Case Study. *Journal submission, under minor revision*, 2021. Technical report available at: `https://www.ccsl.carleton.ca/~chris/publications/2021-IoT-Security-Best-Practices-A-Critical-Analysis.pdf`.

[33] David Barrera, Christopher Bellman, and Paul C. van Oorschot. A Close Look at a Systematic Method for Analyzing Sets of Security Advice. *Journal submission, under review*, 2022.

[34] David Barrera, Ian Molloy, and Heqing Huang. Standardizing IoT network security policy enforcement. In *Workshop on Decentralized IoT Security and Standards (DISS)*, 2018.

[35] Johann Bauer. An Algorithm for IoT Device Identification. In *International Conference on Information Networking (ICOIN)*, 2020.

[36] Christopher Bellman and Paul C. van Oorschot. Analysis, Implications, and Challenges of an Evolving Consumer IoT Security Landscape. In *International Conference on Privacy, Security and Trust (PST)*, 2019.

[37] Christopher Bellman and Paul C. van Oorschot. Systematic Analysis and Comparison of Security Advice Datasets. *Journal submission, under minor revision*, 2022. Technical report available at: `https://arxiv.org/abs/2206.09237`.

[38] Daniel J Bernstein. Curve25519: New Diffie-Hellman Speed Records. In *International Workshop on Public Key Cryptography*, 2006.

[39] Elisa Bertino and Nayeem Islam. Botnets and Internet of Things security. *Computer*, 50(2):76–79, 2017.

[40] Bruhadeshwar Bezawada, Maalvika Bachani, Jordan Peterson, Hossein Shirazi, Indrakshi Ray, and Indrajit Ray. IoTSense: Behavioral Fingerprinting of IoT Devices. *preprint arXiv:1804.03852*, 2018.

[41] Abdulrahman Bin-Rabiah, K K Ramakrishnan, Elizabeth Liri, and Koushik Kar. A Lightweight Authentication and Key Exchange Protocol for IoT. In *Workshop on Decentralized IoT Security and Standards (DISS)*, 2018.

[42] Arnar Birgisson, Joe Gibbs Politz, Úlfar Erlingsson, Ankur Taly, Michael Vrable, and Mark Lentczner. Macaroons: Cookies with Contextual Caveats for Decentralized Authorization in the Cloud. In *NDSS*, 2014.

[43] Matt Bishop. *Computer Security: Art and Science*. Addison-Wesley, 2003.

[44] Harold Boley, Micha Meier, Chris Moss, Michael M Richter, and A. A. Voronkov. Declarative and Procedural Paradigms - Do They Really Compete? In *International Workshop on Processing Declarative Knowledge*, pages 383–398. Springer, 1991.

[45] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In *EUROCRYPT*, 2003.

[46] C. Bormann, M. Ersue, and A. Keranen. Terminology for constrained-node networks. RFC 7228, RFC Editor, May 2014.

[47] Scott Bradner. RFC2119: Key Words for Use in RFCs to Indicate Requirement Levels, 1997. IETF.

[48] Broadband Internet Technical Advisory Group (BITAG). Internet of Things (IoT) Security and Privacy Recommendations, 2016.

[49] CableLabs. A Vision for Secure IoT. `https://www.cablelabs.com/insights/vision-secure-iot/`, 2017.

[50] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David A. Wagner, and Wenchao Zhou. Hidden Voice Commands. In *USENIX Security*, 2016.

[51] Bin Chen, Tanya Ignatenko, Frans M. J. Willems, Roel Maes, Erik van der Sluis, and Georgios Selimis. A Robust SRAM-PUF Key Generation Scheme Based on Polar Codes. In *IEEE Global Communications Conference*, 2017.

[52] Rajarshi Roy Chowdhury, Sandhya Aneja, Nagender Aneja, and Emeroylariffion Abas. Network Traffic Analysis based IoT Device Identification. In *International Conference on Big Data and Internet of Things (BDIoT)*, 2020.

[53] City of New York (NYC) Guidelines for the Internet of Things. Privacy + Transparency, 2019.

[54] City of New York (NYC) Guidelines for the Internet of Things. Security, 2019.

[55] Jeremy Clark and Paul C van Oorschot. SoK: SSL and HTTPS: Revisiting Past Challenges and Evaluating Certificate Trust Model Enhancements. In *IEEE Symp. Security and Privacy*, 2013.

[56] Jeremy Clark, Paul C van Oorschot, Scott Ruoti, Kent Seamons, and Daniel Zappala. SoK: Securing Email—A Stakeholder-Based Analysis. In *Financial Cryptography and Data Security*, 2021.

[57] Cloud Security Alliance (CSA). Security Guidance for Early Adopters of the Internet of Things (IoT), 2015.

[58] Cloud Security Alliance (CSA). Future-proofing the Connected World: 13 Steps to Developing Secure IoT, 2016.

[59] Copper Horse Ltd. Mapping Security & Privacy in the Internet of Things. `https://iotsecuritymapping.uk/wp-content/uploads/Mapping-of-Code-of-Practice-to-recommendations-and-standards_v3.json`, 2019. Version 3 dataset.

[60] Juliet Corbin and Anselm Strauss. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory (3rd ed.)*. SAGE Publications, Inc., 2008.

[61] George Corser, Glenn A. Fink, Mohammed Aledhari, Jared Bielby, Rajesh Nighot, Sukanya Mandal, Nagender Aneja, Chris Hrivnak, and Lucian Cristache. IoT Security Principles and Best Practices. https://internetinitiative.ieee.org/images/files/resources/white_papers/internet_of_things_feb2017.pdf, 2017. IEEE.

[62] Department for Digital, Culture, Media & Sport (DCMS) of the UK Government. Code of Practice for Consumer IoT Security. https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/773867/Code_of_Practice_for_Consumer_IoT_Security_October_2018.pdf, 2018.

[63] Department for Digital, Culture, Media & Sport (DCMS) of the UK Government. Mapping of IoT Security Recommendations, Guidance and Standards to the UK's Code of Practice for Consumer IoT Security. https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/774438/Mapping_of_IoT__Security_Recommendations_Guidance_and_Standards_to_CoP_Oct_2018.pdf, 2018.

[64] Department for Digital, Culture, Media & Sport (DCMS) of the UK Government. Secure by design. https://www.gov.uk/government/collections/secure-by-design, 2020.

[65] Whitfield Diffie, Paul C van Oorschot, and Michael J Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2(2), 1992.

[66] Andrew Dingman, Gianpaolo Russo, George Osterholt, Tyler Uffelman, and L Jean Camp. Good advice that just doesn't help. In *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, 2018.

[67] European Telecommunications Standards Institute (ETSI). CYBER; Cyber Security for Consumer Internet of Things, 2019.

[68] European Telecommunications Standards Institute (ETSI). CYBER; Cyber Security for Consumer Internet of Things: Baseline Requirements (ETSI EN 303 645), 2020.

[69] European Telecommunications Standards Institute (ETSI). CYBER; Cyber Security for Consumer Internet of Things: Conformance Assessment of Baseline Requirements (ETSI TS 103 701 V1.1.1). https://www.etsi.org/deliver/etsi_ts/103700_103799/103701/01.01.01_60/ts_103701v010101p.pdf, 2021.

[70] European Union Agency for Network and Information Security (ENISA). Security and Resilience of Smart Home Environments, 2015.

[71] European Union Agency for Network and Information Security (ENISA). Baseline Security Recommendations for IoT, 2017.

[72] Michael Fagan, Jeffrey Marron, Kevin G. Brady Jr., Barbara B. Cuthill, Katerina N. Megas, and Rebecca Herold. Draft NISTIR 8259C—Creating a Profile Using the IoT Core Baseline and Non-Technical Baseline, 2020. NIST.

[73] Michael Fagan, Jeffrey Marron, Kevin G. Brady Jr., Barbara B. Cuthill, Katerina N. Megas, and Rebecca Herold. NISTIR 8259B—IoT Non-Technical Supporting Capability Core Baseline, 2021. NIST.

[74] Michael Fagan, Jeffrey Marron, Kevin G. Brady Jr., Barbara B. Cuthill, Katerina N. Megas, Rebecca Herold, David Lemire, and Brad Hoehn. SP 800-213—IoT Device Cybersecurity Guidance for the Federal Government: Establishing IoT Device Cybersecurity Requirements, 2020. NIST.

[75] Dirk Fahland, Daniel Lübke, Jan Mendling, Hajo Reijers, Barbara Weber, Matthias Weidlich, and Stefan Zugal. Declarative Versus Imperative Process Modeling Languages: The Issue of Understandability. In *Enterprise, Business-Process and Information Systems Modeling*, pages 353–366. Springer, 2009.

[76] Earlence Fernandes, Amir Rahmati, Kevin Eykholt, and Atul Prakash. Internet of Things Security Research: A Rehash of Old Ideas or New Intellectual Challenges? *IEEE Security & Privacy*, 15(4):79–84, 2017.

[77] Mohamed Amine Ferrag, Leandros A. Maglaras, Helge Janicke, Jianmin Jiang, and Lei Shu. Authentication Protocols for Internet of Things: A Comprehensive Survey. *Security and Communication Networks*, 2017(1), 2017.

[78] FIDO Alliance. FIDO Device Onboard Specification, 2021.

[79] Sebastian Fischer, Katrin Neubauer, Rudolf Hackenberg, Lukas Hinterberger, and Bernhard Weber. IoTAG: An Open Standard for IoT Device IndentificAtion and RecoGnition. In *International Conference on Emerging Security Information, Systems and Technologies (SECURWARE)*, pages 107–113, 2019.

[80] Stephen Fischer, James Mi, and Albert Teng. Pentium III Processor Serial Number Feature and Applications. *Intel Technology Journal*, (Q2):1–6, 1999.

[81] Dinei Florêncio, Cormac Herley, and Baris Coskun. Do strong web passwords accomplish anything? In *Usenix HotSec*, 2007.

[82] Oscar Garcia-Morchon, Sandeep S. Kumar, and Mohit Sethi. State-of-the-Art and Challenges for the Internet of Things Security. `https://datatracker.ietf.org/doc/draft-irtf-t2trg-iot-seccons/`, 2019.

[83] Simson Garfinkel, Gene Spafford, and Alan Schwartz. Chapter 3: Policies and Guidelines, in [84], 2003.

[84] Simson Garfinkel, Gene Spafford, and Alan Schwartz. *Practical UNIX and Internet Security (3rd edition)*. O'Reilly Media, Inc., 2003.

[85] Blaise Gassend, Dwaine Clarke, Marten Van Dijk, and Srinivas Devadas. Silicon physical random functions. In *ACM CCS*, 2002.

[86] Christian Gehrmann, Chris J. Mitchell, and Kaisa Nyberg. Manual Authentication for Wireless Devices. *RSA Cryptobytes*, 7(1):29–37, 2004.

[87] Christian Gehrmann, Joakim Persson, and Ben Smeets. *Bluetooth Security*. Artech House, 2004.

[88] Dieter Gollmann. *Computer Security, 3rd Edition*. Wiley, 2011.

[89] Paul A. Grassi and 12 others. SP 800-63B—Digital Identity Guidelines: Authentication and Lifecycle Management, 2017. NIST.

[90] Christpher Greer, Martin Burns, David Wollman, and Edward Griffor. Cyber-physical systems and Internet of Things. NIST SP (Special Publication) 1900-202, March 2019.

[91] Clémentine Gritti, Melek Önen, Refik Molva, Willy Susilo, and Thomas Plantard. Device Identification and Personal Data Attestation in Networks. *J. Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, 9(4), 2018.

[92] GSM Association. IoT Security Guidelines for Endpoint Ecosystems—Version 2.0. `https://www.gsma.com/iot/wp-content/uploads/2017/10/CLP.13-v2.0.pdf`, 2017.

[93] GSM Association. IoT Security Guidelines. `https://www.gsma.com/iot/iot-security/iot-security-guidelines/`, 2020.

[94] GSM Association. GSM Association. `https://www.gsma.com/`, 2021.

[95] Jorge Guajardo, Sandeep S Kumar, Geert-Jan Schrijen, and Pim Tuyls. FPGA Intrinsic PUFs and Their Use for IP Protection. In *Cryptographic Hardware and Embedded Systems (CHES)*, 2007.

[96] Oscar M. Guillen, Thomas Poppelmann, Jose M. Bermudo Mera, Elena Fuentes Bongenaar, Georg Sigl, and Johanna Sepulveda. Towards post-quantum security for IoT endpoints with NTRU. In *Design, Automation & Test in Europe Conference & Exhibition*, 2017.

[97] Cenk Gündoğan, Peter Kietzmann, Martine Lenders, Hauke Petersen, Thomas C. Schmidt, and Matthias Wählisch. NDN, CoAP, and MQTT: a comparative measurement study in the IoT. In *ACM Conference on Information-Centric Networking (ICN)*, 2018.

[98] Nils Gura, Arun Patel, Arvinderpal Wander, Hans Eberle, and Sheueling Chang Shantz. Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs. In *Cryptographic Hardware and Embedded Systems (CHES)*, 2004.

[99] Oliver Hahm, Emmanuel Baccelli, Hauke Petersen, and Nicolas Tsiftes. Operating Systems for Low-End Devices in the Internet of Things: A Survey. *IEEE Internet of Things Journal*, 3(5):720–734, 2016.

[100] Salma Abdalla Hamad, Wei Emma Zhang, Quan Z. Sheng, and Surya Nepal. IoT Device Identification via Network-Flow Based Fingerprinting and Learning. In *International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE, 2019.

[101] Ghaith Hammouri, Aykutlu Dana, and Berk Sunar. CDs Have Fingerprints Too. In *Cryptographic Hardware and Embedded Systems (CHES)*, 2009.

[102] Ayyoob Hamza, Dinesha Ranathunga, Hassan Habibi Gharakheili, Matthew Roughan, and Vijay Sivaraman. Clear as MUD: Generating, Validating and Applying IoT Behaviorial Profiles. In *Workshop on IoT Security and Privacy*, 2018.

[103] Steve Heath. *Embedded Systems Design*. Elsevier, 2002.

[104] Cormac Herley. So Long, And No Thanks for the Externalities: The Rational Rejection of Security Advice by Users. In *New Security Paradigms Workshop (NSPW)*, 2009.

[105] Nicholas Huaman, Sabrina Amft, Marten Oltrogge, Yasemin Acar, and Sascha Fahl. They Would do Better if They Worked Together: The Case of Interaction Problems Between Password Managers and Websites. In *IEEE Symp. Security and Privacy*, pages 1626–1640, 2021.

[106] Wei Huang, Afshar Ganjali, Beom Heyn Kim, Sukwon Oh, and David Lie. The State of Public Infrastructure-as-a-Service Cloud Security. *ACM Computing Surveys (CSUR)*, 47(4):1–31, 2015.

[107] Jean-Pierre Hubaux, Levente Buttyán, and Srdan Čapkun. The Quest for Security in Mobile Ad Hoc Networks. In *International Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*, 2001.

[108] George F. Hurlburt. The Internet of Things... of all things. *ACM Crossroads*, 22(2):22–26, 2015.

[109] IFTTT. IFTTT. https://ifttt.com/, 2019.

[110] Intrinsic ID. QuiddiKey. https://www.intrinsic-id.com/products/quiddikey/, 2022.

[111] IoT Security Foundation. IoT Security Compliance Framework 1.1, 2017.

[112] IoT Security Foundation. https://www.iotsecurityfoundation.org/, 2021.

[113] IoT Security Initiative. Security Design Best Practices. https://www.iotsi.org/security-best-practices, 2018.

[114] IoTivity. https://iotivity.org/.

[115] A.K. Jain, Arun Ross, and Sharath Pankanti. Biometrics: A Tool for Information Security. *IEEE TIFS*, 1(2):125–143, 2006.

[116] A.K. Jain, Arun Ross, and Salil Prabhakar. An Introduction to Biometric Recognition. *IEEE Trans. on Circuits and Systems for Video Technology*, 14(1), 2004.

[117] Markus Jakobsson and Susanne Wetzel. Security Weaknesses in Bluetooth. In *Cryptographers' Track at the RSA Conference*, 2001.

[118] Erica Johnson. Online Banking Agreements Protect Banks, Hold Customers Liable for Losses, Expert Says. *Canadian Broadcasting Corporation*, Feb 9 2020. https://www.cbc.ca/news/business/online-banking-agreements-1.5453192.

[119] Sheetal Kalra and Sandeep K. Sood. Secure authentication scheme for IoT and cloud servers. *Pervasive and Mobile Computing*, 24:210–223, 2015.

[120] Ruogu Kang, Laura Dabbish, Nathaniel Fruchter, and Sara Kiesler. "My data just goes everywhere:" User Mental Models of the Internet and Implications for Privacy and Security. In *Symposium on Usable Privacy and Security (SOUPS)*, 2019.

[121] Emilia Käsper. Fast Elliptic Curve Cryptography in OpenSSL. In *International Conference on Financial Cryptography and Data Security*. 2011.

[122] Byoungkoo Kim, Seoungyong Yoon, Yousung Kang, and Dooho Choi. PUF based IoT Device Authentication Scheme. In *International Conference on Information and Communication Technology Convergence (ICTC)*, 2019.

[123] Byoungkoo Kim, Seungyong Yoon, and Yousung Kang. PUF-based IoT Device Authentication Scheme on IoT Open Platform. In *International Conference on Information and Communication Technology Convergence (ICTC)*, 2021.

[124] Guy King. Best Security Practices: An Overview. In *National Information Systems Security Conference*, 2000.

[125] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, and Jeffrey Voas. DDoS in the IoT: Mirai and Other Botnets. *Computer*, 50(7):80–84, 2017.

[126] Jaidip Kotak and Yuval Elovici. IoT Device Identification Using Deep Learning. In *International Conference on Computational Intelligence in Security for Information Systems (CISIS 2020)*, 2020.

[127] Katharina Krombholz, Wilfried Mayer, Martin Schmiedecker, and Edgar Weippl. "I Have No Idea What I'm Doing"—On the Usability of Deploying HTTPS. In *Usenix Security Symp.*, 2017.

[128] Philippe Kruchten, Robert L. Nord, and Ipek Ozkaya. Technical Debt: From Metaphor to Theory and Practice. *IEEE Software*, 29:18–21, 2012.

[129] Arun Kumar, Nitesh Saxena, Gene Tsudik, and Ersin Uzun. A comparative study of secure device pairing methods. *Pervasive and Mobile Computing*, 5(6):734–749, 2009.

[130] Sam Kumar, Yuncong Hu, Michael P Andersen, Raluca Ada Popa, and David E Culler. JEDI: Many-to-Many End-to-End Encryption and Key Delegation for IoT. In *Usenix Security Symp.*, 2019.

[131] J. Richard Landis and Gary G. Koch. The Measurement of Observer Agreement for Categorical Data. *Biometrics*, 33(1):159, 1977.

[132] Eliot Lear, Ralph Droms, and Dan Romascanu. Manufacturer Usage Description Specification. RFC 8520, IETF, March 2019.

[133] Amit Levy, Bradford Campbell, Branden Ghena, Daniel B. Giffin, Pat Pannuto, Prabal Dutta, and Philip Levis. Multiprogramming a 64kB Computer Safely and Efficiently. In *Symposium on Operating Systems Principles (SOSP)*, 2017.

[134] Xiong Li, Jianwei Niu, Saru Kumari, Fan Wu, Arun Kumar Sangaiah, and Kim-Kwang Raymond Choo. A three-factor anonymous authentication scheme for wireless sensor networks in internet of things environments. *Journal of Network and Computer Applications*, 103(May 2017):194–204, 2018.

[135] Greg Lindsay, Beau Woods, and Joshua Corman. Smart Homes and the Internet of Things. `https://www.atlanticcouncil.org/wp-content/uploads/2016/03/Smart_Homes_0317_web.pdf`, 2016.

[136] An Liu and Peng Ning. TinyECC: a configurable library for elliptic curve cryptography in wireless sensor networks. In *International Conference on Information Processing in Sensor Networks*, 2008.

[137] Yabing Liu, Will Tome, Liang Zhang, David Choffnes, Dave Levin, Bruce Maggs, Alan Mislove, Aaron Schulman, and Christo Wilson. An End-to-End Measurement of Certificate Revocation in the Web's PKI. In *Internet Measurement Conference (IMC)*, 2015.

[138] Karim Lounis and Mohammad Zulkernine. Bluetooth Low Energy Makes "Just Works" Not Work. In *Cyber Security in Networking Conference (CSNet)*, 2019.

[139] Karim Lounis and Mohammad Zulkernine. Attacks and Defenses in Short-Range Wireless Technologies for IoT. *IEEE Access*, vol. 8:88892–88932, 2020.

[140] Karim Lounis and Mohammad Zulkernine. More Lessons: Analysis of PUF-based Authentication Protocols for IoT. *Cryptology ePrint Archive, Report 2021/1509*, 2021. `https://ia.cr/2021/1509`.

[141] Roel Maes and Ingrid Verbauwhede. *Physically Unclonable Functions: A Study on the State of the Art and Future Research Directions*, pages 3–37. Springer Berlin, Heidelberg, 2010.

[142] Khalid Mahmood, Shehzad Ashraf Chaudhry, Husnain Naqvi, Saru Kumari, Xiong Li, and Arun Kumar Sangaiah. An Elliptic Curve Cryptography Based Lightweight Authentication Scheme for Smart Grid Communication. *Future Generation Computer Systems*, 81, Apr 2018.

[143] Timothy Malche, Priti Maheshwary, and Rakesh Kumar. Secret Key based Sensor Node Security in the Internet of Things (IoT). In *International Conference on Communication and Electronics Systems (ICCES)*, 2020.

[144] Samuel Marchal, Markus Miettinen, Thien Duc Nguyen, Ahmad Reza Sadeghi, and N. Asokan. AuDI: Toward Autonomous IoT Device-Type Identification Using Periodic Communication. *IEEE JSAC*, 37(6), 2019.

[145] Shrirang Mare, Logan Girvin, Franziska Roesner, and Tadayoshi Kohno. Consumer Smart Homes: Where We Are and Where We Need to Go. In *Mobile Computing Systems and Applications*, 2019.

[146] Moxie Marlinspike. Reflections: The ecosystem is moving, 2016. `https://signal.org/blog/the-ecosystem-is-moving/`.

[147] Michael McCool and Elena Reshetova. Distributed security risks and opportunities in the W3C Web of Things. In *Workshop on Decentralized IoT Security and Standards (DISS)*, 2018.

[148] Nora McDonald, Sarita Schoenebeck, and Andrea Forte. Reliability and Inter-rater Reliability in Qualitative Research: Norms and Guidelines for CSCW and HCI Practice. *Proc. ACM Hum.-Comput. Interact*, 3(CSCW):1–23, Nov 2019.

[149] Gary McGraw. *Software Security: Building Security In (1st edition)*. Addison-Wesley Professional, 2006.

[150] Yair Meidan, Michael Bohadana, Asaf Shabtai, Juan David Guarnizo, Martín Ochoa, Nils Ole Tippenhauer, and Yuval Elovici. ProfilIoT: A Machine Learning Approach for IoT Device Identification Based on Network Traffic Analysis. In *ACM Symp. on Applied Computing*, pages 506–509, 2017.

[151] Microsoft. Security best practices for Internet of Things (IoT), 2018.

[152] Markus Miettinen, Samuel Marchal, Ibbad Hafeez, Tommaso Frassetto, N. Asokan, Ahmad Reza Sadeghi, and Sasu Tarkoma. IoT Sentinel: Automated Device-Type Identification for Security Enforcement in IoT. In *International Conference on Distributed Computing Systems*, pages 2511–2514. IEEE, 2017.

[153] Keith Moore, Richard Barnes, and Hannes Tschofenig. Best Current Practices (BCP) for IoT Devices. `https://www.ietf.org/archive/id/draft-moore-iot-security-bcp-01.txt`, July, 2017. IETF Internet-Draft (Expired).

[154] Philipp Morgner and Zinaida Benenson. Exploring Security Economics in IoT Standardization Efforts. In *Workshop on Decentralized IoT Security and Standards (DISS)*, 2018.

[155] Munkenyi Mukhandi, Francisco Damiao, Jorge Granjal, and Joao P. Vilela. Blockchain-based Device Identity Management with Consensus Authentication for IoT Devices. In *Consumer Communications & Networking Conference*, 2022.

[156] Alena Naiakshina, Anastasia Danilova, Christian Tiefenau, Marco Herzog, Sergej Dechand, and Matthew Smith. Why Do Developers Get Password Storage Wrong? A Qualitative Usability Study. In *ACM CCS*, 2017.

[157] Shadi Nashwan. Secure Authentication Scheme Using Diffie–Hellman Key Agreement for Smart IoT Irrigation Systems. *Electronics*, 11(2), 2022.

[158] NIST. Announcing the Advanced Encryption Standard (AES). Technical Report Federal Information Processing Standards Publications (FIPS PUBS) 197, U.S. Department of Commerce, 2001.

[159] NIST. NIST Releases Draft Guidance on Internet of Things Device Cybersecurity, 2020.

[160] Sina Rafati Niya, Benjamin Jeffrey, and Burkhard Stiller. KYoT: Self-sovereign IoT Identification with a Physically Unclonable Function. In *Conference on Local Computer Networks (LCN)*, number November, pages 1–9, 2020.

[161] Hirofumi Noguchi, Tatsuya Demizu, Naoto Hoshikawa, Misao Kataoka, and Yoji Yamato. Autonomous Device Identification Architecture for Internet of Things. In *IEEE World Forum on Internet of Things (WF-IoT)*, 2018.

[162] OASIS. MQTT version 5.0, Mar 2019. https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html.

[163] Charles W. O'Donnell, G. Edward Suh, and Srinivas Devadas. PUF-based random number generation. *MIT CSAIL CSG Technical Memo 481*, 2004.

[164] Online Trust Alliance (OTA). IoT Security & Privacy Trust Framework v2.5, 2017.

[165] Open Web Application Security Project (OWASP). OWASP Secure Coding Practices Quick Reference Guide, 2010.

[166] John Padgette, John Bahr, Mayank Batra, Marcel Holtmann, Rhonda Smithbey, Lily Chen, and Karen Scarfone. SP 800-121 Rev. 2—Guide to Bluetooth Security, 2017. NIST.

[167] Roberto Perdisci, Thomas Papastergiou, Omar Alrawi, and Manos Antonakakis. IoTFinder: Efficient Large-Scale Identification of IoT Devices via Passive DNS Traffic Analysis. In *IEEE EuroS&P*, pages 1–16, 2020.

[168] PlainTextOffenders.com. `https://plaintextoffenders.com/`, 2021.

[169] Jon Postel, Yakov Rekhter, and Tony Li. Best Current Practices. RFC 1818, IETF, August 1995.

[170] PSA Certified. Critical security questions for chip vendors, OS providers and OEMs, 2019.

[171] Indrajit Ray, Diptendu M. Kar, Jordan Peterson, and Steve Goeringer. Device Identity and Trust in IoT-Sphere Forsaking Cryptography. In *International Conf. on Collaboration and Internet Computing (CIC)*, 2019.

[172] Dipankar Raychaudhuri, Kiran Nagaraja, North Brunswick, and Arun Venkataramani. MobilityFirst: A Robust and Trustworthy Mobility- Centric Architecture for the Future Internet. *ACM SIGMobile Mobile Computing and Communication Review*, 16(4):1–12, 2012.

[173] Elissa M. Redmiles, M. Morales, Lisa Maszkiewicz, R. Stevens, Everest Liu, Dhruv Kuchhal, and Michelle L. Mazurek. First Steps Toward Measuring the Readability of Security Advice. In *Workshop on Technology and Consumer Protection*, 2018.

[174] Elissa M. Redmiles, Noel Warford, Amritha Jayanti, Aravind Koneru, Sean Kross, M. Morales, R. Stevens, and Michelle L. Mazurek. A Comprehensive Quality Evaluation of Security and Privacy Advice on the Web. In *Usenix Security Symp.*, 2020.

[175] Karen Renaud. How Smaller Businesses Struggle with Security Advice. *Computer Fraud and Security*, 2016(8):10–18, 2016.

[176] Rodrigo Roman, Cristina Alcaraz, Javier Lopez, and Nicolas Sklavos. Key management systems for sensor networks in the context of the Internet of Things. *Computers & Electrical Engineering*, 37(2):147–159, mar 2011.

[177] Eyal Ronen, Adi Shamir, Achi-Or Weingarten, and Colin OFlynn. IoT Goes Nuclear: Creating a ZigBee Chain Reaction. In *IEEE Symp. on Security and Privacy*, 2017.

[178] Ron Ross, Michael McEvilley, and Janet Carrier Oren. SP 800-160 Vol. 1—Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems (Volume 1). `https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-160v1.pdf`, 2016. NIST.

[179] Ulrich Ruhrmair and Daniel E. Holcomb. PUFs at a glance. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2014.

[180] Ulrich Rührmair, Frank Sehnke, Jan Sölter, Gideon Dror, Srinivas Devadas, and Jürgen Schmidhuber. Modeling Attacks on Physical Unclonable Functions. In *ACM CCS*, 2010.

[181] Ola Salman, Imad H. Elhajj, Ali Chehab, and Ayman Kayssi. A Machine Learning Based Framework for IoT Device Identification and Abnormal Traffic Detection. *Trans. on Emerging Telecommunications Tech.*, (July 2019), 2019.

[182] Jerome H. Saltzer and Michael D. Schroeder. The Protection of Information in Computer Systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975.

[183] Matias R.P. Santos, Rossana M.C. Andrade, Danielo G. Gomes, and Arthur C. Callado. An efficient approach for device identification and traffic classification in IoT ecosystems. In *IEEE Symp. on Computers and Communications*, 2018.

[184] Behcet Sarikaya, Mohit Sethi, and Dan Garcia-Carrillo. Internet Draft: Secure IoT Bootstrapping: A Survey, 2019. Document: draft-sarikaya-t2trg-sbootstrapping-05.

[185] Georgios Selimis, Mario Konijnenburg, Maryam Ashouei, Jos Huisken, Harmke de Groot, Vincent van der Leest, Geert-Jan Schrijen, Marten van Hulst, and Pim Tuyls. Evaluation of 90nm 6T-SRAM as Physical Unclonable Function for Secure Key Generation in Wireless Sensor Nodes. In *International Symposium of Circuits and Systems (ISCAS)*, 2011.

[186] Wentao Shang, Adeola Bannis, Teng Liang, Zhehao Wang, Yingdi Yu, Alexander Afanasyev, Jeff Thompson, Jeff Burke, Beichuan Zhang, and Lixia Zhang. Named Data Networking of Things (Invited Paper). In *International Conference on Internet-of-Things Design and Implementation*, 2016.

[187] Z. Shelby, K. Hartke, and C. Bormann. The constrained application protocol (coap). RFC 7252, RFC Editor, June 2014.

[188] Adam Shostack and Andrew Stewart. *The New School of Information Security*. Pearson Education, 2008.

[189] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini. Security, privacy and trust in Internet of Things: the road ahead. *Computer Networks*, 76:146–164, 2015.

[190] Kushal Singla and Joy Bose. IoT2Vec: Identification of Similar IoT Devices via Activity Footprints. In *International Conf. on Advances in Computing, Communications and Informatics*. IEEE, 2018.

[191] Sean Smith. *The Internet of Risky Things: Trusting the Devices That Surround Us*. O'Reilly Media, 2017.

[192] Saleh Soltan, Prateek Mittal, and H. Vincent Poor. BlackIoT: IoT Botnet of High Wattage Devices Can Disrupt the Power Grid. In *Usenix Security Symp.*, 2018.

[193] Abhay Soorya and 21 others. IoT Security Compliance Framework 2.0, 2018.

[194] Claudio Soriente, Gene Tsudik, and Ersin Uzun. Secure pairing of interface constrained devices. *International Journal of Security and Networks*, 4(1/2), 2009.

[195] Frank Stajano and Ross Anderson. The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks. In *International Workshop on Security Protocols*, 1999.

[196] John A. Stankovic. Research directions for the Internet of Things. *IEEE Internet of Things Journal*, 1(1):3–9, 2014.

[197] Gary Stoneburner, Clark Hayden, and Alexis Feringa. SP 800-27 RevA—Engineering Principles for Information Technology Security (A Baseline for Achieving Security), 2004. NIST.

[198] G. Edward Suh and Srinivas Devadas. Physical Unclonable Functions for Device Authentication and Secret Key Generation. In *Conference on Design Automation*, 2007.

[199] Jani Suomalainen, Jukka Valkonen, and N. Asokan. Standards for security associations in personal networks: a comparative analysis. *International Journal of Security and Networks*, 4(1/2):87, 2009.

[200] Sven Schrecker and 14 others. Industrial Internet of Things Volume G4: Security Framework v1.0, 2016.

[201] Shahin Tajik, Enrico Dietz, Sven Frohmann, Jean-Pierre Seifert, Dmitry Nedospasov, Clemens Helfmeier, Christian Boit, and Helmar Dittrich. Physical Characterization of Arbiter PUFs. In *Cryptographic Hardware and Embedded Systems (CHES)*, 2014.

[202] David R. Thomas. A General Inductive Approach for Analyzing Qualitative Evaluation Data. *American Journal of Evaluation*, 27(2):237–246, Jun 2006.

[203] Mathias Dahl Thomsen, Alberto Giaretta, and Nicola Dragoni. Smart Lamp or Security Camera? Automatic Identification of IoT Devices. In *International Networking Conference*, volume 180 of *Lecture Notes in Networks and Systems*. Springer International Publishing, 2021.

[204] Hannes Tschofenig and Emmanuel Baccelli. Cyberphysical Security for the Masses: A Survey of the Internet Protocol Suite for Internet of Things Security. *IEEE Security & Privacy*, 17(5):47–57, Sep 2019.

[205] Muhamed Turkanović, Boštjan Brumen, and Marko Hölbl. A novel user authentication and key agreement scheme for heterogeneous ad hoc wireless sensor networks, based on the Internet of Things notion. *Ad Hoc Networks*, 20, 2014.

[206] Pim Tuyls, Geert-Jan Schrijen, Boris Škorić, Jan van Geloven, Nynke Verhaegh, and Rob Wolters. Read-Proof Hardware from Protective Coatings. In *Cryptographic Hardware and Embedded Systems (CHES)*, 2006.

[207] Martin Ukrop, Lydia Kraus, Vashek Matyas, and Ahmad Mutleq Wahsheh Heider. Will You Trust This TLS Certificate?: Perceptions of People Working in IT. In *Annual Computer Security Applications Conference (ACSAC)*, 2019.

[208] US National Telecommunications and Information Administration (NTIA). Voluntary Framework for Enhancing Update Process Security, 2017.

[209] Ersin Uzun, Kristiina Karvonen, and N. Asokan. Usability Analysis of Secure Pairing Methods. In *Financial Crypto*, 2007.

[210] Paul C. van Oorschot. *Computer Security and the Internet: Tools and Jewels*. Springer, 2020.

[211] Mathy Vanhoef and Eyal Ronen. Dragonblood: Analyzing the Dragonfly Handshake of WPA3 and EAP-pwd. In *IEEE Symp. Security and Privacy*, 2020.

[212] Arun Venkataramani, Abhigyan Sharma, Xiaozheng Tie, Hardeep Uppal, David Westbrook, Jim Kurose, and Dipankar Raychaudhuri. Design requirements of a global name service for a mobility-centric, trustworthy internetwork. In *International Conference on Communication Systems and Networks*, 2013.

[213] A.S. Wander, Nils Gura, Hans Eberle, Vipul Gupta, and Sheueling Chang Shantz. Energy Analysis of Public-Key Cryptography for Wireless Sensor Networks. In *International Conference on Pervasive Computing and Communications*, 2005.

[214] Hongyuan Wang, Jin Meng, Xilong Du, Tengfei Cao, and Yong Xie. Lightweight and Anonymous Mutual Authentication Protocol for Edge IoT Nodes with Physical Unclonable Function. *Security and Communication Networks*, 2022(1), 2022.

[215] Qi Wang, Wajih Ul Hassan, Adam Bates, and Carl Gunter. Fear and logging in the Internet of Things. In *NDSS*, 2018.

[216] Dan Wendlandt, David G. Andersen, and Adrian Perrig. Perspectives: Improving SSH-style Host Authentication with Multi-Path Probing. In *USENIX Annual Technical Conference*, 2008.

[217] Marilyn Wolf and Dimitrios Serpanos. Safety and Security in Cyber-Physical Systems and Internet-of-Things Systems. *Proceedings of the IEEE*, 106(1):9–20, 2018.

[218] Ford-Long Wong, Frank Stajano, and Jolyon Clulow. Repairing the Bluetooth Pairing Protocol. In *International Workshop on Security Protocols*, 2005.

[219] Felix Wortmann and Kristina Flüchter. Internet of Things. *Business & Information Systems Engineering*, 57:221–224, 2015.

[220] Xilinx. Defense-Grade Zynq UltraScale+ RFSoCs, 2022.

[221] Kaiping Xue, Changsha Ma, Peilin Hong, and Rong Ding. A temporal-credential-based mutual authentication and key agreement scheme for wireless sensor networks. *Journal of Network and Computer Applications*, 36(1), 2013.

[222] Kuiwu Yang, Kangfeng Zheng, Yuanbo Guo, and Dawei Wei. PUF-Based Node Mutual Authentication Scheme for Delay Tolerant Mobile Sensor Network. In *Int. Conf. on Wireless Communications, Networking and Mobile Computing*, 2011.

[223] Yuchen Yang, Longfei Wu, Guisheng Yin, Lijie Li, and Hongbin Zhao. A Survey on Security and Privacy Issues in Internet-of-Things. *IEEE Internet of Things Journal*, 4(5):1250–1258, 2017.

[224] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. Wireless sensor network survey. *Computer Networks*, 52(12):2292–2330, 2008.

[225] Narges Yousefnezhad, Manik Madhikermi, and Kary Framling. MeDI: Measurement-based Device Identification Framework for Internet of Things. In *International Conference on Industrial Informatics (INDIN)*, 2018.

[226] Dan Yu, Peiyang Li, Yongle Chen, Yao Ma, and Junjie Chen. A Time-efficient Multi-Protocol Probe Scheme for Fine-grain IoT Device Identification. *Sensors*, 20(7), 2020.

[227] Lingjing Yu, Bo Luo, Jun Ma, Zhaoyu Zhou, and Qingyun Liu. You Are What You Broadcast: Identification of Mobile and IoT Devices from (Public) WiFi. In *USENIX Security*, 2020.

[228] Meng-Day Yu and Srinivas Devadas. Pervasive, Dynamic Authentication of Physical Items. *ACM Queue*, 14(6), 2016.

[229] Wei Zhang, Yan Meng, Yugeng Liu, Xiaokuan Zhang, Yinqian Zhang, and Haojin Zhu. HoMonit: monitoring smart home apps from encrypted traffic. In *ACM CCS*, 2018.

[230] Zhi Kai Zhang, Michael Cheng Yi Cho, Chia Wei Wang, Chia Wei Hsu, Chong Kuan Chen, and Shiuhpyng Shieh. IoT Security: Ongoing Challenges and Research Opportunities. In *International Conference on Service-Oriented Computing and Applications*, 2014.

Appendix A

A.1 SAcoding Method Software Interface Tool and 1013-Item Dataset

A website interface for the SAcoding method presented in Chapter 4 is available at URL (1) below. At the URL, the results of our coding of the DCMS 1013-item dataset are provided, as well as the software SAcoding method tool. Readers can test the SAcoding method on advice items from the 1013-item dataset, and then compare the tag they gave to an advice item with those of C1 and C2 from Chapters 4 and 5 (provided in the website). Within this website is the full 1013-item dataset (we processed the larger DCMS set [59] down to a slightly smaller set of 1013) and original sources for each advice item.

Separate from the website interface, both coders' results (with related advice items) from Chapters 4 and 5 are also available as a stand-alone JSON file. This file is available through URL (2) below.

Source URLs:

1. SAcoding method web interface:
   `https://www.ccsl.carleton.ca/~chris/sacoding/`
2. Full 1013-item coder results (and advice items):
   `https://github.com/ChristopherBellman/SecurityAdvice/blob/eda8b4fc08f28ee7f7dcfbba92446ca3b011bf2b/cb1013-dataset-thesis.json`