Towards a Secure IoT: Directions for IoT Research

Jean Camp[*], Ryan Henry[‡*], Tadayoshi Kohno[†], Shrirang Mare[*†], Steven Myers[*], Shwetak Patel[†] and Joshua Streiff[*]
[*]Luddy School of Informatics, Computing, and Engineering, Indiana University
[†]Allen School of Computer Science & Engineering, University of Washington
[‡]Department of Computer Science, University of Calgary

***Abstract*—The current state of Internet of Things is woefully insecure, and reaching a secure state requires addressing several serious gaps. Based on the discussions with practitioners and researchers, we identify key gaps and research challenges that must be overcome to chart a path toward a secure IoT.**

***Index Terms*—Cryptography, Internet of Things, Privacy, Security, Usability**

I. Introduction

From light bulbs to refrigerators, children's toys to cars, consumer devices are being increasingly connected to home networks and the Internet. Connectivity offers many benefits to homeowners. For example, homeowners can remotely control their home thermostats to save on utility bills or monitor their homes when they are away. However, it also exposes the home to various online threats: an adversary could compromise or take control of home devices, for example, to collect (and sell) homeowner credentials or to carry out denial of service attacks.

While best practices to secure IoT devices exist, they are often more aspirational than operational. Many of these practices require device owners to change their behavior, e.g., to use a separate Wi-Fi network for guests or certain devices, and to change passwords regularly. However, expecting people to change well-documented past behaviors is arguably unreasonable [1]. It is often easier to change technology than human behavior. Furthermore, many current devices are poorly designed, making it impossible for owners to follow and adopt best practices. Consider the basic practice from the eighties "change the default password," which cannot be adopted for products with unchangeable default passwords or inaccessible password-change interfaces.

How can we move towards a secure Internet of Things? This was the central question we posed to practitioners and researchers gathered at an IoT workshop in late 2017. We discussed the challenges and created a roadmap for a secure IoT. Inspired by these discussions and the conversations that ensued, we highlight in this paper the key challenges and opportunities along four research paths that broadly reflect workshop participants' collective expertise and views: (1) improving the cryptographic infrastructure, (2) designing more usable devices to help people leverage device capabilities, (3) enabling recovery when devices fail or are compromised, and (4) writing secure code when developing the devices.

Pre-print copy.
IEEE Security & Privacy. February 2020.
DOI: 10.1109/MSEC.2020.2970155

Threat Model. Before proceeding, we consider one question that spans all four research directions: *Who is the adversary?* Potential adversaries for a home owner or a device include another occupant in the home (insider), a neighbor, a guest, the developer of an app used on a home device, the device manufacturer, or a nation state. Security requirements vary based on the adversary and the threat that must be addressed. For example, it is easier to protect a device from a nosy neighbor than against a targeted attack from a nation state. Industry practitioners need clarity about the threat model to appropriately protect their devices. From an industry perspective, it may be appropriate to design consumer IoT devices by considering the threat model of an average consumer. However, an attack on home IoT devices may have implications beyond individual consumers' homes and it is important to consider a threat model that is broader than that of an individual home. For example, consumers may not care about a nation state targeting them individually, but a nation-state attacker could use insecure home IoT devices as a springboard to attack a country's critical infrastructure. While it may be futile to protect an individual device/home against a targeted attack from a motivated nation-state attacker, it may be feasible to prevent a simultaneous large-scale compromise of IoT devices. Policymakers and technologists should consider how to make that possible. There is a need for clear guidelines to (and from) both device manufacturers and policymakers on appropriate threat models for IoT devices.

II. Cryptography: Securing the IoT Infrastructure

Cryptography is critical to securing the IoT infrastructure. Generally, infrastructure takes a very long time to change, so it is important to get cryptography right while the infrastructure is still being set up. An important aspect of the IoT infrastructure is developing devices that would meet the security requirements now but also in the future. When developing devices, choices must be made about algorithms and key strengths, and poor choices have long-lasting effects. In devices developed today, we are already seeing poor design choices (e.g., using SHA1, a hash algorithm that was deprecated a decade ago by many organizations, including NIST). Thus, there is an urgency to address cryptographic issues in practice.

Entropy. A core cryptographic challenge in IoT is generating sufficient entropy (i.e., randomness) in encryption keys used by low-power devices. Weak keys are widespread in network devices. One main reason is that these devices have more limited entropy sources than traditional PCs [2]. Keys on IoT devices should be generated for the context of devices use; for certain durable appliances, that context could be the device's lifetime (e.g., decades for refrigerators; years for lightbulbs). Potential solutions to address this problem include seeding devices with entropy at manufacturing or using the physical environment around IoT devices as sources of entropy (e.g., using physically unclonable functions) [2], [3].

Cryptographic agility. IoT goods like televisions, furnaces, water heaters, and refrigerators have long lifetimes. As a result, we should also consider post-quantum cryptography, that is, cryptographic methods that would be relatively secure against quantum computers or major advances in factoring; current cryptographic primitives are based on assumptions about computational capabilities that are not shared by post-quantum primitives. The emergence of quantum attacks should not be able to result in simultaneous mass functional expiration of thermostats, appliances, or other devices in homes across the globe.

One challenge for the adoption of post-quantum cryptography is the lack of universal agreement about which post-quantum standards are acceptable. Identification of such standards would go far to address the issue. While NIST has a complete process, the adoption of lightweight post-quantum cryptography standards does not seem likely in the near term. (Two years after the workshop 32 candidate standards have made it to the second round of this process [4].) Another challenge is that quantum attacks are generally not part of consumers' threat model, and hardware manufacturers may ignore or openly reject any requirements specific it. However, potential exists to bridge pre- and post-quantum cryptographic infrastructures with hash-based signatures and by reusing existing circuits for classical elliptic curve-based cryptography (based on super-singular elliptic curve isogenies).

Discussions of post-quantum cryptography are often proxies for broader discussions on the need for cryptographic agility in the face of other possible advances—e.g., fundamental mathematical breakthroughs that break cryptographic assumptions, or significant changes in algorithms—that would require an immediate, agile response. Uncertainty associated with post-quantum cryptography should not prevent more immediate investments in cryptographic agility. But the nature of *that investment* remains an open research question.

Minimal cryptography. The question of the required minimum level of cryptography, or, more broadly, the minimum level of security, is contextual, and thus an open question. Identifying the cryptographic requirements of a device in a given context may also require addressing questions on contextual usability and threat models. Should devices have a wipe button, or is loss of data worse than loss of privacy? If so, what should trigger wipe functionality? For example, should it be a time- or location-based (e.g., wipe if the device is unused for an extended period or changes location)? Should wipes occur with credentials left intact on the device? Strong encryption versus short-lived keys is an issue that combines usability and context, as does automatic re-keying, re-enrollment, and re-certification. The wide range of processing power, lifetimes, and changing context of IoT devices further highlights the importance of cryptographic agility.

Securing low-power and batteryless devices. It is challenging to add cryptographic primitives in low-power devices, and even more so in batteryless ones that harvest energy. These devices may have intermittent and unreliable power supplies. This constraint raises the tension between adding cryptography in software vs. hardware. Cryptography in software offers the ability to change primitives through software updates, but it can be slow and consume more energy. Cryptography in hardware, more efficient and fast, is difficult to update. It is likely that there will be a tiered category of devices that have different cryptographic capabilities and offer different levels of security and trustworthiness. An IoT network should be able to recognize such devices and treat them accordingly, for example, by limiting network access for less trustworthy devices.

The information gap between cryptographers and device vendors. A significant gap exists between advances in cryptography and implementation practices in the industry. For example, there are several different cryptographic primitives, and the optimal choice varies on factors such as device constraints, the context of device use, and the device's expected longevity. However, device manufacturers and developers may be unaware of which primitive works best in what situation. To inform them of these matters, a *taxonomy* that defines specific security requirements as well as the cost and the benefits of different choices would be helpful. The taxonomy could be used to classify cryptographic primitives (including physical components) according to their applicability to application domains and hardware requirements (e.g., the number of gates needed to implement them in hardware). It would help developers decide, for example, that for their device and application, the optimal choice would be the second most common primitive (e.g., SHA2), not the first (e.g., AES).

III. Usability: Creating Usable IoT Devices

Humans cannot be kept entirely out of the loop [5]. Improving usability by focusing only on improving ease-of-use while ignoring security or privacy is a narrow view that leads to security and privacy failures. Instead, IoT device designers should take a holistic view of usability where the focus is on improving ease-of-use *while* respecting users' privacy and security needs. The concept of usability should be grounded in the *principle of least surprise*, which combines usability, functionality, and risk.

A long-held industry view is that users need to be "trained" to follow good security practices (e.g., password policies). This view is shifting to what the usable security community has known for decades: security should be designed to align with users' expectations and should require minimal (ideally, no) change in user behavior [6], [7]. If end users need to think about security, then arguably we security professionals have already failed. This view extends to developers, as well: if developers need to understand cryptography in the IoT, then the battle for security can be considered similarly lost.

Mental models of IoT. An important research area, and an important first step in developing usable products, is understanding people's mental models of IoT [8], viz., how people currently use technology and what their expectations are in terms of utility, privacy, and security. Doing so lets us design solutions that conform to the least surprise principle, either by designing solutions that align with users' expectations or by informing them when solutions do not so align.

Key questions to explore when studying people's mental models of IoT include: What do people know about device capabilities (e.g., data collection)? How do they envision smart home automation? How do they think devices would behave in the event of an Internet or power outage? What are their access control needs and expectations? What inferences do they believe can be drawn from different devices (using data from either one device or a set of devices)? How do people perceive the security and privacy risks of using IoT devices?

Meaningful user controls. Challenges concerning data control are not unique to IoT devices. They exist for personal devices, apps, and services, but the need for visibility and control is high in the IoT context because these devices enable passive and continuous data collection.

Different stakeholders have different needs based on individual preferences and/or contextual settings. Two people using the same IoT system are likely to have different experiences of agency and different expectations in terms of usability, security, and privacy. For example, one person may value retaining data over privacy, while another may prefer data loss to risk of data exposure. Moreover, IoT devices are often used in shared spaces, which poses an additional challenge: one person's decision to install a device in a shared space could impinge on the security or privacy of others who share that space. Thus, it is necessary to design meaningful controls in IoT devices so that people can tailor devices to their expectations, which may change depending on their social context. Controls could be designed for data collection and use, device maintenance (updates), and/or device access.

The first aspect of data control is *data ownership*. Who owns the data collected by an IoT device—the device owner, the device user (if different), or the device manufacturer? Some companies consider device owners to be data owners; some companies claim data ownership; and some do not specify. Data ownership should be clearly communicated at the time of purchase. Furthermore, data owners should have the ability to observe (ideally, choose) where the data is stored and how it is used, and they should be able to delete their data, because ownership is pointless if data owners lack such control.

Another aspect of control concerns *software updates*. Whether device owners should have control of and responsibility for device updates remains an open question. Automatic updates increase the chances that devices will be updated in a timely way. But mandatory updates—i.e., where a device *stops* working until it is updated—may pose safety risks (e.g., if a smart smoke sensor stops working, a door is unlockable) and may even be considered unethical. One possible resolution to this dilemma is for device manufacturers to release separate updates: updates that change functionality (which would require user intervention or consent) and updates that contain only security patches (which can be automatic). Fundamentally, no one should be forced to choose between a secure but unacceptable state (by doing a security update that also contains new undesirable functionality) and a vulnerable but working state (by not implementing that update).

Lastly, *access control* for IoT devices is an important research area, posing a myriad of open questions. At the core of these questions is the need to determine appropriate access control models for IoT devices. This is further conflated by the fact that many IoT devices lack an interface for authentication and authorization or can be used by multiple people at the same time (e.g., voice assistants, smart speakers). From the usability perspective, the challenge is to design IoT access control models that would *just work*.

Meaningful stakeholder communication. In addition to meaningful controls, there must be meaningful communication about the controls, about risks associated with devices, and ways to mitigate these risks. Ideally, there should be a culture of ongoing communication and feedback—between device users and manufacturers—that informs but does not expose or overwhelm users. Social engineering and human cognitive models illustrate how a failure to design for usability and risk communication is an attack vector.

One of the main uses for IoT devices is home automation: a consumer can string together multiple IoT devices to create an automation that offers convenience. For example, lights in the living room turn on when someone enters the room (walking triggers a motion sensor in the room, which then notifies the lights to activate); or a smart doorlock unlocks when an individual in the living room issues a voice command. But such multi-device automation can lead to unintended consequences and pose new threats. For instance, in the preceding example, if the voice assistant was located near an entrance door, an adversary just outside the door could issue the unlock command, which is clearly a security flaw. The flaw, however, is neither with the voice assistant system, which makes no security claims, nor with the door lock. The flaw comes from using these devices in this combination and in this context. Thus, risk communication about a device should address potential security implications of using the device in combinations with other devices.

Devices may break, malfunction, or be compromised, and device owners should be able to identify these incidents and take appropriate action. A smart hub in a home or an ISP may be able to detect compromised devices in a home network, but communicating this status to the end user is not straightforward. Consider the problem of locating a compromised device in a home. Today, many consumers can easily locate an IoT device in their home because they have only a handful of such devices. However, as the cost and size of devices decreases in the future, people will likely have many devices in their homes, and locating a malfunctioning one may be more difficult. Effective notifications that help smart home users locate malfunctioning devices and guide them to mitigate any risk is a critical research domain.

Another challenge is the allocation of risk and meaningfully communicating risk to consumers. When device owners connect their homes to the Internet, they put themselves at risk, just as they do when driving an automobile or engaging with other technologies. This risk can be reduced by avoiding technology, but doing so may result in fewer benefits from the technology (e.g., home security, health monitoring), and, for an individual, lost benefits may exceed reduced risks. Thus, effective risk-benefit communication is important and remains a challenge. To effectively communicate risk, one possible approach is to use a risk rating system similar to product review ratings, since consumers are already familiar with such systems. But given the different domains and contexts in which IoT devices can be used (e.g., security, health, residential, enterprise), it may be impossible to have meaningful risk ratings for devices across all domains and contexts. An alternate approach could be to provide both ratings and appropriate metaphors [9] that show the pathways to change toward a culture of security and safety.

IV. Recovery: Helping Devices Fail/Recover Gracefully

Devices will break, malfunction, and be compromised. Therefore, it is important to have recovery mechanisms that can: (1) ideally, fix the damaged device, (2) limit the damage from affecting other devices on the network, and/or (3) prevent risk to life and property.

Known safe state. The first step to recovery is resetting the affected device to a known *safe* state, which could be its initial manufacturing state or the last known operational safe state. If a safe state is stored in secure (e.g., tamper-proof) hardware, there can be a high level of trust that the state has not been compromised and that it is indeed a *known* safe state from which full recovery is possible. Yet, most devices today do not ship with secure hardware to store a safe state. The challenge is defining the requirements (e.g., minimal trusted base) and mechanisms to reliably and securely save and load safe states in devices.

Automatic patching. Resetting a device to a known safe state will not result in recovery if that state has a vulnerability. Thus, after resetting a device, patching the vulnerability is critical. The history of software updates suggests that security updates should be automatic to the extent possible. As discussed previously, software updating poses issues of consent and the role of the device owner. Manufacturers may decide not to have automatic patching as an option, but that should not be the default without contextual justification. Not every vulnerability, however, poses a significant risk and requires patching. For example, a user may decide that certain devices are not critical or trustworthy (Section III), or it may be appropriate to not patch a device even with known vulnerability because of additional safeguards already in place. But for devices that are critical (e.g., water heaters) or expensive to replace (e.g., refrigerators), manufacturers should add automatic patching or address the risk generated by the decision not to do so. Addressing that risk requires clear guidelines on developing reliable and secure automatic patching systems; such guidelines or standards would be beneficial to both consumers and manufacturers.

Isolation and transparency. When patching a vulnerable device is not feasible, risk mitigation is required. Mitigation may be based on isolation of the affected device. Isolation via network micro-segmentation is a popular solution to contain information leakage and limit the impact of vulnerabilities. Some home network solutions provide network boundaries for each device (e.g., solutions based on MUD https://tools.ietf.org/html/rfc8520). The winners of the 2017 Federal Trade Commission contest for protecting IoT devices in the home also focused on transparency and isolation (https://www.ftc.gov/iot-home-inspector-challenge). Device isolation could occur during installation or be set dynamically; for example, if a compromised device is detected, a security hub in the home could firewall and isolate it [10]. These are promising approaches, but extending them to a wide range of IoT devices could raise yet unconsidered problems.

Transparency about devices helps reduce surprise and offers value across all stages of their lifecycle. At the time of device purchase, basic facts should be made clear: the expected lifetime, expectations of support from the device manufacturer, and device retirement procedures. During a device's lifetime, if a vulnerability is discovered, manufacturers should adhere to a clear policy of notifying device owners and describing how the vulnerability affects recovery procedures. It should be clear to owners if and when devices need to be updated and when updates are complete. And finally, as device retirement nears, it is critical to communicate clearly when support for the device ends, how to wipe the device (e.g., of any personal data), and how to recycle it.

Device expiration date. The need to support older devices can be cost prohibitive to manufacturers and hinder radical device improvements. To address this problem, one approach is the notion of *expiration dates* for devices and software. Expiration dates can offer consumers some guarantees and provide management cycles that allow consumers to plan. Consider the case of iPhones, where older phones cannot connect to the App Store. The success of the iPhone illustrates that forced upgrades are feasible, at least for high-end devices. The drawback is that expiration dates mandate new purchases. They also create strong perverse incentives to build devices that must be replaced. Imagine the cost if all older OnStar-equipped automobiles had to be replaced instead of patched when vulnerabilities were discovered (https://nyti.ms/2kkonqE).

In addition to perverse incentives are issues of carbon impact. More importantly, it is unclear whether device-expiry dates would be effective in the context of IoT devices, i.e., would users get new secure devices? Again consider iPhones: people continue to use older iPhones in an insecure mode with untrustworthy stores. Thus, expiration dates may address the issue of security to some extent, but they also create strong perverse incentives and come at potentially vast environmental costs.

Failsafe. Events such as Internet outages, power outages, or data/device breaches can and do happen. To ensure safe and reliable operation during such events, critical devices should have built-in safeguards. For example, an Internet-connected thermostat should be prevented from indiscriminately heating/cooling a home beyond safe temperatures because it cannot connect to the Internet or is remotely controlled by an adversary; doors' locking and unlocking mechanisms should remain functional despite power outages. Thus, it is important to design failsafe measures, but they should be designed in a way that prevents common misuse. For instance, in the event of a power failure due to fire, it is safety critical that a smart door should open to let people out; if a thief cuts the power to the home, it is equally safety critical not to unlock the door. Detecting such events and taking appropriate recovery actions, which depend on the device and context should be considered. Failsafe operations for a home-based IoT environment are not well understood, but guidance from safety-critical systems provide a good starting research path [11].

Code integrity and verification. Success and failures in other domains can inform recoverability in IoT. Issues related to roots of trust, economic incentives, and challenges of end-of-life systems existed before IoT, and lessons from those domains should not be neglected. The two primary creators of IoT infrastructure include Internet companies, experts in connectivity and familiar with shipping fast and patching later, and device (or Things) manufacturers, experts in making physical devices but new to connectivity. Learning between the "I"(nternet) and the "T"(hings) requires reaching across industry boundaries. Consider the issue of manifests: How do we create industry-readable manifests? How can we verify them? And how do we create tamper-resistant or tamper-evident manifests? The automotive industry is an example of manufacturers with expertise in certification and manifests, and in this case, the manufacturer is responsible for vehicle functionality. The automotive industry understands how to ship with manifests, and the IT industry understands how to threat model and build roots of trust. Together, these domains could move IoT devices toward code integrity and verification. Further work in this area requires social scientists and industrial economists as much as computer scientists.

V. Code: Supporting Developers in Writing Secure Code

The problem of reducing the amount of insecure code is not unique to IoT devices. But the implications of insecure code may be different in IoT devices compared to, for example, PCs or smartphones, because these devices control home environments and may pose safety risks to life and property. Moreover, their scale and heterogeneity increase the complexity of and constraints on code: There are thousands of different IoT devices that use different platforms. The desire to connect devices gave rise to varied ways of integrating them; for example, via third-party hub devices, smartphone apps, Zigbee, or web services like IFTTT (https://ifttt.com). Each integration method offers a potential attack vector. To further complicate matters, some IoT devices may need built-in autonomy and/or recoverability from possible failures because they have to function in unpredictable environments and without any assistance from people.

There can be no principle of least surprise (Section III) if developers themselves are surprised by their own code, which could happen for several reasons. First, most IoT development occurs in integrated development environments for *opaque* platforms. Second, projects share a tremendous amount of code from which developers may inherit technical debt (e.g., obscure code, vulnerabilities). Finally, the lack of incentives to write secure code and time-to-market pressures force developers to prioritize functionality features over security/privacy features, in part because the former is more visible to end users.

Developer incentives. A range of possible approaches can create incentives to write secure code. One is increased transparency (Sections III and Section IV), which can create marketplace incentives to promote security. At another extreme is strict liability, where developers can be held accountable for security issues in their code. Perhaps, there is a space between today's wild west and the strict regulatory environment that would be optimal for IoT, but both data and research are needed to identify it.

Improving the training and expertise of developers is a rich area for academia-industry collaboration. Companies can outline their developer requirements and academic institutions can design industry-specific courses, certificates, or even degrees. Badges also have great potential to showcase skills where a university degree or industry certification models are too heavy. Developers with the skills to easily identify the minimal library and permissions, use the tools that provide the correct cryptography, annotate, and provide provenance are extremely valuable. Badges can be developed and monetized to improve code quality, and such a system could be incentive-aligned for all the stakeholders.

Usable security APIs and libraries. The ideal development environment makes it easier (or, at least, no more difficult) to write secure code than insecure code. One could achieve this goal by, for example, embedding input validation into APIs, provisioning libraries for common points of failure, and making it trivial to use libraries that allow integration and testing. As noted in Section II, usable and verified cryptographic libraries provide value, but only if they are easily discoverable by developers. Developer forums are filled with insecure and incorrect advice on security implementations, making it difficult for developers to find correct advice and appropriate tools.

Many existing libraries contain insecure code, and developers inadvertently use them. From a developer perspective, it is easiest to import an entire library to access even a single function, which results in overuse of libraries and permissions. This is a chronic problem in mobile development [12]. Developers may not know when they are using insecure libraries because the extent of inclusion of vulnerabilities via reference is itself an open research question. One possible step forward is regular reports on code use and library inclusion, analogous to transportation or other infrastructure reporting. Even annotations, which make it clear which libraries are actually being used and which are included for convenience, would be an improvement over common practices.

Code verification and validation. Consistent, reliable code verification and validation lets developers write secure code. *Code verification* means checking that code meets the program specification. This is usually done using code reviews, static analysis checkers, and occasionally, at compile time. *Code validation* means checking that code meets the needs and expectations of its users. It is generally performed using dynamic testing, but static analysis can at times also validate code. Both are needed; neither is a substitute for the other.

Annotations facilitate automatic review and confirmation, and they also improve code documentation. However, they are not widely used: indeed, code documentation and annotations are unsolved challenges. It is very difficult to validate code that is written without assessment as a goal, and it is rare to find annotated code. Smart home platform vendors are in a position to encourage developers to write secure and annotated code. They could also seek a minimal level of annotation and documentation in their apps and use that annotation for app approval or recommendation. Verifiability and verification of code is an open research challenge.

One possible approach to code assessment is creating mechanisms for annotation so that different stakeholders need not repeat verification. Annotations should also help other developers compare, share, and leverage previously written code. Best practices and community standards are components of the annotation challenge. Badges could exist for code and libraries as well as developers.

Code integration vulnerabilities. In the previously mentioned doorlock example, we noted how secure devices can be used in combination in a way that creates vulnerabilities. Similarly, individually verified code can be combined in a manner that creates emergent failures. For example, an early IoT light switch was coded using a library where the incorrect use of dependencies resulted in overheating and potential risk of fire. This occurred because expectations of library interactions were incorrect. Defining developer expectations of code in a highly variable context is an open question, and the IoT devices' physicality makes this more difficult.

Role of platforms in promoting secure coding practices. Platforms can play a larger role in developer support, feedback, or management. A platform's ability to offer baseline security (e.g., AWS, Azure, iOS) can help raise the bar and encourage developers to adopt certain secure coding practices. Coordination of a few platforms may be a more effective way forward than traditional regulation or best practices; proactive coordination could improve overall security and prevent harm. Further, increased cooperation between academia and industry could also address code flaws.

Shared knowledge of failures. Currently, there is no culture of sharing cybersecurity failures and near misses. In the emerging IoT space, new companies are likely to repeat near misses and mistakes of existing companies in IoT or in other domains. Cross-industry coordination is a role for academia and public-sector leadership; however, finding the right scale and right people is difficult. One step forward would be to curate case studies of security failures and near misses [13]. The goal in disclosing such problems would be to build a body of measurement-based, observed empirical cases in order to avoid future failures. Much can be learned from traditional curating, library practices, and reporting requirements in physical domains. Another dimension where IoT can learn from the large body of work in cyber-physical systems is in modeling the physical effects of devices, which could facilitate better testing environment for IoT systems.

Usability of developer tools. Discussions of usability are often limited to end users, but developers are users, too. Usability cannot be considered separately from the quality of code, just as cryptography cannot be considered separately from usability if either is to be correct. Usability from a code and engineering perspective requires comprehensible annotation to help developers identify security and privacy properties that matter in their own context. A coordinated effort is needed to make it easier to develop systems without the current common, even chronic flaws. Ubiquitous education of developers is a widely supported proposal. However, coding is a global industry spanning industrial development teams working on cryptographic libraries to kids learning to code on tablets. Every developer will not have security expertise, and be no such expectation should be made. To correctly use and integrate a cryptographic or security library in their code, developers should not need to understand cryptography, just like car drivers need not understand how cars work under the hood. Developers learn by example, and better examples are needed. Many developers seek answers to security questions from online guidance, much of which is flawed and insecure [14]. Developer-centered design is a subset of the larger domain of user-centered design and can be informed by broader usability literature [7], [15].

VI. Summary

Reaching a secure state of Internet of Things requires proactively addressing several challenges. This paper offers a path forward, and there is reason for optimism given the foundations that exist for each challenge. We highlighted four core IoT research challenges. First, agile cryptography is critical for IoT devices and can be encouraged by developing a taxonomy of devices and their minimum cryptography needs. Manufacturers and developers could use this taxonomy to choose appropriate cryptographic algorithms. Second, the usability of IoT devices is critical; lacking a usable workflow, end users will not be able to use IoT devices to their full potential. Third, devices will fail or get compromised, so reliable and transparent recovery or mitigation mechanisms are necessary. Finally, developers need usable and appropriate tools to create secure and readable code. Ideally, libraries and tools could even make it difficult for them to create insecure code. None of these challenges is insurmountable, but each requires collaboration and coordination across the IoT ecosystem.

Acknowledgements

We thank our anonymous reviewers for their valuable feedback. We also thank Ivan Evtimov and Jacob Abbott for their helpful feedback on drafts. This research was supported in part by the National Science Foundation under CNS-1565252, CNS-1565375, CNS-1814518, Cisco Research Support, and the Comcast Innovation Fund. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of any other entity. All contributions by Steven Myers were as a faculty member at Indiana University and before February 2018.

References

[1] A. Acquisti, L. Brandimarte, and G. Loewenstein, “Privacy and human behavior in the age of information,” *Science*, vol. 347, no. 6221, pp. 509–514, Jan. 2015. DOI 10.1126/science.aaa1465

[2] N. Heninger, Z. Durumeric, E. Wustrow, and J. A. Halderman, “Mining your Ps and Qs: Detection of widespread weak keys in network devices,” in *Proceedings of the USENIX Security Symposium (USENIX Security)*, 2012, p. 16. Available online: https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/heninger

[3] R. G. M. Landon Curt Noll and S. Sisodiya, “Method for seeding a pseudo-random number generator with a cryptographic hash of a digitization of a chaotic system,” US Patent US5732138A, 1996. Available online: https://patents.google.com/patent/US5732138A/en

[4] M. S. Turan, K. A. McKay, C. Calik, D. H. Chang, and L. E. Bassham, “Status report on the first round of the NIST lightweight cryptography standardization process,” NIST Interagency/Internal Report (NISTIR), Oct. 2019. DOI 10.6028/NIST.IR.8268

[5] L. F. Cranor, “A framework for reasoning about the human in the loop,” in *Proceedings of the Conference on Usability, Psychology, and Security (UPSec)*, 2008, pp. 1:1–1:15. Available online: http://dl.acm.org/citation.cfm?id=1387649.1387650

[6] A. Adams and M. A. Sasse, “Users are not the enemy,” *Communications of the ACM*, vol. 42, no. 12, pp. 40–46, Dec. 1999. DOI 10.1145/322796.322806

[7] L. J. Camp, “Design for trust,” *Trust, Reputation And Security: Theories And Practice*, no. 627610, Dec. 2004. Available online: https://papers.ssrn.com/abstract=627610

[8] ——, “Mental models of privacy and security,” *IEEE Technology and Society Magazine*, vol. 28, no. 3, pp. 37–46, 2009. DOI 10.1109/MTS.2009.934142

[9] T. Denning, T. Kohno, and H. M. Levy, “Computer security and the modern home,” *Communications of the ACM*, vol. 56, no. 1, pp. 94–103, Jan. 2013. DOI 10.1145/2398356.2398377

[10] A. K. Simpson, F. Roesner, and T. Kohno, “Securing vulnerable home IoT devices with an in-hub security manager,” in *Proceedings of IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, 2017. DOI 10.1109/PERCOMW.2017.7917622

[11] N. Leveson, *Engineering a safer world: Systems thinking applied to safety*. MIT press, 2011.

[12] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, “Android permissions demystified,” in *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*. ACM, 2011, pp. 627–638. DOI 10.1145/2046707.2046779

[13] J. Bair, S. M. Bellovin, A. Manley, B. E. Reid, and A. Shostack, “That was close! Reward reporting of cybersecurity ‘near misses’,” University of Colorado Law Legal Studies Research Paper No. 17-28, Dec. 2017, forthcoming in Colorado Technology Law Journal 16.2. Available online: https://papers.ssrn.com/abstract=3081216

[14] Y. Acar, M. Backes, S. Fahl, D. Kim, M. L. Mazurek, and C. Stransky, “You get where you’re looking for: The impact of information sources on code security,” in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2016, pp. 289–305. DOI 10.1109/SP.2016.25

[15] M. Green and M. Smith, “Developers are not the enemy!: The need for usable security APIs,” *IEEE Security & Privacy*, vol. 14, no. 5, pp. 40–46, Sept 2016. DOI 10.1109/MSP.2016.111

Appendix

Jean Camp is a professor in the Luddy School of Informatics, Computing, and Engineering at Indiana University. She received a PhD in engineering and public policy from Carnegie Mellon University. Her research focuses on the intersection of human and technical trust, leveraging economic models and human-centered design to create safe, secure systems. Contact her at ljcamp@indiana.edu.

Ryan Henry is an assistant professor in the Department of Computer Science at the University of Calgary. His focus is on the systems challenges of applied cryptography, with an emphasis on using cryptography to build secure systems that protect the privacy of their users. Contact him at ryan.henry@ucalgary.ca.

Tadayoshi Kohno is a professor in the Paul G. Allen School of Computer Science & Engineering at the University of Washington. He received a PhD in computer science from the University of California San Diego. His research focuses on computer security broadly defined. Contact him at yoshi@cs.washington.edu.

Shrirang Mare is a postdoctoral research associate with a joint appointment in the School of Computer Science & Engineering at the University of Washington and in the School of Informatics, Computing, and Engineering at Indiana University. He received a PhD in computer science from Dartmouth College. His research focuses on usable security and privacy for emerging technologies. Contact him at shri@cs.washington.edu.

Steven Myers received a PhD in computer science from the University of Toronto. He then served in the departments of Informatics and Computer science in the School of Informatics, Computing, and Engineering at Indiana University as an associate professor. His research interests are in all areas of cryptography, and computer and systems security with a specific interest in foundational and applied cryptography, phishing and new heteromorphic attacks. Contact him at steveamyers@gmail.com.

Shwetak Patel is a professor in the Paul G. Allen School of Computer Science & Engineering at the University of Washington. He received his Ph.D. in computer science from Georgia Tech. His research is in the areas of Human-Computer Interaction, Ubiquitous Computing, and Sensor-

Enabled Embedded Systems, with a particular emphasis on the application of computing to health and sustainability. Contact him at shwetak@cs.washington.edu.

Joshua Streiff is a program manager for the Internet of Things House and the Security and Privacy in Informatics, Computing, and Engineering Center at Indiana University. He is also a program manager for a DoD CySP grant for developing secondary educational tools and outreach workshops for cybersecurity awareness. His areas of interest include security education, social contracts and concepts of privacy, and consumer Internet of Things products. Contact him at jostre@iu.edu.