KADIR HAS UNIVERSITY
SCHOOL OF GRADUATE STUDIES
DEPARTMENT OF ENGINEERING AND NATURAL SCIENCES

A NOVEL COMMUNICATION METHOD FOR CONSTRAINED IoT DEVICES

TUĞBERK KOCATEKİN

DOCTOR OF PHILOSOPHY THESIS

ISTANBUL, JUNE, 2022

A NOVEL COMMUNICATION METHOD FOR CONSTRAINED IoT DEVICES

TUĞBERK KOCATEKİN

A thesis submitted to
the School of Graduate Studies of Kadir Has University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in
Computer Engineering

İstanbul, June, 2022

APPROVAL

This thesis titled A NOVEL COMMUNICATION METHOD FOR CONSTRAINED IoT DEVICES submitted by TUĞBERK KOCATEKİN, in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Engineering is approved by

Assoc. Prof. Tamer DAĞ (Advisor) .....
Kadir Has University

Prof. Dr. Cafer ÇALIŞKAN (Co-Advisor) .....
Antalya Bilim University

Assoc. Prof. Habib ŞENOL .....
Kadir Has University

Asst. Prof. Öznur Yaşar DİNER .....
Kadir Has University

Asst. Prof. Murat AK .....
Akdeniz University

Asst. Prof. Aslı BAY .....
Antalya Bilim University

I confirm that the signatures above belong to the aforementioned faculty members.

.....
Prof. Dr. Mehmet Timur Aydemir
Dean of School of Graduate Studies
Date of Approval: 24.06.2022

DECLARATION ON RESEARCH ETHICS AND PUBLISHING METHODS

I, TUĞBERK KOCATEKİN; hereby declare

- that this Ph.D. Thesis that I have submitted is entirely my own work and I have cited and referenced all material and results that are not my own in accordance with the rules;
- that this Ph.D. Thesis does not contain any material from any research submitted or accepted to obtain a degree or diploma at another educational institution;
- and that I commit and undertake to follow the "Kadir Has University Academic Codes and Conduct" prepared in accordance with the "Higher Education Council Codes of Conduct".

In addition, I acknowledge that any claim of irregularity that may arise in relation to this work will result in a disciplinary action in accordance with university legislation.

TUĞBERK KOCATEKİN

.....

24.06.2022

ACKNOWLEDGEMENT

First of all, I would like to express my gratitude to my advisor Prof. Cafer Çalışkan. Without his constant support and guidance, this thesis could not be completed. I also thank the members of my committee, who were very generous with their expertise and time.

I would like to thank my family for all the support and understanding they provided throughout the years. Without their support, I would not be where I am today.

ABSTRACT

Internet of Things (IoT) is becoming an established part of life by interconnecting billions of devices in diverse areas such as healthcare, smart homes, industries, etc. However, these devices are limited in memory, energy and computational capabilities. Being constrained prevents them from applying complex cryptographic encryption algorithms which leads to lack of security and therefore lack of privacy. As a solution, we propose a novel secret sharing scheme based on underlying protocols of visual cryptography to provide a low-cost and secure communication method for constrained IoT devices. Generally, when a device wants to communicate with an outer party, it does so by itself or by using a mediary such as a central hub or gateway; which leads to single point of failure. As a solution, we propose a method where devices collaborate each other and therefore divide the responsibility into multiple, instead of one. We propose two different models: *n-out-of-n* and *k-out-of-n*. In the first model, there is a complete graph where every device is connected to each other. Instead of the original sender, every other device work collaboratively to communicate with the outer party. In the second model, the network is realized as an n-regular graph where a single node has n number of neighbors, which collaborates with each other and here the responsibility is divided into n devices. Results show that this scheme is applicable to constrained devices.

Keywords: IoT, security, secret sharing, visual cryptography, constrained devices

KISITLI IoT CİHAZLARI İÇİN YENİ BİR HABERLEŞME METODU

ÖZET

Nesnelerin İnterneti (IoT), farklı alanlardaki (sağlık, akıllı ev, vb.) binlerce cihazı birbirine bağlayabilme yetisiyle, günlük hayatın yerleşmiş bir parçası olmaya başladı. Ancak, bu cihazlar hafıza, enerji ve hesaplama yeteneği konusunda limitliler. Bu limit nedeniyle IoT cihazlarının karmaşık kriptografik hesaplamalar yapmasının olanaksız hale gelmesi gizlilik ve güvenlik sorunları ortaya çıkarıyor. Buna bir çözüm olarak, görsel kriptoloji altyapısını kullanarak; kısıtlı IoT cihazları için, düşük maliyetli ve güvenli bir haberleşme ve sır paylaşımı yöntemi öneriyoruz. Genellikle, bir cihaz dışarıyla iletişim kurmak istediğinde bunu ya direkt olarak, ya da merkezi bir yapıyı aracı kullanarak kuruyor. Bu da, single point of failure'a neden oluyor. Buna çözüm olarak cihazların işbirliği yaptığı, sorumluluğu tek bir cihaz yerine birkaç cihaza böldüğü bir model öneriyoruz. Önerdiğimiz sistem iki modelden oluşuyor: *k'dan k* ve *n'den k*. İlk modelde, her cihaz birbiriyle bağlantılı ve bilgi göndermek isteyen cihaz dışındaki tüm cihazların işbirliği yapmasıyla bilgi gönderiliyor. İkinci modelde ise, sistemi n-regular graph olarak tanımlayarak, herhangi bir cihazın n sayıda komşusu olduğunu belirledikten sonra; bu n komşuyu işbirliği içerisinde kullanılıyor ve sorumluluk tek cihaz yerine n cihaza bölünüyor. Sonuçlar, bu sistemin kısıtlı cihazlar için güvenli ve uygulanabilir olduğunu gösteriyor.

Anahtar Sözcükler: IoT, güvenlik, görsel kriptoloji, kısıtlı cihazlar

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS AND ABBREVIATIONS

AMQP	Advanced Message Queuing Protocol
BLE	Bluetooth Low Energy
DMIPS	Dhrystone Million Instruction per Second
IETF	Internet Engineering Task Force
IoT	Internet of Things
IP	Internet Protocol
MAC	Message Authentication Code
MITM	Man in the Middle Attack
MQTT	Message Queuing Telemetry Transport
NB-IoT	Narrowband Internet of Things
NFC	Near Field Communication
LoRaWAN	LoRa Wide Area Network
LTE	Long-Term Evolution
RFID	Radio Frequency Identification
SoC	System on a Chip
QoS	Quality of Service
Wi-Fi	Wireless Fidelity
XMPP	Extensible Messaging and Presence Protocol

1. INTRODUCTION

Web technologies improved dramatically over the recent years. This improvement started from simple HTML web pages to Web 2.0 with social networks, online applications, wikis; which became indispensable for our daily and business lives. Currently, Web 2.0 dominates the Internet, and it is hard to find a simple web page with a simple design which only aims to provide information. Web 3.0 however, aims to achieve another goal. Also referred to as Semantic Web, it wants to mark-up content in a standardized way to make it possible for machines to understand. This would make it possible for machines to process data without human interaction. When these developments come together with the recent improvements in sensor networks and near field communication technologies, this led to a new technology and area called *Internet of Things (IoT)* in which it is aimed to combine all together [1]. Moreover, IoT does not only connect computers and mobile phones, but also creates connections in a much larger scale. For instance, automobiles, buildings, cities and even electrical grids are connected to each other. Furthermore, like the physical objects, the virtual objects such as electronic tickets, books etc. also take place in this domain [2].

The term IoT was first proposed in 1998 by Weber as a global Internet-based information architecture for exchanging goods and services in supply chain networks [3]. This is a good use-case for the Internet, because it enables the provider to be alert in case of any lacking goods. This architecture is based on data communication, such as RFID-tagged items. Some popular industry specific proposals for IoT are physical objects with RFID tags [4]. But later, the definition of IoT in [1] has been broadened as *"a core concept where everyday objects can be equipped with identifying, sensing, networking and processing capabilities that will allow them to communicate with one another and with other devices and services over the Internet to achieve*

some useful objective." This new definition is well-suited since IoT has widen its range from supply chain networks to our everyday lives by finding applications in areas such as healthcare, smart infrastructure, social applications, etc.

It seems that IoT devices have been adapting well and their adaptation level is likely to be even higher in near future. As of 2021, the number of active IoT devices has reached to about 10 billion and it's estimated that this number will reach to 25 billion by 2030 [5]. This is because the areas to implement IoT is endless: wearables, health technologies, military applications, smart homes, smart grid, agriculture applications, automobile industry, and so on. Researchers even managed to implant IoT devices into human body in order to monitor organs [6].

To provide examples, a list of best (as of 2018) IoT and smart home gadgets chosen by ZDNet is given in Table 1.1 below [7].

Table 1.1 IoT and Smart home gadgets

Device Name	**What it does**
Ring	A smart doorbell, alerting you even if you are away from your home via mobile application if there is someone at your door.
Wemo	A home automation, which can control smart devices in your home such as light switches, plug outlets, etc.
Foobot	A smart air monitor which monitors air quality including carbon dioxide, temperature and humidity.
August Smart Lock	A smart door lock. It can give access to certain people, track any activity around it through a mobile application and also support traditional keys.
Rachio 3	A smart sprinkler system for gardens.
Nest Thermostat	A learning thermostat used to regulate the temperature. It can learn your preferences and act accordingly.
Nest Cam	A smart indoor security system.
RoboVac	A robot vacuum system. Uses infrared sensors to map the environment and cleans the house.

However, security for IoT devices is a challenge because of the complexity and resource consuming nature of cryptographic algorithms. Since IoT devices are generally composed of low-power and low-memory devices, security and privacy of data in IoT space is considered to be the biggest problem in IoT implementations [8]. Several studies claimed that tailor-made security measures are needed for IoT devices on top of standardization [9, 10]. However, although the number of standard bodies and security organizations increases, studies show that IoT devices deliver poorly implemented security solutions [11, 12]. IoT devices mostly provide a gateway to networks due to their low power capacity and low security. This allows an attacker to gain control of the network by compromising at least some part of this network. For instance in 2017, attackers [13,14] hacked a fish tank in a casino which had internet connection to regulate the temperature, food and cleanliness of the tank. It is stated that 10 GB of data were sent out to a remote computer in a different country. Attackers used the thermostat in the fish tank to find and steal a high-roller database of the casino. Another example is ransomware attacks. Ransomware attacks are frequent in both personal and business life and can be pretty expensive if the user is not taking frequent backups. Usually, a software is installed to a computer and encrypts everything in the device with a private key. Later, a screen pops up and asks for a certain amount of money if the user wants to see their data again. After the amount is paid, the user is given the decryption key. Lack of security standards in commercial IoT devices make these attacks possible. Attackers also designed ransomware specifically designed for IoT devices and referred to as IoT ransomware. Attackers try to control or lock a device and ask for payment to release control.

Our motivation is to find a complete or partial solution for this security issue and therefore to maintain a more secure network for IoT devices. Usually, IoT devices communicate to devices which are out of the network by using a central hub or gateway which can lead to single point of failure. In order to prevent this, we apply visual cryptographic techniques to create a method where every device in a closed IoT network is responsible for the other; by making them work together collectively.

This way, when a device wants to communicate with outer parties remaining or neighboring devices are used instead of the original one.

The rest of the thesis is organized as follows: Chapter 2 includes literature review about security issues on IoT; Chapter 3 includes preliminaries of IoT, cryptography, networking and graph theory, then Chapter 4 contains our model. Chapter 5 has implementation details of the proposed model and finally Chapter 6 summarizes the proposed solution and discusses the future work.

2. HISTORY AND RELATED WORKS

In order for IoT to reach its potential, a secure ecosystem is crucial. There are many security challenges for IoT domains which includes but not limited to: authentication, access control, confidentiality, privacy, trust, secure middleware, mobile security and policy enforcement [15]. There are many attacks done on consumer devices such as attacks on cars, botnets, etc. described in sections below [1, 3, 13, 14, 16–22].

Several detailed studies were done to identify generalized or specific security problems for IoT systems. Y. yang et al. [23] summarized security issues in IoT devices. Din et al. [24] observed several trust mechanisms for IoT such as E-LITHE, GTRS, etc. with their pros and cons. Chen et al. [25] studied security issues specific to location-based services in IoT. Ngu et al. [26] studied and compared several IoT middleware applications and discussed challenges on how to build an effective and secure IoT middleware application.

Security protocols of regular Internet is harder to apply on IoT devices due to device constraints. Trappe et al. [27] identified two main limitations: battery life and low-computation power. Energy requirements for high security calculations such as encryption is very hard to be realized in low-cost solutions.

Single point of failure attacks. Single point of failure is defined as a part of a system that, if fails, render the entire system useless. There are several examples in real-life applications where attackers use the single point of failure problem in order to compromise a network. Several researches stated that IoT devices could be used as an entry point to a network [16, 17]. They are a popular attacking point because by using a third-party or an extension, an attacker may gain control of the network without attacking the network itself, which is probably secured by several security measures. Below, we present several attacks done on IoT devices in subsections.

In an example, attackers found out that a smart bulb uses cloud services to interact with the mobile application instead of Wi-Fi or Bluetooth technologies. Later, they also found out that firmware updates were done with an unsecured communication via HTTP which makes the smart bulb vulnerable to MITM attacks. All these examples show that an IoT device connecting to the Internet without any security measures can create security issues for the users. Moreover, when a device is enabled on the Internet independently without any security measures, it causes a single point of failure where that individual device can affect the entire network in the negative.

In a similar example, researchers from Vectra Threat Labs [17] successfully established an access into a network by using a cheap consumer grade webcam. They reprogrammed the device so that it can serve as a network backdoor, while still operating as a camera. They have shown that cheap consumer devices can impact the attack surface of a network. This also provides a possibility of second-hand items to be used in organized crime and espionage.

SolarWinds, a major information technology company was hacked and infected with malicious code which was installed to every computer via an updating mechanism. Attackers used a method called *supply chain attack* where they insert malicious code into a third party system which has access to an organization's system, so this code made it possible for them to install more malware. By attacking and getting control of a simple extension called Orion, attackers accessed more than 18000 customers and retrieved important information whose value is unable to be calculated. It is important to state that some of the infected are government agencies.

In the recent years, automobiles became highly electrical devices. Although this improves efficiency and safety, it also comes with potential risks. Several studies have evaluated these risks and shown how the cars and drivers are vulnerable to different attacks [18, 28–30].

Modern cars are controlled by a combination of digital components called Electronic Control Units (ECUs) where ECUs are interconnected by internal wired networks

such as CAN and FlexRay buses and they control a broad range of functionality in a car [18]. Although this connection provides additional features, it also provides a broad internal attack surface. Since the internal network (such as CAN bus) connects these device, a single compromised device can offer an effective vector for other components [18]. These are some other good examples of how single point of failure is a serious problem. Moreover, Rouf et al. [19] studied the tire pressure monitoring system (TPMS) which is the first wireless network installed on almost every vehicle. It uses radio frequency (RF) to transmit the data to pressure control unit, which analyzes this data and sends it to the central computer via CAN bus. After a careful investigation, they observed that the central computer trusts the data coming from the TPMS without any authentication. By reverse engineering the process, they were able to disable TPMS, inject spoofed messages to cars and finally activate warning lights on a car.

Koscher et al. [20] studied whether an automobile could be accessed remotely. They showed that it was possible via a broad range of attack vectors varying from CD players and radios to wireless communications; such that it was even possible to remotely track and control the car. They came up with four vulnerability classes: direct physical, indirect physical, short-range wireless and long-range wireless. In the short-range wireless class, they focused on the most general channel: Bluetooth. After thorough investigation, they found a custom code in the telematics ECU and several non-safe function calls. By using a paired mobile device, this flaw can be exploited and the attacker can take control of the ECU. In addition, the attackers made use of the cellular capabilities of the car. Many cars have cellular connection for several use cases such as crash detection and emergency situations. This flaw is such powerful that the attackers could use an iPod with a specially encoded audio file to compromise the car.

There are also some studies done on vehicular ad-hoc networks (VANETs), which are introduced to provide efficient communication between different vehicles. They are used in enhancing traffic safety and reducing congestion [31]. VANETs comprises

of the main elements: trust authority (TA), road unit (RSU) and on-board unit (OSU) [32]. Vehicles are constantly exchanging information between themselves and the RSU. The way these parties communicate to each other is very crucial for security, since it can be attacked and used for damaging purposes. Jamming attacks can also be used to overload the medium and therefore prevent channels from sending or receiving information [33]; timing attacks can be used to delay crucial information being sent to vehicles [34]; hidden vehicle attacks are used to send false position information to neighbor nodes and can forward false messages to vehicles [35, 36]; and many more such as replay attacks, impersonation attacks, wormhole attacks, man in the middle attacks, etc. [32].

Another security issue is about commercial products where the user can easily connect a device on the Internet. However, how this device connects to internet is out of user's control. Moreover, the user is not aware of whether the device is using a secure channel for connection. They also do not have a secure password policy forcing the users to maintain security. Usually, these devices cause issues related to security and privacy. For instance, an attacker may connect to a webcam at your home freely and this may help burglars. Up until now, there have been several botnet attacks in which a huge number of computers were compromised. For instance, Mariposa, Conficker and BredoLab are such botnets in which approximately 10 million computers were infected [1, 3, 22]. One of the most important botnet examples is the Mirai botnet which infects the IoT devices. Let's point out that Mirai botnet is not the first example which uses IoT devices as zombies, as BASHLITE [37] and Carna [21] have such ability to infect the IoT devices. However, Mirai infects a huge number of IoT devices rapidly. In an example, Mirai botnet infected nearly 65,000 IoT devices in the first 20 hours, and eventually infected a total number of 600,000 devices. An analysis shows that Mirai botnet primarily and mostly infected IP cameras, DVRs and consumer routers as well as NAS devices, printers and TV receivers manufactured by different vendors [38].

There are numerous attempts to check for vulnerabilities in IoT devices. One of the

well-known vulnerability was found in an SDK (software development kit) named *ThroughTek Kalay*. It is a plug-and-play system connecting the device to the corresponding mobile application [39]. Researchers found the flaw in the registration phase happening between devices and the corresponding mobile applications. By using this flaw, attacker is able to watch a video feed in real time, insert malicious code in the firmware, shut down devices via denial of service attacks, which could create a very serious safety and privacy concern.

Ronen and Shamir [40] attacked two commercial connected lighting systems, Philips Lux and LimitlessLED in order to achieve a different effect rather than the original functionality of those devices, which is to control the color and intensity of lights. They successfully completed two attacks: First, they extracted data from a secure location by creating a covert LIFI communication system by using smart lights; and were able to read leaked data over 100 meters using available products on market. Second, they showed that an attacker is able to strobe lights at a frequency which may trigger seizures in people with photosensitive epilepsy.

FLocker [41] was first introduced in 2015 targetting mobile Android devices. Later, it is discovered that there is a new variant attacking smart TVs. There are thousands of different variants of the ransomware and the latest one was pretending to be a law enforcement agency, falsely accusing users of crimes and demanding iTunes gift cards worth of 200 US dollars. First, it hides itself as a normal file in order to escape from static code analysis. When it runs, it decrypts that file and executes the code. After a certain amount of time, it asks for admin privileges and if not given, it fakes a system update. After having the admin privileges, it downloads an APK and an HTML file including the ransom page. During this time, system is collecting information in the background such as device information, phone number, contacts, location, etc. and sending it to a remote server. However, this ransomware does not encrypt files but sends information in the TV to a remote location. The ransom is paid to be able to use the TV again. It is possible to fix the computer with a PC and Android developer tools. However, it is very hard for a non-technical person to

deal with this problem by themselves.

Tierney [42] created a fully functional ransomware and took control of a thermostat by local attack. When they were reverse engineering, they saw that every functionality of the device was included in a single directory called root. After investigation, they found out that the code has potential for command injection. From there on, they can implement a ransomware. However, there are also additional problems with such an attack. They were also able to make use of the device and control the buzzer frequency and heating & cooling systems. They concluded that simple security protocols can stop these kind of attacks such as encrypting the firmware or signing it so it cannot be modified.

Margaritelli [43] was able to reverse engineer a mobile application developed to control a smart coffee machine. By learning the commands, he created a terminal application to control the device. However, while pursuing the goal; he observed that although mobile application needs a user to be registered; it was not for authentication purposes but rather analytics. Since there were no real authentication, anyone in that network were able to reach that port and were even able to flash a new firmware.

In near future, standardization for IoT especially towards security aspects will be very important, because the plan is to interconnect several IoT devices from different vendors. For that, there should be standards which vendors are to agree on. However, in the recent years efforts being poured into standardization follow market aspects rather than security measures [44]. It must be stated that standardization in itself is not enough if they are unable to provide solutions for security and privacy concerns for end users. Although there are several international organizations and alliances working on standardization of IoT such as (IEEE, IETF, ITU-T, etc.), there are still open areas to debate and standardize [45, 46].

Examples above show that a compromised device on a network can be used for malicious purposes and create problems. For a network to function well, there

must be trust between participants. This trust can be achieved by authentication. Authentication is defined as the process of confirming and insuring the identity of participants. Although it is a key requirement for IoT, traditional authentication schemes are hard to apply in IoT domain due to constrained nature of these devices [47].

3. PRELIMINARIES

Before going into detail about our proposed model, first we need to define key information about certain topics. We start with graph theory, define what it is and its use in network topologies. Next, we discuss about computer networks, their types and how IoT adapts to networks. Then, we go into detail about how IoT systems work and the existing security issues with IoT. Lastly, we discuss the basics of cryptography, certain cryptographic protocols and visual cryptography.

It is important to note that this chapter do not cover all materials about the topics but focuses mainly on certain concepts to make it easier to understand the rest of the thesis.

3.1 Graph Theory

A graph $G = (V, E)$ is an ordered pair of disjoint sets $V(G)$ and $E(G)$ such that $E(G)$ is a subset of the set $V \times V$. This means that the elements of E are 2-element subsets from $V \times V$. The elements of V are called *vertices* and the elements of E are called *edges*.

An edge $(x, y) \in V \times V$ is usually written as xy and vertices x and y are called *neighbors*. The number of vertices (edges) in a graph G is called the order (size) of G and denoted by $|G|$ ($||G||$). Given a vertex v, the degree of v, denoted by $d_G(v)$, is the number of its neighbors. If every vertex of a graph has the same degree, that graph is called a *regular graph*. Regular graphs can also be referred to as *n-regular graphs*, where n is the degree of each vertex.

A path is a non-empty sequence of distinct vertices in a graph G in which the consecutive vertices are neighbors in G. The number of edges is called the length of

the path. A path with the ending vertex neighboring the initial vertex is called a *cycle*. A cycle of length n is called an n-cycle and denoted by C^n.

A non-empty graph G is called *connected* if there exists a path which connects any two vertices of G. Moreover, a graph with no cycles is called an *acyclic graph* and a connected acyclic graph is called a *tree*. Given a connected graph G, if any edge of a cycle is removed, the resulting graph is still connected. If this procedure is repeated until there are no cycles left, the remaining graph becomes a tree that connects all the vertices of G. This tree is called a *spanning tree*. Figure 3.1 provides an example of a graph and one of the spanning tree of that graph.

Figure 3.1 An example of a graph and its spanning tree

A regular graph is a graph in which every vertex has the same degree. For instance, 3-regular graphs are graphs such that every vertex has degree three. These graphs are also referred to as *cubic or trivalent graphs*. Figure 3.2 presents examples of 2-regular (cycle) and a 3-regular graph.

3.2 Computer Networks

A computer network is a set of computers connected to each other in order to share resources. For computer networks, there are several use cases such as business applications, home applications, mobile users and social issues [48]. In business applications, the main objective is resource sharing. Data and equipment can be used among several parties, even without needing close proximity to each other via Virtual Private Networks (VPN). In home and mobile uses, connectivity to internet is crucial. At home, by using peer to peer connection, users are able to receive and

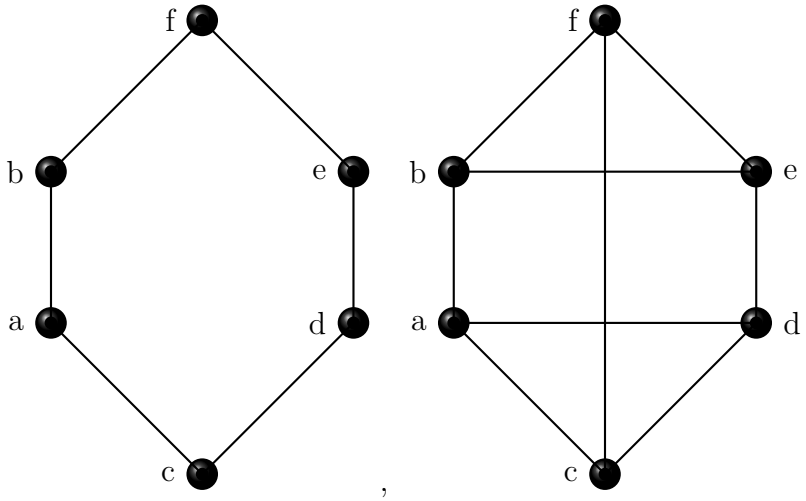


Figure 3.2 2-regular and 3-regular graphs

share information to remote computers which help for entertainment and personal use. In addition, mobile use can be widened into wireless sensor networks (WSN) and GPS connections as well. Social issues on networks are several: privacy and security of personal data, fake news, etc.

There are two types of transmission media in general: *broadcast* and *point-to-point*. In broadcast networks, communication channel is shared within the network. A packet sent by any device can be received by any other device in the network. The receiver can be identified by using a field in the packet. *Wireless networks* are examples of broadcast networks. However, in point-to-point networks, pairs of devices are connected to each other. Blockchain and bittorrent are examples of peer-to-peer networks. Figure 3.3 shows an example for broadcast and point-to-point networks.

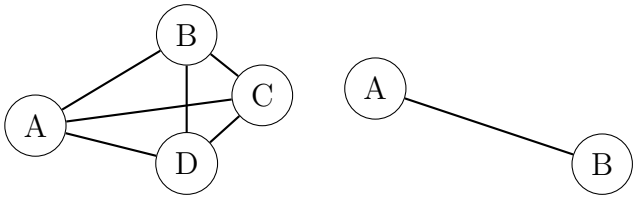


Figure 3.3 Broadcast and point-to-point networks

3.2.1 OSI and TCP/IP Reference Model

Networks are organized as a stack of layers (levels) in order to reduce complexity. This architecture helps simplify the complex structure of systems. Each layer implements some services for upper layers. Number, name and function of these layers may differ from network to network. Every layer has a functionality, a service to be offered to higher layers. Layering approach also limits how much the other layers know about the internal functioning. This way, upper or lower layers only knows the service and not the internal process. This also provides modularity. If a layer changes or updates its implementation of a service, it can be done without effecting the whole network (other layers).

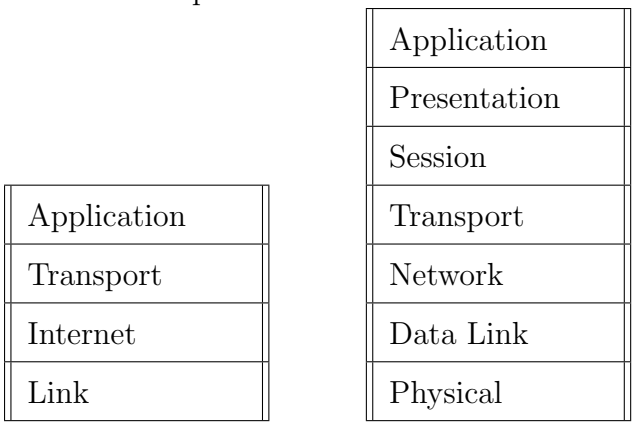
Parties in a network communicate by following some rules and conventions which are called as a *protocol*. Each layer has its own protocol and taking them together comprises a *protocol stack*.

OSI reference model was developed first to address the lack of standardization and officially adapted as a standard in 1979 [49]. It consists of seven layers: *application, presentation, session, transport, network, data link and physical* layers as shown in Figure 3.4. Although Internet uses TCP/IP reference model, OSI reference model is still valid and its described features are important, therefore it is still widely used for educational purposes.

TCP/IP reference model (also known as Internet protocol suite) is based on primary Internet protocols: *IP* and *TCP*. It consists of four layers: *link*, *internet*, *transport* and *application*. It is proposed as a model with fewer defined layers providing an easier fit for real-world protocols. Link layer concerns itself with secure transmission of data packets and it corresponds to first two layers of OSI model. Internet layer is responsible for moving network-layer packets from one host to another. These packets are also referred to as *datagrams*. Transport layer is concerned with reliable exchange of connection-oriented data between different computers in a network. Lastly, application layer serves as an interface where network applications and their

application-layer protocols are stored [50] [51].

Figure 3.4 Internet protocol stack and OSI reference model



The difference between two models can be seen in Figure 3.4. Link layer combines Physical and Data Link layers, and in addition the TCP/IP reference model lacks two layers: *presentation* and *session*. This does not mean that these layers are less important, however it is up to the application developer to build such functionality [50].

Table 3.1 provides several examples for each layer of this reference model.

Table 3.1 Layers of TCP/IP models and examples

Application Layer	DNS, FTP, HTTP, ...
Transport Layer	TCP, UDP, SCTP, ...
Internet Layer	IP (IPv4, IPv6), IPSec, IGMP, ...
Link Layer	PPP, MAC, Ethernet, ...

3.2.2 Computer Network Topologies

A network topology defines how elements of a network are arranged and organized. Several common topologies in computer networks are as follow:

Bus Topology. As shown in Figure 3.5 each node in the network is connected to a single central cable, which is also referred to as **bus**. All data is transmitted through the bus and can be received by any node simultaneously [52]. It should be

kept in mind that since the network depends on a single cable, it is possible that single point of failure can occur.

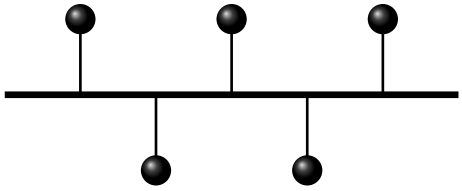


Figure 3.5 Bus topology

Ring Topology. In ring topology, devices in the network are connected to each other in a closed loop. No hierarchical relationship exists between nodes. Figure 3.6 is an example of a ring topology.

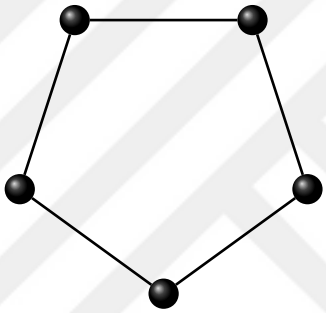


Figure 3.6 Ring topology

Star Topology. Figure 3.7 gives an example of a star topology, in which every peripheral node in the network is connected to a central node. All traffic is controlled through the central node [53].

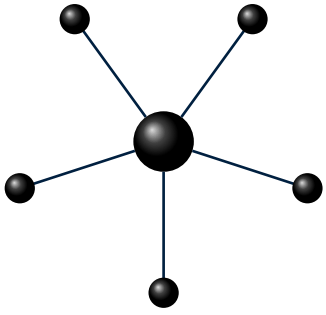


Figure 3.7 Star topology

Mesh Topology. Mesh topology can be implemented in two ways: full mesh and partial mesh [53]. In full mesh, every device in the network is connected to each other. In the latter, some of them are connected but not all. A device in the

network can also act as simple sensor nodes which can route traffic. Figure 3.8 gives an example for both types.

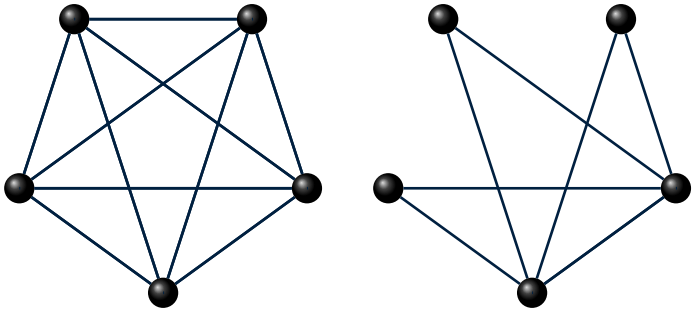


Figure 3.8 Full and partial mesh topology

3.2.3 Graphs as Network Topologies

A topology is defined by a graph $G = (V, E)$ where V represents the set of vertices and E represents the set of edges. Vertices represent devices in a network and edges are used to show the communications.

Table 3.2 Comparison between simple static networks

Name	**Advantage**	**Disadvantage**
Cliques	Nodes are close to each other	Too many edges
Star	Efficient for communication	Weak when central nodes are not available
Rings	Allows to implement pipeline algorithms	Large communication costs
Meshes	Good with image domains and parallel programming	Not regular. Needs special attention for border nodes
Torus	Eases parallel programming	Extra wires can make physical design harder

There are several ways to design a communication architecture. One can either design an architecture where all devices share a common medium such as a *bus* or a *point to point network* where all devices are connected to each other directly. Both have advantages and disadvantages, and the choice depends on the network.

Nielsen states that for a good topology, a trade-off needs to be set between two views: 1. Minimizing the number of links and therefore decreasing the material cost; 2. Maximizing the number of communication links to decrease the communication cost [54].

Networks can be considered in two types: *static* and *dynamic*. Static networks are fixed and they cannot be changed while in operation. Dynamic networks can be adapted to the needs of applications and changing traffic conditions. There are several types of graphs which provide simple static networks that are common topologies: clique, star, ring, grids, torii, cube, etc. Table 3.2 provides a comparison between simple static networks.

3.3 IoT

IoT devices are generally considered in two classes as microcontroller-based devices and microprocessor-based devices. Microprocessor-based devices have decent processing power, memory and connectivity. Some well-known examples for such devices are Beaglebone and Odroid [55]. On the other side, microcontroller-based devices consist of an Integrated Circuit (IC) housing several components such as processing unit, memory and I/O peripherals. These devices are mostly inexpensive due to their limited functionality. A well-known example for this category is Arduino MKR1000 [55].

In this chapter, we discuss IoT networking preliminaries, such as protocols for each layer in TCP/IP model. Later, we explain security, performance and memory capabilities of IoT devices.

Star and mesh topologies are the most commonly used topologies in the IoT domain [56]. Table 3.3 shows some protocols in IoT domain mapped to the TCP/IP reference model. Some of these protocols are going to be explained in this section.

Table 3.3 IoT network protocols and corresponding layers in TCP/IP model

TCP/IP model	IoT Protocols
Application	HTTPS, XMPP, CoAP, MQTT, ...
Transport	TCP, UDP
Internet	IPv6, IPv4; 6LoWPAN, ...
Link	Ethernet, GSM, LTE, ...

3.3.1 Link Layer Protocols

Link layer, also referred to as sensing layer deals with IoT sensors and actuators. While sensors capture physical phenomenon around them, actuators take action on the environment based on the sensed data.

According to Lee et. al. [57], three of the mostly used short-range wireless protocols are ZigBee, Bluetooth, and Wi-Fi that are corresponding to IEEE standards 802.15.1, 802.15.4 and 802.11a/b/g. In the following, we provide a brief description for each these protocols. See Ferro and Potorti [58], Wang et al. [59] and Baker [60] for further details and comparison.

ZigBee. ZigBee is designed for low data-rate and battery-powered applications such as building and home automation applications. It is the de-facto networking standard on the market. It is built on 802.15.4 standard and not IP-based, making it harder to work with Internet Protocol standards such as Wi-Fi, Ethernet, or LTE. A big advantage of ZigBee is that nodes can stay in sleep mode and therefore maintain an improved battery life. Moreover, it is a mesh network protocol and an open standard which operates on 2.4 GHz wireless communication spectrum.

Bluetooth and Bluetooth Low Energy. Bluetooth has been developed with the intention of replacing devices using wired communication at homes and offices in a 10-meter radius. Although it has lower range and transmission speed compared to Wi-Fi, it uses less power and has lower implementation costs. Being low-powered

also helps not causing interference with other wireless devices in the same radio band. BLE (Bluetooth Low Energy) is a low-power version of Bluetooth technology. It is designed for ranges below 100 meters, usually with a single device controlling other devices. It is best suited for devices which are designed to sleep while idle such as wearable health devices [61].

Wi-Fi (IEEE 802.11 - 802.11ah - 802.11ax). Wi-Fi is a wireless network technology which allows devices to communicate with each other. It is based on 802.11 IEEE network standard which uses radio frequency signals to transmit data. Although it provides the highest data throughput, it also has a higher power consumption. Currently, there exist IoT devices which adopt Wi-Fi technology well, however, there are emerging technologies which may have potential to reduce the number of IoT devices using Wi-Fi in near future [61]. This is mainly because 802.11 is a costly protocol with high overhead and power consumption. As a result, a lightweight version 802.11ah is designed to meet with the needs of IoT [62]. Moreover, 802.11ax, also called as Wifi-6, is the most recent version of IEEE 802.11 WLAN standard. It aims to improve its throughput and optimize its performance in large outdoor environments while decreasing power consumption. It also adds new features such as OFDMA, Multi-User MIMO, and spatial reuse [63].

Other than the technologies introduced above, there are also emerging new technologies such as LoRaWAN. It is a long-range wide-area network wireless technology designed for IoT applications. It also offers low-power and low-data rate communication. Unlike other LPWAN technologies such as SigFox and NB-IoT, LoRaWAN enables private network deployments. This and the ability of quick implementation move LoRaWAN further as a candidate for IoT adoption [64].

There is also RFID (Radio Frequency Identification) technology available which is using RFID tags which store identifiers and data for devices or goods, which can be read with RFID reader. The effective range is shorter than 1 meter. A good use case for RFID tags is tracking inventory in retail and industrial IoT applications [61].

Another example of communication protocols with a short range is NFC (Near Field Communication) which has an effective range of just 20 cm. It is available in many mobile devices and has a very high potential in some applications.

In literature, there are several studies comparing these technologies. For instance, Ferro and Potorti [58] compare Bluetooth and Wi-Fi in terms of capacity, network topology, security, QoS support and power consumption. Wang et al [59] compare the MAC of IEEE standard 802.15.3 and 802.11e and show that throughput differences between these standards are very small; and power management of 802.15.3 is easier than 802.11e. Moreover, Baker [60] compares ZigBee and Bluetooth in terms of their strengths and weaknesses for industrial applications. He concludes his study by claiming that Zigbee would be a better choice for IoT in industry because of its long-term battery power and greater range. In practice, the most widely used standards are Zigbee and Bluetooth.

3.3.2 Internet Layer Protocols

Internet layer transmits the information from the link layer to the upper layer.

On the Internet, IPv4 still is the de-facto standard even though IPv6 has been deployed. It uses 32 bits as an identifier, therefore it has potential to provide with 2^{32} addresses. However, IPv6 uses 128 bits identifier with an ability to provide with 2^{128} addresses compared to its predecessor IPv4. In this sense, IPv6 has emerged as an improved technology in order to provide more IP addresses for devices planned to emerge in the future. Although IPv6 is an optimal protocol for resource-rich networking scenarios, it is not well-suited for IoT networks which are limited compared to others. In order to use IPv6 over these constrained networks, implementation of an adaptation layer between the IPv6 and target technology is required [65].

Standards below are trying to encapsulate IPv6 datagrams into small MAC frames for IoT. These standards are important to realize the challenge of working with different networking stack layers [8].

6LoWPAN. The term 6LoWPAN is produced by combining IPv6 and LoWPAN (Low-power Wireless Personal Area Network). It is a wireless mesh network where every node has its own IPv6 address, so it makes small devices with low power join into the IoT domain. This is achieved by making it possible to communicate with 802.15.4 devices on an IP network such as Wi-Fi. PL stands for IPv6 Routing Protocol for Low-Power and Lossy Networks. It is designed for routing IPv6 packets within constrained networks where not all devices are always reachable such as those implemented on 6LoWPAN [61]. It can compute the optimal routing path by building a graph of nodes, and therefore minimizes power consumption and latency.

6Lo. The communication technologies for IoT have been growing every year. That's why, a new combination was created after successful completion of 6LoWPAN. It is called 6Lo, which leverages 6LowPAN and was basically designed for enabling IPv6 over IEEE 802.15.4 networks. With 6Lo, IPv6 is enabled over Bluetooth LE, ITU-T, G.9959, DECT ULE, MS/TP, NFC, IEEE 1091.2 and IEEE 802.11ah [65].

3.3.3 Transport Layer Protocols

There are several messaging protocols used in IoT, however in order for them to be secure, they need encryption. Described below are two de-facto standards for each transport layer protocol, namely each of TCP or UDP.

TLS. TLS is an encryption protocol running on top of TCP, and it aims to provide privacy and data integrity between two communicating applications. TLS consists of two sub-protocols: TLS Record and TLS Handshake [66] [67]. TLS Record protocol is used for encapsulation of other higher-level protocols such as TLS Handshake protocol. It decides how to divide the data, encrypt and later package it into records, so that it can be decrypted by the receiving party [67]. It operates at the presentation layer [67]. This protocol provides a connection security consisting of two properties: Privacy and reliability. The connection is encrypted by a symmetric cryptographic

algorithm with uniquely generated keys. It includes a message integrity check by using a keyed MAC in the message transport and thus reliable. MAC computations are done with secure hash functions such as SHA-1 [66]. TLS Handshake protocol allows the server and client to authenticate each other and negotiate the encryption algorithm and cryptographic keys before any transmission takes place [66]. It operates at the session layer [67]. TLS Handshake protocol provides a connection security consisting of three basic properties: (i) The negotiation of the secret itself is secure, (ii) It is unavailable to eavesdroppers and (iii) the secret cannot be obtained even with a man in the middle attack. The negotiation is reliable as it cannot be modified by any attacker without being detected by the communicating parties. Lastly, the peer's identity can be authenticated by an asymmetric cryptographic method. Although this authentication is optional, it is required for at least one of the peers [66]. As mentioned above, TLS works on TCP, a connection-oriented and reliable protocol. However, for the last few years the number of application layer protocols using UDP transport has increased [68]. Although it is easy to use TLS between application and transport layers to ensure security, it can only be used on a secure channel such as TCP. Therefore, it cannot be used for the datagram traffic [68]. For this reason, a variant of TLS which can be used in unreliable channels is created, namely DTLS. It is created by making some minor changes on TLS and it makes it possible to work under unreliable conditions.

UDP. It is a protocol which is used by application programs to communicate with other programs without a minimum protocol mechanism [40]. It is not a reliable protocol like TCP [69]. It is mainly used in applications where data is delay-sensitive, such as media streaming, Internet telephone and online gaming [68]. Using DTLS protocol to secure communication does not change the behavior of such applications, because DTLS protocol does not reorder or resend lost data.

Datagram protocols are not reliable because there is no procedure for reordering the data or making sure that data are transferred correctly. Since TLS has no internal mechanism to deal with these circumstances, it can only be used in reliable channels

such as TCP. Therefore, a new protocol is needed, which can be simply stated as *TLS over datagram* [68]. Unreliability in the channel creates problems for TLS in two ways: (i) TLS does not allow independent decryption of individual records. But, for TLS to decrypt a record, it must be ordered (ii) TLS handshake layer breaks if handshake messages are lost. Therefore, the handshake messages must be delivered to the other side for TLS to work. However, in an unreliable channel, this creates a problem [68]. DTLS does provide reliability in an unreliable channel by using a retransmission timer to handle *packet loss* [68]. DTLS also solves the sequence problem by assigning a sequence number to each handshake message and therefore determining if the incoming message is the expected one. If the order is broken, it uses a buffer to hold these messages and handle them after receiving all previous messages.

3.3.4 Application Layer Protocols

In this section, we introduce some of the well-known protocols such as MQTT, SMQTT, AMQP, CoAP, HTTP and XMPP.

MQTT. MQTT (Message Queue Telemetry Transport) is a publish/subscribe, lightweight messaging protocol designed for constrained and low-bandwidth, high-latency devices. It was invented in 1999 by IBM and became an OASIS standard as of 2014 [70]. It aims to minimize network bandwidth and resource requirements of the device while ensuring reliability and a degree of insurance for delivery [70]. The architecture of MQTT provides an open and easy implementation, therefore it can support up to thousands of remote clients by using a single server. MQTT keeps message headers as small as possible (2 bytes), it is royalty-free and has built-in support for losses in connection between clients and the server. These properties make MQTT a very good candidate to be used in constrained environments such as M2M and IoT which contain devices with limited power, battery life, and low bandwidth [71].

MQTT protocol has three components: *publishers*, *subscribers* and a *broker*. Publishers can be defined as sensors or IoT devices which tend to send data. Subscribers can be defined as applications which are subscribed to the broker to receive information. It works by publishers sending data to the broker, broker having a queue itself and then sending the data to subscribers. MQTT works over TCP port number 1883 and uses TLS for security. Publishers and subscribers do not know each other's identities.

SMQTT. SMQTT (Secure MQTT) is an extension to the original MQTT protocol, which aims to improve the security of MQTT by implementing a lightweight, attribute-based encryption (ABE). By using this encryption method, the message is secure and can only be read by those who have the necessary key to decrypt the message. MQTT can be implemented in a secure way by using TLS, however this creates additional problems, especially for IoT devices. With the increasing number of devices, storing/managing certificates and key exchanges would be hard [72]. However, SMQTT uses an attribute-based encryption, because the design makes it possible to support broadcast encryption, which is very crucial for M2M communication [72]. The protocol works in four stages: set-up, encryption, publishing, and decryption. Moreover, it has the same components as MQTT. First, subscribers and publishers register themselves to the broker and receive a master secret key. This master secret key depends on the developers' choices of key generation algorithm. Later, the broker encrypts data and publishes it. This encrypted message is received by the subscribers and can be decrypted with the same key.

AMQP. AMQP (Advanced Message Queuing Protocol) is an open standard for passing messages between applications and organizations. It is specifically designed for business enterprises as an alternative to proprietary protocols. It can be defined as a reliable, secure corporate messaging protocol [73]. Moreover, it is a binary protocol designed to support a wide variety of different messaging applications [74]. Unlike MQTT, where the message-broker forwards incoming messages directly to the subscribers; in AMQP the message-broker accepts incoming messages from a

producer in an exchange; and routes messages to a specific queue based on predefined rules. It works on the reliable TCP and uses TLS and SASL for encryption and authentication [75].

CoAP. CoAP (Constrained Application Protocol) is a session layer protocol designed for M2M applications to use with constrained nodes and networks [76]. It aims to keep message overhead small in order to limit the need for fragmentation which causes significant reduction in packet delivery in constrained networks while using REST architecture. Therefore, CoAP aims to realize an optimized subset of REST to be a generic web protocol for constrained environments. It works on datagram protocol UDP for transport and uses DTLS protocol for security. CoAP has four types of messages: Confirmable, Non-confirmable, Acknowledgement and Reset. It provides reliability by marking a message as Confirmable (CON). This message is retransmitted until the recipient sends back an Acknowledgement (ACK) message with the same Message ID, using a default timeout and exponential back-off between retransmissions. If the recipient cannot process a CON message, it replies with a Reset (RST) message instead of ACK. If a more lightweight alternative is requested, the messages can be marked as non-confirmable. However, this will decrease reliability of transmitted messages [76].

HTTP. Although HTTP (Hypertext Transport Protocol) is mostly used as a web messaging protocol, it is an application-level protocol for distributed, collaborative and hypermedia information systems. HTTP can be implemented on top of any protocol assuming that protocol is reliable. Therefore, it usually takes place over TCP/IP protocol and uses port number 80 [77]. The communication between clients and the server is connection-oriented, and like CoAP; client and server send and receive data through URI [73].

XMPP. XMPP, Extensible Messaging and Presence Protocol is an IETF standardized protocol based on XML and is widely used over the internet for chatting and message exchanging [78]. Since it was designed for real-time communications,

it has low latency and looks suitable for IoT communications. However, it is based on XML and creates additional overhead due to heavy use of tags. For this reason, it is not widely adopted in IoT systems [8].

3.3.5 Processing Capabilities

Type and power of processors used in IoT devices depend mainly on the design aim. For example, high processing power may be needed for devices with multimedia-rich use. However, a simple sensing unit does not need that much processing power and it would be unnecessarily expensive to use such a device for that purpose.

Since many devices have default needs such as wireless connectivity, multiple communication standards such as Wi-Fi, Bluetooth and Cellular are integrated into a system of a chip (SOC). This decreases overall cost of materials and provide better battery life [79].

IoT devices are divided into 5 parts: smart sensors, connected audio, connected video, multimedia-rich devices, and high density computer nodes [79].

Smart sensors are microcontrollers with analog interfaces for sensing. They use energy-efficient standards (Bluetooth Smart, Low-power Wi-Fi, etc.) and have CPU requirements between 50-100 DMIPS. Connected audio and video devices are ranged from bluetooth speakers to high-end home cinema systems. CPU requirements range between 300-1000 DMIPS. They generally use Bluetooth and Wi-Fi for wireless communication and connected video devices are used for recording and streaming video such as IP cameras. Internal architecture is very similar to connected audio devices.

Moreover, multimedia-rich devices and high-density computer nodes are considered as high performance computing devices. They can integrate a multicore CPU and multicore GPU depending on the application of the device. Using a smartly configured SoC reduces costs.

3.3.6 Memory Capabilities

For memory, there are several options for IoT developers to choose from: traditional external flash memory, embedded flash memory, multichip package memory and multimedia cards (MMC) [80]. Each of these options is suitable for certain applications.

Traditional external flash memory is widely used in consumer products mainly because of its reliable and low cost nature. There are two types of external flash memory: NAND and NOR flash. NAND flash is widely used in data heavy applications which require high-capacity storage, such as wearable devices; where NOR flash generally finds use in GPS or e-readers with less memory requirements [80]. Also known as eFlash, embedded flash memory is used in IoT devices with critical data and code. Compatibility with microcontrollers due to its high performance makes it a popular memory type. Multichip-package memory (MCP) on the other hand provides more by implementing CPU, GPU, memory and flash storage in a single chip. Embedded multimedia cards (eMMC) contains a controller and flash memory. This controller is among exclusively designed controllers in order to better integrate into application systems. It can process multiple tasks and have a very good performance at reasonable cost [55].

Internet Engineering Task Force (IETF) classified IoT devices into three parts: *low-end*, *middle-end* and *high-end* [55]. Low-end devices are generally used in basic sensing and actuating powers. They are resource challenged and therefore referred to as constrained devices [81]. Since middle-end devices are more powerful, they are more capable than low-end devices. High-end devices on the other hand are mostly single board computers, some of them even capable of storing a GPU to run operating systems [55]. Table 3.4 shows several examples for low-end, middle-end and high-end devices.

Ojo. et al compare several low-end, middle-end and high-end devices thoroughly [55]. Table 3.5 shows the comparison of the memory capabilities of such devices. High-

Table 3.4 Some examples for different IoT device categories

Category	**Examples**
Low-end	Waspmote, Tmote Sky, TelosB
Middle-end	Carambola 2, Tessel 2, Netduino
High-end	Raspberry Pi, PcDuino, BeagleBoard

end devices are not included because many of them uses micro-SD cards as on-board storage and therefore their maximum memory capacity depends on the capacity of micro-SD card. In summary, out of 19 low-end devices, the average memory capability is found to be 262 KB and out of 9 middle-end devices, the average memory capability is found to be 12 MB which is 50 times more compared to low-end average.

Table 3.5 Average memory capabilities of IoT devices

Type	**min**	**max**	**average**
Low-end	48 KB	512 KB	262 KB
Middle-end	1 MB	32 MB	12MB

3.3.7 Security problems in IoT and several solutions

Andrea et al. [82] classified IoT attacks into four main parts, namely: Physical attacks, Network attacks, Software attacks and Encryption attacks.

(i) **Physical attacks**: Node tampering and jamming, RF interference, malicious node and code injection, physical damage, social engineering, sleep deprivation attacks, side channel attacks and booting attacks

(ii) **Network attacks**: Traffic analysis attacks, RFID spoofing and cloning, RFID unauthorized access, sinkhole attack, MITM attack, DoS/DDoS attacks, routing attacks, sybil attack, data transit attacks

(iii) **Software attacks**: Virus, malware, spyware, trojan horse, malicious scripts, DoS, phishing, updates and patches

(iv) **Encryption attacks**: Side channel attacks, cryptanalysis attacks, MITM Attacks

Ronen and Shamir observed that most of the published studies on IoT attacks follows four broad types of behaviour: ignoring the functionality, reducing the functionality, misusing the functionality and extending the functionality [40].

(i) **Ignoring the functionality**: Attacker ignores the standard functionality of the device and just behaves as if the device is a computing device with internet connection. This type of attack doesn't care about the devices' functionality.
(ii) **Reducing the functionality**: Attacker tries to minimize or completely shut down the device. A smart tv may be shut down on purpose. However, in safety critical areas this can be a fatal problem.
(iii) **Misusing the functionality**: Instead of shutting down the device, attacker uses the device for other purposes. An air conditioner can be hacked and can be adjusted to give heat instead of cool in summer.
(iv) **Extending the functionality**: Attacker tries to get a broader and unexpected functionality rather than the original.

Another study by Hassia et al. [83] provides four categories of solutions for securing IoT environments which are mainly blockchain based, fog computing based, machine learning based and edge computing based.

(i) **Blockchain based** approaches provides solutions for privacy [84–86], scalability [87, 88], data loss and spoofing [89].
(ii) **Fog computing based** solutions focus on man in the middle attacks, data transit attacks, eavesdropping, resource-constraint issues and incident response services [83].
(iii) **Machine learning based** solutions are for DoS attack [90, 91], eavesdropping [92, 93], spoofing [94–96], privacy leakage [97, 98] and digital fingerprinting [99].
(iv) **Edge computing based** solutions focus on data breaches [100], data compliance issues [101], safety issues [83] and bandwidth issues [102].

However, it should be kept in mind that applying these solutions may create addi-

tional security issues as well. It is also suggested that for maintaining a secure IoT environment it requires to have: rigorous penetration testing, end-to-end encryption, authorization, scalability testing, secure transmission and cloud encryption.

3.4 Cryptography

Rivets defines cryptography as a practice and study of techniques to communicate in a secure way in the face of third parties [103]. It is about creating secure protocols such that any unwanted parties cannot read or alter the information being passed between original parties. A cryptosystem is a general definition given to a set of cryptographic primitives used to provide information security services.

In cryptography, a message is secured by encryption. This process is succeeded by using a key and is similar to the process of locking doors in real-life. If one uses the same key to encrypt and decrypt a message, this is called *symmetric cryptography* or *single-key cryptography*. However, if there is a pair of keys, where one key is shared publicly for encryption and the other for decryption, that is called *asymmetric cryptography* or *public-key cryptography*.

Let's say two parties, Alice and Bob, want to set up a secure communication channel by using symmetric cryptography. Then, firstly, they need to agree on a key k. If Alice wants to send her message (plaintext) m to Bob, she then encrypts this plaintext by using an encryption algorithm E with the key k. As the output of the encryption process, she obtains the corresponding ciphertext $C = E(k, m)$ and sends it to Bob. Upon receiving this ciphertext, Bob uses the corresponding decryption algorithm D together with the same key k and computes plaintext $m = D(k, C)$.

The encryption/decryption algorithms are publicly known as the only secret part of a cryptosystem is the key. However, in order for parties to use the cryptosystem, they need to exchange a secret key in advance. If both parties know each other and meet prior to communication, they need to use non-cryptographic methods for agreeing on a key. Otherwise, they may use public-key cryptography to exchange

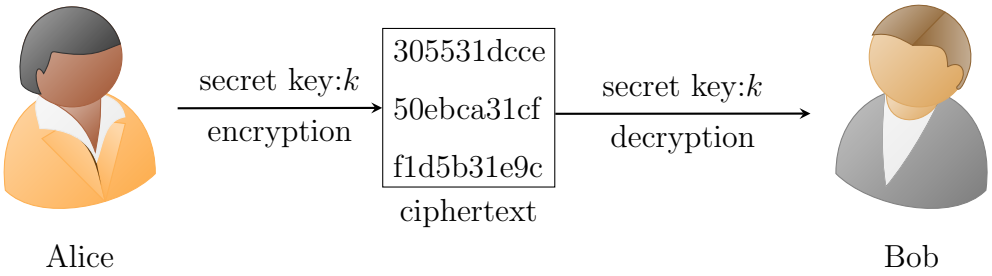


Figure 3.9 Symmetric cryptography

a key before their communication in a symmetric set-up. Moreover, there are also key exchange schemes available, such as Diffie-Hellman key exchange [104].

Symmetric-key encryption is a fast encryption method to implement in both hardware and software, so it is well-suited for encrypting large amounts of data. It can use either stream ciphers or *block ciphers* [105]. Stream cipher encrypts a message character by character by using a key stream. The message is sliced into characters and each one of them becomes a ciphertext character. This key stream is generated from a shared secret key by a pseudorandom generator. For instance, Vernam's one-time pad and Vigenere ciphers are considered in this category. Block ciphers can be found in both symmetric and asymmetric cryptography. They are also building blocks for other cryptographic protocols such as hash functions and pseudo-random number generators. Unlike stream ciphers, block ciphers use blocks to encrypt a plaintext. Given a plaintext consisting of blocks of n-bits, a block cipher maps each of these blocks to their corresponding n-bit ciphertext blocks, where n is called the *block length*. Many modern block ciphers for symmetric cryptography use Feistel ciphers [106].

Some block ciphers have already been cryptanalyzed, however there are some examples, such as AES, which are still used widely today. Advanced Encryption Standard (AES) is a symmetric-encryption algorithm which is standardized by NIST. It is a subset of the Rijndael block cipher with a fixed block size of 128 bits and three different key sizes: 128, 192 and 256 bits [104]. Moreover, it uses 10, 12 or 14 rounds with 128,192 and 256-bit key sizes respectively. AES has found places in many applications, however its inflexibility in the sense of its design makes its implementation

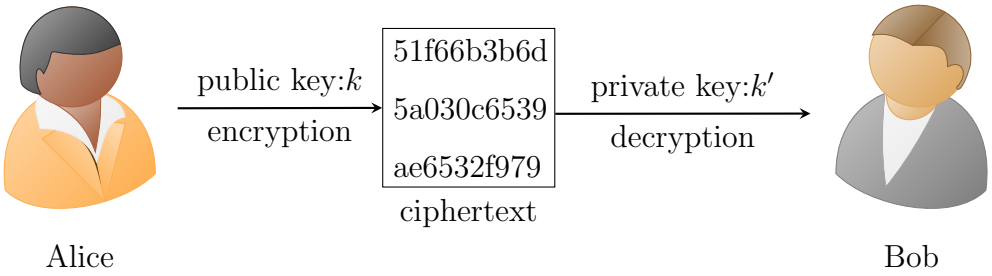


Figure 3.10 Asymmetric cryptography

harder for low-powered devices. Some of its requirements consume serious amount of memory, processing, and battery power which a low-powered device is not capable of supplying. This makes it a non-starter for low-powered embedded devices [107]. However, AES is still used in IoT in communication standards, such as Zigbee and Bluetooth, with dedicated chipset embedded.

In asymmetric cryptography, a key pair is used instead of having a single key to encrypt and decrypt the message. This key pair consists of a *public key* and a *private (secret) key*. Public key is the one which is broadcasted to the public, where anyone can use it to encrypt a message. On the other hand, private key must be kept secret, because this key is used to decrypt the encrypted message. In addition, the role of public and private keys may be switched for some other purposes, i.e. if a message is signed with the private key, it can be verified by using the public key as well. To provide this secrecy on the private key, the key pair should be selected in such a way that no one can construct the private key from the public key.

Most public-key encryption algorithms are based on computationally hard problems in number theory or algebra such as integer factorization and discrete logarithm problems [108]. RSA and Paillier are two of the well-known public key algorithms based on integer factorization. Another well-known example, El-Gamal encryption algorithm, is based on discrete logarithm problem [109–111]. Although speed in symmetric cryptosystems and cryptographic hash functions do not depend on the key and hash length, it is not the same for asymmetric cryptosystems [112]. As stated earlier, many asymmetric cryptosystems depend on either *integer factorization* or *discrete logarithm* problem.

Parties constantly try and communicate with each other via certain mediums such as telephone, internet, and many more. Before communicating, parties may need to verify that they are communicating with the right party. This verification process is called *authentication* and is defined as verifying the identity of communicating parties to each other in the context of computer science [113]. This protects the communication by preventing unauthorized intruders to reach the information.

3.4.1 Other Cryptographic Primivites

Cryptographic Hash Functions. Generally, a hash function is used to compress arbitrary-length strings into shorter strings, in order to achieve $O(1)$ insertion and lookup times for storing a set of elements. However, since the amount of data to be compressed is very large, collisions are unavoidable. Therefore, simple hash functions are not good candidates in cryptography. A *good* hash function has to supply a unique output for every possible input, therefore minimizing the possibility of collision. Some are classed as *collision-resistant hash functions*. Designing a collision-resistant hash function is not as easy as creating a regular hash function, where the main purpose of it is to compress files as a data structure. However, in order to use these hash functions in cryptography, collision-resistance is a must and therefore it requires a more advanced design [114].

According to Katz and Lindell [114], there are three levels of security when considering a cryptographic hash function: collision resistance, second pre-image resistance and pre-image resistance.

- **Collision resistance**: It should be computationally infeasible to find a pair of different input values (m, m') to have the same digest.
- **Second pre-image resistance**: It should be computationally infeasible to find a message m', to hash to the same output as message m.
- **Pre-image resistance**: It should be computationally infeasible to find a message m', which hashes to a specific output, $y = H(m)$.

Here, if a hash function is collision resistant, it is also pre-image resistant; because if there is a second preimage, then that means there is a colliding pair. Also, a pre-image resistant function is called a one-way function, since it is difficult to inverse it.

Cryptographic hash functions are used in many information security areas such as digital signatures, message authentication codes, fingerprinting, checksums, and many more [115–117].

OTP: One Time Passwords. One-time password systems are designed to overcome several drawbacks associated with using a single use password. The security of OTP depends on non-invertibility of a secure hash function [118]. In simplest terms, generation of OTP's consists of three steps: initial step, computation step and output step. First, the user chooses a passphrase which needs to satisfy a standard in size (10 chars) so that it is secure against dictionary attacks. Later, a non-secret seed in a text form is given from the server. This seed and the passphrase are combined, and a secure hash function is applied numerous times on this combination. Lastly, by using a function dependent algorithm, this output is reduced to 64 bits [118].

In recent years, more and more services are using two-factor authentication and OTP is commonly used as a part of this. Either an application or a physical device is used to generate an OTP. A simple design would be using a seed and a timestamp to produce such OTPs and the client uses an application/device to produce it. The server has the same seed, so can produce the same OTP simultaneously. However, it is impossible to synchronize time between two devices. That's why the server produces every possible OTP for the next minute. If the received OTP is included in that list, the client is authenticated.

A standard password is static and can be used multiple times. If an attacker tries and captures this password, then critical and sensitive information can be stolen. Although adding a dynamic password does not decrease the chances of the original

password being captured, it may prevent sensitive information to be stolen. One time passwords are dynamic passwords and they are used only once. Therefore they can provide protection to replay attacks and it is used as an extra layer for security. When OTP is used, stealing the password would not be enough for the attacker to compromise the system. However, the authentication is not limited to the usage of static password as it is also needed for the dynamic passwords, such as one-time password.

There are several ways of creating an OTP: time-synchronized, challenge-based, counter-synchronized, hash-based and SMS-OTP. [119]. After creating an OTP, there are several ways to deliver this to the target device: sending SMS, using mobile apps (push notifications), paper-based, web-based methods, and proprietary tokens [119].

It is highly likely to find OTP in daily life where security is an issue. For instance, banking applications use OTP regularly as two-factor authentication. The type of creating an OTP may change due to the nature of an application. Time-synchronized OTP need both server and client to keep the same time, otherwise it would not be useful. Since the mobile phones are receiving their time from the cell phone network, they are usually synchronized. However, one must keep in mind that when one of the users change time zones, time synchronization will break. Then, a user must enter the OTP within a time period before it expires. Counter-synchronized OTP use a counter which changes every time an OTP is requested. In both of these methods, the user enters the password that he/she sees on the screen of the device creating the OTP whether it is an additional device or a mobile phone. Challenge-based OTP often use an additional hardware with additional PIN. In this, the algorithm does not use time, therefore devices do not need to be synchronized time-wise. The downside of this method is that user must keep the device with him/her at all times. Lastly, hash-based OTP use cryptographic hash functions to produce a fixed-length password.

Lightweight Cryptography. Although conventional cryptographic methods work well in regular computers, it is hard for them to work for low-powered embedded devices such as IoT. This is because those cryptographic methods require much processing power, battery power and memory space. On the other side, embedded systems usually consist of 8-16-32 bit microcontrollers which make it hard for them to use conventional cryptographic methods [107]. For this reason, lightweight cryptography emerged as a field focusing on cryptographic solutions for devices with constrained capabilities in power supply, connectivity, hardware and software [120].

Lightweight cryptography provides a good alternative for low-powered devices by using a smaller block size, shorter keys and less complex rounds. Therefore, it is safe to state that lightweight cryptography is unable to provide a security level such as conventional cryptography [107]. It aims to provide a cryptographic algorithm which uses less memory, less computing resource, and less power supply.

Lightweight cryptography has several promising candidates to replace AES for low powered devices. Table 3.6 below lists some well-known block and stream ciphers.

Table 3.6 Some well-known block and stream ciphers

Lightweight cryptographic methods	**Block size (bits)**	**Key Size(bits)**
PRESENT	64	80,128
XTEA	64	64
RC5	32,64,128	0 to 2040
SIMON	64,72,96,128,144,192,256	32,48,64,96,128
CLEFIA	128	empty
Mickey V2	Stream	80
Trivium	Stream	80
Grain	Stream	80
Enocoro	Stream	80

3.4.2 Visual Cryptography

Visual cryptography was first introduced in 1994 by Naor and Shamir [121] as a novel way to provide secrecy on written material (printed text, notes, images, etc.) without using any complex cryptographic computations. In this, encryption is done by dividing an image into some shares, let's say n shares, and printing them onto transparent films. Then, in order to re-construct the original image, k or more, where $k \leq n$, transparent images must be stacked on top of each other. Otherwise, it is not possible to obtain the image back. Figure 3.11 below provides a basic example of 2-out-of-2 scheme. It can be seen that shares alone do not reveal anything about the original image.

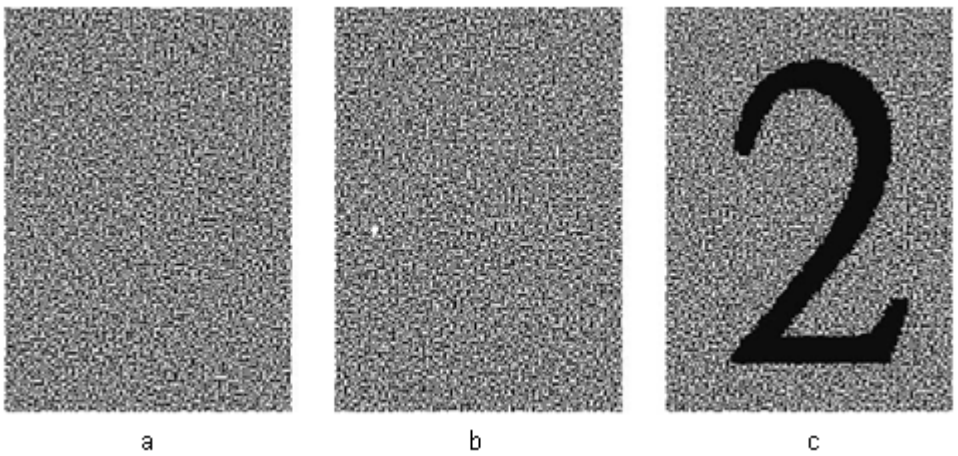


Figure 3.11 a) Share 1, b) Share 2, c) Reconstructing image by stacking shares up

In what follows, we present some related work.

Revenkar et al. [122] compared visual cryptography schemes between 1995 and 2009 based on number of secret images, pixel expansion, image format and types of generated shares. Out of 28 schemes, 19 of them work on binary images and only 5 of them uses meaningful (non-random) shares. It is suggested by Chang et al. [123] meaningful shares should be used because random looking shares could be seen suspicious and therefore vulnerable to attacks. However, number of visual cryptographic schemes using meaningful shares are less than others.

QR Codes are two dimensional matrix codes consisting of black pixels arranged on a white background on a square grid. Since it consists of black and white pixels, it is a very good candidate for visual cryptography applications. Several applications can be found in literature.

Lu et al. [124] use visual cryptography to provide security in QR code mobile payment systems. Authors divide the original QR code of the shop into two shares called shadows. These shadows are further implemented into carrier QR codes to conceal them with the help of error correction system built in QR codes. At this point, there are two carrier QR codes. One of them is stuck on the wall of the shop and the other one is stored in the cloud. When the customer scans the QR code to make a payment, the system immediately downloads the carrier code from the cloud and combines two carrier codes together in order to construct the original QR code. At that point, payment can be done.

Roy and Venkateswaran [125] combine steganography and visual cryptography in order to safeguard customer data, increase customer confidence and prevent identity theft. This proposed method involves a certified authority CA which holds a cover text involving the customer password and customer account number. A snapshot of this information is taken and divided into two shares. One share is given to the CA and the other is stored by the customer. In order to do payment, customer sends its own share to CA where CA combines these two and constructs the original image. Later, CA gives this information to the bank where the customer password is obtained from the cover text and the customer is authenticated. When the customer is authenticated, the bank transfers the money to the merchant. CA also gives the account information of the customer to the merchant, so that he can verify that that specific customer paid.

Hou and Huang [126] propose a novel intellectual property protection scheme for digital images using visual cryptography and statistical property. Authors select two random pixels from the original image and compare the pixel values. If the first

one is bigger, a black pixel is added to the master share; if smaller, a white pixel is added. This happens a number of times and authors claim that law of large numbers satisfies the necessary security demand of visual cryptography. The positions to select random pixels are generated from a private key K. After constructing a master share M; ownership share O is constructed by using contents of master share and a watermark share W. O is stored in a trusted third party. In order to authenticate the ownership, same private key K is used to generate master share M. Since O is stored in a third party, watermark share can be constructed by superimposing M and O. If the watermark share is same as the original watermark, the image ownership is authenticated.

Ying and Yinlan [127] propose a multiparty copyright scheme by using visual cryptography. They propose a system which can work up to 6 users. N, $N < 6$, number of users selects an original image and a watermark image. These two images are later combined and divided into $N+1$ shares where N shares are gone to the owners and 1 share is stored in a server. When a user wants to declare his/her ownership, he sends his own share to the server and by combining these two images, they can construct the watermark image.

Prakasha et al. [128] apply visual secret sharing to authenticate multiple users. Their simulation consists of two users, A and B. Users try to get access to server S. The system also includes a trusted third party T. User A shares key K_{at} with T and user B shares key K_{bt} with T. There is also an additional K_t which is shared among A, B and T. An image P is used as password to authenticate both users, and is divided into two parts called P_1 and P_2. First, A and B identify themselves to T, who encrypts their shares and sends them back. For A, T encrypts P_1 with K_{at} and for B, it encrypts P_2 with K_{bt}. If A and B are not frauds, they are able to decrypt these images. Later, when they want to access to the server, they combine their image shares with their identities with K_{at} and K_{bt} respectively, and encrypts that encryption with K_t. Trusted party T can now decrypt everything, combine shares and check whether that combination is same as original image P. If so, it grants

access.

Tanvir and Qu [129] use a different approach to their multi-user authentication systems. They have decided to use visual cryptography, because it is not a computationally intensive operation; and therefore making it suitable for IoT applications. They used resistive random access memory (RRAM) based circuits to demonstrate the concept of hardware based multi-user authentication. Previously, they have developed a system working with RRAM-based circuits and protocols for single user authentication [130]. While this is the basis for their multi-user authentication, they could not apply this single-user authentication protocol for every k user multiple times, due to scalability and privacy problems [130]. To remedy, they have designed a visual cryptography inspired RRAM-based authentication scheme and showed a small example for 2-out-of-3 user authentication using RRAM-based threshold detector circuit.

Construction of n-out-of-n scheme. In the simplest version of visual secret sharing scheme, a message is a collection of black and white pixels and each black/white pixel is handled separately. Moreover, each pixel appears in n different versions, called *shares*, and each share is a collection of m black and white subpixels. Now, consider an n-by-m binary matrix $S = (s_{ij})$, where $s_{ij} = 1$ if the j^{th} subpixel on i^{th} share is black. Therefore, each row in S represents a different share in the scheme. If one is given two shares, i.e. two rows of S, stacking them together means applying **inclusive OR** operation on these rows. Then the grey level of the resulting stacked vector is determined by the count of 1 bits. This count is basically the Hamming weight $H(v)$, where v is the resulting stacked vector. If this count is at least a fixed threshold value d, then the pixel obtained is assumed to be black. Otherwise, if $H(v) \leq d - \alpha \cdot m$, where $\alpha > 0$ is the relative difference in weight between combined shares from white and black pixels, then it is assumed to be white. In general, the matrix S represents a single pixel and rows of this matrix represent the shares which are to be distributed to n shareholders. After a single pixel (black or white) is splitted into shares, it is reconstructed as follows: First, k (out of n)

rows of S come together and then their joint grey level is computed by counting 1 bits after **inclusive OR** is applied on these k rows. If this number is less than the threshold value, then the pixel is considered as white, otherwise as black. This process requires a selection of a proper matrix S depending on the pixel being black or white. Also, what makes this process interesting and useful is that any randomly chosen k (or more) rows give rise to the same outcome and any $k-1$ (or less) rows do not reveal any info on the grey level of the pixel.

Naor and Shamir [121] present some solutions on how to construct these matrices in their original article. They first consider the case where $k = n$. By adopting their notations, let us consider two sets of binary vectors each of which is of length n, namely $J_1^0, J_2^0, \ldots, J_n^0$ and $J_1^1, J_2^1, \ldots, J_n^1$. Now, it is assumed further that the vectors J_i^0 satisfy the property that any $n-1$ of them are linearly independent, but the entire set is linearly dependent. On the other side, the vectors J_i^1 are linearly independent. For any $n \geq 2$, an easy way to obtain such sets is given below.

$$
\begin{array}{lcllcl}
J_1^0 & : & 1000\ldots00 & J_1^1 & : & 1000\ldots00 \\
J_2^0 & : & 0100\ldots00 & J_2^1 & : & 0100\ldots00 \\
 & & . & & & . \\
 & & . & & & . \\
 & & . & & & . \\
J_{n-1}^0 & : & 0000\ldots10 & J_{n-1}^1 & : & 0000\ldots10 \\
J_n^0 & : & 1111\ldots10 & J_n^1 & : & 0000\ldots01
\end{array}
$$

Then the matrix S^0 (S^1) is constructed as follows: Let's first label the rows of the matrix with the vectors J_i^0 (J_i^1) and the columns with all possible binary vectors of length n, therefore the resulting matrix is of size n by 2^n. Then, the binary entry at the intersection of the i^{th} row and s^{th} column is computed from the inner product of the vector J_i^0 (J_i^1) and the binary vector labeling the s^{th} column.

Once these matrices S^0 and S^1 are constructed, then two collections C^0 and C^1 are produced from these matrices, respectively, by applying all possible permutations on the columns of S^0 and S^1. If one plans on transmitting a 0 bit (or 1 bit), then a random matrix from the collection C^0 (or C^1) is picked and its rows are distributed as the shares. There are two cases to be considered:

(i) One picks a random matrix $S \in C^0$. Since all the columns of S are labelled by all possible binary vectors of length n, there are two columns labelled by all zero vector and the vector $0 \dots 01$. Then these are the only columns (out of 2^n columns) with all zero entries. Therefore, the number of 1 bit entries is $2^n - 2$ when the rows of S are stacked together.

(ii) One picks a random matrix $S' \in C^1$. It is similar to the first case, but there is only one column with all zero entries which is labelled by the zero vector. Therefore, the number of 1 bit entries is $2^n - 1$ in this case.

In both matrices, whenever a smaller number of rows are stacked together, the outcome would have $2^n - 2$ many 1 bit entries and the only time the difference occurs is when all rows of S' are stacked together. Hence, capturing $n - 1$ rows does not reveal any information about the grey level and consequently not reveal whether the rows belong to S or S'. It requires capturing all n rows to determine the color of the pixel. This is an illustration of n-out-of-n scheme. See page 6 on the original article by Naor and Shamir [121] for further details on the proof of this scheme being an n-out-of-n scheme.

Naor and Shamir [121] also came up with an alternative but slightly better way of constructing an n-out-of-n scheme with $m = 2^{n-1}$ columns. Let $W = \{w_1, w_2, \dots, w_n\}$ be a set with n elements, then assume that $E_1, E_2, \dots, E_{2^{n-1}}$ and $O_1, O_2, \dots, O_{2^{n-1}}$ be its subsets of even and odd cardinalities, respectively. Afterwards, n by 2^{n-1} matrices S^0 and S^1 are constructed as follow: For $1 \leq i \leq n$ and $1 \leq s \leq 2^{n-1}$, $S^0[i, j] = 1$ if and only if $w_i \in E_s$ and $S^1[i, j] = 1$ if and only if $w_i \in O_s$. Then, the collections C^0 and C^1 are obtained by permuting all the columns of S^0 and S^1,

respectively.

Construction of k-out-of-n scheme. Although n-out-of-n scheme provides a secure protocol, it would not be practical with increasing number of devices in a network. As we recall, the matrix has a size of $n\ x\ 2^n$. Let's say the number of devices increases up to 100; then it will reach sizes where a device memory is inadequate. In addition, as the number of devices increases, the chances of break downs also increases. This problem can be overcome by using a k-out-of-n secret sharing scheme. Although Naor and Shamir [121] shows that it is possible, Droste [131] improved and provided a separate construction which is explained below:

Later in [131], Droste shows how to construct k-out-of-n visual secret sharing scheme in which $k \leq n$ shares (instead of n shares) suffice to determine whether a pixel is white or black. The construction of the S^0, S^1 matrices is discussed below.

Given a matrix S, a k-restriction of S is a submatrix of S obtained by preserving only k rows of S. Let us assume that a given matrix S has n rows, then we define that

$$ADD(p, S) = \begin{cases} \text{Add every column with } q = p \text{ 1's to S,} & \text{if } p \leq k - p. \\ \text{Add every column with } q = p + n - k \text{ 1's to S,} & \text{if } p > k - p. \end{cases}$$

Now, the matrices S^0 and S^1 are constructed by following the steps given below:

1. For all even $p \in \{0, \ldots, k\}$, add every column with p 1's to each restriction of S^0 by calling $ADD(p, S^0)$.
2. For all odd $p \in \{0, \ldots, k\}$, add every column with p 1's to each restriction of S^1 by calling $ADD(p, S^1)$.
3. While the rests of S^0 and S^1 are not empty:
   (a) Add to S^0 all columns adjusting the rests of S^1 by calling ADD.
   (b) Add to S^1 all columns adjusting the rest of S^0 by calling ADD.

This algorithm gives rise to the constructions of the matrices for the schemes with

$2 \leq k \leq n$. In what follows, it is described how to use it for constructing matrices for the cases $k = 3$ and $k = 4$ in a much shorter way.

Case k=3. In order to construct the matrices necessary for the $k = 3$ case, we have the following steps:

1. Start with S^0 which consists of 1 column of all zeros and all of n columns of weight $n - 1$.
2. Start with S^1 which consists of 1 column of all 1's and all of n columns of weight 1.
3. Adjusting the rest of S^1, add to S^0 $n - 3$ columns of all zeros.
4. Adjusting the rest of S^0, add to S^1 $n - 3$ columns of all 1's.

Note that matrices S^0 and S^1 are both of size n by $2n - 2$. Moreover, the number of 1 bits is equal to n whenever any two rows in S^0 or S^1 are stacked together, but this number stays same when any three rows in S^0 are stacked and it increases to $n+1$ when any three rows in S^1 are stacked together. Below are examples of S^0 and S^1 matrices (for 3-out-of-5 scheme) which are constructed by the method described above.

$$S^0 = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix} \quad S^1 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Case k=4. Similarly, for the $k = 4$ case, we have the following steps:

1. Start with S^0 which consists of 1 column of all zeros, 1 column of all 1's, and all of $\binom{n}{2}$ columns of weight 2.
2. Start with S^1 which consists of all of n columns of weight 1 and all of n columns of weight $n - 1$.

3. Adjusting the rest of S^0, add to S^1 all of n columns of weight 1 for $n-4$ times.
4. Adjusting the rest of S^1, add to S^0 $n-4$ columns with all zeros, $n-4$ columns with all 1's and then again $\frac{1}{2} \cdot (n^2 - 7n + 12)$ columns with all zeros.

Note that matrices S^0 and S^1 are both of size n by $n \cdot (n-2)$. Moreover, the number of 1 bits is equal to $4n-9$ whenever any three rows in S^0 or S^1 are stacked together, but this number is equal to $5n - 13$ when any four rows in S^0 are stacked and it increases to $5n - 12$ when any four rows in S^1 are stacked together.

4. MODEL

As IoT devices take place in our daily lives, their security plays a central role in their intended usage. Without some security assumptions, it is very likely to have a chaotic medium due to the attacks, etc. As shown in given examples, IoT devices are being used without considering their security. Therefore, there have been many attacks such as botnet attacks, attacks on celebrated brands of cars, etc. One of the reasons why some of these attacks occurred in an easier manner is the (lack of) security assumption which is directly influenced by the way the system is set-up. The security of many of these known examples depends only on the security of a central unit. This creates a single point of failure which is defined earlier. Instead, a decentralized(distributed) model for IoT devices could present better solutions in security. This requires IoT devices to take active roles in protocols instead of just handling basic commands sent by the central unit.

In this chapter, we build on the previously mentioned visual cryptographic base and provide models and contributions. First, we describe how to use n-out-of-n and k-out-of-n models for our own scenarios. In addition, we further ease the process by using XOR operation instead of OR operation which is easier to construct on computer systems.

4.1 n-out-of-n Approach

Let us consider a secure closed network $\mathcal{N}$ of n (where $n > 2$) IoT devices and denote these devices in the network by $d_1, d_2, \ldots, d_n$. Assume that devices can communicate with each other pairwise in $\mathcal{N}$. Moreover, each of these devices has individual connection to Internet through a dummy router. It means that there is no central hub which processes and transmits messages collected from IoT devices to a remote master device on the behalf of IoT devices. In other words, the messages

are sent to a remote master device by the IoT devices themselves in a collaborative environment. When an IoT device d would like to send a message to a remote master device, it first triggers a communication with other IoT devices in $\mathcal{N}$ and then these devices, other than d, use their own shares to transmit the message of d to a remote master device. Although a master device, which is responsible for the distribution of shares, can communicate with each of the IoT devices securely in $\mathcal{N}$, the transmission of messages to a remote master device is assumed to be in only one direction through an insecure channel, that is, remote master device does not send messages back to IoT devices through this insecure channel.

Moreover, messages of the IoT devices are assumed to be predetermined and an enumeration applied to these messages. For instance, consider a message corresponding to a overheating issue belonging to a device, then a certain number is assigned to this type of issue. When this issue occurs, the bitstring corresponding to the assigned number is transmitted to the master device through an insecure channel. This tranmission is done bitwise in a collaborative way by the IoT devices. Let us assume that each device d_i has a fixed number t_i of predetermined messages that are enumerated by numbers 0 through $t_i - 1$. If $t_{\max} = \text{Maximum}\{t_1, \ldots, t_n\}$, then each message can be represented by a bitstring of length $s = \lceil log\ t_{\max} \rceil$ by using a proper padding for shorter bitstrings.

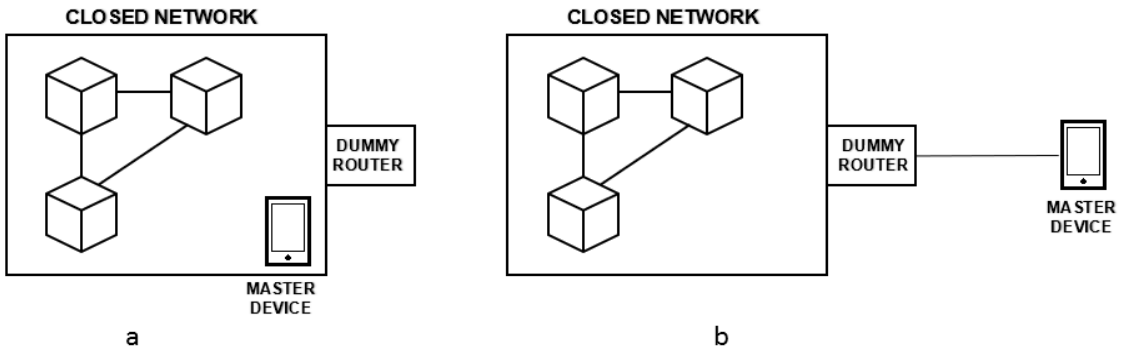


Figure 4.1 Illustration of a single master device a) near case b) away case

Now consider a master device which communicates with each of the IoT devices securely in $\mathcal{N}$, so this master device can distribute same number of shares to each device. Afterwards, each device can send a message, that is picked from their lists

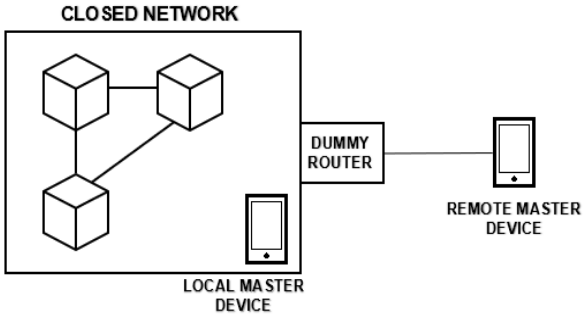


Figure 4.2 Illustration of local and remote master devices

of predetermined messages, to a master device through an insecure channel. Note that the master device to which messages are sent by devices is not necessarily the same master device which distributes the shares. In one case, one may think of a master device which is near these devices on a regular basis, so it can frequently communicate with the devices in $\mathcal{N}$ and IoT devices send their messages to this master device when it is away. In another case, there may be a remote master device to which the IoT devices send their messages. But then it requires another unit, such as a second local master device, which can distribute shares to devices in $\mathcal{N}$ and synchronize with the remote master device. In this scenario, a synchronization between these master devices is required and how it is achieved will be discussed later. See Figure 4.1 and Figure 4.2 for illustration of these two cases.

In what follows, we discuss the model from the IoT devices perspective, so we may refer to the same or different master devices during share distribution and process of sending messages. In which, having one or more master devices does not change the actions or their order taken by the IoT devices.

We first assume a network topology for $\mathcal{N}$ that is based on a complete graph $\mathcal{G}$ of order n in which the IoT devices are represented by the nodes in $\mathcal{G}$ and the adjacency relation defines a neigborhood among the IoT devices. Since $\mathcal{G}$ is complete, any node is a neighbor of any other. This neighborhood takes role in our model whenever a device transmits a message to a remote master device. In this process, only the neighboring devices collaborate with each other in sending the messages. It is obvious in this case that all $n-1$ neighboring nodes collaborate to send a message of

a particular node. It means that whenever an IoT device plans on sending a message to a remote master device, all members of $\mathcal{N}$ become aware of this and they all keep the track of messages contentwise and countwise.

Before any device starts sending messages to a remote master device, a master device (same as or different than the remote master device) distributes IoT devices' shares in $\mathcal{N}$. For sending a bitstring of length s, each participating IoT device uses s shares in total by consuming exactly one share for every bit of the message. However, since it is not known in advance which bitstring will be sent, the master device distributes $2 \cdot s$ shares to each device by considering all potential messages of length s. The master device takes the expected number of messages sent by the IoT devices overall until next scheduled share distribution process into account, so it distributes same and enough number of shares in $\mathcal{N}$. During share distribution, the network topology determines the type of the matrices whose rows to be distributed by the master device. In general, for a k-regular graph $\mathcal{G}$ defining the network topology, a $(k+1)$-out-of-n scheme is used. In our case, n-out-of-n scheme is constructed since we adopt the complete graphs ((n-1)-regular graph) to define a topology over $\mathcal{N}$. To do so, the master device constructs S^0, S^1 matrices as described above and then obtains the corresponding collections C^0, C^1 by permuting the columns of these matrices. For a message of length s, the master device picks s pairs of random matrices from $C_0 \times C_1$ with the condition that the matrices in a pair differ from each other by at least two rows. Note that each of these matrices has exactly n rows, then, to each of the IoT devices, the master device distributes exactly one row from each. For instance, let us assume that, for the j^{th} bit of a potential message, the rows of M_j^0 and M_j^1 have been distributed to the IoT devices. Then, in the actual message, each device uses their shares belonging to M_j^0 if the j^{th} bit is a 0-bit and shares belonging to M_j^1 if the j^{th} bit is a 1-bit.

Now, assume that an IoT device d plans to send the message $m = m_1 m_2 \dots m_s$, where $m_i \in \{0, 1\}$, to a remote master device after all shares are distributed by a master device. We also assume that the neighboring devices for d are denoted by

$d_{i_1}, d_{i_2}, \ldots, d_{i_{n-1}}$, where $1 \leq i_j \leq n$. Then d broadcasts the first bit of m, namely m_1, to all other devices in $\mathcal{N}$. Upon receiving m_1, each d_{i_j} sends its current share (row) belonging to the matrix M_1^0 if $m_1 = 0$ or belonging to the matrix M_1^1 if $m_1 = 1$ to the remote master device through an insecure channel. Note that device d itself does not send its share to the remote master device. This process continues similarly for m_2, m_3 and so on until the shares for m_s are all sent to the remote master device. Upon receiving the shares sent for the message m, the remote master device checks whether these shares belong to the matrices M_j^0 or M_j^1 and it determines the device that initiated the communication since its share is not sent to the remote master device. Since matrices in a pair differ from each other by at least two rows, we avoid the ambiguity on determining to which matrix the received shares belong. After receiving all bits of m belonging to d, the remote master device can decide for which of the predetermined messages of d the bitstring m has been sent.

To illustrate the model, we give the following small example. Let's construct the matrices S^0 and S^1 and generate the collections C^0 and C^1 as follows:

$$S^0\text{: } \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \xrightarrow[columns]{Permuting} C^0\text{: } \left\{ ..., \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}, ..., \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}, ..., \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}, ... \right\}$$

$$S^1\text{: } \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \xrightarrow[columns]{Permuting} C^1\text{: } \left\{ ..., \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}, ..., \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}, ..., \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}, ... \right\}$$

We assume that pairs of matrices below are picked from the collections C^0 and C^1, then without loss of generality the first rows of matrices are all distributed to device d_1, the second rows to device d_2 and third rows to device d_3.

$$M_1^0 : M_1^1 = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} : \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}, M_2^0 : M_2^1 = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} : \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

,

$$M_3^0 : M_3^1 = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} : \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}.$$

Now let's say d_2 plans on sending the message $m = m_1m_2m_3 = 001$, then devices d_1 and d_3 send the following shares to the remote master device in the order given below:

$$d_1 : \begin{bmatrix} 0 & 0 & 1 & 1 \end{bmatrix}_{m_1} \begin{bmatrix} 0 & 0 & 1 & 1 \end{bmatrix}_{m_2} \begin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix}_{m_3};$$
$$d_3 : \begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix}_{m_1} \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix}_{m_2} \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix}_{m_3}$$

When the remote master device receives the shares for m_1; $r_{1,1} = \begin{bmatrix} 0 & 0 & 1 & 1 \end{bmatrix}_{m_1}$ and $r_{3,1} = \begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix}_{m_1}$, then it can decide the bit value of m_1 as follows: Firstly, it stores the pairs of matrices in the order the shares have been distributed, so it knows that the shares are from either M_1^0 or M_1^1. Moreover, these two matrices differ from each other by at least two rows, so the master device can locate $r_{1,1}$ and $r_{3,1}$ as the first and third rows of M_1^0. This implies that $m_1 = 0$. Also, the remote master device can detect that $r_{1,1}$ is sent by d_1, $r_{3,1}$ sent by d_3 and d_2 has not sent its share so the message belongs to d_2. A similar process is conducted for m_2 and m_3 so finally the master device obtains the message as $m = 001$.

4.2 k-out-of-n Approach

This approach has a similar set-up as the previous one, however the main difference is the topology defined over the IoT network $\mathcal{N}$. In this approach, the network topology adapts a k-regular graph where $k < n - 1$. This means every node is not a neighbor of any other node in the network. As described in the previous model if a device wants to send a message, its neighbors are informed and they send their shares to a remote master device. The device itself does not share its own share.

One other difference in this approach is the synchronization among the IoT devices in

$\mathcal{N}$ overall. When an IoT device triggers a communication, we described that only its neighbors send their shares. However, other non-neighboring devices must be aware of this communication to keep track of their shares. As assumed before, master device distributes the same number of shares to IoT devices. For a particular device, if there is a communication initialized by one of its k neighbors, this device pass its current share to the remote master device. Otherwise, it ignores its current share and its next share becomes the current share. To achieve this, a synchronization is among the IoT devices is described later.

Similarly, the master device constructs S^0, S^1 matrices and then obtains the corresponding collections C^0, C^1. For a message of length s, the master device picks s pairs of random matrices from $C_0 \times C_1$ with the condition that the matrices in a pair overlap at most one row. Note that each of these matrices has still exactly n rows, then, to each of the IoT devices, the master device distributes exactly one row from each.

Now, assume that an IoT device d plans to send the message $m = m_1 m_2 \dots m_s$, where $m_i \in \{0, 1\}$, to a remote master device after all shares are distributed by the master device. We also assume that the neighboring devices for d are denoted by $d_{i_1}, d_{i_2}, \dots, d_{i_k}$, where $1 \leq i_j \leq n$. Then d broadcasts the first bit of m, namely m_1, to all its neighboring devices in $\mathcal{N}$. Upon receiving m_1, each d_{i_j} sends its current share (row) belonging to the matrix M_1^0 if $m_1 = 0$ or belonging to the matrix M_1^1 if $m_1 = 1$ to the remote master device through an insecure channel. Note that neither device d itself nor the non-neighboring devices do not send their shares to the remote master device. This process continues similarly for m_2, m_3 and so on until the shares for m_s are all sent to the remote master device. Upon receiving the shares sent for the message m, the remote master device checks whether these shares belong to the matrices M_j^0 or M_j^1 and it determines the device that initiated the communication since its share is not sent to the remote master device and there is only once device neighboring the devices which have sent their shares. Since matrices in a pair differ from each other almost entirely, we avoid the ambiguity on determining to which

matrix the received shares belong. After receiving all bits of m belonging to d, the remote master device can decide for which of the predetermined messages of d the bitstring m has been sent.

In the following sections, we illustrate this approach for $k = 3$ and $k = 4$.

4.2.1 Construction of 3-out-of-n secret sharing scheme

For k=3; the following steps are followed to construct 3-out-of-n secret sharing scheme:

1. Let S^0 consist of 1 column of all 0's and n columns of weight $n - 1$.
2. Let S^1 consist of 1 column of all 1's and n columns of weight 1.
3. Adjusting the rest of S^1, add to S^0 $n - 3$ columns of all 0's.
4. Adjusting the rest of S^0, add to S^1 $n - 1$ columns of all 1's.

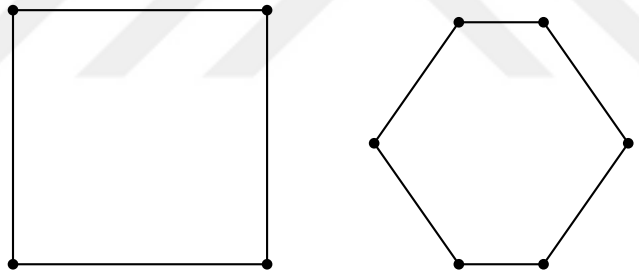


Figure 4.3 2-regular graphs

For this particular example, a cycle (2-regular graph) shown in Figure 4.3 is used as a topology, so every device has exactly 2 neighbors.

When a 3-out-of-n scheme is constructed, rows are distributed to IoT devices. If one of these IoT devices wants to send a message to the remote master device, it lets its two neighboring devices know and these two devices send their shares to the master device. Upon receiving two shares (rows) the remote master device directly determines the device which triggers the communication since it knows the topology. Also, with our original assumption that two matrices overlaps at at most only one row, the master device can decide which matrix these shares (rows) belong to.

$$M_1^0 : M_1^1 = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix} : \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

For the sake of simplicity, let's say that a device d_1 wants to send a single bit, $m = 0$. Neighbors of d_1 are d_2 and d_5. These devices are going to send following shares to the remote master device:

$$d_2 : \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}_{m_1}$$
$$d_5 : \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}_{m_1}$$

Remember that matrices only have one row in common. Therefore, when the remote master device receives the shares for m_1, $r_{2,1} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}_{m_1}$ and $r_{5,1} = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}_{m_1}$; by checking with the stored pairs of matrices, it can identify whether the shares are from M_1^0 or M_1^1, and locate $r_{2,1}$ and $r_{5,1}$ as second and fifth rows of M_1^0.

4.2.2 Construction of 4-out-of-n secret sharing scheme

Similar to the case k=3; the following steps are for k=4:

1. Let S^0 consist of 1 column of all 0's, 1 column of all 1's and $\binom{n}{2}$ columns of weight 2.
2. Let S^1 consist of n columns of weight 1 and n columns of weight $n-1$.
3. Adjusting the rest of S^1, add to S^1 n columns of weight 1 for $n-4$ times.
4. Adjusting the rest of S^0, add to S^0 $n-4$ columns with all 0's and $n-4$ columns with all 1's. Add $(n^2-7n+12)/2$ columns with all 0's.

For $k = 4$, we need to adapt 3-regular (cubic or trivalent) graphs as the network

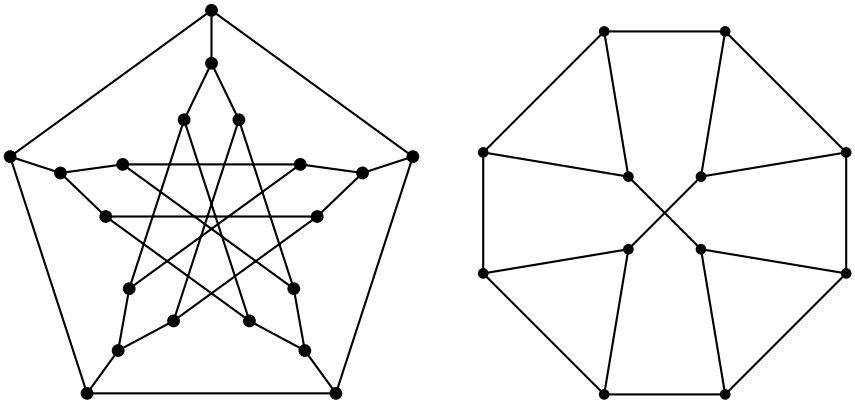


Figure 4.4 3-regular graphs

topology. Figure 4.4 provides two different examples for cubic graphs. The existence of cubic graphs is discussed for even orders. For odd n values one may have resolution by using a dummy node to make the order even. Then, the rest is very similar to the previous case.

For even $n > 3$, there exist cubic graphs of order n [132]. However, for odd n, we may use a dummy node to make n look like an even number. Here, we add a dummy node and join this with one of the other nodes, i.e. d. Therefore, d receives twice as many shares as other devices shares. Since dummy node and d may share the same nodes as neighbors, the final graph may not be simple. The dummy node would not send a message, so d does not need to send message on behalf of the dummy node. However, d sends the shares of the dummy node to the remote master device if one of the neighbors of the dummy node triggers a communication.

5. IMPLEMENTATION

In this chapter, we give further information about implementation details of our method. First, we talk about how it provides secrecy. Next, we define how to synchronize multiple devices by using OTP and enumeration methods. Lastly, we analyze our method in performance and memory aspects.

5.1 Secrecy & Security

The proposed model depends on the visual secret sharing scheme originally introduced by Naor and Shamir [121]. In their study, Naor and Shamir provided with two constructions for an n-out-of-n scheme. In their first construction the resulting matrices S^0 or S^1 both are of size $k \times 2^k$ and the latter method produces matrices of size $k \times 2^{k-1}$. Although the second construction method gives rise to a more efficient result, we use mainly the first construction in our model for practical purposes in construction. For both constructions, Naor and Shamir also include the proofs showing that the matrices satisfy some certain conditions. In the first part of their discussion, the number of all zero columns leads to the conclusion that stacking less than k rows does not make any difference on S^0 or S^1, but stacking k rows on S^1 yields more 1-bit entries compared to the matrix S^0. The continuation of the discussion shows that the distributions of submatrices with $k-1$ rows or less in collections C^0 and C^1 are same. This means that if one captures $k-1$ or less rows from a matrix, then this information is not sufficient to decide to which collection this matrix belongs.

Naor and Shamir posed some questions in their original work. Although they included some discussions on the existence of k-out-of-n schemes, Droste [131] resolved this problem. In his paper, Droste shows explicitly how to construct such schemes

and includes the necessary proofs. Similarly, he shows how stacking rows on S^0 and S^1 makes difference when k of them are used instead of k or less case. His approach on security aspect is also similar as he discusses the distribution of the submatrices in collections C^0 and C^1.

In our proposed model, the secrecy and security mainly depend on the fact that matrices S^0 and S^1 result in the desired schemes. When an IoT device triggers a communication, this device does not send its shares. Missing a share (a row), an eavesdropping third party on an insecure channel cannot decide whether the sent shares (rows) belong to a matrix from C^0 or C^1 even if it captures all other $k-1$ rows. As it is discussed above, it is proved that finding a matrix with the captured $k-1$ rows in C^0 or C^1 are equally likely, so it cannot determine the bit value.

Moreover, it is assumed in n-out-of-n approach that matrices in each pair picked from the collections C^0 and C^1 differ from each other by at least two rows, so transmitting just $n-1$ shares (rows) to the master devices does not create any ambiguity on its side. It can still determine the matrix (and so the corresponding bit value) upon receiving $n-1$ rows (shares) since it stores all the shares. Similarly in k-out-of-n approach, the assumption of having matrices differing from each other by at most 1 row takes care of a potential ambiguity on the master device side. The existence of such matrix pairs follow from the fact that the rows of S^0 and S^1 in both approaches are all distinct.

Up to this point, it is known that a third party cannot reveal the secrecy by obtaining the information sent over the insecure channel. However, it can try to attack in some other ways. At this point, it is beneficial to remember and summarize inner workings of the protocol.

Firstly, it is previously assumed that IoT devices reside in a secure network, so they are assumed not to be compromised by third parties unless they are physically in the network. Even so, other than doing their own sensing tasks, IoT devices are not doing anything instead of incrementing a simple counter.

Dummy router is open to the internet and it is possible that it may be compromised by a third party. However, dummy router itself cannot reveal the secrecy because it relays the information sent to it by the IoT devices. Since the original sender only broadcasts the intention to other devices and not dummy router; compromising dummy router does not help the adversary.

A third party may try to intervene the communication, alter some shares and can send different data to the remote master device. Since our method uses sequence numbering, master device can easily detect that there is an anomaly and discard the message. IoT devices and master device both have the same sequence and master device only compares incoming shares to the sequence the system is on. Moreover, a third party can try a replay attack but since the master device chooses different matrix pairs from the collections for every sequence, this case will be similar to the previous attack and master device will discard it.

In addition, impersonation issues can also be detected since the master device authenticates the senders via the missing shares. The shares are distributed to the IoT devices by the master device and an IoT device does not produce its own share.

A third party can try to disturb the communication line by sending random values. However, our proposed method does not aim to provide a solution for this problem. It aims to provide secrecy on an insecure line. The only way a third party is able to reveal the secret is to have the same database as the master device. However, the database of the master device changes at every distribution. Therefore, if a third party is able to compromise the master device and its database for a given distribution, it can reveal the secret.

If a third party continously listens to the insecure line and stores that information, it is unable to reveal a meaningful message from those submatrices. The probability of a given submatrix to belong in either collection is the same, since each collection has the same number of a given submatrix.

Our proposed system is secure as long as IoT devices are communicating within a closed network and master device keeps the distribution database to itself. If a master device works maliciously and shares distribution details, the secret can be revealed by a third party.

5.2 Synchronization

In this section we discuss various aspects of the implementation of the proposed model. The intention is to point out some aspects related to implementation of the model.

In the proposed model the topology defined over $\mathcal{N}$ is adopting a complete graph or a regular graph. In the case of complete graph, we can easily assume that any device in $\mathcal{N}$ can communicate with any other securely through their neighborhood. This maintains an environment in which any device is aware of any communication initiated by a member in $\mathcal{N}$. Moreover, devices are all aware of which messages are transmitted to the remote master device and in which order. Therefore, devices can follow the order of the shares to be used for every communication and this maintains the synchronization among the IoT devices in $\mathcal{N}$. Note that adding or removing device to/from network requires a new share distribution by the master device. On the otherside, if a regular graph is adapted as a network topology, we may still assume that any device is aware of the initiation of sending a message even if it is not a neighbor of the device which triggered the communication. If there is no neighborhood among the devices, the message is counted but no action is taken. An alternative method for distributing this knowledge over the entire network is to use a predefined spanning tree over the network topology. By this spanning tree structure, any device on $\mathcal{N}$ can follow the messages. this approach may be more useful within larger networks.

Another aspect of the implementation is the concurrent communication initiations by different IoT devices in $\mathcal{N}$. In the case of two different devices broadcasting

their messages in $\mathcal{N}$ concurrently, an implementation requires some scheduling or priority mechanism to point out which message is handled first within the network. In this queueing mechanism, one may want to consider adding time stamps to the broadcasted messages or using the type of the messages to be transmitted in order to determine the priorities. It is essential to synchronize IoT devices, so they all use correct corresponding shares for the same message. This can be implemented in two ways: first one allowing shares to be sent by IoT devices whenever bit values are received, and the second, IoT devices waiting for transmission until the entire message is received. We assume that all messages are of the same length, so whenever a device broadcasts the bits in its message, other devices in $\mathcal{N}$ can wait for the entire set of bits received then transmit their corresponding shares to the remote master device. If they choose to send their shares whenever they receive a bit value without any delay, the remote master device can wait for a certain number of shares received, then it processes the shares to compute the message. Both scenarios may have pros and cons, but it is critical to maintain synchronization among the IoT devices, so the remote master device does not fail due to some ambiguity on which sent share belongs to which message.

In the proposed model, there may be one or two master devices present as mentioned previously. In the case of a single master device, synchronization on share distribution is not an issue since there is only one device handling this. However, if there are two master devices, namely a local one and a remote one, then it requires some synchronization method among them to maintain the distribution of the same shares on two sides. This may be achieved as follows:

We first assume that both of master devices can construct matrices S^0 and S^1 as their constructions are public. As described before, the collections C^0 and C^1 are obtained by permuting the columns of the matrices S^0 and S^1, respectively. However, there is no need to contruct these collections in advance, instead one may apply a permutation on S^0 or S^1 whenever it is necessary. This requires generating the same permutations on both ends simultaneously and results in the same matrices

(so same shares) on both sides. To achieve this, a randomly generated seed can be used together with a method which maintains a one-to-one corresponding between seeds (integers) and permutations. In other words, the seed can be used to determine a permutation from, let's say, a set of lexicographically enumerated permutations.

If the matrices S^0 or S^1 are of size $n \times m$, the number of permutations of required is $m!$ Richard [133] shows that it is possible to determine the corresponding permutation for a given randomly generated seed, where it is between 1 and $m!$. Surely, this seed must be same for both local and master devices. To have this, both devices must use the same seed generating algorithm. This algorithm will be a proprietary algorithm which will use the timestamp information, similar to the OTP methods. The integer, as this seed, will be used as a step to get the correspoding permutation which is unavailable to third parties. By applying this permutation on both of S^0 and S^1 matrices, both local and master devices will come up with the same matrix for share distribution.

In order to generate the seed, we need a number generator. However, this generator must take into account some value which is shared by local and master device. This can be a timestamp, or time information. Generally, in settings such as this (OTP), a timestamp is used. However, it is used for authentication purposes. The client sends a value, and the server calculates 60 values representing every outcome for every second in a minute; and checks whether the given value is in that range. However, this cannot be used in our scenario. A possible solution may be just using a timestamp up to minute value. e.g. let's say the local master device sent "ready" message to remote master device at 22:37:33. Since there will be a time delay, remote master device will receive this at a time 22:37:33 + t. However, we can use 22:30:00 or 22:00:00 as a timestamp and generate a seed according to that. In addition, we can use some simple confirmation between these two devices so that they both know they are generating "a" seed. However, this algorithm must be proprietary and must not be known by the third party. Otherwise, they can intercept the timestamp and try and guess the exact value in order to generate the same random number.

After successfully generating the same seed in integer, master and local devices will use that number as an order to find a permutation. In order for both devices to find the same permutation, they must abide by the same enumeration method. By using lexicographic enumeration, both devices can find the same permutations and construct the same matrix from the initial S^0 and S^1 matrices. Moreover, factoradics is a short term for factorial number system. This system is a mixed radix numeral system adapted to numbering permutations. In order to find the k^{th} lexicographical permutation, k needs to be written in the factorial base first and here is a pseudocode on how to write k in factorial base:

```
create stack s;
int i = 1;
k = input()
while k != 0:
  stack.push(k%i);
  int k = k / i;
  i++;
while s:
  s.pop()
```

Starting from 1, we divide k into 1 and update k as the dividend. We also store the remainder in a stack. By increasing the quotient by 1 in every step, we do the same calculation until the dividend is 0. When the dividend is 0, we push the last remainder into the stack. Reading the values from stack will give us the factorial number representation, which is the k^{th} order in lexicographic enumeration.

Lehmer code carries out the conversion between the factorial number system and the corresponding permutation. Each digit in the factorial number system represents a corresponding digit from the permutation number. Let's say that we have three devices in the network and our RNG created integer 23. The factorial number representation of 23 is 3210. Here, each digit gives information about the permutation.

Since we have three devices, our matrix is going to have the size of 3 x 4, which means there are 4 columns. Our initial value will be 1234 (0123). Table 5.1 below is an example of how to apply 3210 into this.

digit	remaining elements	chosen element
3	1234	4
2	123	2
1	13	3
0	1	1

Table 5.1 How it works

Here, up until the last digit of our integer, we select that index from the original string and remove it. Joining the removed elements will give us the final permutation.

5.3 Authentication, Performance and Memory Requirements

Authentication. In the proposed model, the IoT device which starts the communication, i.e. which intends to send a message, does not send its shares to the remote master device. Instead, this particular device just broadcasts the bits of its message in $\mathcal{N}$. Therefore, the remote master device can figure out the owner of the message by going through the shares received and realizing that its shares are missing. For a message of length s, the remote master device will not receive this device's s many shares as it is receiving others'. Since the master device has all the shares of IoT devices in storage, it will detect the owner of the sent messages. Moreover, it will also detect if one of the received messages or their order is changed or manipulated.

Performance. The performance of the master device is irrelevant, because it is chosen to be a device with relatively adequate computing power, such as a mobile phone or a PC. However, it is important for the IoT devices to have adequate computational power to realize this model. However, IoT devices are expected to handle basic steps such as triggering a communication by broadcasting bit values of its message in $\mathcal{N}$, keeping the track of bits shared with itself and its own shares,

upon receiving a bit value sending its next share to the remote master device, etc., so none of these operations are costly and consequently do not require high computation power on an IoT device.

Porting modern cryptography over to ultra-low-end devices are very difficult [27]. The reason is that these devices has very limited resources to use for security. They also state that lightweight cryptography or cryptography in general is very hard to successfully used and implemented on low-end IoT devices without any weaknesses.

Zhang et al. [134] presents a performance and energy consumption analysis of three AES implementations. These three AES implementations are: exclusively software original AES, exclusively software AES with optimized table lookup and hardware supported AES. Analysis and tests were done on a MicaZ sensor node running TinyOS 2.1.0. This sensor node has 8-bit ATmega128L microcontroller, 128KB RAM, 512KB ROM and Chipcon CC2420 RF transceiver with AES-128 support. Table 5.2 below summarizes their findings:

Table 5.2 Total RAM & ROM usage for key setup, encryption and decryption

AES Implementation	**ROM (byte)**	**RAM (byte)**
Original	26336	4660
With table lookup	25130	4898
Hardware Accelerated	30655	1634

Table 5.3 shows that using these implementations increase both time and energy of these devices. All three methods can be successfully implemented on the MicaZ node and the hardware implementation observed to be the fastest. However, authors commented that the speed advantage of the hardware is negated by its energy consumption.

However, it is also possible in recent microcontrollers to reduce both duration and energy consumptions of AES operations. By using the hardware accelerator in ATmega128RFA1 microcontroller with an integrated transceiver, Panait and Dragomir [135] provided an optimized AES implementation for four modes of op-

Table 5.3 Execution times & energy consumption for key setup and encryption

AES Implementation	**Time (ms)**	**Energy (μJ)**
Original	1,4217	30,31
With table lookup	1,3272	28,64
Hardware Accelerated	0,5751	43,94

eration (ECB, CBC, CFB & CTR). Compared to the previous studies, by using a microcontroller with an integrated transceiver, they were able to decrease the cost of encryption. Authors also shown that although average power consumption is similar between software implementation compared to their hardware implementation, energy consumption differs dramatically. Table 5.4 shows the approximated energy consumption for increasing plaintext size.

Table 5.4 Energy consumption of software vs. hardware AES implementation

AES Implementation	**Energy (μJ)**
Software (0 bytes)	10
Software (128 bytes)	≈ 60
Hardware (0 bytes)	≈ 0
Hardware (128 bytes)	≈ 15

Tables above show that although encryption can be applicable in certain scenarios, it is obvious that they come with increase in running time and energy consumption. Our proposed method does not employ traditional encryption algorithms and can work faster with less energy consumption, making it available to even lesser capable devices.

Memory Requirements. As discussed in the earlier section, assuming that $t_{\max} = \text{Maximum}\{t_1, \ldots, t_n\}$, where t_i is the number of predetermined messages for a device d_i, then each message can be represented by a bitstring of length $s = \lceil log\ t_{\max} \rceil$. Moreover, for every bit of a message, the master device distributes a share (row) of length 2^{n-1}, where n is the number of devices in $\mathcal{N}$. Furthermore, if there are approximately p messages that are sent to the remote master device from one share distribution to the next one, then an IoT device requires to keep at least $p \cdot s \cdot 2^{n-1}$

bits in a given certain order. For instance, in a scenario with an IoT network with n devices in which overall one message out of 128 predetermined messages is sent every minute and share distribution process is conducted on daily basis, each device needs to keep at least $2^{n+12.3}$ bits or approximately 2^n kB. In a small-sized network, this requires IoT devices to have low memory capacities and, as the number of devices in a network increases, their individual memory capacities are required to increase accordingly. This may be considered as a limitation during the implementation of the model, however one may come up with a resolution by dividing the original network into smaller-sized subnetworks and managing them. Another resolution may be adopting a k-regular graph, where $k << n$, as a topology and adjusting the model with respect to this topology.

6. CONCLUSION

IoT is a concept where objects we use everyday are equipped with capabilities such as identifying, sensing, networking and processing. These capabilities allow IoT devices to communicate with one another. With increases in technological space, IoT devices are being used in many diverse areas: supply chain networks, healthcare, smart infrastructure, social applications and much more. However, this fast movement brings security vulnerabilities with it, making IoT an open target for botnets and such, which is shown to be a big problem for both IoT and other devices on the Internet. One of the security vulnerabilities is single point of failure which is defined as a crucial part of the system such that if it fails, it renders the entire system useless. There are several examples in real-life applications where attackers use this problem to compromise a network.

Since IoT devices are constrained in performance and power, it is hard to apply security protocols. Therefore, our motivation is to find a suitable solution which eliminates single point of failure and provide a secure communication without the application of power consuming cryptographic techniques.

To overcome this problem, we propose a *distributed* model for IoT devices where every IoT device in the network take active roles in communication instead of being independent from each other. By working together, the need for a central gateway or hub diminishes and this eliminates single point of failure. We propose two different approaches, namely n-out-of-n and k-out-of-n approaches. In the first approach, the topology is a complete graph where every device is responsible for the other. This approach is more useful for areas with small number of devices such as smart home, etc. In the second approach, the topology is a k-regular graph, where $k < n$. Here, unlike the first approach; only neighboring devices are participating in the secrecy scheme. Moreover, we provide easier constructing methods for 3-out-of-n and 4-out-

of-n secret sharing schemes. This approach is more useful for larger areas such as factories, farms, etc.

Lastly, our proposed method provides secure communication on an insecure line without any use of complex cryptographic algorithms and satisfies necessary memory conditions; thus making it suitable for constrained devices.

BIBLIOGRAPHY

[1] A. Whitmore, A. Agarwal, and L. Da Xu, "The internet of things—a survey of topics and trends," *Information systems frontiers*, vol. 17, no. 2, pp. 261–274, 2015.

[2] S. Zeadally, A. K. Das, and N. Sklavos, "Cryptographic technologies and protocol standards for internet of things," *Internet of Things*, vol. 14, p. 100075, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660519301799

[3] R. H. Weber, "Internet of things – new security and privacy challenges," *Computer Law & Security Review*, vol. 26, no. 1, pp. 23–30, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0267364909001939

[4] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[5] B. Jovanovic, "Internet of things statistics for 2022 - taking things apart," *DataProt*. accessed: Jul 8, 2018. [Online]. Available: https://dataprot.net/statistics/iot-statistics/

[6] G. Yang, M. Jiang, W. Ouyang, G. Ji, H. Xie, A. M. Rahmani, P. Liljeberg, and H. Tenhunen, "Iot-based remote pain monitoring system: From device to cloud platform," *IEEE journal of biomedical and health informatics*, vol. 22, no. 6, pp. 1711–1719, 2017.

[7] ZDNet, "The best iot, smart home gadgets in 2018," *ZDNet*. accessed: Jul 5, 2018. [Online]. Available: https://www.zdnet.com/pictures/the-best-iot-smart-home-gadgets-in-2018

[8] T. Salman and R. Jain, "A survey of protocols and standards for internet of things," *arXiv preprint arXiv:1903.11549*, 2019.

[9] S. Babar, A. Stango, N. Prasad, J. Sen, and R. Prasad, "Proposed embedded security framework for internet of things (iot)," in *2011 2nd International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology (Wireless VITAE)*. IEEE, 2011, pp. 1–5.

[10] O. Garcia-Morchon, S. Kumar, and M. Sethi, "Internet of things (iot) security: State of the art and challenges," *Internet Res. Task Force (IRTF)*, p. RFC8576,

2019.

[11] T. Hughes, "A world with more iot standards bodies than iot standards," *TechTarget*. blog. accessed: Jul 5, 2018. [Online]. Available: https://internetofthingsagenda.techtarget.com/blog/IoT-Agenda/A-world-with-more-IoT-standards-bodies-than-IoT-standards

[12] F-Secure, "Pinning down the iot," *F-Secure*. accessed: Jul 5, 2018. [Online]. Available: https://www.forbes.com/sites/forbestechcouncil/2017/06/06/best-smart-home-devices-and-how-iot-is-changing-the-way-we-live

[13] A. Schiffer, "How a fish tank helped hack a casino," *Washington Post*. accessed: Jul 5, 2018. [Online]. Available: https://www.washingtonpost.com/news/innovations/wp/2017/07/21/how-a-fish-tank-helped-hack-a-casino

[14] O. Williams-Grut, "Hackers once stole a casino's high-roller database through a thermometer in the lobby fish tank," *Business Insider*. accessed: Jul 5, 2018. [Online]. Available: http://www.businessinsider.com/hackers-stole-a-casinos-database-through-a-thermometer-in-the-lobby-fish-tank-2018-4

[15] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in internet of things: The road ahead," *Computer networks*, vol. 76, pp. 146–164, 2015.

[16] A. Muravitsky, V. Dashchenko, and R. Sako, "Iot hack: how to break a smart home again."

[17] V. T. Labs, "Turning a webcam into a backdoor," *Vectra Thread Labs*. accessed: Apr 5, 2022. [Online]. Available: https://web.archive.org/web/20180805134121/https://blog.vectra.ai/blog/turning-a-webcam-into-a-backdoor

[18] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive experimental analyses of automotive attack surfaces," in *20th USENIX Security Symposium (USENIX Security 11)*, 2011.

[19] I. Rouf, R. Miller, H. Mustafa, T. Taylor, S. Oh, W. Xu, M. Gruteser, W. Trappe, and I. Seskar, "Security and privacy vulnerabilities of {In-Car} wireless networks: A tire pressure monitoring system case study," in *19th USENIX Security Symposium (USENIX Security 10)*, 2010.

[20] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham *et al.*, "Experimental security analysis of a modern automobile," in *2010 IEEE symposium on security and pri-*

vacy. IEEE, 2010, pp. 447–462.

[21] C. Botnet, "Internet census 2012: Port scanning/0 using insecure embedded devices," *SourceForge, White Paper*, 2012.

[22] P. Sinha, A. Boukhtouta, V. H. Belarde, and M. Debbabi, "Insights from the analysis of the mariposa botnet," in *2010 Fifth International Conference on Risks and Security of Internet and Systems (CRiSIS)*, 2010, pp. 1–9.

[23] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao, "A survey on security and privacy issues in internet-of-things," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1250–1258, 2017.

[24] I. U. Din, M. Guizani, B.-S. Kim, S. Hassan, and M. K. Khan, "Trust management techniques for the internet of things: A survey," *IEEE Access*, vol. 7, pp. 29 763–29 787, 2018.

[25] L. Chen, S. Thombre, K. Järvinen, E. S. Lohan, A. Alén-Savikko, H. Leppäkoski, M. Z. H. Bhuiyan, S. Bu-Pasha, G. N. Ferrara, S. Honkala *et al.*, "Robustness, security and privacy in location-based services for future iot: A survey," *IEEE Access*, vol. 5, pp. 8956–8977, 2017.

[26] A. H. Ngu, M. Gutierrez, V. Metsis, S. Nepal, and Q. Z. Sheng, "Iot middleware: A survey on issues and enabling technologies," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 1–20, 2016.

[27] W. Trappe, R. Howard, and R. S. Moore, "Low-energy security: Limits and opportunities in the internet of things," *IEEE Security & Privacy*, vol. 13, no. 1, pp. 14–21, 2015.

[28] U. E. Larson and D. K. Nilsson, "Securing vehicles against cyber attacks," in *Proceedings of the 4th annual workshop on Cyber security and information intelligence research: developing strategies to meet the cyber security and information intelligence challenges ahead*, 2008, pp. 1–3.

[29] Y. Zhao, "Telematics: safe and fun driving," *IEEE Intelligent systems*, vol. 17, no. 1, pp. 10–14, 2002.

[30] M. Wolf, A. Weimerskirch, and T. Wollinger, "State of the art: Embedding security in vehicles," *EURASIP Journal on Embedded Systems*, vol. 2007, pp. 1–16, 2007.

[31] M. S. Sheikh and J. Liang, "A comprehensive survey on vanet security services in traffic management system," *Wireless Communications and Mobile*

Computing, vol. 2019, 2019.

[32] M. Arif, G. Wang, M. Z. A. Bhuiyan, T. Wang, and J. Chen, "A survey on security attacks in vanets: Communication, applications and challenges," *Vehicular Communications*, vol. 19, p. 100179, 2019.

[33] J. Ben-Othman and L. Mokdad, "Modeling and verification tools for jamming attacks in vanets," in *2014 IEEE Global Communications Conference*. IEEE, 2014, pp. 4562–4567.

[34] V. H. La and A. R. Cavalli, "Security attacks and solutions in Vehicular Ad Hoc Networks : a survey," *International journal on AdHoc networking systems (IJANS)*, vol. 4, no. 2, pp. 1 – 20, Apr. 2014. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01262473

[35] W. Ahmed and M. Elhadef, "Securing intelligent vehicular ad hoc networks: A survey," in *Advances in Computer Science and Ubiquitous Computing*. Springer, 2017, pp. 6–14.

[36] V. H. La and A. R. Cavalli, "Security attacks and solutions in vehicular ad hoc networks: a survey," *International journal on AdHoc networking systems (IJANS)*, vol. 4, no. 2, pp. 1–20, 2014.

[37] A. Marzano, D. Alexander, O. Fonseca, E. Fazzion, C. Hoepers, K. Steding-Jessen, M. H. Chaves, Í. Cunha, D. Guedes, and W. Meira, "The evolution of bashlite and mirai iot botnets," in *2018 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2018, pp. 00 813–00 818.

[38] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis *et al.*, "Understanding the mirai botnet," in *26th USENIX security symposium (USENIX Security 17)*, 2017, pp. 1093–1110.

[39] L. H. Newman, "Millions of web camera and baby monitor feeds are exposed," *Wired*. accessed: Apr 5, 2022. [Online]. Available: https://www.wired.com/story/kalay-iot-bug-video-feeds/

[40] E. Ronen and A. Shamir, "Extended functionality attacks on iot devices: The case of smart lights," in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2016, pp. 3–12.

[41] K. Y. Echo Duan, Veo Zhang, "Flocker mobile ransomware crosses to smart tv," *TrendMicro*. accessed: May 1, 2022. [Online]. Available: "https://www.trendmicro.com/en_us/research/16/f/flocker-ransomware-cro

sses-smart-tv.html"

[42] A. Tierney, "Thermostat ransomware: a lesson in iot security," *PenTestPartners*. accessed: May 1, 2022. [Online]. Available: "https://www.pentestpartners.com/security-blog/thermostat-ransomware-a-lesson-in-iot-security/"

[43] S. Margaritelli, "Reversing the smarter coffee iot machine protocol to make coffee using the terminal," *EvilSocket*. accessed: May 1, 2022. [Online]. Available: "https://www.evilsocket.net/2016/10/09/IoCOFFEE-Reversing-the-Smarter-Coffee-IoT-machine-protocol-to-make-coffee-using-terminal/index.html"

[44] P. Morgner and Z. Benenson, "Exploring security economics in iot standardization efforts," *arXiv preprint arXiv:1810.12035*, 2018.

[45] A. Pal, H. K. Rath, S. Shailendra, and A. Bhattacharyya, "Iot standardization: the road ahead," *Internet of Things-Technology, Applications and Standardization*, pp. 53–74, 2018.

[46] A. Pal and B. Purushothaman, *IoT technical challenges and solutions*. Artech House, 2016.

[47] M. El-Hajj, A. Fadlallah, M. Chamoun, and A. Serhrouchni, "A survey of internet of things (iot) authentication schemes," *Sensors*, vol. 19, no. 5, p. 1141, 2019.

[48] A. S. Tanenbaum and D. Wetherall, *Computer Networks*, 5th ed. Boston: Prentice Hall, 2011. [Online]. Available: https://www.safaribooksonline.com/library/view/computer-networks-fifth/9780133485936/

[49] M. M. Alani, "Osi model," in *Guide to OSI and TCP/IP Models*. Springer, 2014, pp. 5–17.

[50] K. James F. and K. W. Ross, *Computer Networking: A Top-Down Approach*. Pearson, 2001.

[51] C. Meinel and H. Sack, "The foundation of the internet: Tcp/ip reference model," in *Internetworking*. Springer, 2013, pp. 29–61.

[52] D. Groth and T. Skandier, *Network+ study guide*. SYBEX Inc., 2005.

[53] R. Jiang, "A review of network topology," in *4th International Conference on*

Computer, Mechatronics, Control and Electronic Engineering. Atlantis Press, 2015, pp. 1167–1170.

[54] F. Nielsen, "Topology of interconnection networks," in *Introduction to HPC with MPI for Data Science*. Springer, 2016, pp. 63–97.

[55] M. O. Ojo, S. Giordano, G. Procissi, and I. N. Seitanidis, "A review of low-end, middle-end, and high-end iot devices," *IEEE Access*, vol. 6, pp. 70 528–70 554, 2018.

[56] J. R. Anna Gerber, "A guide to internet of things (iot) processors," *IBM*. accessed: Apr 12, 2022. [Online]. Available: https://developer.ibm.com/articles/iot-lp101-connectivity-network-protocols/

[57] J.-S. Lee, Y.-W. Su, and C.-C. Shen, "A comparative study of wireless protocols: Bluetooth, uwb, zigbee, and wi-fi," in *IECON 2007-33rd Annual Conference of the IEEE Industrial Electronics Society*. Ieee, 2007, pp. 46–51.

[58] E. Ferro and F. Potorti, "Bluetooth and wi-fi wireless protocols: a survey and a comparison," *IEEE Wireless Communications*, vol. 12, no. 1, pp. 12–26, 2005.

[59] X. Wang, Y. Ren, J. Zhao, Z. Guo, and R. Yao, "Comparison of IEEE 802.11e and IEEE 802.15.3 MAC," in *Proceedings of the IEEE 6th Circuits and Systems Symposium on Emerging Technologies: Frontiers of Mobile and Wireless Communication (IEEE Cat. No.04EX710)*, vol. 2, 2004, pp. 675–680 Vol.2.

[60] N. Baker, "Zigbee and bluetooth: Strengths and weaknesses for industrial applications," *Computing & Control Engineering Journal*, vol. 16, pp. 20 – 25, 05 2005.

[61] A. Gerber and J. Romeo, "Connecting all the things in the internet of things," *IBM Developer.—2020.—URL: https://developer. ibm. com/technologies/iot/articles/iot-lp101-connectivity-networkprotocols/(accessed: 30.03. 2021)*, 2017.

[62] M. Park, "Ieee 802.11ah: sub-1-ghz license-exempt operation for the internet of things," *IEEE Communications Magazine*, vol. 53, no. 9, pp. 145–151, 2015.

[63] E. J. Oughton, W. Lehr, K. Katsaros, I. Selinis, D. Bubley, and J. Kusuma, "Revisiting wireless internet connectivity: 5g vs wi-fi 6," *Telecommunications Policy*, vol. 45, no. 5, p. 102127, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S030859612100032X

[64] J. Haxhibeqiri, E. De Poorter, I. Moerman, and J. Hoebeke, "A survey of lorawan for iot: From technology to application," *Sensors*, vol. 18, no. 11, p. 3995, Nov 2018. [Online]. Available: http://dx.doi.org/10.3390/s18113995

[65] C. Gomez, J. Paradells, C. Bormann, and J. Crowcroft, "From 6lowpan to 6lo: Expanding the universe of ipv6-supported technologies for the internet of things," *IEEE Communications Magazine*, vol. 55, no. 12, pp. 148–155, 2017.

[66] T. Dierks and E. Rescorla, "Rfc 5246-the transport layer security (tls) protocol version 1.2," *The Internet Engineering Task Force (IETF)*, 2008.

[67] S. Vandeven, "Ssl/tls: What's under the hood," *SANS Institute InfoSec Reading Room*, vol. 13, 2013.

[68] E. Rescorla, N. Modadugu *et al.*, "Rfc 4347: Datagram transport layer security," *IETF, Request For Comments*, 2006.

[69] J. Postel, "Rfc0768: User datagram protocol," 1980.

[70] O. S. I. A. Errata, "Mqtt version 3.1. 1 plus errata 01," 2015.

[71] V. Lampkin, W. T. Leong, L. Olivera, S. Rawat, N. Subrahmanyam, R. Xiang, G. Kallas, N. Krishna, S. Fassmann, M. Keen *et al.*, *Building smarter planet solutions with mqtt and ibm websphere mq telemetry*. IBM Redbooks, 2012.

[72] M. Singh, M. Rajan, V. Shivraj, and P. Balamuralidhar, "Secure mqtt for internet of things (iot)," in *2015 fifth international conference on communication systems and network technologies*. IEEE, 2015, pp. 746–751.

[73] N. Naik, "Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP," in *2017 IEEE International Systems Engineering Symposium (ISSE)*, 2017, pp. 1–7.

[74] A. Foster, "Messaging technologies for the industrial internet and the internet of things," *PrismTech Whitepaper*, vol. 21, 2015.

[75] J. E. Luzuriaga, M. Perez, P. Boronat, J. C. Cano, C. Calafate, and P. Manzoni, "A comparative evaluation of AMQP and MQTT protocols over unstable and mobile networks," in *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, 2015, pp. 931–936.

[76] Z. Shelby, K. Hartke, C. Bormann, and B. Frank, "Rfc 7252: The constrained application protocol (coap)," *Internet Engineering Task Force*, 2014.

[77] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Rfc2616: Hypertext transfer protocol–http/1.1," 1999.

[78] V. Karagiannis, P. Chatzimisios, F. Vazquez-Gallego, and J. Alonso-Zarate, "A survey on application layer protocols for the internet of things," *Transaction on IoT and Cloud computing*, vol. 3, no. 1, pp. 11–17, 2015.

[79] A. Voica, "A guide to internet of things (iot) processors," *MIPS*. accessed: Apr 11, 2022. [Online]. Available: https://www.mips.com/blog/a-guide-to-iot-processors

[80] C. Zevala, "Iot memory: An overview of the options," *JAXEnter*. accessed: Apr 11, 2022. [Online]. Available: https://jaxenter.com/iot-memory-overview-options-131270.html

[81] C. Bormann, M. Ersue, and A. Keranen, "Terminology for constrained-node networks," *Internet Engineering Task Force (IETF): Fremont, CA, USA*, pp. 2070–1721, 2014.

[82] I. Andrea, C. Chrysostomou, and G. Hadjichristofi, "Internet of things: Security vulnerabilities and challenges," in *2015 IEEE symposium on computers and communication (ISCC)*. IEEE, 2015, pp. 180–187.

[83] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, and B. Sikdar, "A survey on iot security: application areas, security threats, and solution architectures," *IEEE Access*, vol. 7, pp. 82 721–82 743, 2019.

[84] O. Novo, "Blockchain meets iot: An architecture for scalable access management in iot," *IEEE internet of things journal*, vol. 5, no. 2, pp. 1184–1195, 2018.

[85] P. Lv, L. Wang, H. Zhu, W. Deng, and L. Gu, "An iot-oriented privacy-preserving publish/subscribe model over blockchains," *IEEE Access*, vol. 7, pp. 41 309–41 314, 2019.

[86] U. Javaid, M. N. Aman, and B. Sikdar, "Blockpro: Blockchain based data provenance and integrity for secure iot environments," in *Proceedings of the 1st Workshop on Blockchain-enabled Networked Sensor Systems*, 2018, pp. 13–18.

[87] K. R. Ozyilmaz and A. Yurdakul, "Designing a blockchain-based iot with ethereum, swarm, and lora: the software solution to create high availability with minimal security risks," *IEEE Consumer Electronics Magazine*, vol. 8, no. 2, pp. 28–34, 2019.

[88] V. Sharma, "An energy-efficient transaction model for the blockchain-enabled internet of vehicles (iov)," *IEEE Communications Letters*, vol. 23, no. 2, pp. 246–249, 2018.

[89] "How blockchain can change the future of iot," *VentureBeat*. accessed: Apr 15, 2022. [Online]. Available: https://venturebeat.com/2016/11/20/how-blockchain-can-change-the-future-of-iot/#:~:text=Blockchain%20technology%20will%20enable%20the,as%20device%20spoofing%20and%20impersonation.

[90] K. Pavani and A. Damodaram, "Intrusion detection using mlp for manets," 2013.

[91] R. V. Kulkarni and G. K. Venayagamoorthy, "Neural network based secure media access control protocol for wireless sensor networks," in *2009 international joint conference on neural networks*. IEEE, 2009, pp. 1680–1687.

[92] L. Xiao, C. Xie, T. Chen, H. Dai, and H. V. Poor, "A mobile offloading game against smart attacks," *IEEE Access*, vol. 4, pp. 2281–2291, 2016.

[93] L. Xiao, Q. Yan, W. Lou, G. Chen, and Y. T. Hou, "Proximity-based security techniques for mobile users in wireless networks," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 12, pp. 2089–2100, 2013.

[94] L. Xiao, Y. Li, G. Han, G. Liu, and W. Zhuang, "Phy-layer spoofing detection with reinforcement learning in wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 12, pp. 10 037–10 047, 2016.

[95] C. Shi, J. Liu, H. Liu, and Y. Chen, "Smart user authentication through actuation of daily activities leveraging wifi-enabled iot," in *Proceedings of the 18th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2017, pp. 1–10.

[96] L. Xiao, X. Wan, and Z. Han, "Phy-layer authentication with multiple landmarks with reduced overhead," *IEEE Transactions on Wireless Communications*, vol. 17, no. 3, pp. 1676–1687, 2017.

[97] Z. Yan, P. Zhang, and A. V. Vasilakos, "A survey on trust management for internet of things," *Journal of network and computer applications*, vol. 42, pp. 120–134, 2014.

[98] C. Li and G. Wang, "A light-weight commodity integrity detection algorithm based on chinese remainder theorem," in *2012 IEEE 11th international conference on trust, security and privacy in computing and communications*. IEEE, 2012, pp. 1018–1023.

[99] K. Spirina, "Biometric authentication: The future of iot security solutions," *VentureBeat*. accessed: Apr 15, 2022. [Online]. Available: "https://www.iotevolutionworld.com/iot/articles/438690-biometric-authentication-future-iot-security-solutions.htm"

[100] G. Premsankar, M. Di Francesco, and T. Taleb, "Edge computing for the internet of things: A case study," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1275–1284, 2018.

[101] L. Rosencrance, "6 significant issues that edge computing in iot solves," *IoTAgenda*. accessed: Apr 15, 2022. [Online]. Available: "https://www.techtarget.com/iotagenda/feature/6-significant-issues-that-edge-computing-in-IoT-solves"

[102] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2017.

[103] R. L. Rivest, *Cryptography*. Elsevier, 1990, vol. 1, ch. 13, pp. 717–755.

[104] J. Daemen, "Aes proposal : Rijndael," 1998.

[105] C. Paar and J. Pelzl, *Understanding cryptography: a textbook for students and practitioners*. Springer Science & Business Media, 2009.

[106] V. Nachef, J. Patarin, and E. Volte, "Feistel ciphers," *Cham: Springer International Publishing*, 2017.

[107] W. J. Buchanan, S. Li, and R. Asif, "Lightweight cryptography methods," *Journal of Cyber Security Technology*, vol. 1, no. 3-4, pp. 187–201, 2017.

[108] P. Gaudry, "Integer factorization and discrete logarithm problems," *Les cours du CIRM*, vol. 4, no. 1, pp. 1–20, 2014.

[109] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, p. 120–126, feb 1978. [Online]. Available: https://doi.org/10.1145/359340.359342

[110] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology — EUROCRYPT '99*, J. Stern, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 223–238.

[111] T. Elgamal, "A public key cryptosystem and a signature scheme based on

discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, 1985.

[112] A. K. Lenstra, "Key length. contribution to the handbook of information security," 2004.

[113] R. M. Needham and M. D. Schroeder, "Using encryption for authentication in large networks of computers," *Communications of the ACM*, vol. 21, no. 12, pp. 993–999, 1978.

[114] J. Katz and Y. Lindell, "Introduction to modern cryptography," 2020.

[115] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Advances in Cryptology — CRYPTO '87*, C. Pomerance, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1988, pp. 369–378.

[116] H. Krawczyk, M. Bellare, and R. Canetti, "Hmac: Keyed-hashing for message authentication," 1997.

[117] J. Oostveen, T. Kalker, and J. Haitsma, "Feature extraction and a database strategy for video fingerprinting," in *Recent Advances in Visual Information Systems*, S.-K. Chang, Z. Chen, and S.-Y. Lee, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 117–128.

[118] N. Haller, C. Metz, P. Nesser, and M. Straw, "A one-time password system," *Network Working Group Request for Comments*, vol. 2289, 1998.

[119] K. Aravindhan and R. Karthiga, "One time password: A survey," *International Journal of Emerging Trends in Engineering and Development*, vol. 1, no. 3, pp. 613–623, 2013.

[120] C. Manifavas, G. Hatzivasilis, K. Fysarakis, and K. Rantos, "Lightweight cryptography for embedded systems – a comparative analysis," in *Data Privacy Management and Autonomous Spontaneous Security*, J. Garcia-Alfaro, G. Lioudakis, N. Cuppens-Boulahia, S. Foley, and W. M. Fitzgerald, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 333–349.

[121] M. Naor and A. Shamir, "Visual cryptography," in *Advances in Cryptology — EUROCRYPT'94*, A. De Santis, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 1–12.

[122] P. S. Revenkar, A. Anjum, and W. Gandhare, "Survey of visual cryptography schemes," *International Journal of Security and Its Applications*, vol. 4, no. 2, pp. 49–56, 2010.

[123] C.-C. Chang, C.-S. Tsai, and T.-S. Chen, "A new scheme for sharing secret color images in computer network," in *Proceedings Seventh International Conference on Parallel and Distributed Systems (Cat. No. PR00568)*. IEEE, 2000, pp. 21–27.

[124] J. Lu, Z. Yang, L. Li, W. Yuan, L. Li, and C.-C. Chang, "Multiple schemes for mobile payment authentication using QR code and visual cryptography," *Mobile Information Systems*, vol. 2017, 2017.

[125] S. Roy and P. Venkateswaran, "Online payment system using steganography and visual cryptography," in *2014 IEEE Students' Conference on Electrical, Electronics and Computer Science*, 2014, pp. 1–5.

[126] Y.-C. Hou and P.-H. Huang, "Image protection based on visual cryptography and statistical property," in *2011 IEEE Statistical Signal Processing Workshop (SSP)*, 2011, pp. 481–484.

[127] Y. Shen and Y. Ye, "Visual cryptography based multiparty copyright protect scheme," in *2010 2nd International Conference on Advanced Computer Control*, vol. 2, 2010, pp. 223–226.

[128] K. Krishna Prakasha, B. Muniyal, D. Shetty *et al.*, "Multi user authentication protocol using visual sceret sharing," 2016.

[129] M. T. Arafin and G. Qu, "Secret sharing and multi-user authentication: From visual cryptography to RRAM circuits," in *Proceedings of the 26th edition on Great Lakes Symposium on VLSI*, 2016, pp. 169–174.

[130] M. Arafin and G. Qu, "RRAM based lightweight user authentication," in *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2015, pp. 139–145.

[131] S. Droste, "New results on visual cryptography," in *Advances in Cryptology — CRYPTO '96*, N. Koblitz, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 401–415.

[132] F. Harary, *Graph Theory*. CRC Press, 1994.

[133] R. A. Brualdi, *Introductory combinatorics*. Pearson Education India, 1977.

[134] F. Zhang, R. Dojen, and T. Coffey, "Comparative performance and energy consumption analysis of different aes implementations on a wireless sensor network node," *International Journal of Sensor Networks*, vol. 10, no. 4, pp. 192–201, 2011.

[135] C. Panait and D. Dragomir, "Measuring the performance and energy consumption of aes in wireless sensor networks," in *2015 Federated Conference on Computer Science and Information Systems (FedCSIS)*. IEEE, 2015, pp. 1261–1266.

CURRICULUM VITAE

Personal Information

Name and surname: : Tuğberk Kocatekin

Academic Background

Bachelor's Degree Education : Electrical and Electronics Engineering (2010)
Yeditepe University

Post Graduate Education : Computer Engineering (2013)
Bahçeşehir University

Foreign Languages : English