Obfuscation And Diversification For Securing The Internet Of Things (IOT)

INTRODUCTION

- Information sharing in Internet of Things (IoT) is the element that makes the cooperation of the devices feasible, but on the other hand, it raises concerns about the security of the collected data and the privacy of the users.
- Some of the collected data contain (sensitive) personal and business information about the users.
- Therefore, it is highly significant to appropriately protect this data while it is stored and transmitted.

INTRODUCTION (CONTINUED)

- Nonetheless, the IoT service providers principally are tackling the availability and interoperability of the IoT services, so the security of the devices and services has never been the main focus.
- A report by claims that nearly 70% of the devices participating in IoT are vulnerable to security exploits, which make the network prone to security attacks.
- Therefore, there should be effective techniques to protect these devices and the shared information over the network, in order to ensure the liability of the system in addition to the availability of it.

INTRODUCTION (CONTINUED)

- Due to the fact that IoT is based on the Internet, it is subject to the traditional security threats existing for the Internet.
- The dynamic nature of the IoT environment, along with the heterogeneity and large scale of devices, make the traditional security issues more critical and also present new security challenges.
- Additionally, devices in IoT are exceedingly constrained in capacity and computational power.
- Hence, the security measure considered is required to be lightweight, to be tolerable by the devices, and to be compatible with the limitations of the participating nodes in IoT.

INTRODUCTION (CONTINUED)

- Tothe best of our knowledge, there is no research existing that studies the security of IoT through the potential techniques of obfuscation and diversification.
- In this chapter, we propose two novel approaches to address the security threats in IoT based on these two promising techniques.
- Obfuscation and diversification have been successful in mitigating the risk of malware in various domains.

We propose:

(1) applying the two techniques, obfuscation and diversification, for protecting the operating systems and APIs of the IoT devices, and

(2) applying the two techniques on the communication protocols among the devices.

DISTINGUISHING CHARACTERISTICS OF IOT - OPERATING SYSTEMS AND SOFTWARE IN IOT

- IoT is made up of a wide variety of heterogeneous components, including sensors, devices, and actuators.
- Some of these components are supplied by more powerful 32-bit processors (eg, PCs and smartphones), whereas some others are equipped with only lightweight 8-bit micro-controllers.
- On that account, the chosen software should be compatible with all ranges of devices, including the low-powered ones.
- Moreover, the software should not only be able to support the functionality of the devices, but also should be compatible with the limitation of the participating nodes of this network, in terms of computational power, memory, and energy capacity.

DISTINGUISHING CHARACTERISTICS OF IOT – OPERATING SYSTEMS AND SOFTWARE IN IOT (CONTINUED)

The following items are the generic prerequisites of software running on IoT devices:

- Heterogeneous hardware constraints
- Programmability
- Autonomy

DISTINGUISHING CHARACTERISTICS OF IOT – OPERATING SYSTEMS AND SOFTWARE IN IOT (CONTINUED)

Heterogeneous hardware constraints:

- The chosen software for IoT should require a fairly low amount of memory, and operate with low complexity, so that the IoT devices with limited memory and computational power would be able to support the operations.
- Additionally, due to the variety of the hardware in IoT, the software should be able to support a wide range of hardware platforms, including the constrained ones.

DISTINGUISHING CHARACTERISTICS OF IOT – OPERATING SYSTEMS AND SOFTWARE IN IOT (CONTINUED)

Programmability:

- From the development point of view, the chosen software should provide a standard application-program interface (API), and should also support the standard programming languages.
- For example, the operating system should make C and C++ available, as high-level languages, for the application developers.

DISTINGUISHING CHARACTERISTICS OF IOT – OPERATING SYSTEMS AND SOFTWARE IN IOT (CONTINUED)

Autonomy:

- For energy efficiency means,

(1)the software should allow sleep cycles for saving energy when the hardware is in the idle mode;

(2)the network stack should be adaptive to the constrained characteristic of the devices in IoT, and also should allow the protocols to be replaced at each layer; and

(3) the chosen software should be sufficiently robust and reliable.

DISTINGUISHING CHARACTERISTICS OF IOT – OPERATING SYSTEMS AND SOFTWARE IN IOT (CONTINUED)

- These factors motivated the developers to construct generic software and operating systems that are compatible with all ranges of devices in IoT that have diverse capacities and capabilities.
- This means that the software, on one hand, is capable of leveraging the capabilities of the more powerful devices, and, on the other hand, can run on the power-restricted devices.
- The following table lists some of the embedded operating systems that are designed to meet the requirements of the heterogeneous constrained nodes (sensors and devices) in this network.

DISTINGUISHING CHARACTERISTICS OF IOT - OPERATING SYSTEMS AND SOFTWARE IN IOT

Table 14.1 Operating Systems for Embedded Systems

Operating System	Overview	Characteristics	Language	Open Source
Contiki	An open-source multitasking OS designed for wireless sensor network (WSN) and memory-efficient embedded systems network.	Modular structure, multithreading, event-driven.	C	✓
TinyOS	An open-source OS, intended for the low-power wireless devices	Monolithic structure, multithreading, event-driven, support for TOS threads.	NesC	✓
RIOT OS	Real time	Modular structure, multithreading	C and C++	✓
Mantis	An open-source operating system designed for WSN. It presents C API with Linux and Windows development environments.	Threads	C	✓
Nano-RK	This OS has a lightweight embedded resource kernel (RK) and networking support to be used in WSN.	Threads	C	✗
LiteOS	An open-source UNIX-like OS for WSN	Threads and events	LiteC++	✓
FreeRTOS	A real-time OS designed for embedded devices		C	✗
Linux		Monolithic structure, event-driven	C and C++	✓

DISTINGUISHING CHARACTERISTICS OF IOT – OPERATING SYSTEMS AND SOFTWARE IN IOT (CONTINUED)

- Among all, Contiki and TinyOS are the most commonly used operating systems for IoT devices.
- Contiki is an open-source operating system developed in C programming language and designed to operate on low-power memory-restricted devices.
- Contiki is considered to be a lightweight operating system and reasonably memory efficient, requiring only a few kilobytes of memory.
- By supporting various networking standards, such as IPV4, IPV6, and CoAP, it connects the low-power microcontrollers to the Internet.

DISTINGUISHING CHARACTERISTICS OF IOT - OPERATING SYSTEMS AND SOFTWARE IN IOT (CONTINUED)

- For energy efficiency means, it has a set of mechanisms that enables the system to run in a lower-power mode, which consumes less power but is still able to send and receive messages.
- For memory-efficiency purposes its design is based on a protothreads model, which is a combination of a multithreading and event-driven approach.
- Protothreads provide blocking event handlers.
- Therefore, this demonstrates that multithreading does not always have to be on the kernel's lowest levels, but can be at the application library on top of the event-driven kernel.
- Moreover, Contiki has a dynamic nature, that is, it allows the dynamic loading and unloading of applications at runtime.

DISTINGUISHING CHARACTERISTICS OF IOT – OPERATING SYSTEMS AND SOFTWARE IN IOT (CONTINUED)

- TinyOS is an open-source operating system developed in nesC, which is an extension of the C language.
- Similar to Contiki, TinyOS is multithreading and event-driven, and it is designed according to a component-based programming model and monolithic structure.
- TinyOS is specifically designed for the devices distributed in the sensor networks with limited resources, for example, 512 bytes of RAM and 8 Kbytes of program memory.

DISTINGUISHING CHARACTERISTICS OF IOT - IOT NETWORK STACK AND ACCESS PROTOCOLS

- IoT is built up of a large number of objects, including sensors, devices, and applications, connected to one another.
- There are three different types of links for connecting these objects to each other:

(1) device-to-device (D2D), which is the connection between the devices;

(2)device-to-server (D2S), which is the connection between the devices and the servers, and

(3) server-to-server (S2S), which is the connection between different servers to share collected data.

- For making the connections feasible, various communication protocols are employed.

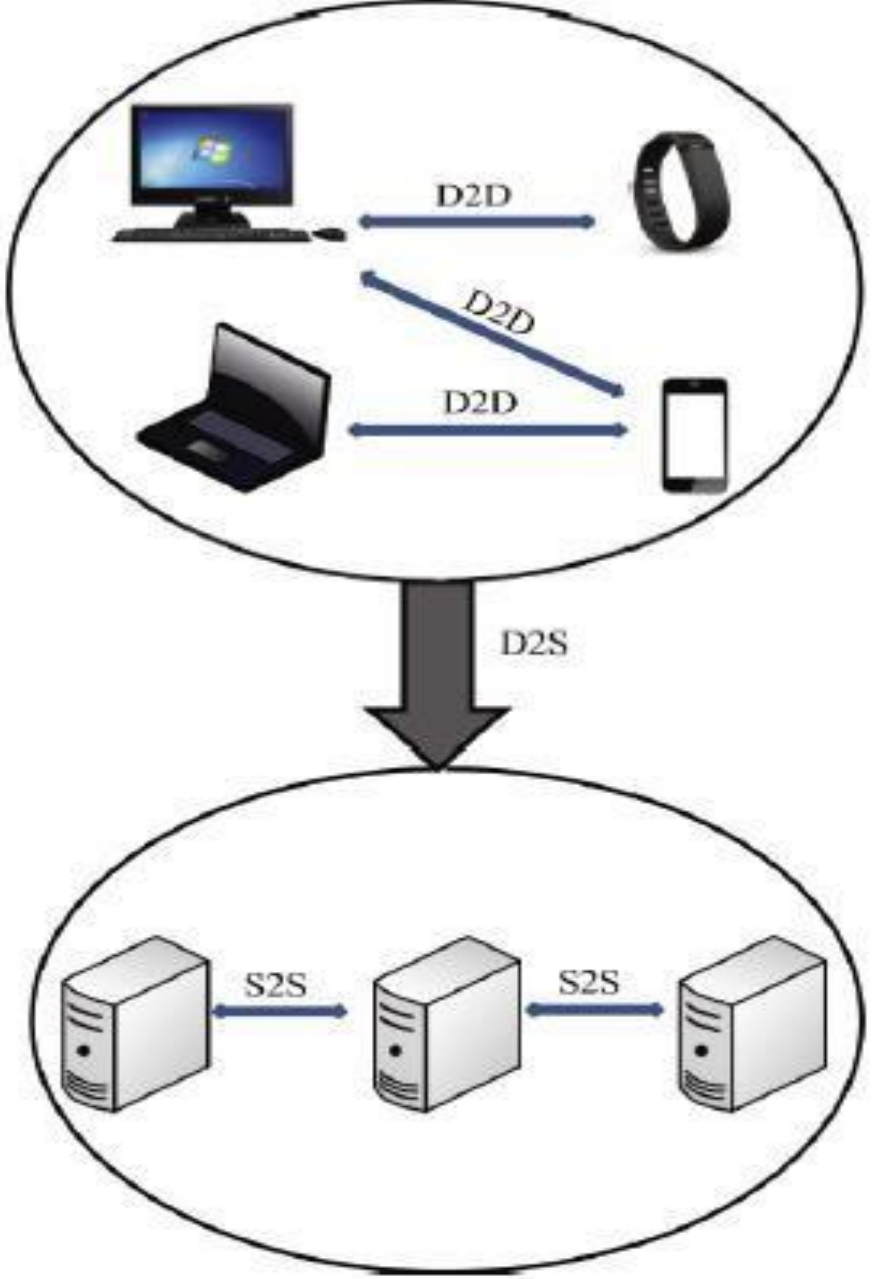

FIGURE 14.1 Communication Links

There are three different types of communication links among the different components of IoT. (A) *D2D*, device to device; (B) *D2S*, device to server; (C) *S2S*, server to server.

DISTINGUISHING CHARACTERISTICS OF IOT - IOT NETWORK STACK AND ACCESS PROTOCOLS (CONTINUED)

- Considering the TCP/IP as the de facto standard for the communication networks, some are in the opinion that it could be used also for IoT in the future, to provide flexible IP based architecture.
- Currently, the low capacity of the resource-constrained devices makes it challenging to deploy IPv6 in IoT and in Low power and Lossy Networks (LLNs).
- LLNs are types of networks in which both routers and nodes are constrained in terms of memory, energy, and processing power.
- These networks are characterized as being unstable, having a high rate of loss, and having a low data-rate.

DISTINGUISHING CHARACTERISTICS OF IOT - IOT NETWORK STACK AND ACCESS PROTOCOLS (CONTINUED)

- On that account, the Internet Engineering Task Force (IETF) has presented protocols adaptable to this environment, such as Constrained Application Protocol (CoAP) and IPv6 Routing Protocol for Low power and Lossy Networks (RPL).
- With the help of these standard protocols, the normal IP-based devices (eg, PCs and smartphones) can connect to the sensor devices.
- The developed protocols are designed in accordance with the requirements and characteristics of IoT.
- Users select the proper set of protocols based on the requirements of their applications.

DISTINGUISHING CHARACTERISTICS OF IOT - IOT NETWORK STACK AND ACCESS PROTOCOLS (CONTINUED)

- Due to the fact that components in IoT utilize various protocols to communicate to the network (eg, CoAP, MQTT, DDS, XMPP), to enable them to communicate to each other the protocols need to be translated to a standard protocol through XML, JSON, or RESTful APIs.
- These standard protocols support the scalability, interoperability, and the low power and lossy behavior of the nodes in IoT.
- The scalability deals with the issues of adding an extra node to the network, and interoperability ensures that the devices in IoT are able to communicate with each other.

DISTINGUISHING CHARACTERISTICS OF IOT - IOT NETWORK STACK AND ACCESS PROTOCOLS (CONTINUED)

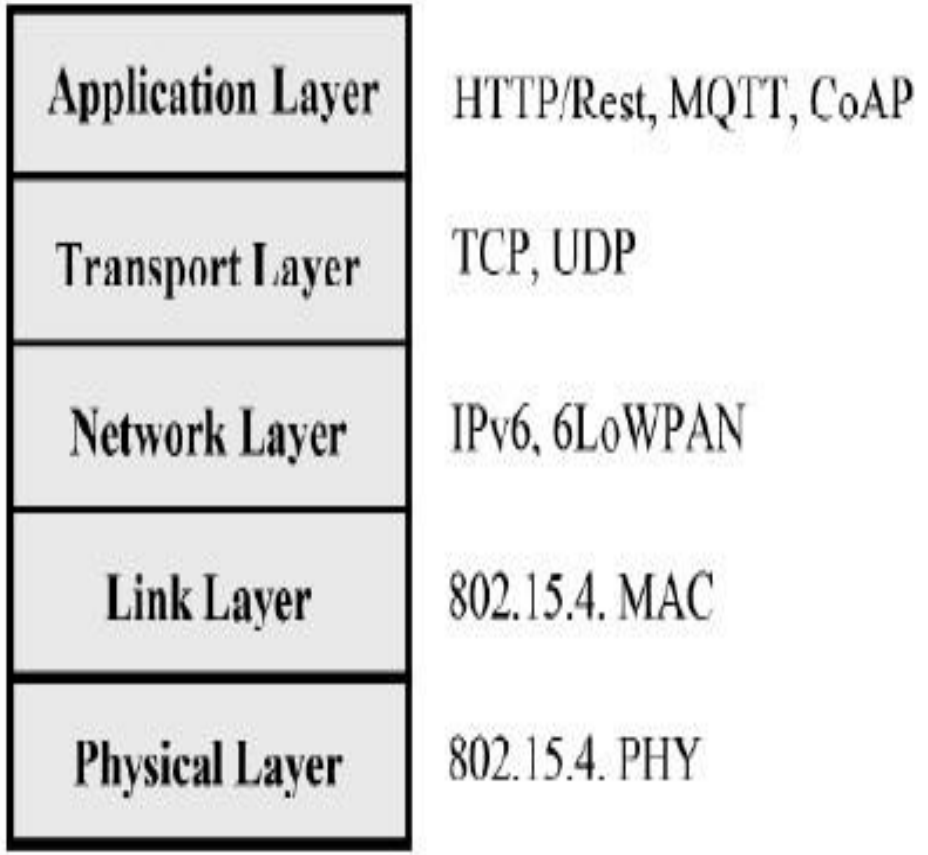

FIGURE 14.2 IoT Network Stack and Some of the Communication Protocols Used at Each Layer

DISTINGUISHING CHARACTERISTICS OF IOT - IOT NETWORK STACK AND ACCESS PROTOCOLS (CONTINUED)

- The above diagram depicts the network stack that is currently used in IoT, with some examples of the communication protocols at each layer.
- The following are the most commonly used protocols:
- **CoAP** is an application-layer protocol that is built on UDP and is used in resource constrained nodes. As HTTP is fairly complex for the LLNs, CoAP is proposed as a web-transfer protocol, which is simply translated to HTTP and simplifies the integration with the Web. Due to the fact that CoAP is designed over UDP and not TCP, the common SSL/TLS cannot be used for providing security. For this purpose, Datagram Transport Layer Security (DTLS) is available to provide the same services as TLS.

DISTINGUISHING CHARACTERISTICS OF IOT - IOT NETWORK STACK AND ACCESS PROTOCOLS (CONTINUED)

- **User Datagram Protocol (UDP) and Transmission Control Protocol (TCP)** are the protocols that are used on the Internet. TCP has proven to be quite complex for the LLNs. Thus, UDP is the most common protocol used at this layer for the LLNs.

DISTINGUISHING CHARACTERISTICS OF IOT - IOT NETWORK STACK AND ACCESS PROTOCOLS (CONTINUED)

- On the traditional Internet IP diagram, **IPv4 and IPv6** are used for controlling the messaging at the network layer. In IoT architecture, in order to allow IPv6 packets to be transmitted over IEEE 802.15.4-based networks, 6LoWPAN [15] comes as an adaptation layer between the link layer and the network layer. It presents packet fragmentation and header compression to decrease the datagram size.

DISTINGUISHING CHARACTERISTICS OF IOT - IOT NETWORK STACK AND ACCESS PROTOCOLS (CONTINUED)

- **RPL** is a routing protocol standardized for the LLNs. It supports the traffic flow between the devices of a network (point-to-point), the devices and a central node (multipoint-to-point), and the central node with devices (point-to-multipoint).

DISTINGUISHING CHARACTERISTICS OF IOT - IOT NETWORK STACK AND ACCESS PROTOCOLS (CONTINUED)

- The **MAC/PHY layers** that had been traditionally used on the Internet (eg, WiFi and Ethernet) typically had high bandwidth and required high power, which made them incompatible for IoT. IEEE 802.15.4 is the standard that is mainly used for IoT to specify the MAC layer and the physical layer. It has lower bandwidth that supports smaller packet size

DISTINGUISHING CHARACTERISTICS OF IOT - IOT NETWORK STACK AND ACCESS PROTOCOLS (CONTINUED)

- **Message Queue Telemetry Transport (MQTT)** is a publish/subscribe messaging protocol for communicating the collected data from the devices to the servers (D2S). MQTT is considered to be a many-to-many protocol that passes the messages from one client to another through a central broker. MQTT is designed on top of the TCP/IP, so the connection could be encrypted with SSL/TLS.

DISTINGUISHING CHARACTERISTICS OF IOT - IOT NETWORK STACK AND ACCESS PROTOCOLS (CONTINUED)

- **Extensible Messaging and Presence Protocol (XMPP)** is a protocol to connect the devices and their users to the server (D2S). It is based on XML (Extensible Markup Language) and provides services for instant messaging and presence functionality.

DISTINGUISHING CHARACTERISTICS OF IOT - IOT NETWORK STACK AND ACCESS PROTOCOLS (CONTINUED)

- **Data Distribution Service (DDS)** is a fast bus for connecting the publisher to the subscriber (D2D) in real-time systems. It is known as interoperable, dependable, scalable, and having high performance.

DISTINGUISHING CHARACTERISTICS OF IOT - IOT NETWORK STACK AND ACCESS PROTOCOLS (CONTINUED)

- **Advanced Message Queuing Protocol (AMQP)** is an application-layer protocol with features of flexible routing and a queuing system for connecting one server to another (S2S). It can reuse the underlying transport models, such as TCP/IP and UDP

DISTINGUISHING CHARACTERISTICS OF IOT - SECURITY AND PRIVACY IN IOT

- Considering the fact that IoT is founded on the Internet, it is susceptible to the traditional security attacks threatening the Internet.
- Furthermore, the key characteristics of IoT not only make the traditional security challenges more severe, but they also introduce new challenges.

The characteristics are

- Heterogeneity
- Low capacity
- Scale
- Wireless connection
- Embedded use
- Mobility

DISTINGUISHING CHARACTERISTICS OF IOT - SECURITY AND PRIVACY IN IOT (CONTINUED)

Heterogeneity:

- IoT embraces a diverse set of devices with different capabilities that communicate with each other.
- An extremely constrained device should open up a secure communication channel with a more powerful device, for example, a smartphone.
- Therefore, the security mechanism used (such as a cryptography approach or key management scheme) needs to be compatible with both communicating parties.

DISTINGUISHING CHARACTERISTICS OF IOT - SECURITY AND PRIVACY IN IOT (CONTINUED)

Low capacity:

- The devices in IoT are fairly limited in terms of computational power, memory, and battery capacity, which make them unable to handle complex security schemes.

Scale:

- The number of participating nodes in IoT is exceedingly growing, which makes it harder to control the collected data.

DISTINGUISHING CHARACTERISTICS OF IOT - SECURITY AND PRIVACY IN IOT (CONTINUED)

Wireless connection:

- Devices connect to the Internet through various wireless links (eg, ZigBee, Bluetooth).
- The wireless connection increases the chance of eavesdropping.
- Thus, the links need to be secured so that the intruders cannot intercept the communication.

DISTINGUISHING CHARACTERISTICS OF IOT - SECURITY AND PRIVACY IN IOT (CONTINUED)

Embedded use:

- most of the IoT devices are designed for a single purpose with different communication patterns.
- This makes it quite challenging to find a security scheme compatible with these varied patterns.

Mobility:

- The devices in IoT are mobile and are connected to the Internet through various providers.

DISTINGUISHING CHARACTERISTICS OF IOT - SECURITY AND PRIVACY IN IOT (CONTINUED)

- The aforementioned properties make IoT more prone to the security threats compared to the Internet and traditional sensor networks.

The following are the security challenges in IoT that need to be tackled

- Authentication and authorization
- Communication security
- Software vulnerabilities
- Malware
- Privacy

DISTINGUISHING CHARACTERISTICS OF IOT - SECURITY AND PRIVACY IN IOT (CONTINUED)

- The tiny sensors and devices of IoT are typically unable to handle the traditional **authentication and authorization** techniques.
- Furthermore, the existing authentication mechanisms proposed previously for the sensor networks assume that sensors are part of a network that connect to the Internet through a central gateway; however, the idea in IoT is that the nodes connect to the Internet directly.
- This makes the authentication more challenging, as the nodes need to be authenticated individually.

DISTINGUISHING CHARACTERISTICS OF IOT - SECURITY AND PRIVACY IN IOT (CONTINUED)

- **Communication security** is an important factor in any communication network to ensure the integrity of the transmitted data and to guarantee that the data will not fall into the wrong hands.
- Cryptography is a successful technique always used for securing the communication; however, typical cryptographic algorithms take up a high amount of computational power and bandwidth.

DISTINGUISHING CHARACTERISTICS OF IOT - SECURITY AND PRIVACY IN IOT (CONTINUED)

- The programming bugs caused by the developers at the development stage cause **software vulnerabilities**.
- Software vulnerabilities, if exposed, can result in security attacks.
- Program analysis is a way of discovering these vulnerabilities before the software is released, which, however, requires computing power that is not compliant with the constraints of the IoT.

DISTINGUISHING CHARACTERISTICS OF IOT - SECURITY AND PRIVACY IN IOT (CONTINUED)

- **Malware (malicious software)** is a set of instructions injected in the user's computer to manipulate the system maliciously toward the attacker's desires.
- The connectivity of the devices in IoT makes it easier for the attackers to widely propagate the malware over the network.
- The first malware instance against IoT was reported by Symantec in November 2013, which indicated the significance of having solutions to address this security issue.
- To the best of our knowledge, there has not been much research work on the malware targeting IoT.

DISTINGUISHING CHARACTERISTICS OF IOT - SECURITY AND PRIVACY IN IOT (CONTINUED)

- **Privacy** is another issue in IoT to be addressed. Because of the increase in the use of IoT in people's daily lives, more and more (personal) data is collected.
- Additionally, IoT captures the behavioral pattern of the daily actions that a user takes, in order to provide customized services based on the user's preferences.
- Hence, it is highly significant to protect all of this information, while it is either stored or transmitted over the network, in order to maintain the privacy of the user.
- Nonetheless, the existing privacy-preserving approaches, such as data anonymity and location privacy, require high-powered equipment and higher bandwidth that are not always adaptable to the IoT network.

OBFUSCATION AND DIVERSIFICATION TECHNIQUES

- **Code obfuscation** is to transform the program's code into another version, which is syntactically different but semantically the same.
- That is, the program still produces the same output although its implementation differs from the original one.
- The purpose of this transformation is to make the code more complex and difficult to understand, in order to make the malicious reverse-engineering of the code harder and costlier for the attacker.

OBFUSCATION AND DIVERSIFICATION TECHNIQUES (CONTINUED)

- The following diagram illustrates a piece of obfuscated code, which is scrambled in a way so that the purpose of the code is not easy to understand.
- Certainly, with given time and resources, the attacker may succeed in comprehending and breaking the obfuscated code; however, it requires more time and energy compared to breaking the original version.
- There have been several different obfuscation mechanisms proposed.
- Each of these mechanisms applies the obfuscation transformations at various parts of the code, and also at various phases of the program development.

OBFUSCATION AND DIVERSIFICATION TECHNIQUES (CONTINUED)

(A)

```
function setText(data) {
document.getElementById("myDiv").innerHTML = data;
}
```

(B)

```
function ghds3x(n) {
h = "\x69\u006En\u0065r\x48T\u004DL";
a="s c v o v d h e , n i";x=a.split(" ");b="gztxleWentBsyf";
r=b.replace("z",x[7]).replace("x","E").replace("s","").replace("f","I")
["repl" + "ace"]("W","m")+"d";
c="my"+String.fromCharCode(68)+x[10]+"v";
s=x[5]+x[3]+x[1]+"um"+x[7]+x[9]+"t";d=this[s][r](c);if(+![])
{ d[h]=n; } else { d[h]=c; } }
```

FIGURE 14.3

(A) Original version of a piece of JavaScript code, and (B) obfuscated version of the same code.

OBFUSCATION AND DIVERSIFICATION TECHNIQUES (CONTINUED)

- For instance, obfuscation through opaque predicates is a technique used to alter the control flow of the program.
- Opaque predicates are Boolean expressions that are always executed in the same way, and the outcome is always known for the obfuscator, but not for the attacker in priori.
- Evaluating these Boolean expressions at runtime makes the analysis of the code harder.

OBFUSCATION AND DIVERSIFICATION TECHNIQUES (CONTINUED)

- Diversification aims to make diverse unique instances of a program that are syntactically different but functionally equivalent.
- Currently, the software and operating systems are developed and distributed in a monocultural manner.
- Their identical design makes them suffer from the same types of vulnerabilities, and they are prone to the same security attacks.
- An intruder, by exploiting these vulnerabilities, can simply undertake a vast number of systems.
- Diversification, by introducing multiculturalism to the software design, aims at impeding the risk of massive-scale attacks.
- The way that a program is diversified is unique, and it is kept secret.

OBFUSCATION AND DIVERSIFICATION TECHNIQUES (CONTINUED)

- Assuming that the attacker discovers the diversification secret of one instance of the program, it can possibly attack against that specific version, whereas the others would still be safe.
- That is to say, a single-attack model does not work on several systems, and the attacker needs to design system-specific attack models.
- For this reason, diversification is considered to be a promising technique to defend against massive-scale attacks and protect the largely distributed systems.
- The following diagram shows the distribution of the uniquely diversified replicas of Program P among its users.
- The attacker, by designing an attack model, can only undertake one replica, and the other replicas are safe.

OBFUSCATION AND DIVERSIFICATION TECHNIQUES (CONTINUED)

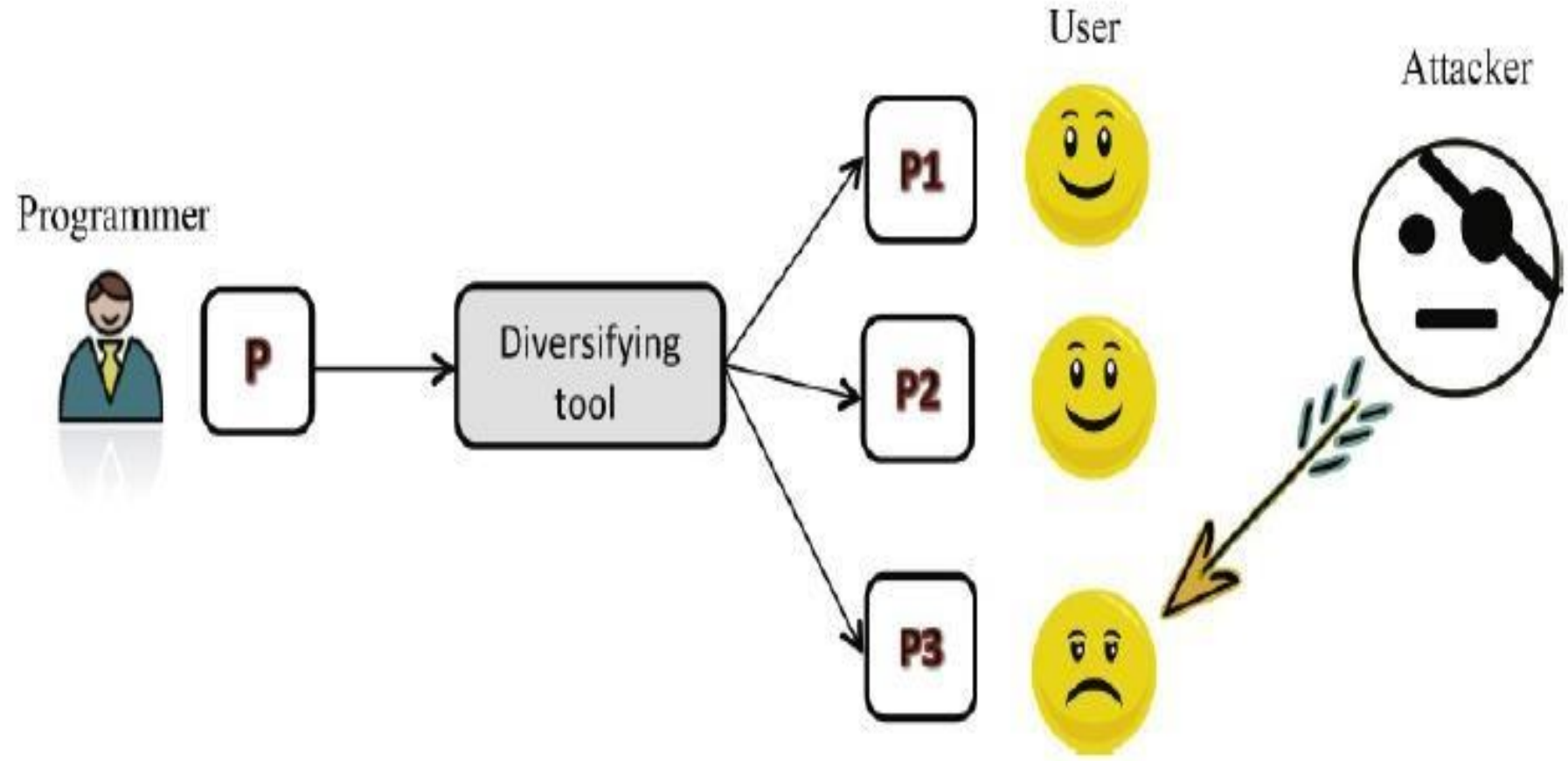

FIGURE 14.4 Diversification Generates and Distributes Unique Replicas of a Program

Thus, if an attacker manages to attack one copy of the software, the other copies are safe.

OBFUSCATION AND DIVERSIFICATION TECHNIQUES (CONTINUED)

- Program bugs caused by the developers at the time of development are unavoidable and lead to software vulnerabilities.
- In theory, the injected malware uses the knowledge it has picked up from the system's vulnerabilities to run its code.
- Diversification of the internal system interfaces makes it difficult for the malware to gain knowledge about the vulnerabilities of the system and exploit those vulnerabilities to perform an attack, as code-injection attacks are based on using some knowledge of the internal implementation details.

OBFUSCATION AND DIVERSIFICATION TECHNIQUES (CONTINUED)

- Moreover, after diversification, because the malware does not have enough knowledge about the interfaces, it is harder for it to call them and execute its malicious code.
- Thus, eventually, the malware becomes ineffective.
- The general idea in code obfuscation and program diversification is not to remove the vulnerabilities of the software, but to elude the attacker from taking advantage of them.
- Some of these vulnerabilities are not even known at the time of the software release.
- These techniques help with preventing the zero-day type of attacks that take advantage of the unknown vulnerabilities.

ENHANCING THE SECURITY IN IOT USING OBFUSCATION AND DIVERSIFICATION TECHNIQUES

- The majority of the security threats in IoT base their exploits on the vulnerabilities that exist at the application layer and the network layer.
- The vulnerabilities caused at the development phase of the applications and software are unavoidable, and some of them remain unknown until an attack occurs (ie, zero-day attacks).
- Therefore, there should be security measures considered to prevent intruders from taking advantage of these vulnerabilities.

ENHANCING THE SECURITY IN IOT USING OBFUSCATION AND DIVERSIFICATION TECHNIQUES (CONTINUED)

- Two novel techniques that make it difficult for an attacker to exploit the system's vulnerabilities to conduct a successful attack.

We propose

(1)obfuscating/diversifying the operating systems and APIs used in the IoT devices, and

(2)obfuscating/diversifying some of the access protocols among nodes of this network.

- The earlier approach secures the IoT at the application layer, whereas the later introduces security at the network layer.

ENHANCING THE SECURITY IN IOT USING OBFUSCATION AND DIVERSIFICATION TECHNIQUES (CONTINUED)

- The sensors and devices participating in IoT function with the help of the operating systems and software on them, and, like any other software system, they have vulnerabilities that make them prone to security attacks.
- An intruder seeks to exploit existing vulnerabilities on the system by finding his or her entry to the system.
- For instance, a piece of malware can capitalize on these vulnerabilities to inject its malicious code to spy on or manipulate the targeted system.
- In our proposed approach, we do not aim at removing these vulnerabilities, but we aim at preventing or making it difficult for the attacker to learn about these vulnerabilities and to utilize them.

ENHANCING THE SECURITY IN IOT USING OBFUSCATION AND DIVERSIFICATION TECHNIQUES (CONTINUED)

- We achieve this goal by applying obfuscation and diversification techniques on the operating systems and APIs of the devices in IoT.
- Obfuscation of the operating systems and APIs make them complicated to comprehend, thus an attacker needs to spend more time and effort to understand the program in order to design an attack model.
- Diversification of the operating systems and APIs used on the devices improve the security of the devices by creating a unique internal structure for them.
- This implies that even if an attacker finds out the diversification secret for the software of one of the devices, it can undertake only that specific device, and other devices are safe.

ENHANCING THE SECURITY IN IOT USING OBFUSCATION AND DIVERSIFICATION TECHNIQUES (CONTINUED)

- This is because their operating systems and APIs are diversified with a different diversification secret, that is, although devices might have similar software with similar functionality, their interfaces are uniquely diversified and the attacker needs to design various attack models for each of these systems.
- In our previous work, through obfuscating the operating systems (eg, Linux) and diversifying the APIs, we successfully made it harder for the malware to interact with the interfaces and access the resources.
- We believe that the same approaches are effective in IoT to protect the operating systems and APIs of the devices.

ENHANCING THE SECURITY IN IOT USING OBFUSCATION AND DIVERSIFICATION TECHNIQUES (CONTINUED)

- Obfuscation makes it difficult for the malware to gain knowledge about the environment, and thus cannot interact with the resources.
- Diversification makes the software of the devices unique, thus thwarting the massive-scale attacks.
- The second part of our idea is to apply obfuscation and diversification on the access protocol of the communication links among the nodes of the network.
- The application level protocol of a network identifies the interfaces and the shared protocols that the communicating nodes use in the network.
- Protocol identification refers to identifying the protocol used in a communication session by the communicating parties.

ENHANCING THE SECURITY IN IOT USING OBFUSCATION AND DIVERSIFICATION TECHNIQUES (CONTINUED)

- Static analysis could be used to capture the protocol used in the communication and compare it to the common existing protocols.
- The knowledge gained about the protocol used can be misused by an intruder to break into the communication, to eavesdrop, or to manipulate the data being sent over the network.
- Our idea is to make it hard for an attacker to gain this knowledge and identify the protocol used.
- We propose to obfuscate the protocols, in order to make the protocol unintelligible and difficult to identify.
- Protocol obfuscation removes/scrambles the characteristics that make the protocol identifiable, such as byte sequence and packet size (eg, by making them look random).

ENHANCING THE SECURITY IN IOT USING OBFUSCATION AND DIVERSIFICATION TECHNIQUES (CONTINUED)

- Cryptography is a common way to obfuscate the protocol.
- Upon the security need and the capacity of the network, different levels of encryption could be employed.
- For instance, in Plain mode, only the headers are encrypted and the payload is transmitted unencrypted, whereas in RC4, stronger cryptography is applied, for which a more powerful attack model is required to break.
- Certainly, RC4 is a stronger approach than Plain, but consumes more CPU time.
- We propose to obfuscate the communication protocol of a subset of nodes (eg, communicating devices in a home).

ENHANCING THE SECURITY IN IOT USING OBFUSCATION AND DIVERSIFICATION TECHNIQUES (CONTINUED)

- The way in which the obfuscated protocol is kept is a secret among these nodes, in a manner that the nodes need to know the obfuscation secret, in order to be able to communicate with each other.
- Changing/complicating the form of the protocol makes it dissimilar from the typical format.
- With the help of obfuscation, we aim at generating a large number of unique, diversified protocols that function the same as, but look different from, the reference protocol.
- We have already applied this to the SQL query protocol.

MOTIVATIONS AND LIMITATIONS OF THE PROPOSED IDEAS

- The techniques, obfuscation and diversification, have been shown to bring a high level of security in various domains.
- This motivated us to propose the use of these techniques in an IoT environment to boost the security of the participating nodes of this network and also the communication links among them.
- We believe our approaches are fairly successful in impeding the risk of malicious reverse-engineering, unknown zero-day attacks, targeted attacks, and massive-scale types of attacks.

MOTIVATIONS AND LIMITATIONS OF THE PROPOSED IDEAS (CONTINUED)

In the following, we consider the advantages of the proposed ideas, and we will continue with the limitations and drawbacks that these approaches may bring along.

- Additional security at the device level
- Energy efficiency
- No complexity for the manufacturer
- Mitigates the risk of malware
- Mitigates the risk of massive-scale attacks
- Amend the update limitation in embedded devices

1. ADDITIONAL SECURITY AT THE DEVICE LEVEL

- The existing security measures in IoT mainly focus on securing the network.
- The proposed ideas present security at the device level, which is an orthogonal proactive security measure.
- In this manner, even if the malware makes its way to one node, it is stopped at that point and has no way to propagate to the whole network.
- This is because in order to communicate with the other nodes, it has to know the obfuscation method, which is secret.
- Therefore, the malware that is unable to talk to the other nodes becomes ineffective.

2. ENERGY EFFICIENCY

- The sensors and devices in IoT are quite resource-constrained, meaning that they are extremely limited with regard to computational power, memory, and energy capacity.
- The considered security measure is required to be lightweight and compatible with these limitations.
- For this reason, the strong cryptograph y mechanisms and the anti-virus programs cannot be used on these devices, due to their high complexity, energy consumption, and the impact on performance.
- API diversification will not have any substantial execution overhead, whereas protocol diversification and also obfuscation may slow down the execution to some extent.

3. NO COMPLEXITY FOR THE MANUFACTURER

- The sensors and devices in IoT contain tiny chips embedded in them, which are intolerant to a complex design.
- Our proposed security techniques do not introduce any additional complexity to the manufacturing process.

4. MITIGATES THE RISK OF MALWARE

- The participating devices in IoT function with the help of lightweight operating systems on them.
- In order to handle the operations, the operating systems execute codes.
- Similar to any other software, code execution is a potential attack surface for malicious software to access the code, read, or modify it as the attacker desires.
- To this end, the operating systems of the devices should be protected from the malicious software.

4. MITIGATES THE RISK OF MALWARE (CONTINUED)

- We believe that obfuscation and diversification of the operating systems are effective techniques to render the malware ineffective to interact with the environment and execute its code.
- To make our approach compatible with the limitations of the devices, we employ the less complicated obfuscation and diversification mechanisms, such as identifier renaming.

5. MITIGATES THE RISK OF MASSIVE-SCALE ATTACKS

- The sensors, actuators, and devices are designed, manufactured, and distributed identically in “monoculture” manner.
- This means that they are produced with similar layout, and therefore, with similar security vulnerabilities.
- Thus, an attacker, by getting knowledge about the vulnerabilities of a component, and then designing an attack model, can simply invade a large number of devices.
- We believe that diversification presents a “multicultural” behavior in software deployment, and is a potent security mechanism for a largely distributed environment such as IoT.

6. AMEND THE UPDATE LIMITATION IN EMBEDDED DEVICES

- Typically, the software on the embedded devices cannot be updated or receive the security patches.
- We believe that obfuscation and diversification techniques can protect these devices so as to be prepare them for the zero-day type of attacks, because the basic idea of these two techniques does not try to remove the vulnerabilities and security holes of the software, but it avoids (or makes it hard for) an attacker to take advantage of them.

MOTIVATIONS AND LIMITATIONS OF THE PROPOSED IDEAS (CONTINUED)

- In spite of the security advances that our proposed approach presents to the system, it has some limitations and also brings along some costs.
- Code obfuscation protects the code through scrambling and complicating it, which causes costs in terms of code-size increase and execution overhead, thus affecting the performance of the system to some extent.
- Protocol obfuscation is done by changing/complicating the form of the protocol and making it different from the default format (for instance, by changing the datatypes, states, and number of message exchanges).

MOTIVATIONS AND LIMITATIONS OF THE PROPOSED IDEAS (CONTINUED)

- Therefore, when obfuscating the communication protocol, it is required that the communicating parties are capable of supporting the obfuscated protocol, that is, that they know the obfuscation secret and how the protocol is obfuscated.
- Diversification of the applications makes the development and distribution of the software more challenging.
- At the deployment phase, there comes an additional phase to diversify the software to make unique versions.
- Also, at the distribution level, managing the update patches might be a challenge.

MOTIVATIONS AND LIMITATIONS OF THE PROPOSED IDEAS (CONTINUED)

- Considering these challenges, limitations, and costs, depending on the need of the system, different levels of obfuscation and diversification could be applied to achieve the security.
- The more obfuscation/diversification is applied, the higher security and for sure the more overhead we will attain.
- There is always a trade-off between the level of security and performance.
- Taking the low capacity of the IoT devices into account, lightweight obfuscation/diversification mechanisms are the most suitable to apply, such as renaming the identifiers of callable entities in APIs and propagating the changes to legal applications of IoT devices.

DIFFERENT USE-CASE SCENARIOS ON SOFTWARE DIVERSIFICATION AND OBFUSCATION

- In the following, we will discuss two use-case scenarios in which program diversification and code obfuscation are applied together with protocol obfuscation.
- The first use-case scenario describes a security sensor network used to monitor public spaces.
- The second use-case illustrates a medical sensor network used to observe patients.

DIFFERENT USE-CASE SCENARIOS ON SOFTWARE DIVERSIFICATION AND OBFUSCATION (CONTINUED)

- Most of our modern cities are already overlooked by a legion of sensors.
- Nowadays most of these are digital video recorder (DVR) cameras mounted in the walls to monitor public streets and parks.
- Nevertheless, in the future, it is likely that a wide range of sensors will be used to observe and adjust cities.
- For example, air- and water-quality sensors can be installed into the population centers.
- Similarly, it is likely that traffic and transportation will be monitored closely in the future.
- In public areas, some sensors could be used for environmental monitoring, and some others could be used for detecting explosives.

DIFFERENT USE-CASE SCENARIOS ON SOFTWARE DIVERSIFICATION AND OBFUSCATION (CONTINUED)

- A public monitoring system is a tempting target for a hacker.
- Whereas hacking a public security sensor network would be an easy way to achieve great publicity, it is also a strategically important objective for cyber terrorists as well as criminals.
- One recent example of a security breach in these cameras was announced recently.
- The security cameras were hijacked and used to mine bitcoins.
- Thus, it is highly significant to protect these kinds of surveillance-device networks.

DIFFERENT USE-CASE SCENARIOS ON SOFTWARE DIVERSIFICATION AND OBFUSCATION (CONTINUED)

- As with all IoT devices, the security-surveillance-device networks are prone to attacks due to their limited hardware and computational capabilities.
- Hence, there is a great need for security measures that are tolerable by these constraint devices.
- Diversification of the operating systems of the devices is an achievable way to break device monoculturalism.
- With this, for example, the previously described bitcoin-mining attack could have been easily avoided: although the attackers would still have been able to capture a single device, they would not have been able to paralyze the whole network.
- Furthermore, if the program code and interfaces of sensor devices are obfuscated, then creating an attack against those devices would be an tiring task.

DIFFERENT USE-CASE SCENARIOS ON SOFTWARE DIVERSIFICATION AND OBFUSCATION (CONTINUED)

- Another interesting possibility enabled by IoT technologies is remote health-monitoring.
- The technology offers a wide range of different vital signs that can be monitored.
- For example, blood pressure, heart rate, blood glucose level, and other signs can be remotely observed.
- These kinds of systems could, for example, send a notification in case of an emergency or collect information to a database for later use.
- In countries where the average age of citizens has been constantly growing, the remote health-monitoring would be able to bring efficiency and cost savings to the healthcare districts and societies.

DIFFERENT USE-CASE SCENARIOS ON SOFTWARE DIVERSIFICATION AND OBFUSCATION (CONTINUED)

- In the domain of healthcare, privacy and security are crucial.
- In remote health monitoring, securing the confidentiality of information is an important aspect.
- Although diversification and obfuscation help to protect the monitors against large-scale attacks, securing the privacy of the communication between the end-points is still an open question.
- In our use case, we propose utilizing the protocol obfuscation in addition to other techniques to secure the communication.
- The protocol obfuscation makes the communication between end-points more arbitrary and harder to break than using only, for example, cryptography.

DIFFERENT USE-CASE SCENARIOS ON SOFTWARE DIVERSIFICATION AND OBFUSCATION (CONTINUED)

- There are, however, certain open problems that need to be addressed when utilizing diversification in IoT networks.
- First, what must be resolved is how the controlling unit knows and stores the way that each monitor is diversified, and how it should be contacted.
- Second, although the program code diversification adds only a little, if any, performance penalty, it is currently not known as to what these penalties might be, which are caused by the protocol obfuscation.