Exploring Potential 6LoWPAN Traffic Side Channels

Yan Yan
Department of Computer Science
University of Bristol
y.yan@bristol.ac.uk

Elisabeth Oswald
Department of Computer Science
University of Bristol
elisabeth.oswald@bristol.ac.uk

Theo Tryfonas
Department of Computer Science
University of Bristol
theo.tryfonas@bristol.ac.uk

Abstract

The Internet of Things (IoT) has become a reality: small connected devices feature in everyday objects including childrens' toys, TVs, fridges, heating control units, etc. Supply chains feature sensors throughout, and significant investments go into researching next-generation healthcare, where sensors monitor wellbeing. A future in which sensors and other (small) devices interact to create sophisticated applications seems just around the corner. All of these applications have a fundamental need for security and privacy and thus cryptography is deployed as part of an attempt to secure them. In this paper we explore a particular type of flaw, namely side channel information, on the protocol level that can exist despite the use of cryptography. Our research investigates the potential for utilising packet length and timing information (both are easily obtained) to extract interesting information from a system. We find that using these side channels we can distinguish between devices, different programs running on the same device including which sensor is accessed. We also find it is possible to distinguish between different types of ICMP messages despite the use of encryption. Based on our findings, we provide a set of recommendations to efficiently mitigate these side channels in the IoT context.

Categories and Subject Descriptors

H.5.m [**Information interfaces and presentation**]: Miscellaneous

General Terms

IoT Privacy and Security

Keywords

LoWPAN; Side Channel Attacks; Traffic Analysis

1 Introduction

The expression 'Internet of Things' (IoT) can refer to a multitude of objects and protocols, which share that they have been purposefully designed for resource constraint environments. Whereas the typical TCP/IP network stack produces considerable overhead to achieve quality of service for applications that are based on it, the nature of many IoT 'things' is such that a full implementation of it would not be practical. Often 'things' are sensor, which are devices that have to function on little resources (most importantly power). Thus a whole host of new networking protocols have been developed over the years to cater for such resource constrained devices: 6LoWPAN is the 'tiny' version of IPv6, UDP tends to be used instead of TCP/IP, DTLS can be used for end-to-end security or one can directly invoke 802.15.4 security which is part of 6LoWPAN, and finally CoAP(s) is the replacement for HTTP(s). Thus there are two options (802.15.4, and DTLS) to secure communications between the 'things' and a server/gateway.

Implementing cryptography correctly and securely has proven to be a massive challenge as evidenced by the multitude of implementation attacks over the years. Triggered off by research that showed how to utilise additional information via timing and power side channels [14], many different flavours of side channel attacks were discovered over the last decade. Many attacks use phyiscal information (such as low level execution timings or power consumption) to recover secret keys, but many other attacks use protocol level information (such as packet lengths, types of packets or protocol messages) to recover information about plaintexts, devices in the network, or the network itself. There exists a considerable body of work in the context of conventional, i.e. HTTPs over TCP/IP network, but the applicability of (some) of these attacks in the context of a typical IoT protocol stack is lacking. This is the gap that we would like to address with this work.

This paper is structured as follows: after reviewing some relevant attack paths for HTTPs over TCP/IP in the following subsection, we briefly explain our experimental network in Section 2. We discuss the impact of packet length leakage in Section 3, followed by an analysis of the response time leakage in Section 4. We summarise our work in Section 5.

1.1 Related Work

Traffic Analysis is well studied in the context of encrypted Internet traffic, especially for web applications based on

HTTPs and TCP/IP. The landmark study by Chen et al. [1] discussed different side channel attacks against web applications and [28] studied the practicability of an attack specifically targeted Google and Bing search boxes. Later work by Mather and Oswald [21] proposed the use of Mutual Information to pinpoint the potential leakage points in web traffic. For non-HTTPs applications, the papers [4], [36] and [2] described attacks against encrypted text, voice and video traffic respectively. Machine learning is widely used to analyse the traffic, and behaviours of different classifiers are studied by [11] and [7]. Based on all these published works we can conclude that two features, the packet length and response time, are the most exploited ones among all attacks. Different countermeasures were studied by [37], [20] and [8].

Reflecting on IoT applications, we stipulate that most of these attacks may still be applicable, as we intend to demonstrate in this paper. Considering the future vision that IoT devices could be indeed connected to the Internet with even more sensitive data flowing over different networks, the task of designing secure IoT applications becomes increasingly challenging.

With regard to the aspect of protocol design, the recent paper [24] summarised some known flaws of 6LoWPAN, including its susceptibility to the Fragmentation Attack [13], Sinkhole Attack [17], Hello Flood Attack [31], Wormhole Attack [12] and Blackhole Attack [34]. In addition, [27] reported certain problematic designs in 802.15.4 security [10]. However we do not discuss further these particular design flaws as they touch on a different aspect of the security issues in 6LoWPAN compared to what we address in this paper.

2 Our Experimental Network

Our experimental network is constructed using two different devices. These are a TelosB and a CC2538. The TelosB is a low cost sensor powered by an MSP430 with an AES co-processor. It represents typical low-end devices. The CC2538 is the high end device powered by an ARM Cortex-M3 with multiple cryptographic processors including AES, RSA, SHA-2 and ECC, suggesting that it is suitable to develop secure applications.

Both devices are supported by the Contiki OS. We adopted the default settings of the Contiki OS, except for enabling 802.15.4 security [10] for some experiments. Note that the Contiki MAC [5] is chosen by default over TSCH [32]. For Layer 4 [9] and above protocols, we went with the widely accepted combination of CoAP [30], and DTLS [26](optional) over UDP [25][1]. Table 1 summarises our choice of protocol stack.

Table 1. Protocol stack for our experiments(* is optinal)

Physical	802.15.4
Link	
Network	6LoWPAN
Transmission	UDP
	DTLS*
Application	CoAP / CoAPs*

[1] CoAPs is equivalent to CoAP over DTLS.

2.0.1 802.15.4 and DTLS

In our setting, there are two standards available for packet encryption, namely 802.15.4 security [10] and DTLS [26]. 802.15.4 security is provided by the noncoresec [16] API, which implements 802.15.4 authenticated encryption with AES-128 CCM* [6] using a hard-coded key shared by the whole 6LoWPAN network. We chose tinyDTLS as library for the DTLS protocols, because it provides a minimum DTLS implementation that supports two ciphersuites which are TLS_PSK_WITH_AES_128_CCM_8 [22] and TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 [22] respectively. Evidently, they both utilise AES-128 CCM* as the packet encryption method.

3 Exploiting Packet Length Information

As our brief survey of traffic analysis via exploiting packet lengths showed in Section 1.1, the packet length has proven to be a powerful side channel for the classical Internet protocols. It is worth noting that this side channel is 'noisy' in the classical Internet setting: websites or web applications in this setting typically feature advertisements, which impact on packet lengths; TCP/IP allows to fragment packets and then reassembles them, a feature which is not presented in UDP. Thus, due to the nature of UDP exploiting the packet length as side channel should be easier in the IoT setting.

Clearly then, any web application style implementations involving an IoT device will thus be extremely vulnerable to attacks such as [1]. In the absence of this scenario for state-of-the art IoT applications, it still sends a cautionary warning to developers: binary responses (e.g. 'yes' vs. 'no', or 'on' vs. 'off') must always be coded via a binary variable and not via strings because these will have different lengths, which are directly visible via the packet length.

In the remainder of this section we will highlight further problems that arise if packet lengths leak information.

3.1 Distinguishing ICMP Messages

The Internet Control Message Protocol(ICMP) [3] performs the management tasks in a network, such as link establishment and routing information exchange. As explained before we utilise the open source system Contiki, which supports a (sub)set of the ICMP standard (we list the supported ICMP messages in Table 2). Many ICMP messages are ideal for network discovery and exploration, although the purpose of ICMP is to send error messages to the source IP address if standard IP packets fail to be transmitted correctly.

Generally, ICMP messages can be protected by either using the secure ICMP messages as described in [3], or relying on the lower layer encryption provided by 802.15.4. Contiki OS does not have the former implemented, hence 802.15.4 security is the only option currently. We simulated a 6LoWPAN network with 802.15.4 security enabled (with strongest encryption and authentication). We configured the nodes to also generate random UDP packets. Despite the fact that all ICMP messages were encrypted, our experiments show that several ICMP messages can be identified by their packet size and MAC destination. Table 2 summarises the packet features. The value x denotes the size of user defined data in bytes.

Table 2. Metadata of Contiki Supported Packets

	Packet Size (bytes)	MAC Destination
DIS	85	broadcast
DIO	118/123	broadcast/unicast
DAO	97	unicast
NS	87	broadcast/unicast
NA	87	unicast
PING	$101+x$	unicast
UDP Multicast	$85+x$	broadcast
UDP Unicast	$107+x$	unicast

Among the unicast packets, PING and UDP have at least 101 and 108 bytes[2]. Therefore, DAO can be uniquely identified as the shorter unicast packet of 97 bytes. For the same reason NA and unicast NS can also be distinguished from other packets by filtering packets of 87 bytes. Considering that NA is sent as a response to NS according to the protocol, one can always identify the first being NS and second being NA.

Similarly, unicast DIO can be identified as the 123 bytes packet followed by DIS, where the later has a unique 85 byte size. However, there is a potential of false positive induced by PING or UDP packets with user defined data crafted to have the same packet length[3]. PING could be recognised by its pair-wised appearance, as the response would have nearly the same meta data as the original request, except the exchanged source and destination. For broadcast packets, DIS can be easily identified by its unique 85 bytes packet size. Others like broadcast NS can be identified by the followed characteristic NA response; and packets of 118 bytes those are periodically broadcasted are likely to be DIOs.

In summary, among all the packets, DAO, NA, NS, DIS can be identified with certainty. DIO and PING cannot be certainly identified but they both have significant characters. Notice that the above contained all ICMPv6 messages supported by Contiki; therefore UDP packets can be reversely filtered, although in some cases they get mixed with DIO and PING.

Although leakage in ICMP messages does not directly lead to any breach of application data, it would still be harmful by providing the adversary with information about the state of the network, including which nodes recently joined etc. Specifically DAO is always sent from a child to its parent and can be uniquely identified; therefore together with MAC addresses the adversary may exploit it to draw a graph that shows the parental relations in the network. In addition, these information can also be exploited by attacks as in [19].

3.2 Distinguishing Different Devices

In the classical Internet world, ICMP has been well known for its use for OS fingerprinting [33]. In the case of the IoT, this could be possible as well (as different OS support different subsets of ICMP), however an additional attack vector exists. This is because different IoT devices have different hardware limitations or drivers. We noticed that our TelosB [23] discards all packets exceeding 127 bytes[4] whereas our CC2538 handles packets even up to 160 bytes. Therefore an adversary can immediately rule out TelosB whenever a packet larger than 127 bytes processed by the target.

4 Exploiting Response Time Information

The response time is another major feature that has been previously exploited in Internet traffic analysis attacks. Like in the case of exploiting packet lengths, we would expect that the same attacks (as in the classical Internet setting) can be applied to 6LoWPAN traffic. Indeed, like in the previous section, we would expect that they will work even better because the accuracy of timing measurements can be greatly improved for 6LoWPAN traffic: this is because there are fewer noise sources in the traffic, the devices are physically close to each other and uses RF to communicate, the adversary can remove the RTT noises by measure the packets on the server side, and the performance of the constrained devices is low and hence gives a better resolution of the execution time.

4.1 Distinguishing Different Sensors

The first application of timing analysis that we describe is to distinguish between different sensors that are accessed on a device. For this purpose we set up an experiment on a CC2538, which has three on-board sensors: Vdd, temperature, and an Ambient Light Sensor (short ALS). We access these via CoAP [30], which is a protocol designed for constrained devices that provides an universal interface for accessing resources. CoAPs is the secure version which stands for CoAP with DTLS.

Due to the different physical characteristics of the sensors, there could be a variance of time that is required for reading the measurements. We investigated whether such variances could be observed through the packet response latency. If this was the case, then an adversary could learn the nature/purpose of sensors on a network by observing their response time.

We thus set up an experiment on CC2538, using all three sensors from "cc2538-demo". We used CoAP from the "erest-example" in the Contiki OS source code, as there is no CoAPs implementation available. Although DTLS processing would definitely have an impact on the response latency, we argue that such impact would be independent to the sensors being accessed; hence similar result can be equally expected for CoAPs. We carefully controlled other factors, including URIs, data representation and code flow, to be uniform for all three sensors in order to guarantee a controlled environment.

Table 3 summarises the result. It shows that ALS takes about 2ms longer and hence can be easily distinguished. Vdd and temperature have much more strongly overlapping distributions, and thus are more difficult to distinguish. Nevertheless these results confirm our hypothesis: different sensors have different latencies and these leak through the response time. An adversary who is interested in finding out information about devices on a network might thus be able to match

[2] PING can be sent without user defined data and UDP packets requires at least 1 byte.

[3] 22 bytes for PING and 16 bytes for UDP.

[4] MTU specified by 802.15.4 standard.

Table 3. CoAP Response Latency for Sensor Readings on CC2538

	Average (ms)	Range(ms)
Vdd	9.622	[9.388, 10.318]
Temperature	9.835	[9.525, 10.318]
ALS	11.651	[11.338, 12.031]

Table 4. PING Response Latency

	CC2538	TelosB
Average(ms)	9.56	17.03
Range(ms)	[9.16, 10.06]	[16.49, 17.68]

the (known) behaviour of 'interesting' sensors to what they observe on the network. We remark that this could be useful even in the setting where the sensors transmit their data unencrypted: after all they might return only some reading without a unit of measurement; thus seeing their return data might not as such reveal their nature.

4.2 Distinguishing Different Devices

As we observed before, different devices have different underlying hardware and thus different computational power. This implies that there could be the potential that different devices take different amounts of time to process the same message. Because ICMP messages are standardised, they are particularly suitable for this purpose. Among the different ICMP messages, PING is especially ideal for two reasons:

1. It is mandatory in the ICMP standard.
2. It only swaps the source and destination address of the packet; thus minimises different code path in protocol processing.

Table 4 shows the PING response latency on CC2538 and TelosB. The result confirms that these devices can be distinguished by PING response latency.

4.3 Distinguishing Programs

We remarked before that the functionality of a sensor is potentially valuable information. For instance some sensors might be predominantly passive, e.g. they might read the temperature and report it back periodically, whereas some sensors might control something upon receiving commands. Thus knowing the functionality enables an adversary to make (more) sense of the observed traffic in the network. This could be done if a 'fingerprint' could be produced for different programs. From an adversary's perspective a positive result would imply that they could 'fingerprint' products which are on the market and thus use this information to infer what program is running on a target device.

To illustrate why this might work, we now look at Figure 1. It illustrates two sensors receiving the same service request. In our example, at the time of receiving the request, Sensor Node 1 was idle and hence responded immediately, whilst Sensor Node 2 postponed the request for reading a sensor. Clearly, the response time on Sensor Node 2 would appear longer than that of Sensor Node 1.

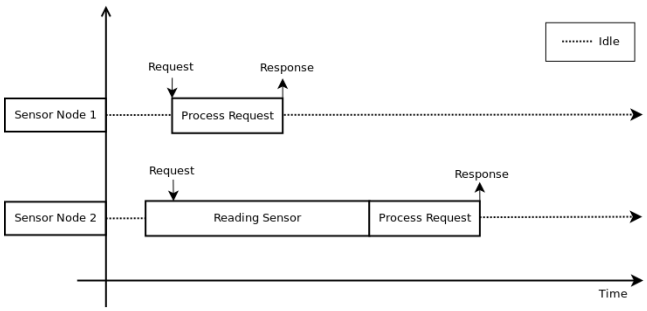


Figure 1. Variations in Response Time

In real life, most sensors are programmed in a loop; therefore the same code fragments are repeated through the life time of a sensor. Each code fragment takes different time to execute and hence the response times vary. This behaviour could be statistically analysed and the resulting distribution could be stored as a 'fingerprint' .

For this fingerprinting scenario, we must assume the adversary has the pre-knowledge of potential programs and can fingerprint them (or that they have access to a database that contains this information). To identify an unknown program running on target sensor, the adversary collects a new fingerprint and then matches it to available fingerprints. Clearly, to effectively launch the attack, the adversary needs to be able to send the request to a targeted sensor (requests with short predictable processing time are preferable as they induce less noise).

In practice, the request can be instantiated by several messages defined in the sensor network protocols. PING is exceptionally ideal as it is mandatory in the ICMP standard [18] and has only negligible computation. Other options but not excluded are Heartbeat in DTLS [29], Reset in CoAP [30], etc.

Figure 2 shows an example of PING packets captured on an CC2538 running Contiki OS. The response time, which refers to PING Response Interval, PRI, is defined to be the time between a PING response and its last paired request.

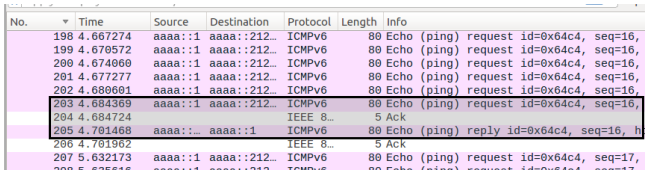
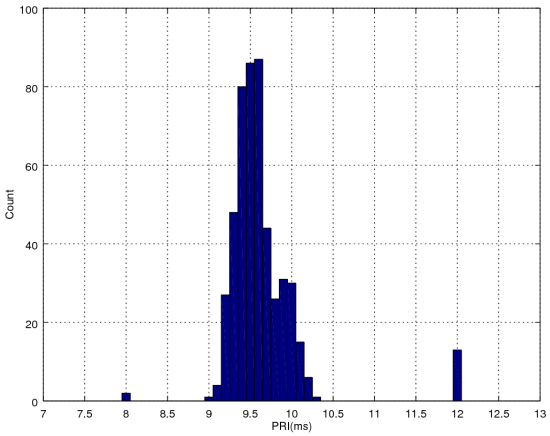


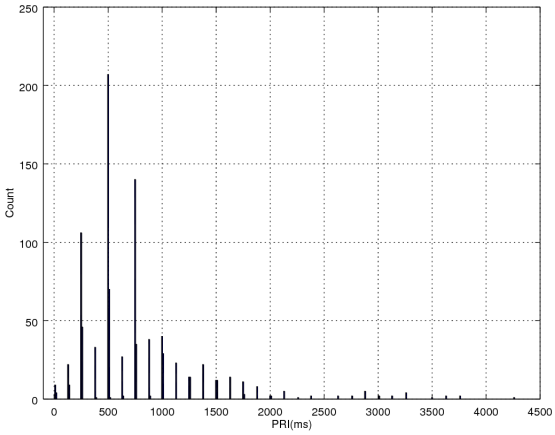
Figure 2. Example PRI

Figure 3a shows the histogram of PRIs collected on the "helloworld" example from Contiki OS. Values $\geq$12ms are collected at 12ms. The result shows that most PRIs are clustered around 9.5ms which consists with our result in Table 4. The majority, roughly ranged [9.0, 10.3]ms, corresponds to the usual response time as depicted by Sensor Node 1 in Figure 1.

We further plotted the upper outliers, mostly ranged [12, 2000]ms, in Figure 3b. Unfortunately we do not have a solu-



(a) PRIs of helloworld



(b) PRIs outliers of helloworld

Figure 3. helloworld PRIs

tion to investigate the exact cause of such delay, as we were unable to control the code execution that requires environmental interaction within a timing critical context. Nevertheless, we suppose these outliers correspond to the extended response time as depicted by Sensor Node 2 in Figure 1. The distribution described by Figure 3b is the fingerprint of the "helloworld" example.

The result in Figure 3 shows a clear gap between the usual PRIs and extended PRIs. In fact other applications we experimented also showed the same property. This implies that an adversary can easily draw a threshold by observing the whole PRI distribution and then filter out the fingerprint. In our experiments the threshold is set to 12ms but any other values within the gap would also work.

We collected the fingerprints for three programs taken from the Contiki OS examples:

broadcast This program periodically broadcasts a constant message.

powertrace This program records the power consumption and broadcasts a constant message.

Sensorpayload This program is based on the "er-rest-example" embedded together with sensor accesses taken from "cc2538-demo". It captures a real case scenario where three different sensors, namely Temperature, Vdd and ALS, are being accessed through CoAP.

Specifically for "Sensorpayload" we collected fingerprints for 8 different scenarios where different sensors are being accessed. For each program we independently collected 2 fingerprints for comparison.

During the experiments we realised that most of the fingerprints do not adhere to common distributions; therefore we used a non parametric test, the Kolmogorov-Smirnov Distance [15], as our test statistic. This is a well understood statistic with previous uses in side channel analysis [35].

By adapting our distinguisher to utilise the minimum KS distance, we were able to identify 13 out of 20 fingerprints successfully. The 'overlapping' fingerprints are mainly due to the "Sensorpayload" program, which access different sensors, but otherwise has identical program code. Thus we did expect that the different instantiations of it would lead to very similar fingerprints.

5 Conclusion

In this paper we explore, for the first time, the use of packet lengths and response times, which are protocol level side channels, as means to recover information about IoT 'things'. We do this experimentally, which we base on two extremely popular devices running on a popular open source OS, with a typical stack of protocols. Whilst we do not cover a wide range of devices, the fact that two of the most popular devices show the characteristics that we hypothesise, gives credibility to our results. Our results show that it is possible (in principle) to recover information about a device and its function (i.e. the hardware and the software that runs on it) via inspecting encrypted traffic that it produces. We also point out that ICMP messages can be distinguished from each other despite the use of encryption.

Although 6LoWPAN is a relatively experimental standard and most smart devices today are still based on WiFi, we reasonably argue that the same attacks could be mounted on these devices as well since WiFi packets contains all the same leakage. For instance, in IFTTT based applications, such as WeMo, such leakage may reveal the user specified "receipts" which results into a severe privacy and security issue.

In order to mitigate the leakage that is given by packet lengths, previous works recommend padding [7]. We echo this recommendation. Whilst padding to MTU is considered inefficient for the Internet, it is in fact highly appropriate for 6LoWPAN because:

- It completely hides the length of original plaintext.
- 6LoWPAN has only a low MTU of 127 bytes; therefore the overhead is acceptable.
- It induces negligible computational overhead.

With regard to the leaking information about the device or OS, we suggest strictly applying the standard MTU to eliminate the differences in drivers. Although there is a potential

of performance downgrade, it will also improve the compatibility among different devices.

In order to mitigate the leakage given by response times, the natural countermeasure is to write time-constant code, which is known to be notoriously difficult. But two approaches are available to a software developer:

- Randomly delay the response. This essentially adds noise to the measurements of the adversary.
- Use a threshold response time, i.e. a request is either responded at a predefined time or not responded at all.

Within the context of 6LoWPAN the second method is recommended as most 6LoWPAN application would tolerate missing packets and timer is available on most platforms. However, the threshold must be carefully chosen to preserve the functionality of the 6LoWPAN application.

6 Acknowledgements and Disclaimer

This work was in part supported by EPSRC via grant EP/N011635/1 (LADA). No research data was created for this paper.

7 References

[1] S. Chen, R. Wang, X. Wang, and K. Zhang. Side-channel leaks in web applications: A reality today, a challenge tomorrow. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 191–206. IEEE, 2010.

[2] R. B. Chris Wampler, A. Selcuk Uluagac. Information leakage in encrypted ip video traffic. ieee-globecom 2015, 2015.

[3] A. Conta, S. Deering, and M. Gupta. Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. RFC 4443 (Draft Standard), Mar. 2006. Updated by RFC 4884.

[4] S. Coull and K. Dyer. Privacy failures in encrypted messaging services: Apple imessage and beyond. *arXiv preprint arXiv:1403.1906*, 2014.

[5] A. Dunkels. The contikimac radio duty cycling protocol. *SICS Report*, 2011.

[6] M. J. Dworkin. Sp 800-38c. recommendation for block cipher modes of operation: The ccm mode for authentication and confidentiality. Technical report, National Institute of Standards & Technology, Gaithersburg, MD, United States, 2004.

[7] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton. Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, SP '12, pages 332–346, Washington, DC, USA, 2012. IEEE Computer Society.

[8] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton. Protocol misidentification made easy with format-transforming encryption. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 61–72. ACM, 2013.

[9] I. O. for Standardization ISO. ISO/IEC 7498-1Information technology - Open Systems Interconnection - Basic Reference Model: The Basic Model. Technical report, ISO, June 1994.

[10] I. . W. Group. IEEE Standard for Information Technology- Telecommunications and Information Exchange Between Systems- Local and Metropolitan Area Networks- Specific Requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs). Technical report, IEEE 802.15.4 Working Group, 2006.

[11] D. Herrmann, R. Wendolsky, and H. Federrath. Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier. In *Proceedings of the 2009 ACM workshop on Cloud computing security*, pages 31–42. ACM, 2009.

[12] Y.-C. Hu, A. Perrig, and D. B. Johnson. Wormhole attacks in wireless networks. *Selected Areas in Communications, IEEE Journal on*, 24(2):370–380, 2006.

[13] R. Hummen, J. Hiller, H. Wirtz, M. Henze, H. Shafagh, and K. Wehrle. 6lowpan fragmentation attacks and mitigation mechanisms. In *Proceedings of the Sixth ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec '13, pages 55–66, New York, NY, USA, 2013. ACM.

[14] P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *Advances in cryptologyCRYPTO99*, pages 789–789. Springer, 1999.

[15] A. N. Kolmogorov. *Sulla determinazione empirica di una legge di distribuzione*. na, 1933.

[16] K.-F. Krentz, H. Rafiee, and C. Meinel. 6lowpan security: Adding compromise resilience to the 802.15.4 security sublayer. In *Proceedings of the International Workshop on Adaptive Security*, ASPI '13, pages 1:1–1:10, New York, NY, USA, 2013. ACM.

[17] I. Krontiris, T. Giannetsos, and T. Dimitriou. Launching a sinkhole attack in wireless sensor networks; the intruder side. In *Networking and Communications, 2008. WIMOB'08. IEEE International Conference on Wireless and Mobile Computing,*, pages 526–531. IEEE, 2008.

[18] M. Kulkarni, A. Patel, and K. Leung. Mobile IPv4 Dynamic Home Agent (HA) Assignment. RFC 4433 (Proposed Standard), Mar. 2006.

[19] V. Kumar, G. Oikonomou, and T. Tryfonas. Traffic forensics for ipv6-based wireless sensor networks and the internet of things. In *Internet of Things (WF-IoT), 2016 IEEE 3rd World Forum on*, pages 633–638. IEEE, 2016.

[20] X. Luo, P. Zhou, E. W. Chan, W. Lee, R. K. Chang, and R. Perdisci. Httpos: Sealing information leaks with browser-side obfuscation of encrypted flows. In *NDSS*, 2011.

[21] L. Mather and E. Oswald. Pinpointing side-channel information leaks in web applications. *Journal of Cryptographic Engineering*, 2(3):161–177, 2012.

[22] D. McGrew and D. Bailey. AES-CCM Cipher Suites for Transport Layer Security (TLS). RFC 6655 (Proposed Standard), July 2012.

[23] Online: `http://www.willow.co.uk/html/telosb_mote_platform.php`.

[24] P. Pongle and G. Chavan. A survey: Attacks on rpl and 6lowpan in iot. In *Pervasive Computing (ICPC), 2015 International Conference on*, pages 1–6. IEEE, 2015.

[25] J. Postel. User Datagram Protocol. RFC 768 (INTERNET STANDARD), Aug. 1980.

[26] E. Rescorla and N. Modadugu. Datagram Transport Layer Security Version 1.2. RFC 6347 (Proposed Standard), Jan. 2012. Updated by RFCs 7507, 7905.

[27] N. Sastry and D. Wagner. Security considerations for ieee 802.15.4 networks. In *Proceedings of the 3rd ACM Workshop on Wireless Security*, WiSe '04, pages 32–42, New York, NY, USA, 2004. ACM.

[28] A. Schaub, E. Schneider, A. Hollender, V. Calasans, L. Jolie, R. Touillon, A. Heuser, S. Guilley, and O. Rioul. Attacking suggest boxes in web applications over https using side-channel stochastic algorithms. In *Risks and Security of Internet and Systems*, pages 116–130. Springer, 2014.

[29] R. Seggelmann, M. Tuexen, and M. Williams. Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension. RFC 6520 (Proposed Standard), Feb. 2012.

[30] Z. Shelby, K. Hartke, and C. Bormann. The Constrained Application Protocol (CoAP). RFC 7252 (Proposed Standard), June 2014.

[31] V. P. Singh, A. S. A. Ukey, and S. Jain. Signal strength based hello flood attack detection and prevention in wireless sensor networks. *International Journal of Computer Applications*, 62(15), 2013.

[32] P. Thubert, T. Watteyne, M. R. Palattella, X. Vilajosana, and Q. Wang. Ietf 6tsch: Combining ipv6 connectivity with industrial performance. In L. Barolli, I. You, F. Xhafa, F.-Y. Leu, and H.-C. Chen, editors, *IMIS*, pages 541–546. IEEE Computer Society, 2013.

[33] F. Veysset, O. Courtay, O. Heen, I. Team, et al. New tool and technique for remote operating system fingerprinting. *Intranode Software Technologies*, 4, 2002.

[34] M. Wazid, A. Katal, R. Singh Sachan, R. Goudar, and D. P. Singh. Detection and prevention mechanism for blackhole attack in wireless sensor network. In *Communications and Signal Processing (ICCSP), 2013 International Conference on*, pages 576–581. IEEE, 2013.

[35] C. Whitnall, E. Oswald, and L. Mather. An exploration of the kolmogorov-smirnov test as a competitor to mutual information analysis. In *International Conference on Smart Card Research and Advanced Applications*, pages 234–251. Springer, 2011.

[36] C. V. Wright, L. Ballard, S. E. Coull, F. Monrose, and G. M. Masson. Spot me if you can: Uncovering spoken phrases in encrypted

voip conversations. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 35–49. IEEE, 2008.

[37] C. V. Wright, S. E. Coull, and F. Monrose. Traffic morphing: An efficient defense against statistical traffic analysis. In *NDSS*, 2009.