Security in the Internet of Things: A Challenge of Scale

Patrick Schaumont
Bradley Department of Electrical and Computer Engineering
Virginia Tech
Blacksburg, USA
{schaum}@vt.edu

***Abstract*—Technological scaling has offered a windfall of benefits to electronics design. Increased transistor density has offered an exponential increase in computing capabilities over time, but without a corresponding increase in system cost. Information security has its own success story with scaling. Cryptographic algorithms become exponentially harder to break through a mere linear increase in encryption complexity or in key-length.**

In the Internet of Things, scaling is as much a security liability as it is an advantage. These security liabilities are new, poorly understood and poorly regulated. Some examples include the following: privacy of IoT data in the cloud; the safety consequences of poor information security in cyber-physical systems; the liabilities of long-lifetime devices that use outdated or poorly tested information security; the performance-limited information security in devices that run on the outskirts of the IoT using nothing but harvested energy.

In this contribution we consider the security landscape for IoT. We consider the technological consequences of securely extending the Internet into the physical world of things. We identify current limitations, ongoing research efforts, and open challenges for the design community.

***Index Terms*—Internet of Things; Lightweight Cryptography; Scaling Effects**

I. Introduction

Our connected lifestyle and our reliance on highly cyberized environments leads to the Internet of Things: a collection of connected devices and the cloud services to support them. The scale of the Internet of Things is enormous. CISCO estimates that by 2020, up to 50 billion devices will be connected to the Internet, or 6.6 devices for every person on earth [1]. We consider the security consequences of connecting electronics on this enormous scale. Among the many perspectives one can take on this problem, we will emphasize a technical perspective with emphasis on integrated and embedded electronics. One should recognize that there are equally important social, economical, and regulatory perspectives in IoT [2], [3].

A. An Example: Technology Chain of eHealth

Figure 1 illustrates a typical scenario of an application domain that makes part of the *Internet of Things* paradigm: the use of electronics to improve health care. A patient is monitored and controlled through a combination of implanted and wearable devices configured in a body-area network. These devices monitor and relay vital parameters such as blood glucose level and heart rate in real-time. The devices may also, through external control, take life-saving actions such as delivering an electric shock in response to heart fibrillation [4]. eHealth applications extend well beyond the body area network. Through a local gateway near the user, user data measurements are forwarded to various other service providers for monitoring, analysis or storage. In this scenario, the user literally becomes the endpoint of an information network.

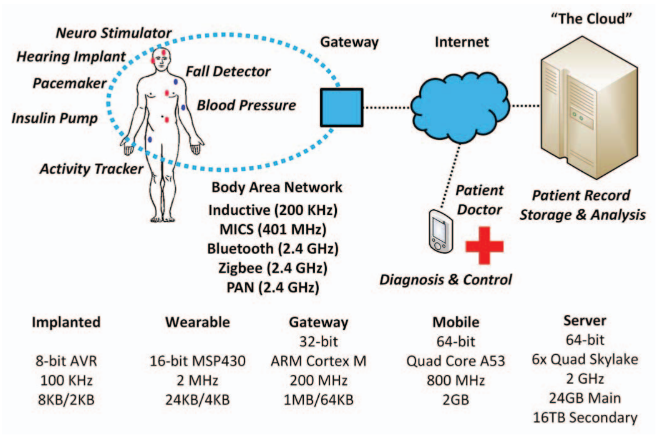

Figure 1: Platforms for the eHealth IoT scenario

This information ecosystem runs on a broad collection of computer architectures with a wide range of capabilities, from a tiny microcontroller in an implant to a high-end multi-core processor on a cloud server. This enormous variation in processing capabilities is reflected in the variation of processing speeds and memory footprints over orders of magnitude. The processing configuration at each point in this chain reflects the balance between the demand for computational capacity versus the energy efficiency of the architecture that can deliver it. In each case, of course, cost is minimized. Near the user, data processing speeds are low, and devices need to be especially good at being inactive. In the cloud, the data from thousands of users are aggregated, and processors are optimized for sharing and batch processing.

While scaling greatly affects the underlying architectures, its impact on the security requirements for data is much lower. A digitized heartbeat, for example, remains a digitized heartbeat no matter where it is stored in this information network. If the data confidentiality of a medical measure-

Table I: Security objectives for IWMD [5]

Security	**Safety**	**Privacy**
Data Confidentiality	Device Settings	Device Existence
Access Authorization	Device Access	Device Type Tracking
Data Authentication	Device Update	Device PHY Tracking
Data Integrity	Device Availability	Patient Data
Data Availability		

ment is a concern when it is transmitted wirelessly from a wearable device, then this confidentiality also applies when that measurement is stored in the cloud. If there is a privacy policy that requires encrypted storage of patient records in the cloud, then the mobile apps that handle such private user data have to apply the same care. Indeed, the expectations with respect to information security remain uniform throughout the IoT. This is a challenge, because it implies that information security algorithms (encryption for confidentiality, signatures for authentication, etc) cannot be scaled as easily as the architectures.

B. Security Objectives in the eHealth Scenario

Rushanan *et. al.* make an extensive review of the security concerns for implants and wearable medical devices (IWMD), and they distinguish (information) security, safety and privacy as summarized in Table I [5].

- *Information security* is needed because of the sensitive nature of medical data, which does not allow for errors or information leaks. Devices need to apply data encryption and/or authenticated encryption to support confidentiality and data integrity. They need to vouch for the origin of data through electronic signatures, and they need to authorize the access to external users. To accomplish this, a medical device needs a complete stack of information security algorithms.
- *Safety* concerns relate to actions taken on the patient through the device. The internal state of a medical device, its software version and its parameter settings need to be fully trusted. This must be enforced though a proper trusted computing base.
- *Privacy* concerns relate to the need for patient privacy once medical data is digitized and stored on the device, or traveling around the Internet. Privacy is achieved through a combination of policy, information security and physical measures.

To implement IWMD devices, these security objectives must be met using a limited set of resources, such that each threat is addressed through a proper security technique.

II. Security Challenges for embedded IoT

In this section, we review the technical consequences of a secure IoT. We emphasize the case of the embedded device - a small micro-controller that executes a constrained cyclic executive or a small operating system. Such a device has limited computational capabilities and limited storage capabilities. Once it is connected as an end-point on the Internet, it brings requirements with respect to security, safety and privacy. Among the many concerns, we highlight the risks of a large-scale computing infrastructure, the risks of new attack models and the challenge of powering IoT devices. A review of IoT security challenges from a production and manufacturing perspective may be found in [6].

A. The Risks of Large Numbers

The massive amount of embedded devices that make up the *things* in the Internet brings two immediate consequences. First, the effect of a security flaw in a device or a software component on a device is dramatically amplified when the device is deployed in large numbers. Second, the tight cost constraints on connected devices implies that even simple security precautions may be considered expensive.

The first issue implies that large numbers of identical connected devices increase risk. An example is the recent DDOS network attack (Oct 2016) on several major websites, which was coordinated by exploiting a weak login in the firmware of connected DVR machines and cameras [7]. In addition, even if a flaw is present in just a subset of connected devices, it may still be identified through network scanning, such as for example when looking for outdated SSH/TLS libraries [8], [9] or weak cryptographic keys [10]. Furthermore, embedded firmware images can be scanned for embedded vulnerabilities [11], or listed in a database of vulnerabilities [12]. An adversary can therefore study a device in depth before the attack, knowing that any small weakness found can be replicated on a large number of identical devices in the field.

Deploying a large quantity of connected devices also means that each single device will be constructed under tight cost constraints, with the bare minimum of features and security defenses. Francillon explains that this creates a tension in the design process, between cutting costs in the short term (as is it hard to justify the additional cost of security over functional features) and stability in the long term (as sooner or later, security flaws will bite back and trigger major product recalls) [13].

B. Attacks on All Fronts

The adversary may have a wide range of objectives (such as changing the firmware, controlling the software execution, or exfiltrating confidential data or keys). The possible actions of the adversary are classified in an *attacker model*, the set of assumptions that one makes regarding the capabilities of the adversary. Insight into the attacker model is needed to define the proper defenses in hardware and software. We refer to the discussion by Piessens *et. al.* for an in-depth discussion on the software-oriented attacker models [14].

Figure 2 demonstrates a typical IoT embedded system along with three different attacker models. The embedded system includes hardware and I/O, and a software stack consisting of an embedded operating system along with one or more tasks. We assume that one of these tasks is a *secure task* which has assets that deserve protection.

- The *input/output attacker* is the best understood, and most common adversary. This attacker is able to control and

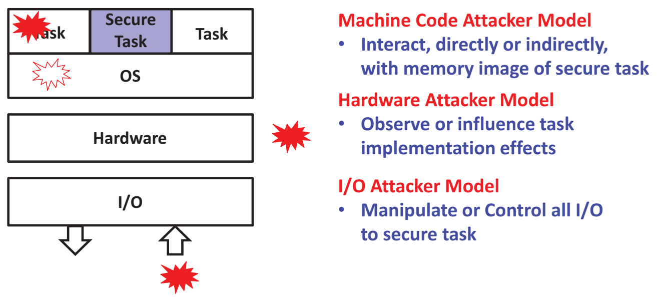
Task
Secure Task
Task
OS
Hardware
I/O
Machine Code Attacker Model
• Interact, directly or indirectly, with memory image of secure task
Hardware Attacker Model
• Observe or influence task implementation effects
I/O Attacker Model
• Manipulate or Control all I/O to secure task

Figure 2: Three attacker models.

Table II: Energy-harvester Powered two-pass Authentication using MSP430 10MHz and CC2500 RF. MAC is a shared-secret authentication protocol; ECDSA and WNITZ are public-key protocols [24]

		Harvester (1 Hour)			Fixed Source	
		RF	Piezo	Solar	Supercap 3V/0.22F	Batt AAA
$\mu W \rightarrow$		0.1	10	1000		
$mJ(/hr) \rightarrow$		3.6	36	3600	660	3500M
Protocol	mJ	**Authentications/Hour**			**Authentications**	
MAC	0.15	2.4	240	24K	4400	23.3M
ECDSA	14	0.03	3	257	47	250K
WNITZ	20.15	0.02	2	179	33	174K

modify all input to the embedded system, thereby triggering buffer overflow and memory corruption [15]. Many attack techniques use this vulnerability, such as return-oriented-programming [16], overwriting code pointers, or executing data as code. These threats are well understood, and they are addressed through safe-memory programming techniques.

- The *memory attacker* goes beyond the input/output boundary. This adversary can operate within the same memory space as the trusted embedded software. For example, a code update on an embedded system may insert malicious code, or the operating system on the embedded system may install a malicious driver. Protecting against the memory attacker requires the ability of the embedded system to *isolate* a portion of the memory state. This is harder on small embedded systems, which have not been designed with this capability in mind. Several recent proposals have demonstrated feasible solutions, such as TrustZone [17], TrustLite [18], and SANCUS [19]. These provide logical isolation, but not physical isolation. This leaves an opportunity for the hardware attacker.
- The *hardware attacker* is an adversary who is able to observe and/or alter the physical execution of software and hardware in the embedded system. Side-channel analysis is a passive attack that targets information leakage [20], while fault-injection analysis is an active attack that targets the availability of the embedded system [21]. In contrast to the earlier mentioned attack models, a *comprehensive* countermeasure against the hardware attacker remains an elusive goal. There are many point solutions, although none of these solutions has claimed to be composable. For example, a well known technique against fault injection is time-redundant execution (executing code multiple times) [22], but such time-redundancy increases side-channel leakage through power consumption, and thus it makes a side-channel analysis attack more likely.

C. *The Energy-starved IoT*

In a discussion on IoT security issues, we cannot ignore the ultimate enabler of this ubiquitous connectivity: energy. The Internet already consumes massive quantities of energy. In 2007, Greenpeace estimated that the cloud and its connected devices used more electricity than India, Germany, Canada or France [23]. The Climate Group estimated that connected devices will become the dominant electricity consumer in the Internet, taking up 57% of the Internet's electricity by 2020. This applies, or course, to the entire internet infrastructure *combined* (servers, access network, devices). Most of the interesting applications in IoT are in places with constrained energy provisioning. Implanted devices, such as in the eHealth scenario covered earlier, aim for multi-year operation out of necessity. Other applications, such as for structural health monitoring, may be fully integrated and inaccessible, and have to be fully self-powered for years.

We consider the energy needs for a wireless device that performs a challenge-response authentication protocol. In this case, a wireless node computes a response to a challenge, and the response is computed using a Message Authentication Code (MAC) or a signature. The energy cost is determined by the wireless transmission cost of receiving a challenge from a verifier, cryptographically computing the response, and transmitting it back to the verifier. We rely on measurements made on an MSP430-based wireless node with a 2.4GHz frontend [24]. A MAC-based authentication can be completed in less than millijoule, while public-key based authentication can be completed in 20 millijoules or less. This energy has to be balanced against the source of energy – a battery or an energy harvester. Since the energy harvested through an energy-harvesting transducer is limited, the IoT device has to reduce its duty cycle by periodically entering a sleep mode [25]. Table II shows the number of authentications that can be obtained on a full battery charge, or else from a typical energy harvester using an hour of harvesting time. The numbers indicate that, while energy harvesting is unable to deliver the power for continuous operation, it can provide energy for a reasonable duty cycle provided that we can efficiently store the harvested energy until it is needed.

Herein lies a catch, as energy storage of harvested sources on supercapacitors is not very efficient. Supercapacitors are used in place of rechargeable (lithium-ion) batteries because they offer higher power density and peak-power delivery, but with a low energy density and a high rate of self-discharge [26]. Therefore, the harvested energy must be consumed quickly.

Fig. 3 demonstrates the consequences. Fig. 3a demonstrates

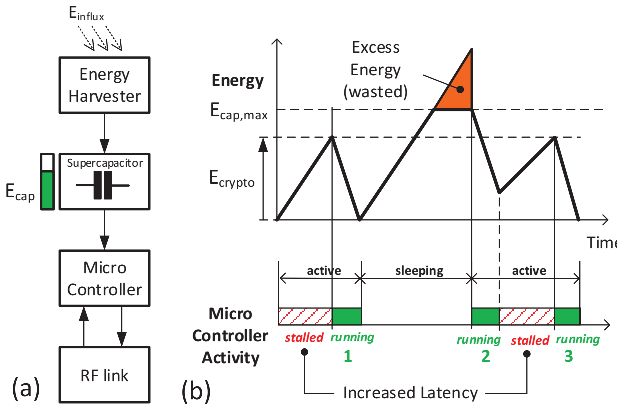

Figure 3: *(a) Layout of an energy-harvested IoT node, (b) Operation of crypto.*

the architecture of a generic, energy-harvested IoT device. We assume a steady influx of energy, such that the harvester charges a supercapacitor slowly but continuously. When E_{crypto} Joules of energy is available, the device can complete a single iteration of the cryptographic protocol. Compared to the rate of energy influx, the value of E_{crypto} is rather large. When an iteration of the protocol is desired but the energy store is discharged, the device has no other choice but to *stall* until the required energy is harvested. Fig. 3b demonstrates three different iterations of this protocol, all of which show inefficiency. The first time, the protocol is stalled because less than E_{crypto} Joules is available on the supercapacitor. This delay will be experienced, at application level, as increased protocol latency. The second time, the protocol can proceed immediately because the supercapacitor is fully charged. However, because it reaches full charge well before the second iteration is initiated, some harvested energy goes to waste. The third time, the IoT device again needs to wait until E_{crypto} is available, despite having wasted harvested energy moments before. Although a larger supercapacitor could be used, this would also increase the IoT device cost and size. These observations can be made on the measurements in Table II as well. For example, the 0.22F supercap can be fully charged by a 1mW solar cell in just about 10 minutes. Anything beyond that is wasted. Furthermore, a piezo harvester delivers only sufficient energy for a single ECDSA authentication in a time-span of 20 minutes, assuming that there's no self-discharge and that the energy conversion is ideal (80 % conversion efficiency would be a more realistic estimate [27]).

We summarize the design challenges of a secure IoT as follows. As traditional computer networking extends into the embedded, physical space, it carries significant requirements with respect to security, safety and privacy. These expectations have to be met by micro-controllers and microsystems that face tight constraints in cost and energy budget, and that face an extensive set of threats. The next section highlights some of the recent advances into this dark tunnel.

III. Security Solutions for the IoT

We highlight three promising developments towards a secure IoT: latency-optimized design, energy-optimized design, and secure embedded computing architectures.

A. Latency-optimized Cryptography for the IoT

Traditional information security has emphasized the always-on, throughput-optimized, power-optimized computing paradigm. In the devices on the outskirts of the IoT, other concerns take hold. Devices are occasionally-on and then compute on a few data items. This has to complete as quickly as possible, as it allows the IoT devices to turn off again. This means that low *latency*, the ability to compute a single response quickly, is more important than high throughput, the ability to compute many reponses at a high rate. Some examples of security protocols that benefit from low-latency design are authentication using challenge-response, the encryption of low-rate data streams, and inline memory-integrity techniques in secure computer architectures. Recent research results in this direction include low-latency ciphers such as PRINCE [28], and single-cycle encryption designs [29]. However, much work remains to be done. Public key-algorithms still remain notoriously complex. A highly-optimized signature algorithm for 80-bit secure (160-bit prime-field) ECC requires close to ten million cycles on an 8-bit design [30]. Another example of a 128-bit-security `curve25519` ECC curve on an 8-bit, 16-bit and 32-bit microcontroller requires 14 million, 5.5 million and 3.5 million clock cycles respectively [31].

B. Energy-optimized Cryptography for the IoT

A second promising development is the design of energy-optimized rather than power-optimized algorithms. While power dissipation reflects the rate of work, energy reflects the quantity of work done. The battery-powered operation of an IoT device is clearly energy-limited, and the energy footprint of a security protocol directly translates to the lifetime of the device. The energy-harvesting operation of an IoT device is energy-limited as well because the device only operates in multiples of energy quanta. A recent proposal for an energy-optimized block cipher is Midori [32]. The design choices of Midori include a minimum number of rounds in a round-parallel implementation with components with low switching (4-bit Sboxes instead of 8-bit Sboxes). Midori demonstrates original thinking into how low-energy hardware design can be used to steer algorithmic design choices. As a result, Midori is currently the most energy-efficient hardware block cipher, when compared to other lightweight designs. However, as with low-latency security, much work remains to be done. There are currently no solutions of public-key ciphers specifically optimized for low energy.

C. Secure Embedded Computer Architectures

A third area that has demonstrated valuable results is secure embedded computer architectures. They deal with the Machine-code Attacker Model (Figure 2) at the level of the architecture. Examples include TrustLite [18], HAFIX [33]

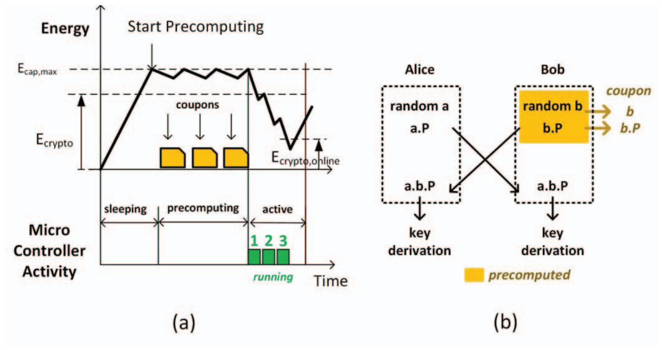

Figure 4: *(a) Pre-computed operations (b) Precomputed Diffie-Hellman.*

and SANCUS [19]. The purpose of these designs is to bring the features of cloud-level secure computing to embedded platforms. For example, they provide isolation (hardware-based access control for code and data), attestation (demonstration of the internal state of node), the ability to build a secure communication channel between the IoT application and a cloud server, and the ability for secure code update. While such secure embedded computer architectures require modifications to the micro-controller hardware and software toolchain, they demonstrate that the trusted computing concept of the Internet can also be supported in the IoT.

IV. Open Challenges

We conclude with some open challenges in IoT security, and remind that this paper has taken a specific viewpoint towards the support of security in embedded computing platforms. Among the many open problems, we highlight two open major challenges for the secure embedded computing field.

A. Public Key Cryptography for the IoT

Public-key protocols are essential for key-exchange. Without these algorithms, the management of keys becomes a logistical challenge, and we need to recur to pre-distributed keys. In fact, one could define the boundary of the secure Internet, in practical terms, at the level of the connected platform that is able to efficiently support a public-key exchange. Secure connectivity builds on open-source, community-supported efforts. A recent scan for SSH servers, for example, found that 95% of about 16M internet-connected servers supported `OpenSSH` or `dropbear` while less than 5% used another package [8]. Using open-source software increases trust and reliability in the long term. While micro-controller designs can use hardware-accelerators for public-key primitives, this creates non-portable code and the secure Internet protocol stack needs to be hand-ported to every embedded accelerator. This is only useful for specialized, large-volume applications such as smart-cards. Hence, we need energy-aware, constraint-aware mechanisms in standard public-key packages.

Figure 4a demonstrates an alternate route towards public-key crypto on energy-harvested IoT, which relies on doing computations on excess harvested energy. Public-key algorithms are especially suited for this because a significant portion of computations is related to key material preparation. The precomputing process is activated when the supercap is fully charged while there is no online activity, and therefore it operates completely transparent to the IoT device user. The intermediate results of precomputing, called *coupons*, are stored securely in non-volatile memory. These coupons will then reduce the online latency of the key exchange. Furthermore, the IoT can now use a smaller supercapacitor and, because excess energy is no longer wasted, the IoT device will do more work under a limited energy influx.

Figure 4b shows the application of this concept to Diffie-Hellman key-exchange, used during initialization of a secure link between Alice and Bob. They select random numbers a and b, perform an ECC point multiplication $a.P$ and $b.P$, exchange the result and repeat the point multiplication to derive a common secret $a.b.P$. From Figure 3, we observe that half effort of an ECDH key exchange can be precomputed, which will cut the latency and energy-cost for online operations in half. Other pre-computed ECC schemes are for example for key pair generation [34] and signatures [35].

B. Breaking the Stovepipe Model

The second major challenge for a secure IoT is dealing with the *stovepipe model* of secure design. This means that in current practice, a single security issue is handled through a single defense, a point solution. The problem with this strategy is that it remains unknown whether defenses are composable, that is, if they can be combined with other defenses. The attacker model in Figure 2 defines three different adversaries. In the current state of research, it's hard, if not impossible, to tell if a defense against one attacker will be an advantage or a limitation to another attacker. It's easy to show examples of non-composable solutions.

- One class of side-channel countermeasures, *hiding*, aims at reducing variations in the behavior of the software (using constant-time design). This means that a given sensitive instruction will appear always at exactly the same time, which in turn simplifies the injection of faults. Hence, a side-channel countermeasure may make successful fault attacks more probable. Note that this example is the dual of the example discussed in II.B.
- The secure embedded computing architectures illustrated earlier [18], [19], [33] simply ignore the hardware attacker, even though they rely on a hardcoded key for their security claims.
- In a recent Trojan attack, the privilege bit in a processor, which enforces isolation between software processes, was tampered at the hardware level, such that privilege escalation could be triggered by an adversarial software process [36].

The problem with these examples is not that we fail in designing perfect security; the problem is that we cannot realistically tell if a defense will maintain any level of security once we change the attacker model.

V. Conclusions

In this paper we pointed out the significant challenges ahead in designing a secure IoT, from an architectural perspective, from a security protocol perspective, and from an energy perspective. The complexity of this problem points to a *codesign-everything* methodology: we need to gain insight into the adversary at multiple levels of abstraction. Security engineers, computer architecture engineers, and cryptographic engineers need to join forces to tackle the challenge of secure IoT.

References

[1] D. Evans, "The Internet of Things: How the Next Evolution of the Internet Is Changing Everything," White Paper, April 2011, https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf.

[2] S. L. Keoh, S. S. Kumar, and H. Tschofenig, "Securing the Internet of Things: A Standardization Perspective," *IEEE Internet of Things Journal*, vol. 1, no. 3, pp. 265–275, June 2014.

[3] O. Vermesan and P. Fries, "Internet of Things: Position Paper on Standardization for IoT technologies," IERC, January 2015, http://www.internet-of-things-research.eu/pdf/IERC_Position_Paper_IoT_Standardization_Final.pdf.

[4] W. Burleson, S. S. Clark, B. Ransford, and K. Fu, "Design Challenges for Secure Implantable Medical Devices," in *The 49th Annual Design Automation Conference 2012, DAC '12, San Francisco, CA, USA, June 3-7, 2012*, 2012, pp. 12–17.

[5] M. Rushanan, A. D. Rubin, D. F. Kune, and C. M. Swanson, "SoK: Security and Privacy in Implantable Medical Devices and Body Area Networks," in *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, 2014, pp. 524–539.

[6] N. Cam-Winget, A. Sadeghi, and Y. Jin, "Invited - Can IoT be Secured: Emerging Challenges in Connecting the Unconnected," in *Proceedings of the 53rd Annual Design Automation Conference, DAC 2016, Austin, TX, USA, June 5-9, 2016*, 2016, pp. 122:1–122:6.

[7] B. Krebs, "Hacked Cameras, DVRs Powered Todays Massive Internet Outage," Krebs on Security Blog, October 2016, https://krebsonsecurity.com/2016/10/hacked-cameras-dvrs-powered-todays-massive-internet-outage/.

[8] M. R. Albrecht, J. P. Degabriele, T. B. Hansen, and K. G. Paterson, "A Surfeit of SSH Cipher Suites," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, 2016, pp. 1480–1491.

[9] R. Holz, J. Amann, O. Mehani, M. A. Kâafar, and M. Wachs, "TLS in the Wild: An Internet-wide Analysis of TLS-based Protocols for Electronic Communication," in *23nd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*, 2016.

[10] 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 21th USENIX Security Symposium, Bellevue, WA, USA, August 8-10, 2012*, 2012, pp. 205–220.

[11] A. Costin, J. Zaddach, A. Francillon, and D. Balzarotti, "A large-scale analysis of the security of embedded firmwares," in *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014.*, 2014, pp. 95–110.

[12] Y. Jin, "IoT/CPS Security Vulnerabilities Database," University of Central Florida, 2016 November, http://www.hardwaresecurity.org/iot/database/.

[13] A. Francillon, "Trust, but Verify: Why and How to Establish Trust in Embedded Devices," in *2016 Design, Automation & Test in Europe Conference & Exhibition, DATE 2016, Dresden, Germany, March 14-18, 2016*, 2016, pp. 1178–1182.

[14] F. Piessens and I. Verbauwhede, "Software security: Vulnerabilities and Countermeasures for Two Attacker Models," in *2016 Design, Automation & Test in Europe Conference & Exhibition, DATE 2016, Dresden, Germany, March 14-18, 2016*, 2016, pp. 990–999.

[15] L. Szekeres, M. Payer, T. Wei, and R. Sekar, "Eternal War in Memory," *IEEE Security & Privacy*, vol. 12, no. 3, pp. 45–53, 2014.

[16] H. Shacham, "The Geometry of Innocent Flesh on the Bone: Return-into-libc Without Function Calls (on the X86)," in *Proc. of the 2007 ACM Conf. on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*, 2007, pp. 552–561.

[17] P. Wilson, A. Frey, T. Mihm, D. Kershaw, and T. Alves, "Implementing Embedded Security on Dual-Virtual-CPU Systems," *IEEE Design & Test of Computers*, vol. 24, no. 6, pp. 582–591, 2007.

[18] P. Koeberl, S. Schulz, A. Sadeghi, and V. Varadharajan, "TrustLite: a Security Architecture for Tiny Embedded Devices," in *Ninth Eurosys Conference 2014, EuroSys 2014, Amsterdam, The Netherlands, April 13-16, 2014*, 2014, pp. 10:1–10:14.

[19] J. Noorman, P. Agten, W. Daniels, R. Strackx, A. V. Herrewege, C. Huygens, B. Preneel, I. Verbauwhede, and F. Piessens, "Sancus: Low-cost Trustworthy Extensible Networked Devices with a Zero-software Trusted Computing Base," in *Proc. of the 22th USENIX Security Symposium, Washington, DC, USA, August 14-16, 2013*, 2013, pp. 479–494.

[20] R. Spreitzer, V. Moonsamy, T. Korak, and S. Mangard, "SoK: Systematic Classification of Side-Channel Attacks on Mobile Devices," arXiv:1611.03748v1, 2016, https://arxiv.org/abs/1611.03748.

[21] M. Joye and M. Tunstall, Eds., *Fault Analysis in Cryptography*, ser. Information Security and Cryptography. Springer, 2012.

[22] K. Wu and R. Karri, "Fault Secure Datapath Synthesis using Hybrid Time and Hardware Redundancy," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 23, no. 10, pp. 1476–1485, 2004.

[23] Greenpeace International, "Make IT green: Cloud computing and its contribution to climage change," White Paper, March 2010, http://www.greenpeace.org/international/en/publications/reports/make-it-green-cloud-computing/.

[24] P. Schaumont, B. Yuce, K. Pabbuleti, and D. Mane, "Secure Authentication with Energy-harvesting: A Multi-dimensional Balancing Act ," *Sustainable Computing: Informatics and Systems*, pp. –, 2015.

[25] M. T. Penella-Lopez and M. Gasulla-Forner, "Introduction," in *Powering Autonomous Sensors*. Springer Netherlands, 2011, pp. 1–7.

[26] A. Nasiri, S. A. Zabalawi, and G. Mandic, "Indoor Power Harvesting Using Photovoltaic Cells for Low-Power Applications," *IEEE Trans. Industrial Electronics*, vol. 56, no. 11, pp. 4502–4509, Nov 2009.

[27] S. Ray, Y. Jin, and A. Raychowdhury, "The Changing Computing Paradigm With Internet of Things: A Tutorial Introduction," *IEEE Design & Test*, vol. 33, no. 2, pp. 76–96, 2016.

[28] J. Borghoff, A. Canteaut, T. Güneysu, E. B. Kavun, M. Knezevic, L. R. Knudsen, G. Leander, V. Nikov, C. Paar, C. Rechberger, P. Rombouts, S. S. Thomsen, and T. Yalcin, "PRINCE: A Low-latency Block Cipher for Pervasive Computing Applications," in *ASIACRYPT'12*. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 208–225.

[29] P. Maene and I. Verbauwhede, "Single-Cycle Implementations of Block Ciphers," Cryptology ePrint Archive, Report 2015/658, 2015, http://eprint.iacr.org/2015/658.

[30] Z. Liu, E. Wenger, and J. Großschädl, *MoTE-ECC: Energy-Scalable Elliptic Curve Cryptography for WirelessSensorNetworks*. Cham: Springer International Publishing, 2014, pp. 361–379.

[31] M. Düll, B. Haase, G. Hinterwälder, M. Hutter, C. Paar, A. H. Sánchez, and P. Schwabe, "High-speed curve25519 on 8-bit, 16-bit, and 32-bit microcontrollers," *Des. Codes Cryptography*, vol. 77, no. 2-3, pp. 493–514, 2015.

[32] S. Banik, A. Bogdanov, T. Isobe, K. Shibutani, H. Hiwatari, T. Akishita, and F. Regazzoni, *Midori: A Block Cipher for Low Energy*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 411–436.

[33] L. Davi, M. Hanreich, D. Paul, A. Sadeghi, P. Koeberl, D. Sullivan, O. Arias, and Y. Jin, "HAFIX: Hardware-assisted Flow Integrity Extension," in *Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, June 7-11, 2015*, 2015, pp. 74:1–74:6.

[34] W. Whyte, A. Weimerskirch, V. Kumar, and T. Hehn, "A Security Credential Management System for V2V Communications," in *2013 IEEE Vehicular Networking Conference, Boston, MA, USA, December 16-18, 2013*, 2013, pp. 1–8.

[35] G. Bianchi, A. Capossele, C. Petrioli, and D. Spenza, "AGREE: exploiting energy harvesting to support data-centric access control in wsns," *Ad Hoc Networks*, vol. 11, no. 8, pp. 2625–2636, 2013.

[36] K. Yang, M. Hicks, Q. Dong, T. M. Austin, and D. Sylvester, "A2: analog malicious hardware," in *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, 2016, pp. 18–37.