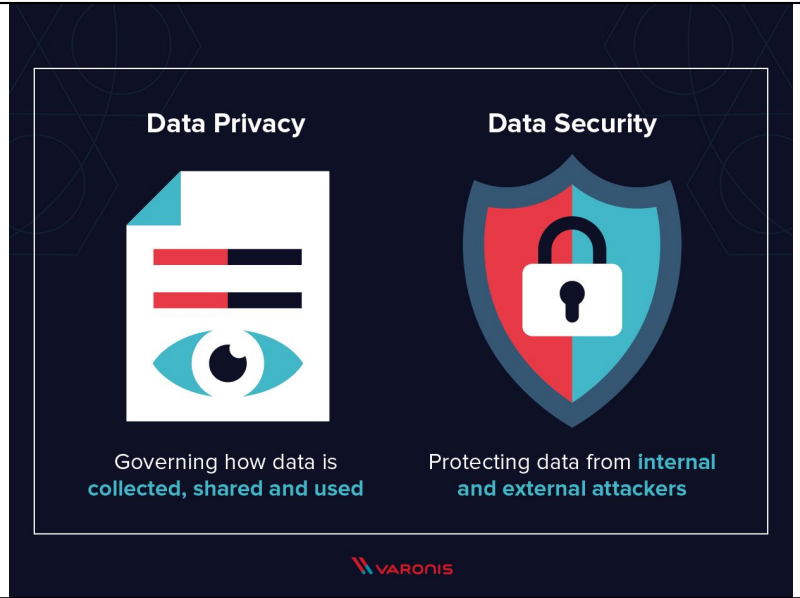
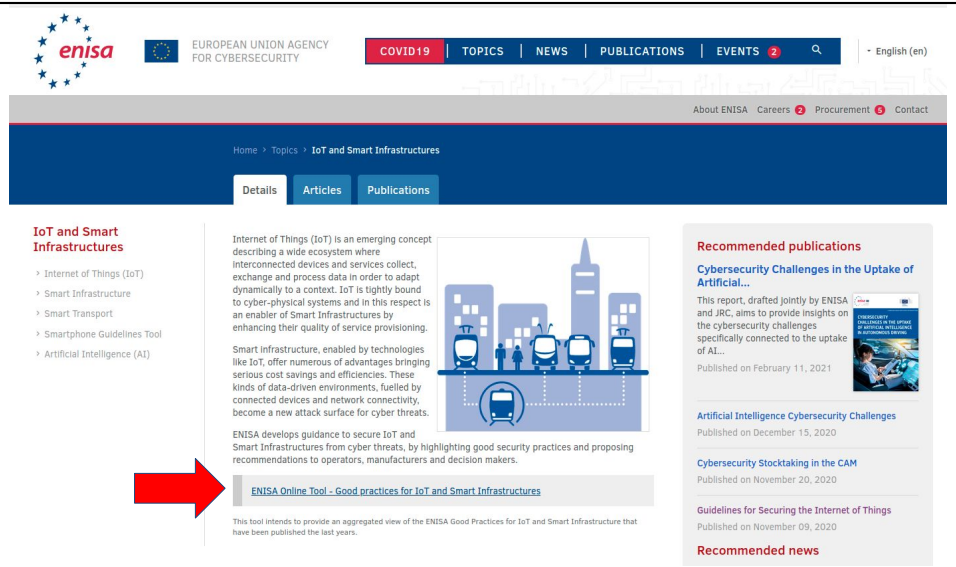
IoT Security and Privacy

Andrea Vitaletti

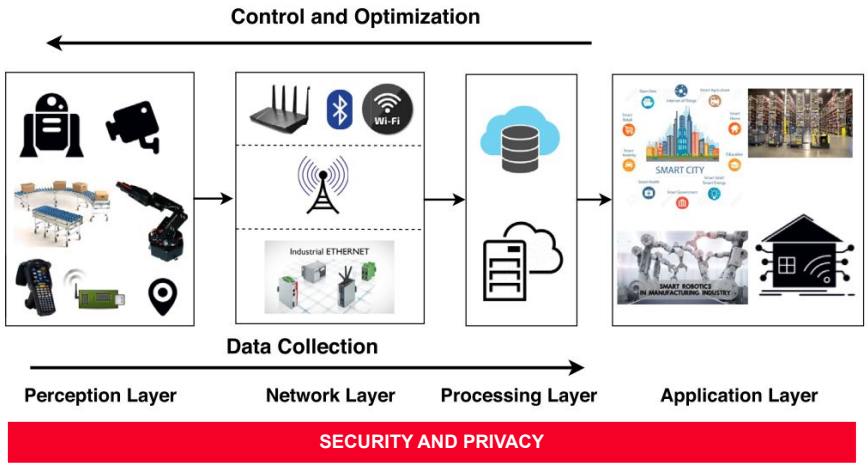
Source: www.cbronline.com





https://www.enisa.europa.eu/topics/iot-and-smart-infrastructures

https://www.sciencedirect.com/science/article/pii/S1084804519303418



20 Vulnerabilities in Samsung SmartThings Hub

- Smart locks controlled by the SmartThings Hub could be unlocked, allowing for physical access to the home.
- Cameras deployed within the home could be used to remotely monitor occupants.
- The motion detectors used by the home alarm system could be disabled.
- Smart plugs could be controlled to turn off or on different things that may be connected.
- Thermostats could be controlled by unauthorized attackers.
- Attackers could cause physical damage to appliances or other devices that may be connected to smart plugs deployed within the smart home.

LOOK HERE

https://blog.talosintelligence.com/2018/07/samsung-smartthings-vulns.html

Are Companies Ready?

Surveyed more than 5,000 enterprises around the world

- 85% of enterprises are in the process of or intend to deploy IoT devices.
- Yet a mere 10% of those surveyed feel confident that they could secure those devices against hackers.

source: AT&T's Cybersecurity Insights Report

Considering the most popular devices: TVs, webcams, thermostats, power outlets, sprinkler controllers, hubs for controlling multiple devices, door locks, home alarms,scales, and garage door openers

- 90% collected personal data
- 70% used unencrypted network services

source: HP Fortify 2014 IoT security report

An unprecedented amount of Data

The sheer amount of data that IoT devices can generate is staggering.
Fewer than 10,000 households can generate 150 million discrete data points every day.
This creates more entry points for hackers and leaves sensitive information vulnerable.

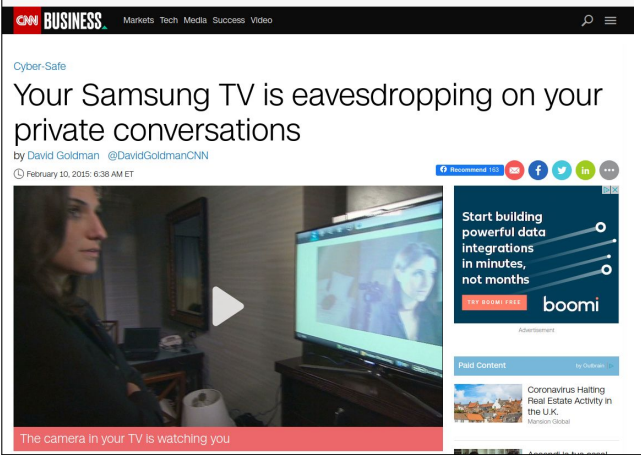
source: Federal Trade Commission report entitled "Internet of Things: Privacy & Security in a Connected World"

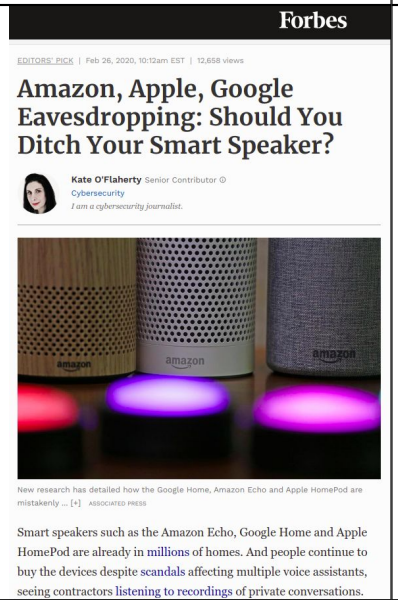
Unwanted Public Profile

You've undoubtedly agreed to terms of service at some point, but have you ever actually read through an entire document?
An insurance company might gather information from you about your driving habits through a connected car when calculating your insurance rate.
The same could occur for health or life insurance thanks to fitness trackers.

source: Federal Trade Commission report entitled "Internet of Things: Privacy & Security in a Connected World"

Eavesdropping

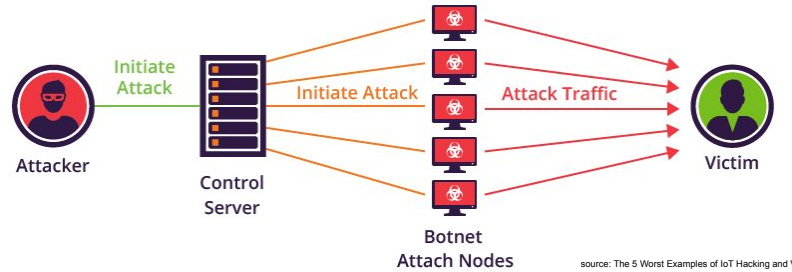




MIRAI (DDOS)

(aka Dyn Attack)

Unlike other botnets, which are typically made up of computers, the Mirai botnet is largely made up of so-called (IoT) devices such as digital cameras and DVR players.



source: The 5 Worst Examples of IoT Hacking and Vulnerabilities in Recorded History

According to PC Magazine, here are four straightforward IoT security lessons from MIRAI

- “Devices that cannot have their software, passwords, or firmware updated should never be implemented.
- Changing the default username and password should be mandatory for the installation of any device on the Internet.
- Passwords for IoT devices should be unique per device, especially when they are connected to the Internet.
- Always patch IoT devices with the latest software and firmware updates to mitigate vulnerabilities.”

source: The 5 Worst Examples of IoT Hacking and Vulnerabilities in Recorded History

FDA U.S. FOOD & DRUG ADMINISTRATION

Firmware Update to Address Cybersecurity Vulnerabilities Identified in Abbott's (formerly St. Jude Medical's) Implantable Cardiac Pacemakers: FDA Safety Communication

Date Issued

August 29, 2017

“The FDA confirmed that St. Jude Medical’s implantable cardiac devices have vulnerabilities that could allow a hacker to access a device. Once in, they could deplete the battery or administer incorrect pacing or shocks, the FDA said. The devices, like pacemakers and defibrillators, are used to monitor and control patients’ heart functions and prevent heart attacks.”

source: The 5 Worst Examples of IoT Hacking and Vulnerabilities in Recorded History

TRENDnet Webcam Hack

“Faulty software that let anyone who obtained a camera’s IP address look through it — and sometimes listen as well.

Further, from at least April 2010 [until about January 2012], TRENDnet transmitted user login credentials in clear, readable text over the Internet, and its mobile apps for the cameras stored consumers' login information in clear, readable text on their mobile devices, the FTC said.

It is basic security practice to secure IP addresses against hacking and to encrypt login credentials or at least password-protect them, and TRENDnet's failure to do so was surprising.”

source: The 5 Worst Examples of IoT Hacking and Vulnerabilities in Recorded History

The Jeep Hack

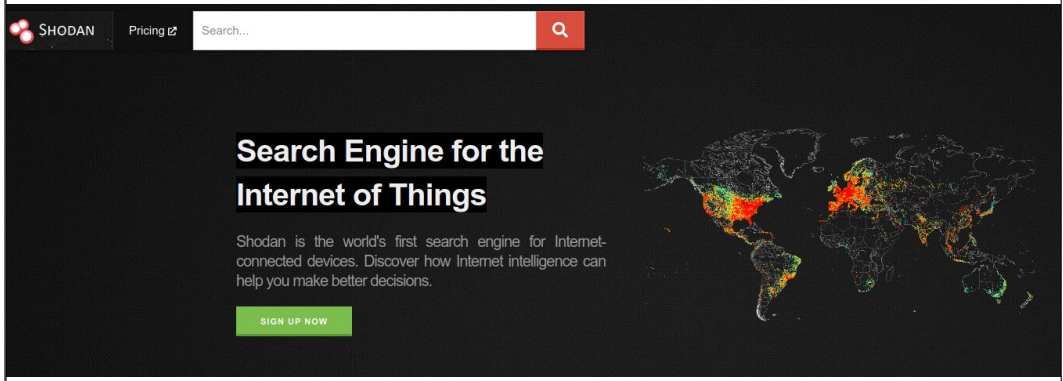
In July [2015], a team of researchers was able to take total control of a Jeep SUV using the vehicle’s CAN bus. By exploiting a firmware update vulnerability, they hijacked the vehicle over the Sprint cellular network and discovered they could make it speed up, slow down and even veer off the road. Its proof of concept for emerging Internet of Things (IoT) hacks: While companies often ignore the security of peripheral devices or networks, the consequences can be disastrous.”

source: The 5 Worst Examples of IoT Hacking and Vulnerabilities in Recorded History

Common Vulnerabilities and Exposures (CVE) collected by MITRE

https://cve.mitre.org/

https://www.shodan.io/



https://www.safetydetectives.com/blog/what-is-shodan-and-how-to-use-it-most-effectively/

OWASP Top 10 IoT Security Weaknesses

1. Weak, Guessable, or Hardcoded Passwords
2. Insecure Network Services
3. Insecure Ecosystem Interfaces
4. Lack of Secure Update Mechanism
5. Use of Insecure or Outdated Components
6. Insufficient Privacy Protection
7. Insecure Data Transfer and Storage
8. Lack of Device Management
9. Insecure Default Settings
10. Lack of Physical Hardening

Source: ***OWASP Internet of Things Project***

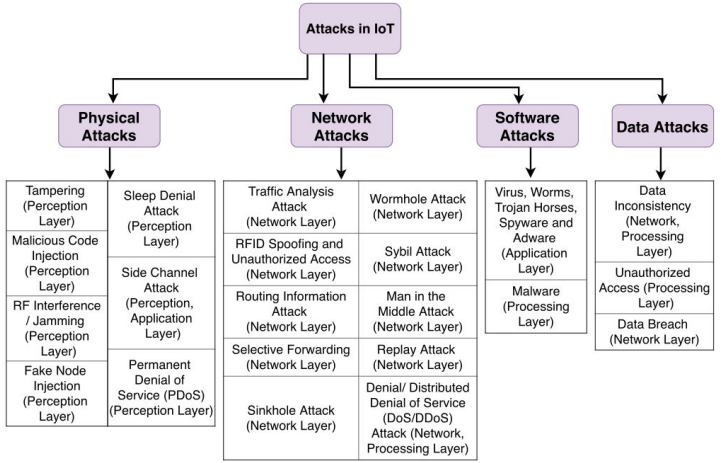


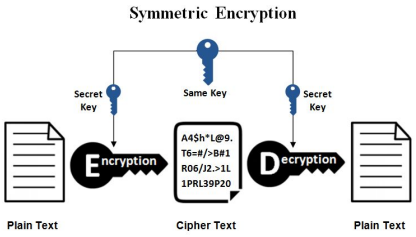
Fig. 2. Attacks in IoT.

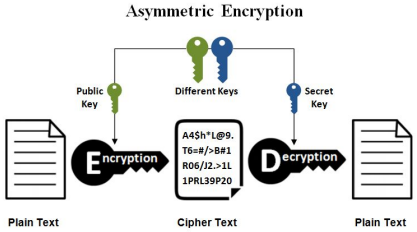
https://www.sciencedirect.com/science/article/pii/S1084804519303418

- ENCRYPTION → Confidentiality
- HASH → Integrity
- SIGNATURE → Authenticity + Non repudiation
- DTLS
- OTA REPROGRAMMING

Security Property	Meaning
Confidentiality	Information is only available to the people intended to use or see it.
Integrity	Information is changed only in appropriate ways by the people authorized to change it.
Availability	Apps and services are ready when needed and perform acceptably.
Authentication	A person's identity is determined before access is granted if anonymous people are not allowed.
Authorization	People are allowed or denied access to the app or app resources.
Nonrepudiation	A person cannot perform an action and then later deny performing the action.

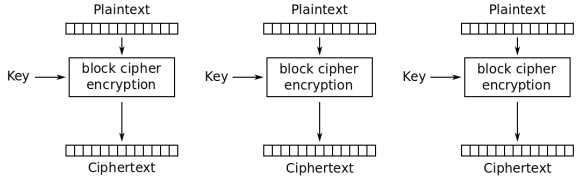
Source: Mike Gualtieri, Principal Analyst, Forrester Research



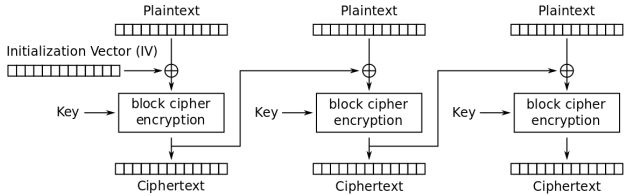


	Symmetric	Asymmetric
Complexity/Speed/Resources	+	-
Key Distribution	-	+

Block cipher

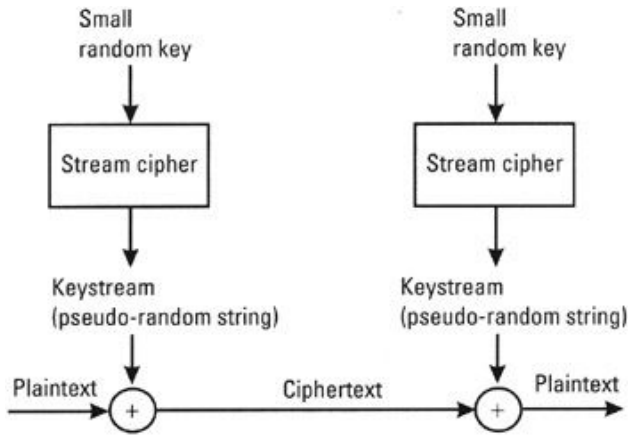


Electronic Codebook (ECB) mode encryption



Cipher Block Chaining (CBC) mode encryption

Stream cipher



[Docs] [txt|pdf] [draft-ietf-lwig...] [Tracker] [Diff1] [Diff2]

INFORMATIONAL

Internet Engineering Task Force (IETF)
Request for Comments: 7228
Category: Informational
ISSN: 2070-1721

C. Bormann
Universitaet Bremen TZI
M. Ersue
Nokia Solutions and Networks
A. Keranen
Ericsson

May 2014

Terminology for Constrained-Node Networks

Abstract

The Internet Protocol Suite is increasingly used on small devices with severe constraints on power, memory, and processing resources, creating constrained-node networks. This document provides a number of basic terms that have been useful in the standardization work for constrained-node networks.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7228.

CONSTRAINED NODES

- constraints on the maximum code complexity (ROM/Flash),
- constraints on the size of state and buffers (RAM),
- constraints on the amount of computation feasible in a period of time ("processing power"),
- constraints on the available power, and
- constraints on user interface and accessibility in deployment (ability to set keys, update software, etc.).

3. Classes of Constrained Devices

Despite the overwhelming variety of Internet-connected devices that can be envisioned, it may be worthwhile to have some succinct terminology for different classes of constrained devices. In this document, the class designations in Table 1 may be used as rough indications of device capabilities:

Name	data size (e.g., RAM)	code size (e.g., Flash)
Class 0, C0	<< 10 KiB	<< 100 KiB
Class 1, C1	~ 10 KiB	~ 100 KiB
Class 2, C2	~ 50 KiB	~ 250 KiB

Table 1: Classes of Constrained Devices (KiB = 1024 bytes)

Class 0 devices are very constrained sensor-like motes. They are so severely constrained in memory and processing capabilities that most likely they will not have the resources required to communicate directly with the Internet in a secure manner (rare heroic, narrowly targeted implementation efforts notwithstanding). Class 0 devices will participate in Internet communications with the help of larger devices acting as proxies, gateways, or servers. Class 0 devices generally cannot be secured or managed comprehensively in the traditional sense. They will most likely be preconfigured (and will be reconfigured rarely, if at all) with a very small data set. For management purposes, they could answer keepalive signals and send on/off or basic health indications.

Class 1 devices are quite constrained in code space and processing capabilities, such that they cannot easily talk to other Internet nodes employing a full protocol stack such as using HTTP, Transport Layer Security (TLS), and related security protocols and XML-based data representations. However, they are capable enough to use a protocol stack specifically designed for constrained nodes (such as the Constrained Application Protocol (CoAP) over UDP [COAP]) and participate in meaningful conversations without the help of a gateway node. In particular, they can provide support for the security functions required on a large network. Therefore, they can be integrated as fully developed peers into an IP network, but they need to be parsimonious with state memory, code space, and often power expenditure for protocol and application usage.

Class 2 devices are less constrained and fundamentally capable of supporting most of the same protocol stacks as used on notebooks or servers. However, even these devices can benefit from lightweight and energy-efficient protocols and from consuming less bandwidth. Furthermore, using fewer resources for networking leaves more resources available to applications. Thus, using the protocol stacks defined for more constrained devices on Class 2 devices might reduce development costs and increase the interoperability.

Constrained devices with capabilities significantly beyond Class 2 devices exist. They are less demanding from a standards development point of view as they can largely use existing protocols unchanged. The present document therefore does not make any attempt to define classes beyond Class 2. These devices can still be constrained by a limited energy supply.

Performance comparison

```
[dave@hal9000 ~]$ openssl speed aes-128-cbc
Doing aes-128 cbc for 3s on 16 size blocks: 26126940 aes-128 cbc's in 3.00s
Doing aes-128 cbc for 3s on 64 size blocks: 7160075 aes-128 cbc's in 3.00s
...
The 'numbers' are in 1000s of bytes per second processed.
type             16 bytes     64 bytes    256 bytes   1024 bytes   8192 bytes
aes-128 cbc      139343.68k   152748.27k   155215.70k   155745.61k   157196.29k


[dave@hal9000 ~]$ openssl speed rsa2048
Doing 2048 bit private rsa's for 10s: 9267 2048 bit private RSA's in 9.99s
Doing 2048 bit public rsa's for 10s: 299665 2048 bit public RSA's in 9.99s
...
                  sign    verify    sign/s verify/s
rsa 2048 bits 0.001078s 0.000033s    927.6  29996.5
```

In 3 sec ~26Mln 16 byte

In 3 sec ~ (26*16) 416Mln bytes

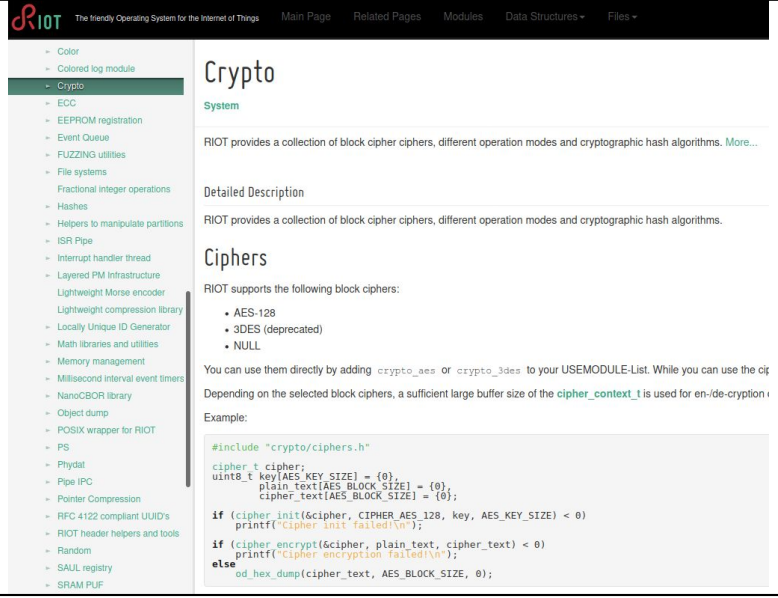
In 1 sec ~ (416/3) 138Mln

256 bytes = 2048 bit

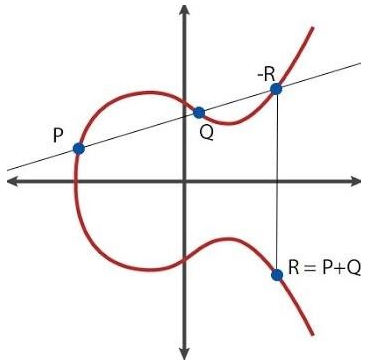
TABLE IV. COMPARISON AMONG DIFFERENT RECENT (2015-2018) AES IMPLEMENTATIONS.

Ref.	Year	Tech.	Data-path	Frequency (MHz)	Area (kGEs)	Area (mm2)	Cycles/ Encryption	Power (mW)	Throughput	Energy/Bit (pJ/bit)
[14]	'17	130nm	-	50	-	-	-	10.74	2.601 Gbps	-
[17]	'17	130nm	-	-	-	0.64	-	5.47	-	-
[18]	'17	28nm	32 bit	10	8.6	-	44	0.02	28 Mbps	0.65
[20]	'18	28nm	8 bit	50	-	0.0028	213	0.045	30.05 Mbps	1.50
[21]	'15	22nm	8 bit	76	1.95	0.0022	336	0.17	29 Mbps	5.6
[22]	'16	40nm	8 bit	122	2.2	0.00429	337	0.1	46.2 Mbps	2.2
[24]	'15	90nm	32 bit	10	5.5	-	44	-	28 Mbps	6.2
[23]	'15	65nm	8 bit	32	4	0.012	160	0.0617	25.6 Mbps	2.3

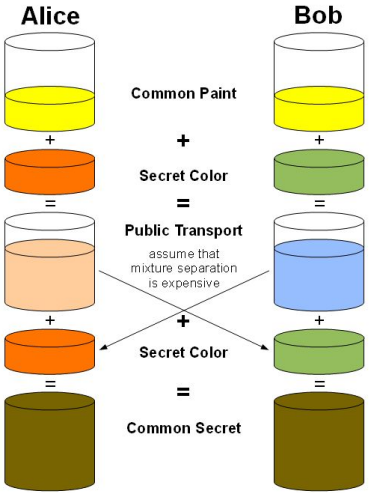
I. K. Dutta, B. Ghosh and M. Bayoumi, "Lightweight Cryptography for Internet of Insecure Things: A Survey," *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, 2019, pp. 0475-0481, doi: 10.1109/CCWC.2019.8666557.

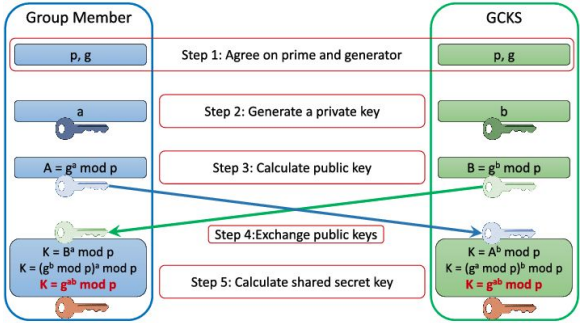


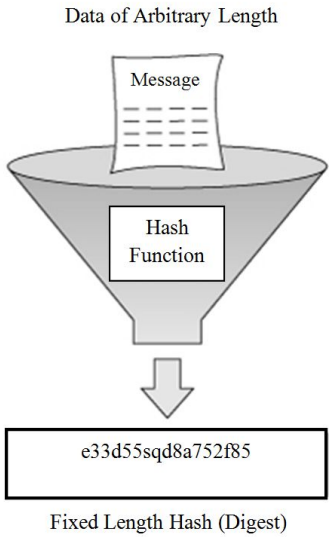
WHAT ABOUT ASYMMETRIC CRYPTO?

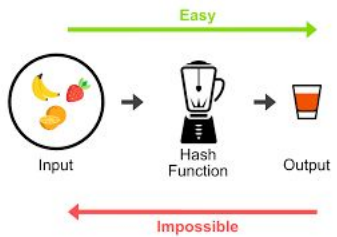


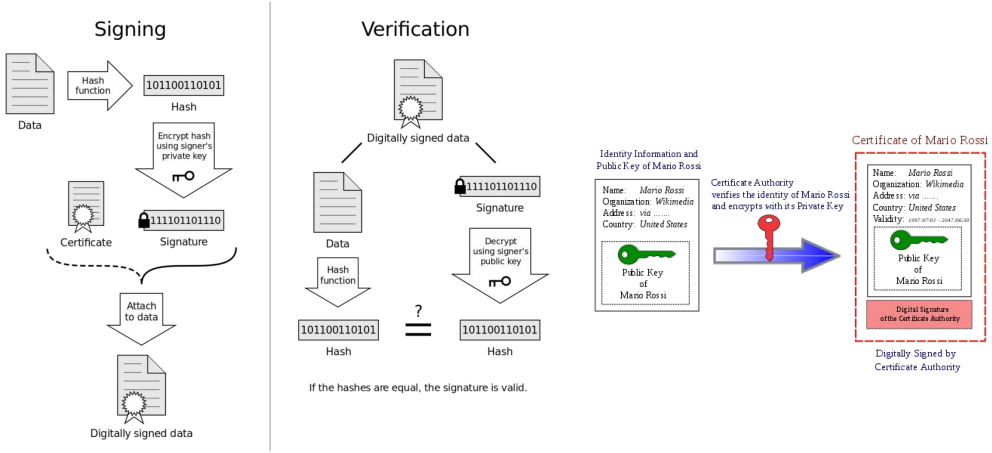
Diffie-Hellman

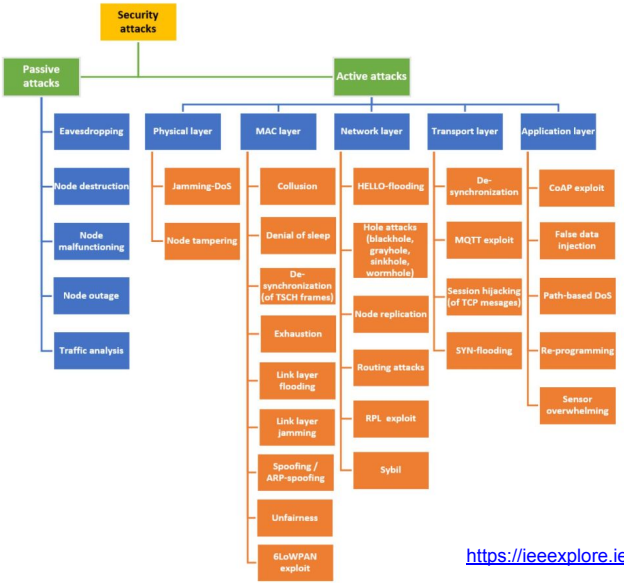






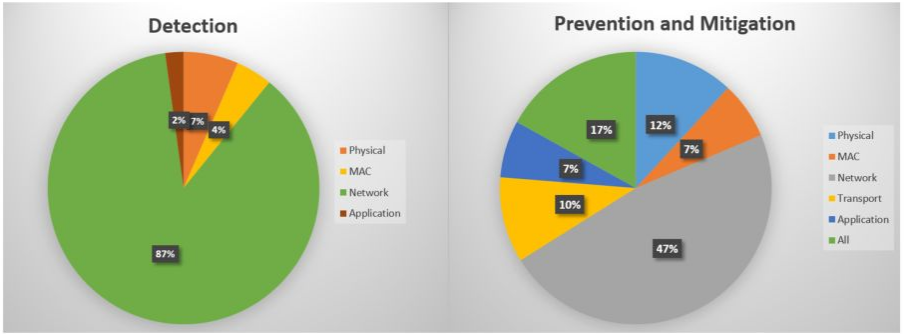






https://ieeexplore.ieee.org/document/8897627

DoS attack	Defense strategy
Radio interference	Usage of spread-spectrum communication
Physical tampering	Usage of tamper-resistant nodes
Denying channel	Usage of error correction codes
Blackhole	Usage of multiple routing paths
Misdirection	Usage of source authorization
Flooding	Limiting the total number of connections



Distribution of the cyber-security solutions according to OSI representation.

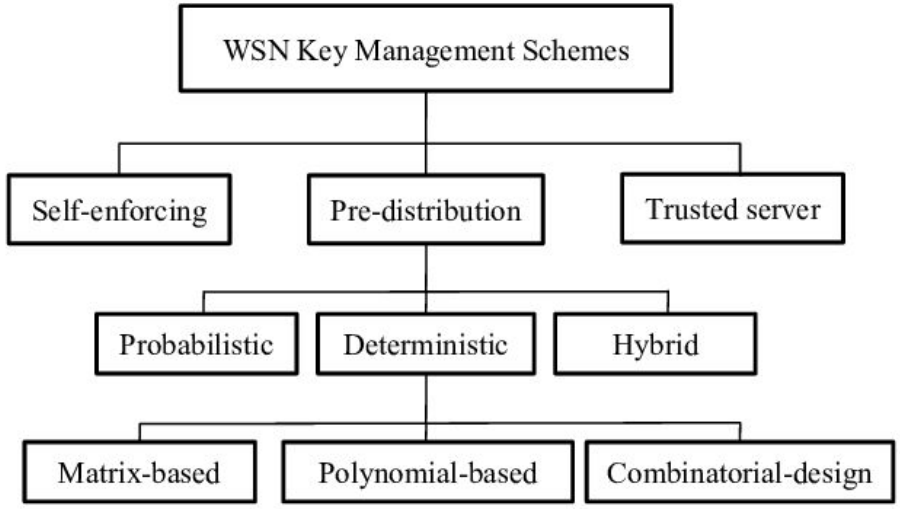
TABLE II
LAYERED CATEGORIZATION OF CYBER-SECURITY ATTACKS TOWARDS WSNS AND IOT* ALONG WITH THE PROPOSED SOLUTIONS TO DEFEND AGAINST THOSE ATTACKS

Attack type	Layer	Priority	Proposed Solutions for Detection	Proposed Solutions for Prevention/Mitigation
Eavesdropping	All layers	Medium	N/A	Link-layer encryption [63], [89], [90], [91], [92], [93], SensorWare communication multicast model [94], Key pre-distribution [95], [96], [97]
Jamming-DoS	Physical	High	Swarm intelligence [99], JAM (mapping) [100]	Usage of spread-spectrum communication [98], JAM (re-routing) [100], Wormhole technique [101]
Tampering	Physical	Low	Routinely executing physical checks	Tamper resistant hardware, disabling JTAG and/or protecting bootstrap loader [67], camouflaging [18]
Col.& Exhaus.	MAC	Medium	Error detection codes [18]	TDM, Error correction codes [18]
Denial of sleep	MAC	Medium	Anomaly detection on motes [70]	N/A
De-Synch.	MAC	Low	N/A	6TiSCH [102]
Unfairness	MAC	Medium	N/A	Usage of small frames [61]
6LoWPAN expl.	MAC	Medium	N/A	6LowPSec [103], Content chaining scheme [72]

According to PC Magazine, here are four straightforward IoT security lessons from MIRAI

- "Devices that cannot have their software, passwords, or firmware updated should never be implemented.
- Changing the default username and password should be mandatory for the installation of any device on the Internet.
- Passwords for IoT devices should be unique per device, especially when they are connected to the Internet.
- Always patch IoT devices with the latest software and firmware updates to mitigate vulnerabilities."

source: The 5 Worst Examples of IoT Hacking and Vulnerabilities in Recorded History



Source: https://www.researchgate.net/figure/Classification-of-key-management-schemes-in-WSN_fig8_266083307

Key distribution schemes

- Pre-Distribution Schemes: sensor nodes store some initial keys before the nodes are deployed
  - Deterministic
    - single mission-key: the capture of any sensor node may compromise the entire network. Revocation is ineffective
    - pair-wise private sharing + revocation: n-1 keys in each sensor, n(n-1) in the whole network
  - Probabilistic: common pre-distribution keys w.h.p

Probabilistic

- A ring of keys is distributed to each sensor node before deployment. Each key ring consists of a randomly chosen k keys from a large pool of P keys, which is generated offline.
- A pair of nodes can communicate if they share any key among their key rings.
- Although a pair of nodes may not always have a shared key, if a path between them exists, they can use that path to exchange a key that establishes a direct link.

A Key-Management Scheme for Distributed Sensor Networks*

Laurent Eschenauer
Electrical and Computer Engineering Department
University of Maryland
College Park, MD, USA
laurent.e@mail.com

Virgil D. Gligor
Electrical and Computer Engineering Department
University of Maryland
College Park, MD, USA
gligor@eng.umd.edu

ABSTRACT

Distributed Sensor Networks (DSNs) are ad-hoc mobile networks that include sensor nodes with limited computation and communication capabilities. DSNs are dynamic in the sense that they allow addition and deletion of sensor nodes after deployment to grow the network or replace failing and unreliable nodes. DSNs may be deployed in hostile areas where communication is monitored and nodes are subject to capture and surreptitious use by an adversary. Hence DSNs require cryptographic protection of communications, sensor-capture detection, key revocation and sensor disabling. In this paper, we present a key-management scheme designed to satisfy both operational and security requirements of DSNs. The scheme includes selective distribution and revocation of keys to sensor nodes as well as node re-keying without substantial computation and communication capabilities. It relies on probabilistic key sharing among the nodes of a random graph and uses simple protocols for shared-key discovery and path-key establishment, and for key revocation, re-keying, and incremental addition of nodes. The security and network connectivity characteristics supported by the key-management scheme are discussed and simulation experiments presented.

Categories and Subject Descriptors

C.2.0 [**Computer-communication networks**]: General—*security and protection*

Keywords

key management, sensor networks, random graphs, probabilistic key sharing

1. INTRODUCTION

Distributed Sensor Networks (DSNs) share several characteristics with the more traditional embedded wireless networks [13]. Both include arrays of sensor nodes that are battery powered, have limited computational capabilities and memory, and rely on intermittent wireless communication via radio frequency and, possibly, optical links. Both include data-collection nodes, which cache sensor data and make it available for processing to application components of the network, and control nodes, which monitor the status of and broadcast simple commands to sensor nodes. Although in both networks most nodes have limited, if any, mobility after deployment, some nodes are highly mobile (e.g., data collection and control nodes placed on humans, vehicles, aircraft). However, DSNs differ from the traditional embedded wireless networks in several important areas, namely: their scale is orders of magnitude larger than that of embedded wireless networks (e.g., tens of thousands as opposed to just tens of sensor nodes); they are dynamic in the sense that they allow addition and deletion of sensor nodes after deployment to extend the network or replace failing and unreliable nodes without physical contact; and they may be deployed in hostile areas where communication is

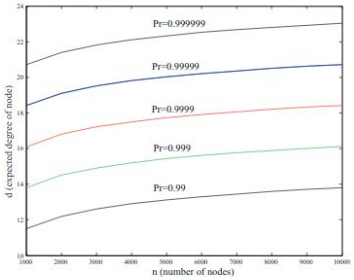


Figure 1: Expected degree of node vs. number of nodes, where $P_c = Pr[G(n,p)$ is connected]

neighbors). To establish DSN shared-key connectivity, we need to answer the following two questions:

- what value should the expected degree of a node, d, have so that a DSN of n nodes is connected? and,
- given d and the neighborhood connectivity constraints imposed by wireless communication (e.g., the number of nodes n' in a neighborhood), what values should the key ring size, k, and pool, P, have for a network of size n? In particular, if memory capacity of each sensor limits the key ring size to a given value of k, what should the size of the key pool, P, be?

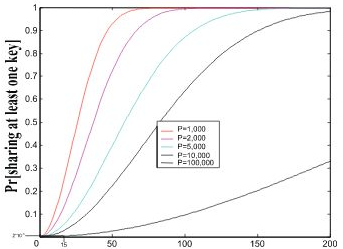
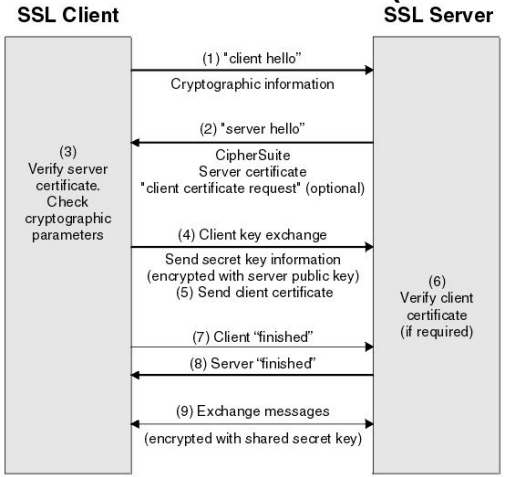


Figure 2: Probability of sharing at least one key when two nodes choose k keys from a pool of size P

TRUSTED SERVER (SSL RSA)



TRUSTED SERVER

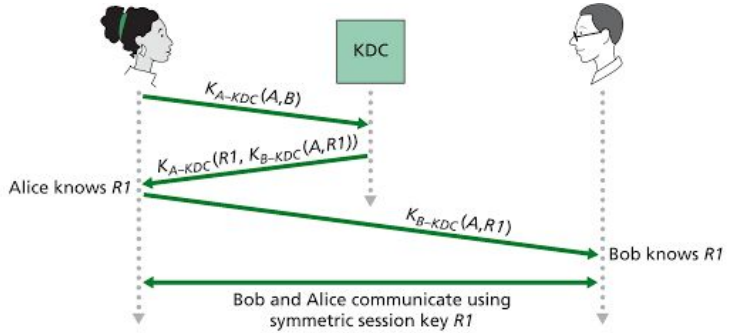


Figure 8.19 ♦ Setting up a one-time session key using a key distribution center

Trusted Execution Environment (TEE)

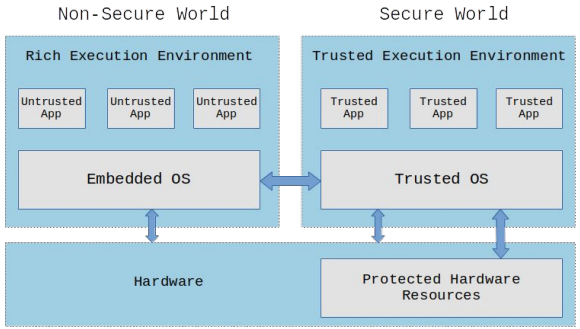


TABLE III. COMPARISON BETWEEN HARDWARE AND SOFTWARE SECURITY SOLUTION

Hardware Security Solution	Software Security Solution
Keys are segregated within an isolated security environment	Numerous copies of keys live across system and backups
Can contain internally managed memory space, gives more memory space protection	Memory access is not secure enough. It can't facilitate their own physical memory, it uses externally available memory or secondary memory, where memory protection is difficult
More data integrity assurance	Less data integrity assurance
Less susceptible to reverse engineering	Software implementations are more easily readable by adversaries and are therefore more susceptible to reverse engineering
Can mask the fluctuation in power consumption to prevent Power Analysis attacks	Have a defined pattern in terms of power consumption, not resistant to Power analysis attacks
Not dependent on high level operating system services	Dependent on Operating System Security

A Trusted Execution Environment (TEE) is an isolated environment where, even if the operating system is compromised, your data is protected.

Blackhole	Network	High	Anomaly detection on motes [70], REWARD [104], ActiveTrust [109], Packet count [110], TinyOS beaconing [111], Honeypot [106], Watchdog [107], Pseudo clustering algo. [112]	REWARD routing [104], Multi-path routing [63], [98], [126], Mesh net. topology [105], ActiveTrust routing [109], Isolation [110], BAMBi [111], MAODV [108]
HELLO flooding	Network	Medium	Bidirectional verification technique [113]	Identity verification protocol [63], Multi-path multi-base station routing [113], μ-TESLA [89]
Node-Replication (Clone)	Network	High	*Centralized solutions:* SET [116], Random pair wise key pre-distribution [117], Social fingerprinting [118], Speed test [119], Multi-level clustering [120] *Distributed solutions:* HIP-HOP [75], N2NB, DM, RM and LSM [76], SDC and P-MPC [121], RED [122], MEM [123], RDE [124]	ID-based public keys [114], Location-based key management [115], Multi-level clustering [120]
RPL DODAG	Network	Medium	N/A	Integrity check [80], VeRA [140]
RPL local repair	Network	Medium	N/A	Inclusion of timer in link repair messages, VeRA [140]
RPL rank	Network	Medium	N/A	TRAIL [139], VeRA [140]
Rushing	Network	Medium	N/A	Rushing Attack Prevention (RAP) [77]
Selective forwarding (Grayhole)	Network	High	Anomaly detection on motes [70], Acknowledgment monitoring [125], Neighbor knowledge [127], Reporting packet drops [128], Failure detection framework [129], Watchdog [107]	Multi-path routing [63], [126], Source authorization [98]
Sinkhole	Network	High	Network flow graph [130], Geo-statistical sampling approach and distributed monitoring approach [131], Redundancy mechanism [132]	Secure routing algorithm [133]
Sybil	Network	Medium	Radio resource testing, ID-based symmetric keys, registration, position verification, code attestation [134], RADS [135]	Indirect validation [63], Identity verification [134], Isolation [135], ID-based public keys [114]
Wormhole	Network	Medium	Packet Leashes [74], Directional antennas [136]	Location-based keys [114], Centralized computing [137], DAWWSEN [138]

De-Synch.	Transport	Low	N/A	Authentication via transport layer protocol headers [61]
SYN-flooding	Transport	Medium	N/A	SYN-cookies [64], Client puzzles [141]
MQTT exploit	Transport	Medium	N/A	Enforcement of security policies [142], SMQTT [84]
Session hijacking	Transport	Medium	N/A	Light-weight user authentication algorithm for optimized routing in mobile networks [144]
CoAP exploit	App.	Medium	N/A	CoAPs, employment of DTLS [86]
False data injec.	App.	Low	SET [145]	Collective secret [145]
Path-based DoS	App.	Medium	N/A	One-way hash chains [87]

* IoT specific attacks and their corresponding solutions are highlighted with the red colored text!

future internet

MDPI

Review

Security of IoT Application Layer Protocols: Challenges and Findings

Giuseppe Nebbione * and Maria Carla Calzarossa

Department of Electrical, Computer and Biomedical Engineering, University of Pavia, I-27100 Pavia, Italy; mcc@unipv.it

* Correspondence: giuseppe.nebbione01@universitadipavia.it

Received: 17 January 2020; Accepted: 14 March 2020; Published: 17 March 2020

check for updates

Abstract: IoT technologies are becoming pervasive in public and private sectors and represent presently an integral part of our daily life. The advantages offered by these technologies are frequently coupled with serious security issues that are often not properly overseen or even ignored. The IoT threat landscape is extremely wide and complex and involves a wide variety of hardware and software technologies. In this framework, the security of application layer protocols is of paramount importance since these protocols are at the basis of the communications among applications and services running on different IoT devices and on cloud/edge infrastructures. This paper offers a comprehensive survey of application layer protocol security by presenting the main challenges and findings. More specifically, the paper focuses on the most popular protocols devised in IoT environments for messaging/data sharing and for service discovery. The main threats of these protocols as well as the Common Vulnerabilities and Exposures (CVE) for their products and services are analyzed and discussed in detail. Good practices and measures that can be adopted to mitigate threats and attacks are also investigated. Our findings indicate that ensuring security at the application layer is very challenging. IoT devices are exposed to numerous security risks due to lack of appropriate security services in the protocols as well as to vulnerabilities or incorrect configuration of the products and services being deployed. Moreover, the constrained capabilities of these devices affect the types of security services that can be implemented.

Keywords: IoT; security; threat; mitigation; application layer protocols; CVE; MQTT; CoAP; mDNS; SSDP; AMQP; DDS; XMPP; good practices

Table 2. Summary of the security services supported by the messaging protocols.

Protocol	Authentication		Authorization	Confidentiality	
	SASL	Custom	Custom	TLS	DTLS
MQTT		•		•	
CoAP					•
AMQP	•			•	
DDS		•	•	•	•
XMPP	•		•	•	

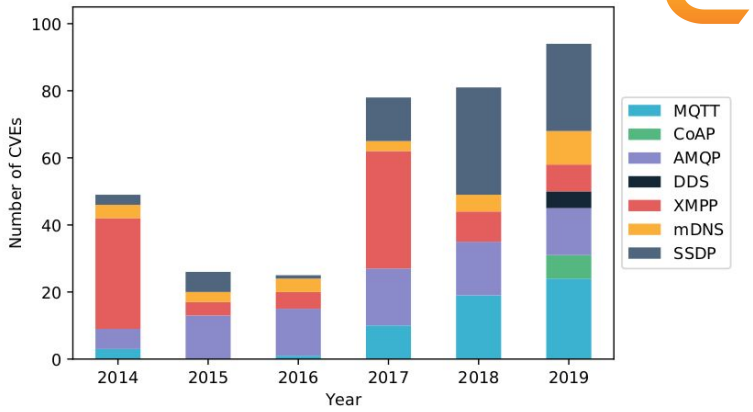


Figure 3. Breakdown of the CVEs per year and protocol.

Table 4. Per protocol breakdown of the number of CVEs according to their severity and overall CVSS2 score.

Protocol	Severity			CVSS2 Score
	Low	Medium	High	
MQTT	3	42	12	5.6
CoAP	0	5	2	6.6
AMQP	11	50	17	5.2
DDS	0	5	0	5.0
XMPP	5	70	19	5.6
mDNS	0	16	13	6.4
SSDP	5	49	27	5.9

It is also important to outline that security risks and vulnerabilities expose IoT devices to a wide range of threats and attacks (see Table 5) that could have very serious effects.

Table 5. Summary of the major attacks affecting the application layer protocols analyzed in this paper.

Protocol	Eavesdropping Attacks	IP Spoofing Attacks	DoS/DDoS Attacks	MiTM Attacks	Poisoning Attacks
MQTT			•	•	
CoAP		•	•	•	
AMQP			•		
DDS			•		
XMPP			•	•	
mDNS	•	•	•	•	•
SSDP	•	•	•	•	•

MQTT

- *Authentication*: the MQTT broker does not properly check the publisher/subscriber identity and does not block repeated authentication attempts. These vulnerabilities could grant an attacker the access to MQTT devices or could overload the broker and eventually make it crash;
- *Authorization*: the MQTT broker does not properly set the publishing/subscribing permissions. This vulnerability could grant an attacker the control over data or functions of MQTT devices;
- *Message delivery*: a publisher sends messages that cannot be delivered because of the lack of subscribers. This vulnerability could lead to significant degradation of broker performance;
- *Message validation*: a publisher sends messages containing disallowed characters that are not properly interpreted by brokers and subscribers. This vulnerability could be exploited to perform many different malicious attacks;
- *Message encryption*: clients and servers exchange messages in plaintext, thus allowing an attacker to eavesdrop and spoof the messages in transit. This vulnerability could be exploited to perform Man-in-The-Middle (MiTM) attacks.

https://blog.avast.com/mqtt-vulnerabilities-hacking-smart-homes

COAP

- *Message parsing*: the processing logic of client and server parsers does not properly handle incoming messages. This vulnerability could affect CoAP node availability because of overload conditions and even open the ability to remotely execute arbitrary code on the node under attack;
- *Proxying and caching*: the access control mechanisms of proxies and caches are not properly implemented. This vulnerability could compromise their content, thus breaking confidentiality and integrity of CoAP messages;
- *Bootstrapping*: the setup of new CoAP nodes is not properly implemented. This vulnerability could grant unauthorized nodes the access to a CoAP environment;
- *Key generation*: the generation of cryptographic keys is not sufficiently robust. The usage of these keys could compromise CoAP nodes;
- *IP address spoofing*: by forging the IP addresses of CoAP nodes, an attacker could perform a variety of side attacks including the generation of spoofed response messages and acknowledgments as well as reflection/amplification attacks;
- *Cross-protocol exchanges*: an attacker sends a CoAP node a message with a spoofed IP address and a fake source port number; the response of this node will reach the node under attack and force it to interpret the received message according to the rules of the target protocol.

The analysis of the few CVEs affecting products and services based on CoAP suggests that these vulnerabilities materialize differently. In particular, according to our classification, the most common security issue refers to improper message parsing. For example, some CoAP libraries mishandle invalid options or certain exceptions when receiving specifically crafted messages (e.g., CVE-2018-12679, CVE-2018-12680). Other libraries are affected by overflow vulnerabilities while processing an incoming message (e.g., CVE-2019-17212). The exploitation of these vulnerabilities could

TABLE IV
PROMISING CYBER-SECURITY SOLUTIONS FOR IOT

Security proposal	OSI layer	Cyber-defense strategy
PUFs [166]	Physical	Hardware embedded security functions
NOMA-based mMTC network [167], [168]	Physical	Introducing interference on the communication channels to ensure security
IDS framework [169]	Physical	Linear regression method by using radio signal measurements
6LowPSec [103]	MAC	End-to-end security solution that works on 6LoWPAN protocol
SoftThings [170]	Network	Machine learning is used at the SDN controller
IDS framework [169]	Network	Usage of a strict hierarchy and a verification process of nodes and blacklisting
CoAP protocol [85]	Transport	Employment of DTLS which provides TLS equivalent security
MQTT-Security [142]	Transport	Enforcement of security policy rules for IoT
SMQTT [84]	Transport	Employment of encryption (ABE) on transmitted data
ITS [171]	Application	Byzantine-resilient state machine approach for CPHS
Privacy-preserving HAN [172]	Application	Usage of encrypted overlay network over commercially available VPN
Trust establishment [173]	Application	Blending both hardware- and software-based remote attestation schemes
IoT security framework [174]	All layers	A threat model for IoT to be employed for smart cyber-infrastructures

DTLS

DTLS is a secure data transfer protocol used to encrypt data transferred over datagram protocols(typycally UDP).

The DTLS protocol provides communications privacy for datagram protocols.

It is based on the Transport Layer Security (TLS) protocol and provides equivalent security guarantees.

DTLS is defined in RFC 4347 and RFC 6347.

DTLS

SSL/TLS is designed to be working over reliable transport channel(typically TCP).

SSL/TLS cannot tolerate data loss and out of sequence data records.

If a data record is coming in out of sequence, the data record may not be decrypted correctly but with MAC verification error.

What if an application is built based on UDP?

The data record may be out of sequence or may be lost, what if the data security of this kind also needs to be guaranteed?

This is why DTLS is designed.

DTLS

- Add explicit sequence number field to record. This ensures data can be reordered correctly when it's out of sequence. In TLS, the sequence number is in the encrypted record MAC and can be verified only after the decryption. In DTLS, this is not possible since data may get lost
- Drop support of some cipher suites used in TLS. Stream ciphers are banned in DTLS since the decryption of data in stream cipher depends on the previous decrypted data. These ciphers cannot be used in DTLS as data may get lost. Typically, RC4 cannot be used in DTLS.
- Add retransmission mechanism in case of packet loss. There is a retransmission timer which will retransmit a packet after some timeout if it sending side doesn't get the correct response.
- Change the alert mechanism for MAC verification failure. In TLS, if a MAC verification fails, then a fatal error will be sent and the connection will be invalidated. In DTLS, if a MAC verification fails, the recommended method is just drop the record received without aborting the connection.
- A new record message is added -- **HelloVerifyRequest**. HelloVerifyRequest is designed to prevent DoS attacks. Since data security protocols are vulnerable to DoS attacks, to prevent these attacks. HelloVerifyRequest will be used to identify the sender of the ClientHello is not faked with invalid IPs. The HelloVerifyRequest is designed to be small and it contains a cookie which is used to identify the client. After the client sends the ClientHello message, server SHOULD reply with a HelloVerifyRequest and then the client will send the ClientHello again with the cookie from HelloVerifyRequest attached. Then the rest of handshake will continue.

4.2. The DTLS Handshake Protocol

DTLS uses all of the same handshake messages and flows as TLS, with three principal changes:

1. A stateless cookie exchange has been added to prevent denial of service attacks.

2. Modifications to the handshake header to handle message loss, reordering, and fragmentation.

3. Retransmission timers to handle message loss.

With these exceptions, the DTLS message formats, flows, and logic are the same as those of TLS 1.1.

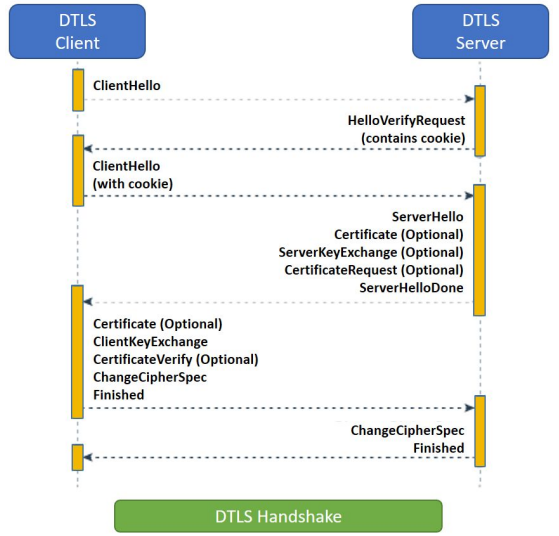
4.2.1. Denial of Service Countermeasures

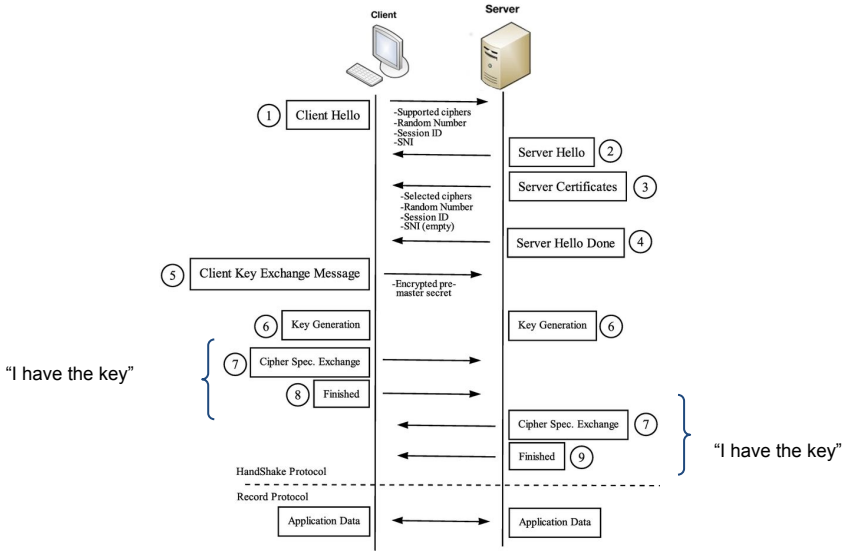
Datagram security protocols are extremely susceptible to a variety of denial of service (DoS) attacks. Two attacks are of particular concern:

1. An attacker can consume excessive resources on the server by transmitting a series of handshake initiation requests, causing the server to allocate state and potentially to perform expensive cryptographic operations.

2. An attacker can use the server as an amplifier by sending connection initiation messages with a forged source of the victim. The server then sends its next message (in DTLS, a Certificate message, which can be quite large) to the victim machine, thus flooding it.

In order to counter both of these attacks, DTLS borrows the stateless cookie technique used by Photuris [PHOTURIS] and IKE [IKE]. When the client sends its ClientHello message to the server, the server MAY respond with a HelloVerifyRequest message. This message contains a stateless cookie generated using the technique of [PHOTURIS]. The client MUST retransmit the ClientHello with the cookie added. The server then verifies the cookie and proceeds with the handshake only if it is valid. This mechanism forces the attacker/client to be able to receive the cookie, which makes DoS attacks with spoofed IP addresses difficult. This mechanism does not provide any defense against DoS attacks mounted from valid IP addresses.



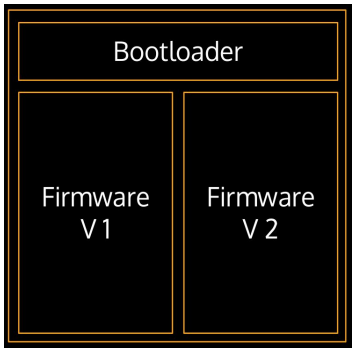


"You can't secure what you can't update"

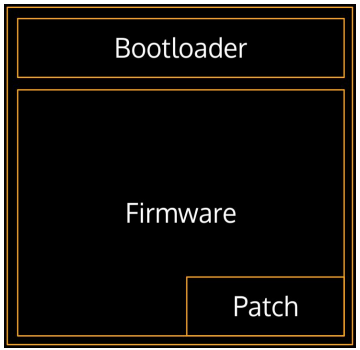
According to PC Magazine, here are four straightforward IoT security lessons from MIRAI

- "Devices that cannot have their software, passwords, or firmware updated should never be implemented.
- Changing the default username and password should be mandatory for the installation of any device on the Internet.
- Passwords for IoT devices should be unique per device, especially when they are connected to the Internet.
- Always patch IoT devices with the latest software and firmware updates to mitigate vulnerabilities."

source: The 5 Worst Examples of IoT Hacking and Vulnerabilities in Recorded History



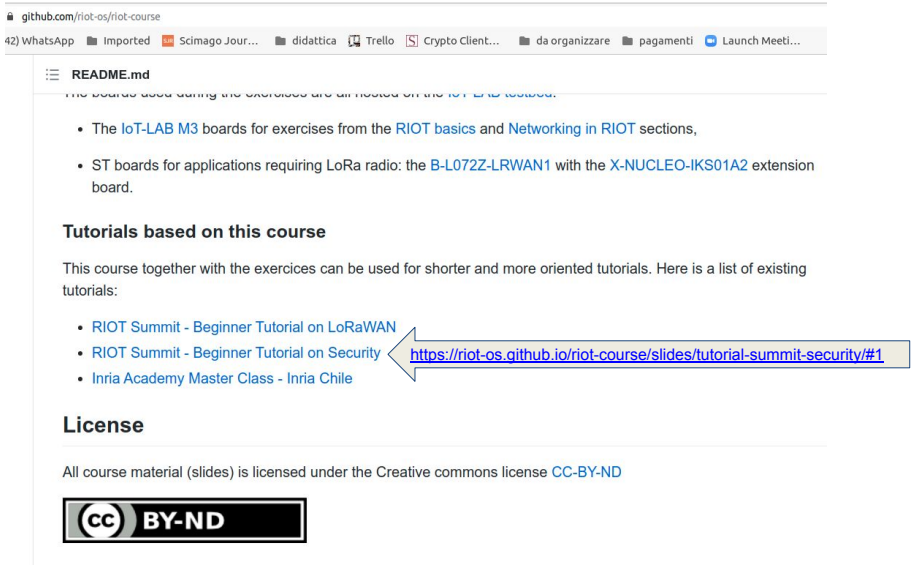
FLASH and SWITCH

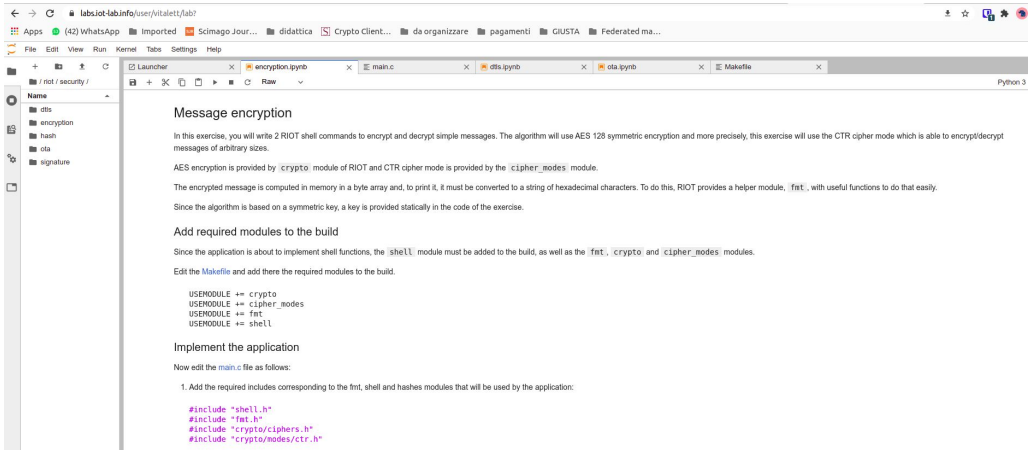


Reboot & Patch update

Must be bulletproof

- Upgrading is hard:
  - NAND flash errors
  - Unexpected power loss
  - Network errors
  - Unexpected incompatibilities
  - Checksum, cryptographic signature
- A 0.1% failure rate on a 1m fleet is 1000 bricked devices

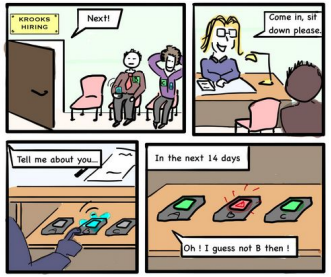




Very serious implications also in crowdsensing … future class!

Anonymous tracing, a dangerous oxymoron

A risk analysis for non-specialists



The KROOKS company intends to recruit a temporary employee. They want to make sure that the candidate does not fall sick between the job interview and signing the contract. They therefore use a dedicated phone that is switched on only during the interview, and which will receive an alert if the candidate later tests positive for the disease.

Automated contact tracing using a *smartphone* application yields many risks that are independent from the specifics of the inner-workings of said application. We are specialists in cryptography, security or technology law. Our expertise lies notably in our ability to anticipate the various abuses, misuses, and other ill-intentioned behaviours that could arise. **We offer an analysis of the risks posed by such an application that is based on the study of concrete scenarios, and which is aimed at non-specialists.**

Download the full document (.pdf)