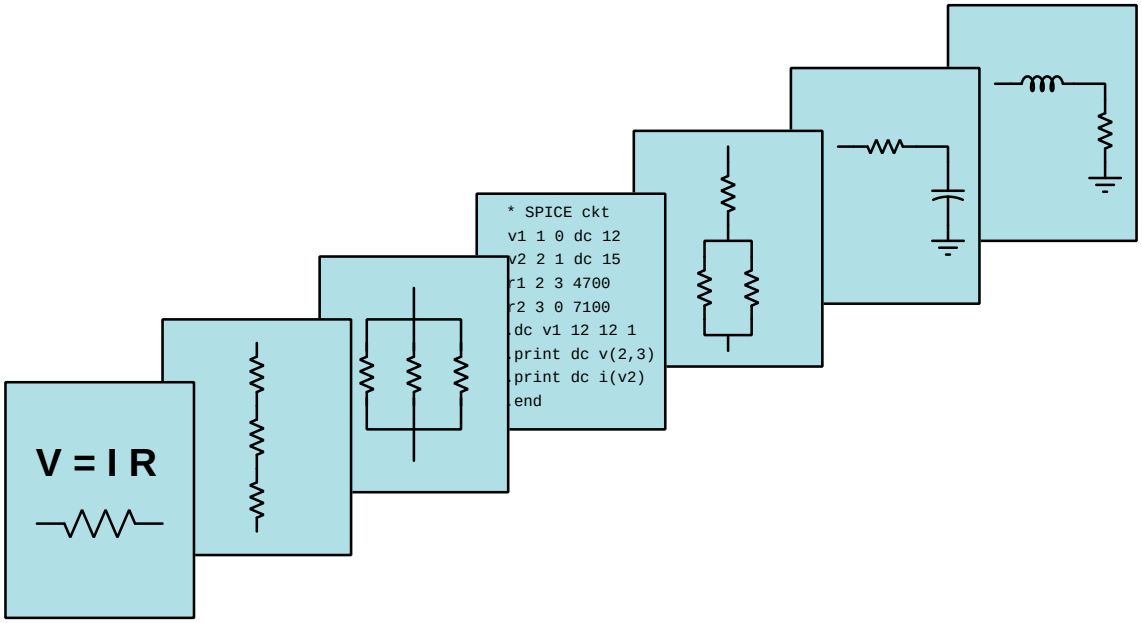
Modular Electronics Learning (ModEL) project



Digital Security

© 2016-2022 by Tony R. Kuphaldt – under the terms and conditions of the Creative Commons Attribution 4.0 International Public License

Last update = 29 November 2022

This is a copyrighted work, but licensed under the Creative Commons Attribution 4.0 International Public License. A copy of this license is found in the last Appendix of this document. Alternatively, you may visit `http://creativecommons.org/licenses/by/4.0/` or send a letter to Creative Commons: 171 Second Street, Suite 300, San Francisco, California, 94105, USA. The terms and conditions of this license allow for free copying, distribution, and/or modification of all licensed works by the general public.

Contents

1 Introduction **3**

2 Tutorial **5**

2.1 Industrial digital security . . . 5
2.2 Motives for industrial cyberattacks . . . 6
2.2.1 Technical challenge . . . 6
2.2.2 Profit . . . 6
2.2.3 Espionage . . . 7
2.2.4 Sabotage . . . 8
2.2.5 Terrorism . . . 9
2.3 Design-based fortifications . . . 9
2.3.1 Strong authentication . . . 10
2.3.2 Air gaps . . . 12
2.3.3 Firewalls . . . 14
2.3.4 Demilitarized Zones . . . 18
2.3.5 Anti-virus . . . 20
2.3.6 Encryption . . . 21
2.3.7 Read-only system access . . . 23
2.3.8 Control platform diversity . . . 24
2.4 Policy-based fortifications . . . 25
2.4.1 Foster awareness . . . 25
2.4.2 Employ dedicated security personnel . . . 26
2.4.3 Utilize effective authentication . . . 27
2.4.4 Cautiously grant authorization . . . 29
2.4.5 Maintain good documentation . . . 29
2.4.6 Close unnecessary access pathways . . . 30
2.4.7 Maintain operating system software . . . 31
2.4.8 Routinely archive critical data . . . 32
2.4.9 Create response plans . . . 32
2.4.10 Limit mobile and personal device access . . . 33
2.4.11 Secure all toolkits . . . 33
2.4.12 Close abandoned accounts . . . 34

3 Historical References **35**
3.1 Spread-spectrum radio patent . . . 36
3.2 Stuxnet . . . 39
3.2.1 A primer on uranium enrichment . . . 40
3.2.2 Gas centrifuge vulnerabilities . . . 46
3.2.3 The Natanz uranium enrichment facility . . . 48
3.2.4 How Stuxnet worked . . . 49

4 Derivations and Technical References **51**
4.1 The OSI Reference Model . . . 52
4.2 Lexicon of cyber-security terms . . . 54

5 Questions **61**
5.1 Conceptual reasoning . . . 65
5.1.1 Reading outline and reflections . . . 66
5.1.2 Foundational concepts . . . 67
5.1.3 Personal computer security . . . 69
5.1.4 Microcontroller security fuse . . . 70
5.1.5 Vulnerability databases . . . 70
5.2 Quantitative reasoning . . . 71
5.2.1 Miscellaneous physical constants . . . 72
5.2.2 Introduction to spreadsheets . . . 73
5.2.3 Password strength . . . 76
5.2.4 Exclusive-OR encryption . . . 76
5.3 Diagnostic reasoning . . . 77
5.3.1 Power grid vulnerabilities . . . 78
5.3.2 Fortifying a generic SCADA system . . . 80
5.3.3 Fortifying a natural gas SCADA system . . . 82

A Problem-Solving Strategies **85**

B Instructional philosophy **87**
B.1 First principles of learning . . . 88
B.2 Proven strategies for instructors . . . 89
B.3 Proven strategies for students . . . 91
B.4 Design of these learning modules . . . 92

C Tools used **95**

D Creative Commons License **99**

E References **107**

F Version history **111**

Index **112**

Chapter 1

Introduction

As digital technology finds greater application in personal, commercial, and industrial realms, the threats to these systems from malicious intrusion also grow. Cyber-security, which used to be strictly limited to information technology (IT) systems such as those used in office and research environments (e.g. desktop computers, printers, internet routers), is now a pressing concern for all our lives in the modern world. This module seeks to explain fundamental concepts essential to digital security.

Important concepts related to digital security include: **authentication**, **authorization**, **encryption**, the **OSI Reference Model**, **defense-in-depth**, **ransomware**, **air gaps**, **firewalls**, **SCADA** systems, **denial of service** attack, **LAN**, **Access Control List**, **IP addresses**, **blacklisting** versus **whitelisting**, **ping**, **handshaking**, **deep packet inspection**, **demilitarized zones**, **man-in-the-middle** attack, **tunneling**, **spread-spectrum** radio, **data diode**, **password** strength, and **dictionary** versus **brute-force** attacks.

Here are some good questions to ask of yourself while studying this subject:

- How might an experiment be designed and conducted to measure the latency (time lag) introduced by a network firewall? What hypothesis (i.e. prediction) might you pose for that experiment, and what result(s) would either support or disprove that hypothesis?
- What are some of the motives for unauthorized entry into a computer system?
- What are some design-based protections that may be implemented into a digital system?
- What are some policy-based protections that may be implemented into a digital system?
- How are computers used in industrial applications that might be vulnerable to attack?
- What are some different styles of attacks against a digital system?
- What does a firewall do?
- Why do firewall policies typically differ between general computer networks versus industrial control system networks?
- How does a demilitarized zone (DMZ) function?

- How does a policy of "blacklisting" differ from a policy of "whitelisting" for a firewall?
- How do Virtual Private Networks work to secure communications over a public channel?

Chapter 2

Tutorial

2.1 Industrial digital security

Security of digital computer and digital network systems has been a concern of Information Technology (IT) professionals for many years, propelled in large part by the expansion of the internet, both geographically and in terms of personal computing platforms and applications. Many good references exist for this general field of concern, but this Tutorial will tend to focus on *industrial* digital systems.

Why this focus, you ask? It is generally understood that digital system security is not as strong as it needs to be in a great many industrial facilities, and also that the consequences of a "hack" on an industrial system may very well wreak greater havoc than on personal or commercial systems. Unlike general-purpose IT systems which are maintained by IT professionals well-versed in digital security principles and technologies, industrial computing systems are often maintained by people primarily charged with matters other than security. The industrial electrician who needs to configure a digital motor speed control, or the chemical engineer tasked with optimizing the efficiency of a crude oil distillation unit, is not as well-positioned as the average IT engineer or technician for identifying and mitigating security risks. The years of study and practice necessary to become a competent electrician, chemical engineer, or IT professional is daunting enough for just one of those careers. It is therefore unrealistic to expect every industrial electrician or engineer to *also* possess the full competence of an IT professional, or vice-versa. Yet, the problem to be solved in the arena of industrial cyber-security is one requiring *all* these fields of expertise.

This Tutorial is not intended to make you a digital security expert. However, it is intended to brief you on some of the fundamental principles and practical applications, so that you may productively team with IT professionals to secure industrial digital systems from malicious attacks. It is divided into two sections, one on *design-based* methods of security and another on *policy-based* methods.

2.2 Motives for industrial cyberattacks

There are multiple motives for compromising the security of an industrial control system, some of which overlap motives for attacking IT systems, and some of which are unique to the industrial world. This section details some of the reasons why people might wish to attack an industrial control system.

2.2.1 Technical challenge

Computer experts tend to be a demographic of people motivated by technical challenges and problem-solving. To this type of person, the challenge of breaking in to a computer system designed to foil intruders may be too tempting to resist.

To the person interested in compromising a digital system just for the sake of seeing whether it can be done, the reward is in achieving access, not necessarily inflicting any damage. These people are generally not a direct threat, but may pose an indirect threat if they share their expertise with others harboring sinister motives.

Other individuals motivated by the technical challenge of accessing a digital system are interested in seeing just how much havoc they can wreak once they gain access. Such individuals are analogous to *digital arsonists*, interested in starting the biggest fire that they can simply for the sake of the fire's size.

2.2.2 Profit

The major motive driving IT cyber-attacks today is *profit*: the theft of credit card and other sensitive digital information which may be sold on the black market. Criminal organizations benefit from this style of digital attack, with many attackers becoming millionaires by way of their digital exploits.

Another form of profit-driven attack is commonly called *ransomware*, where an attacker inserts malicious software on the victim's computer(s) preventing access to the system or encrypting files such that they become unusable. This malware then presents a message to the victim asking for monetary payment in exchange for normal system access.

Neither of these attacks is novel to industrial systems, and in fact are commonplace in the IT world. What is novel in industrial systems is the severity of the repercussions. One might imagine the response from an oil drilling rig's management team to ransomware preventing startup-up of a new oil well, where downtime may be in the range of millions of US dollars per day of production. Not only is the imperative to get back online stronger than it would be for a private individual whose home computer was being held ransom, but the ability for an oil company to immediately pay the attacker is much greater than any private individual.

Another potential application of the profit motive in industrial system attacks is *commodities trading*. Traders who profit from the purchase and sale of commodities produced by industrial manufacturers might stand to gain by knowing the day-to-day operational status of those manufacturers. If such people were to access the production and inventory logs residing in a facility's digital control system, for example, they may be able to make more profitable trading decisions based on this privileged information. Eavesdropping on industrial control system data therefore poses another mode of *insider trading*.

2.2.3 Espionage

Aside from gathering data from industrial systems for the direct purpose of profit, less direct motives for attacking industrial control systems exist. One such motive is the theft of proprietary process data, for example recipes and formulae for producing chemical products such as craft foods and drinks, as well as pharmaceuticals.

Special control strategies and process designs critical to the manufacture of certain products are valuable to competing organizations as well. A chemical company eager to discover how to control a temperamental new chemical reaction process might wish to sample the controller algorithms and instrument configurations used by a successful competitor. Even if these design details were not stolen outright, the attacker may gather valuable test data and learn from the developmental mistakes of their competitor, thereby saving time and money pursuing their own design.

Militaries also stand to gain from espionage of industrial measurement and control systems, since the military capabilities of other nations are founded on industrial-scale operations. A country interested in tracking the development of an adversary's nuclear weapons potential, for example, would have a motive to perform digital espionage via the control systems of those foreign nuclear facilities.

2.2.4 Sabotage

Here, at least in my view, is where cyber-security as it relates to industrial control systems becomes really interesting. The major factor distinguishing digital control system security from IT system security is the former's supervision of a real physical process. This means a control system cyber-attack has far more *direct potential for harm* than any IT cyber-attack.

Corporations and nation-states both have an interest in industrial sabotage if it means they may diminish the economic productivity of a competitor. A country, for example, whose export market is dominated by a single product may be tempted to launch cyber-attacks against facilities producing that same product in other countries, as a means to either maintain or elevate their power in the world economy. Corporations have the exact same interest, just at a different level within the global economy.

Certain activists may also have an interest in sabotaging an industrial facility. Shutting down production of a facility they deem dangerous or unethical, or perhaps just causing the company financial loss through poor product quality and/or non-compliance, are potential motivators for activists to target specific industrial processes.

Military interest in industrial sabotage is practically a "given" assumption, as such a cyber-attack merely constitutes a new type of weapon to add to their existing arsenals. Unlike conventional weapons, cyber-weapons are relatively inexpensive.

Another category of sabotage relevant to cyber-attacks is that perpetrated by *malicious insiders*. This last category is especially troubling, as it involves personnel with in-depth knowledge of the digital systems in question. This simple fact makes defense against such attacks extremely challenging, because these are people normally authorized to access the system and therefore are able to bypass most (if not all) security measures. A few notable examples of internal sabotage are listed here:

- Secret agents of foreign nations
- Recently discharged (former) employees
- Disgruntled employees within a corporation

The destructive potential of a government operative with access to critical systems needs no further explanation. Employees, however, do. An employee who gets laid off or fired may still have access to their former employer's critical systems if their system account is not promptly closed. The same is true if the company maintains a lax password policy, such as multiple people sharing a common user account. Even current employees may be motivated to sabotage their employer's systems, especially where there might be an economic advantage[1] to doing so.

[1] Consider what forms of sabotage *striking* employees might be willing to do in order to gain leverage at the bargaining table.

2.2.5 Terrorism

This last motive is especially troubling when one considers the proliferation of digital technology and the disconcerting rise of terror-related attacks around the world. The goal of terrorists is quite simply to instill terror as a means of manipulating and/or punishing perceived enemies. Driven by ideology, terrorists tend not to discriminate when selecting their targets. Like arsonists previously mentioned, success is measured by the magnitude of terror and carnage instilled by the event. Common concerns of ethics are trumped by the dictates of the ideology.

The attacks of 11 September 2001 on the twin towers of the World Trade Center in New York City taught the world how ordinary technologies and systems (in that case, fully-fueled jet passenger aircraft) may be exploited as weapons capable of killing and injuring thousands of people. Industrial process designers would do well to think in similar terms, examining their systems not just from the perspective of their intended purpose but also as potential weapons wielded by terrorists.

2.3 Design-based fortifications

A *design-based* fortification is one rooted in technical details of system architecture and functionality. Some of these are quite simple (e.g. air gaps) while others are quite complex (e.g. encryption). In either case, these fortifications are ideally implemented at the inception of a new system, and at every point of system alteration or expansion.

The design-based fortifications discussed in the following subsections are summarized here:

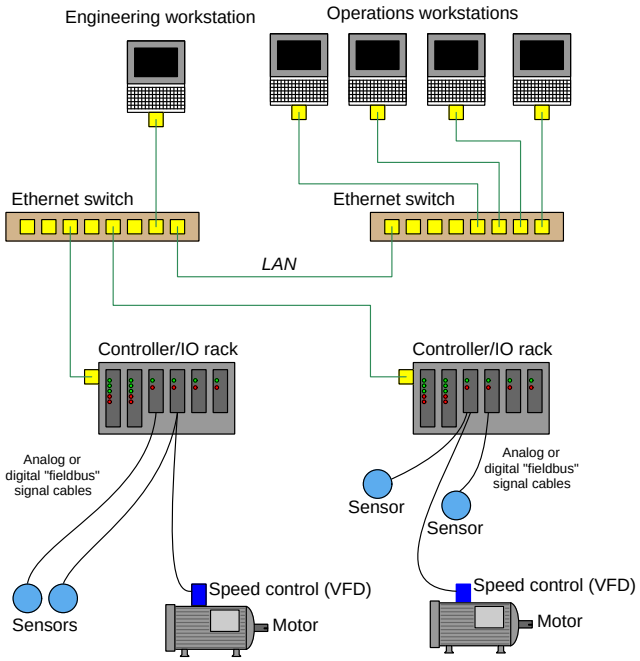
- **Authentication**: accurate identification of the person or device requesting access to a digital system
- **Air gaps**: physically separating networks to limit connectivity
- **Firewalls**: hardware and/or software designed to selectively block certain network transactions
- **Demilitarized Zones**: a technique employing multiple firewalls and *server* computers to further limit network transactions
- **Encryption**: scrambling digital messages in very specific ways to make them difficult for (only) the wrong parties to interpret
- **Read-only access**: permitting data flow in one direction only to prevent active manipulation
- **Diversity**: using different hardware/software for critical purposes so that a single-platform vulnerability cannot compromise all portions of the system

A foundational design principle in digital system security is *defense-in-depth*, which refers to a design philosophy employing multiple layers of protection. The goal of defense-in-depth is to decrease risk by designing the system in such a way that multiple things would have to fail (or become compromised) rather than any single thing. We will see how many of the design-based fortifications described in this section employ defense-in-depth.

2.3.1 Strong authentication

The authentication security provided by passwords, which is the most basic and popular form of authentication at the time of this writing, may be greatly enhanced if the system is designed to not just reject incorrect passwords, but to actively inconvenience the user for entering wrong passwords.

Consider for example the following diagram showing a simplified control system network for an industrial facility. Field instruments such as sensors and motors connect to I/O (Input/Output) modules directly connected to microprocessor-based controllers. These controllers may be SCADA[2] RTUs (Remote Terminal Units), DCS (Distributed Control System) nodes, PLCs (Programmable Logic Controllers), microcontrollers, or any other form of digital system designed to automatically measure and/or control physical processes. Human operators require access to the data collected by these controllers, and also access to parameters necessary for regulation (e.g. setpoint values), which in this case is provided by a set of personal computers called *workstations*. A separate workstation PC exists for maintenance and engineering use, loaded with appropriate software applications for accessing low-level parameters in the control system nodes and for updating control system software. In this example, Ethernet is the network standard of choice used to link all these devices together for the mutual sharing of data:



It would be wise to configure the PC workstations with password authentication to ensure only authorized personnel have access to the functions of each. The engineering workstation in particular

[2] *SCADA* is a common term used to describe industrial computer networks, where the principal purpose of the network is the exchange of data related to physical systems such as electric power control, pipeline flow measurement, etc. It is an acronym standing for *Supervisory Control And Data Acquisition*. Computer nodes located in remote areas of the SCADA system are called RTUs (Remote Terminal Units), while central computer nodes located in a control room are called MTUs (Main Terminal Units).

needs to be protected so that unauthorized personnel do not (either accidently or maliciously) alter critical parameters within the control system essential for proper regulation which could damage the process or otherwise interrupt production.

Password timeout systems introduce a mandatory waiting period for the user if they enter an incorrect password, typically after a couple of attempts so as to allow for innocent entry errors. *Password lockout* systems completely lock a user out of their digital account if they enter multiple incorrect passwords. The user's account must then be reset by another user on that system possessing high-level privileges.

The concept behind both password timeouts and password lockouts is to greatly increase the amount of time required for any dictionary-style or brute-force password attack to be successful, and therefore deter these attacks. Unfortunately timeouts and lockouts also present another form of system vulnerability to a *denial of service* attack. Someone wishing to deny access to a particular system user need only attempt to sign in as that user, using as many incorrect passwords as necessary to trip the automatic lockout. The timeout or lockout system will then delay (or outright deny) access to the legitimate user.

Authentication based on the user's knowledge (e.g. passwords) is but one form of identification, though. Other forms of authentication exist which are based on the possession of physical items called *tokens*, as well as identification based on unique features of the user's body (e.g. retinal patterns, fingerprints, facial features) called *biometric* authentication. The strength of a digital system's security will be improved through the use of *multi-factor authentication*, i.e. not relying solely on one method to authenticate users.

Token-based authentication requires all users to carry tokens on their person. This form of authentication so long as the token does not become stolen or copied by a malicious party.

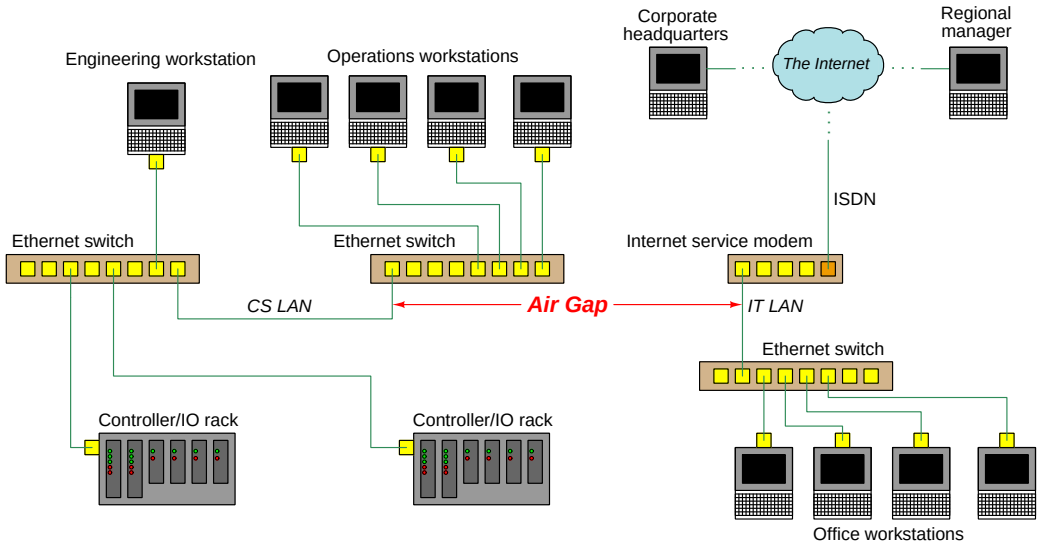
Biometric authentication enjoys the advantage of being extremely difficult to replicate and nearly[3] impossible to lose. The hardware required to scan fingerprints is relatively simple and inexpensive. Retinal scanners are more complex, but not beyond the reach of organizations possessing expensive digital assets. Presumably, there will even be DNA-based authentication technology available in the future.

[3]Before you laugh at the idea of losing one's own body, consider something as plausible as a fingerprint scanner programmed to accept the image of all fingers on one hand, and then that user suffering an injury to one of the fingers on that hand either obscuring the fingerprint or destroying the finger entirely.

2.3.2 Air gaps

An *air gap* is precisely what the name implies: a physical separation between the critical system network and any other data network preventing communication. Although it seems so simple that it ought to be obvious, an important design question to ask is whether or not the system in question really needs to have connectivity at all. Certainly, the more networked the system is, the easier it will be to access useful information and perform useful operational functions. However, connectivity is also a liability: that same convenience makes it easier for attackers to gain access.

Consider the following diagram, showing a simplified example of an industrial control system network (the Control System Local Area Network, or *CS LAN*) "air gapped" from the facility's IT network (the *IT LAN*):



The air gap between the two Ethernet-based networks not only preclude any direct data transfer between one and the other, but also ensure their respective data traffic never collides. This simple design should be used whenever possible, as it is simple and effective on multiple fronts.

While it may seem as though air gaps are the ultimate solution to digital security because they absolutely prohibit direct network-to-network data transfer, they are not invincible. A control system that *never* connects to a network other than its own is still vulnerable to cyber-attack via detachable programming and data-storage devices. For example, one of the controllers in the example control system may become compromised by way of an infected flash data drive plugged into the Engineering Workstation computer.

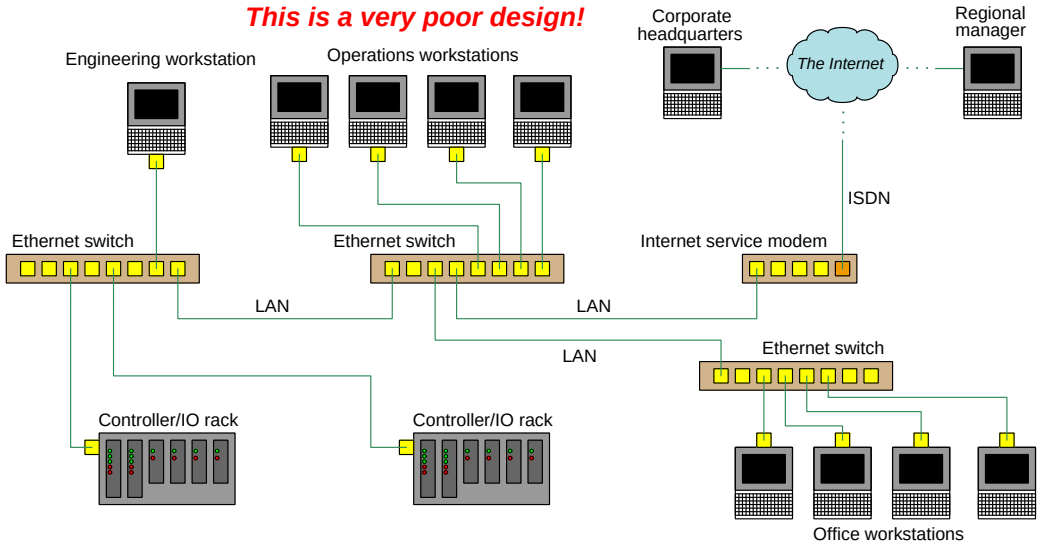
In order for air gaps to be completely effective, they must be permanent and include portable devices as well as network connections. This is where effective security policy comes into play, ensuring portable devices are not allowed into areas where they might connect (intentionally or otherwise) to critical systems. Effective air-gapping of critical networks also necessitates physical security of the network media: ensuring attackers cannot gain access to the network cables themselves, so as to "tap" into those cables and thereby gain access. This requires careful planning of cable routes and possibly extra infrastructure (e.g. separate cable trays, conduits, and access-controlled equipment rooms) to implement.

Wireless (radio) data networks pose a special problem for the "air gap" strategy, because the very purpose of radio communication is to bridge physical air gaps. A partial measure applicable to some wireless systems is to use *directional antennas* to link separated points together, as opposed to *omnidirectional* antennas which transmit and receive radio energy equally in all directions. This complicates the task of "breaking in" to the data communication channel, although it is not 100 percent effective since no directional antenna has a perfectly focused radiation pattern, nor do directional antennas preclude the possibility of an attacker intercepting communications directly between the two antennae. Like all security measures, the purpose of using directional antennas is to make an attack *less probable*.

2.3.3 Firewalls

Digital networks should be separated into different *areas* or *layers* in order to reduce their exposure to sources of harm. A network "air gap" is an extreme form of network segregation, but is impractical when some data must be communicated between networks.

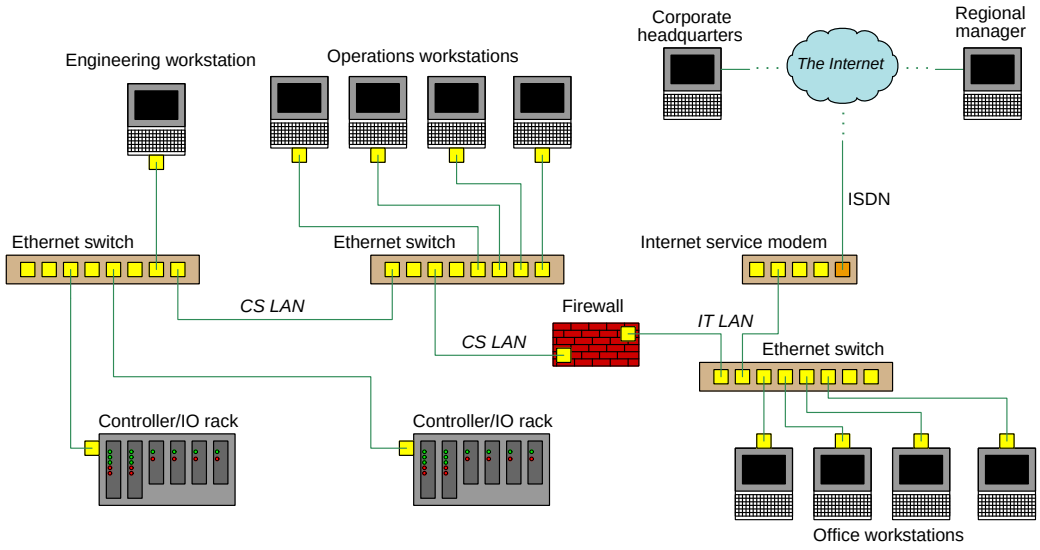
At the opposite end of the network segregation spectrum is a scenario where all digital devices, control systems and office computers alike, connect to the facility's common Local Area Network (LAN). This is a universally bad policy, as it invites a host of problems not limited to cyber-attacks but extending well beyond that to innocent mistakes and routine faults which may compromise system integrity. At the very least, control systems deserve their own dedicated network(s) on which to communicate, free of traffic from general information technology (IT) office systems. The following illustration shows a very poorly-designed network for an industrial facility, where all computers share a common LAN, and are all connected to the internet:



In facilities where control system data absolutely must be shared on the general LAN, or shared with an external network such as a WAN or the internet, a *firewall* should be used to connect those two networks. Firewalls are either software or hardware entities designed to filter data passed through based on pre-set rules. These rules are stored in a list called an *Access Control List*, or *ACL*. In essence, each network on either side of a firewall is a "zone" of communication, while the firewall is a "conduit" between zones allowing only certain types of messages through. A rudimentary firewall might be configured to "blacklist" any data packets carrying hyper-text transfer protocol (HTTP) messages, as a way to prevent web-based access to the system. Alternatively, a firewall might be configured to "whitelist" only data packets carrying Modbus messages for a control system and block everything else.

Firewalls are standard in IT networks, and have been used successfully for many years. They may exist as discrete hardware devices with multiple network cable jacks (at minimum one in and one out) screening data traffic between two or more LAN segments, or as software applications running under the operating system of a personal computer to screen data traffic in and out of that PC.

A revised version of the previous industrial network diagram shows how a firewall device could be inserted in such a way as to segregate the LAN into two sub-networks, one for the control system and another for general use:



With this firewall in place between the CS and IT networks, and configured with appropriate rules governing the passage of data between the two networks, the control system will be more secure from outside eavesdropping or attack than it was before. For example, data packets received by the firewall from questionable sources on the internet may be denied, while data packets received from known-legitimate sources may be permitted. Certain destination addresses, such as the IP addresses of the controllers themselves, may be blocked from receiving any data originating on the IT LAN, since only the Operations and Engineering workstations should ever need to send data to the controllers.

Similarly, another firewall could be inserted between the IT LAN's Ethernet switch and the Internet service modem for the purpose of screening data flowing between the IT LAN and the outside world. This would add a measure of security to the facility's IT network.

Firewall configuration is an area where stark differences may be seen between the control system (CS) versus information technology (IT) worlds. In the IT world, the job of a firewall is to permit passage of an extremely diverse legitimate data traffic while blocking very specific forms of data. In the CS world, most legitimate data is of a very limited type and occurs between a very limited number of devices, while all other data is considered illegitimate. For this reason, it is more common to see IT firewalls employ a "blacklist" policy where all data is permitted *except* for types specifically blacklisted in the ACL rules, while CS firewalls commonly employ a "whitelist" policy where all data is denied unless specifically permitted in the ACL rules. The phrase *deny by default* is synonymous with whitelisting.

For example, below you will see a few "blacklist" rules taken from a typical `iptables`[4] entry for the native firewall within a Linux operating system, intended to reject ("DROP") any data packets entering the computer from an internet modem connected to Ethernet port 0 (`eth0`) bearing an IP address within any of the "private" ranges[5] specified by the Internet Corporation for Assigned Names and Numbers (ICANN):

```
iptables -A INPUT -i eth0 -s 10.0.0.0/8 -j DROP
iptables -A INPUT -i eth0 -s 172.16.0.0/12 -j DROP
iptables -A INPUT -i eth0 -s 192.168.0.0/16 -j DROP
```

The `-s` option in `iptables` specifies a *source* IP address (or range of addresses as shown above), meaning that such a rule is screening data packets based on layer 3 of the OSI Reference model. Firewalls also provide the means to screen data based on TCP ports which exist at layer 4 of the OSI model, as seen in this next example:

```
iptables -A INPUT -p tcp --dport http -j ACCEPT
iptables -A INPUT -p tcp --dport ssh -j ACCEPT
```

Here, the `-p` option (*protocol*) specifies screening based on TCP port identification, while the `--dport` (*destination port*) option specifies which TCP port will be identified. Together, these two rules tell the firewall to permit ("ACCEPT") all data packets destined for HTTP (web page) or SSH (Secure SHell) ports on external devices, and serve as examples of "whitelist" rules in an ACL.

[4] For the curious, `iptables` is an administration-level utility application for Linux operating systems, used to edit the ACL rulebase of the operating system's built-in software firewall. Each line of text in these examples is a command that may be typed manually at the command-line interface of the operating system, or more commonly written to a *script* file to be automatically read and executed upon start-up of the computer. The `-A` option instructs `iptables` to Append a new rule to the ACL. These rules are organized into groups called "chains" which are given names such as `INPUT` and `OUTPUT`. While the specific format of ACL rules are unique to each firewall, they share many common features.

[5] No device connected directly to the internet should bear an IP address within any of these three ranges, and therefore any data packets received from devices with such an address is immediately suspect.

Basic firewall behavior is based on screening packets based on IP address (either source or destination), and/or based on TCP port, and as such provide only minimal fortification against attack. An example of a crude denial of service attack thwarted by a simple firewall rule is a *ping flood*. Ping is a network diagnostic utility that is part of the ICMP (Internet Control Message Protocol) suite used to test for connection between two IP-aware devices, and it works by having the receiving device reply to the sending device's query. These queries and replies are very simple and consist of very small amounts of data, but if a device is repeatedly "pinged" by one or more machines it may become so busy answering ping requests that it cannot do anything else on the network. An example of an ACL rule thwarting this crude attack is as follows:

```
iptables -A INPUT -p icmp --icmp-type echo-request -j DROP
```

With this rule in place, the firewall will deny ("DROP") all echo-request (i.e. ping) queries. This, of course, will prevent anyone from every using `ping` to diagnose a connection to the firewalled computer, but it will also prevent ping flood attacks.

Many modern firewalls offer *stateful inspection* of data packets, referring to the firewall's ability to recognize and log the state of each connection. TCP, for example, uses a "handshaking" procedure involving simple SYN ("synchronize") and ACK ("acknowledge") messages sent back and forth between two devices to confirm a reliable connection before any transmitting any data. A stateful firewall will track the progress of this SYN/ACK "handshake" sequence and reject any data from reaching the destination device if they do not agree with the firewall's logged state of that sequence. Such screening ability filters other types of denial of service attacks which are based on exploitation of this handshake (e.g. a TCP SYN flood attack[6]).

Stateful inspection is only useful, of course, for state-based protocols such as TCP. UDP is a notably *stateless* protocol that is often used for industrial data because the protocol itself is much simpler than TCP and therefore easier to implement in limited hardware such as within the processor of a PLC.

Some specialized firewalls are manufactured specifically for industrial control systems. One such firewall at the time of this writing (2016) is manufactured by Tofino, and has the capability to screen data packets based on rules specific to industrial control system platforms such as popular PLC models. Industrial firewalls differ from general-purpose data firewalls in their ability to recognize control-specific data, which exists at layer 7 of the OSI Reference model. This is popularly referred to as *Deep Packet Inspection*, or *DPI*, because the firewall inspects the contents of each packet (not just source, destination, port, and connection state) for legitimacy.

Two significant challenges complicate Deep Packet Inspection for any industrial control system. The first challenge is that the firewall must be fluent in the control system's command structure to be able to discern between legitimate and illegitimate data. Thus, DPI firewalls must be pre-loaded with files describing what legitimate control system data and data exchange sequences looks like. Any upgrade of the control system's network involving changes to the protocol necessitates upgrading of the DPI firewall as well. The second challenge is that the firewall must perform this deep inspection fast enough that the added latency will not compromise control system performance. The more comprehensive the DPI algorithm, the longer each message will be delayed by the inspection, and

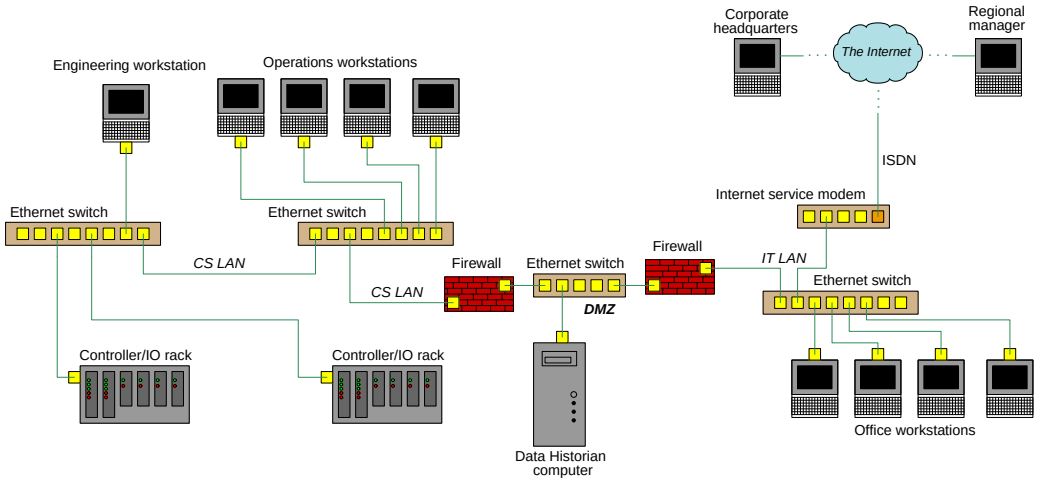
[6] If a TCP-capable device receives too many SYN ("synchronize") messages in rapid succession, it may lock up and refuse to accept any others.

the slower the network will be.

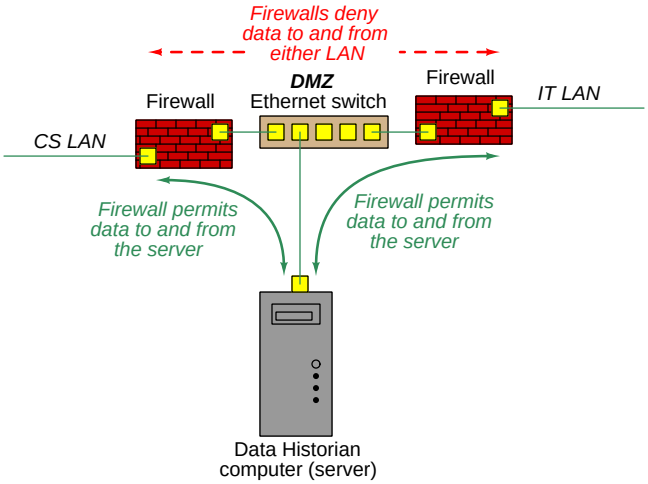
2.3.4 Demilitarized Zones

For all its complexity, a network firewall really only provides limited security in limiting data communication between two networks. This is especially true of stateless firewalls which only screen data based on such criteria as IP addresses and TCP ports. One way to augment the effectiveness of firewalls is to use multiple firewalls to build something referred to as a *DeMilitarized Zone*, or *DMZ*. A DMZ consists of three basic elements: a *data server* or *proxy* computer sandwiched between two firewalls. The purpose of a DMZ is to force all data traffic between the segregated networks to pass through the server/proxy device and prohibit any form of direct network-to-network communication. Meanwhile, the server/proxy device is programmed with limited functionality to only process legitimate data between the two networks.

An example of a DMZ applied to our industrial control system appears in the following diagram. Please note that the two firewall symbols shown here merely represent firewall *functions* and need not exist as two physical devices. It is possible to build a DMZ using a single special-purpose device with multiple Ethernet ports and dual firewall ACLs:



For each firewall at the boundary of the DMZ, its respective ACL must be configured so as to only permit ("whitelist") the Data Historian computer, which is the "server" device within the DMZ. The purpose of a Data Historian is to poll process data from the control system at regular intervals, store that data on a high-capacity data drive, and provide that data (i.e. "serve the data") to external computers[7] requesting it.



Given the fact that it's highly unlikely anyone in a corporate office needs access to real-time process data, the Data Historian's function[8] will be limited to polling and archiving process data over long periods of time, typically years. This data, when requested by any user on the IT LAN or beyond, will be provided in convenient form by the Data Historian without need for direct reading from the controllers. One way to view the function of a server inside of a DMZ is to see it as an *intentional man-in-the-middle* dispensing information strictly on a need-to-know basis.

A DMZ installed in an IT network must convey a much broader range of information, everything from email messages and web pages to large document files and bandwidth-intensive streaming video. A corporation's *web server*, for example, which is the computer upon which web page files are hosted for view by the outside world, is typically located within a DMZ network.

Any cyber-attack on a system shielded behind a DMZ must first compromise one or more of the devices lying within the DMZ before any assault may be launched against the protected network, since the dual firewalls prohibit any direct network-to-network communication. This naturally complicates the task (though, as with any fortification, it can never *prevent* a breach) for the attacker. The DMZ device's security therefore becomes an additional layer of protection to whatever fortifications exist within the protected LAN.

[7] These external computers are called *clients*, and in this network could include the office workstations as well as workstation PCs at corporate headquarters and the regional manager's office.

[8] Data Historians have existed in Distributed Control Systems (DCSs) for many years, and in fact pre-date DMZs. Their purpose during those halcyon days prior to network security concerns was to provide operations and maintenance personnel with long-term data useful for running the process and diagnosing a range of problems. DCS controllers are typically limited in memory, and simply cannot archive the vast quantities of process data capable within a general-purpose computer. Their function in modern times as part of an industrial control system DMZ is simply an extension of their original purpose.

2.3.5 Anti-virus

While firewalls restrict the flow of data in to and/or out of a digital network, another design-based fortification called *anti-virus* guards against malicious software (generally called *malware*) that makes it through other defenses. The term *virus* in this case refers to software written for a purpose contrary to the intended use of a digital system, usually equipped with the ability to replicate itself and thereby "infect" as many systems as possible. Firewalls generally cannot inspect the content of network data closely enough or with enough "intelligence" to properly identify digital viruses, and so it falls to another application to perform this task.

Correctly detecting a digital virus is no small feat, and generally depends on the anti-virus software being continually updated with a database of known digital viruses against which it may check downloaded files for matches. Without a fresh and accurate database, an anti-virus application will be helpless to detect novel threats. The pace at which new viruses are written is dauntingly swift, and so the task for anti-virus software managers is to stay as far ahead of the threats as possible, updating their user base with new database entries as often as practical.

All digital fortifications are imperfect, and as such they may make both *false positive* and *false negative* determinations. A "false positive" determination for anti-virus detection means the anti-virus application mistakenly flags a file as malicious when in fact it is safe. A "false negative" determination, by contrast, means the anti-virus software mistakenly clears a file as safe when in fact it is malware. Consumer-grade anti-virus applications generally work on the principle of *blacklisting*, which means they block recognized malware files and pass everything else. Such a policy is errs on the side of making false negatives, because any downloaded virus not in the database will be assumed safe. A more conservative approach, generally favored for industrial network systems, is to use anti-virus software with a *whitelisting* option, permitting the download of known-safe files while blocking everything else. Whitelisting errs on the side of making false positive determinations, because a safe file that isn't on the "whitelist" is assumed to be malware.

2.3.6 Encryption

Encryption refers to the intentional scrambling of data by means of a designated code called a *key*, a similar (or in some cases identical) key being used to un-scramble (decrypt) that data on the receiving end. The purpose of encryption, of course, is to foil passive attacks by making the data unintelligible to anyone but the intended recipient, and also to foil active attacks by making it impossible for an attacker's transmitted message to be successfully received.

The strength of any encryption method lies in the key used to encrypt and decrypt the protected data, and so these keys should be managed with similar care as passwords. If the key becomes known to anyone other than the intended users, the encryption becomes worthless. It is for this reason that many encryption systems provide the feature of *key rotation*, whereby keys are periodically randomized. Just prior to switching to a new key, the new key's value is communicated with all other devices in the cryptographic system as a piece of encrypted data.

A popular form of encrypted communication is a *Virtual Private Network* or *VPN*. This is where two or more devices use VPN software (or multiple VPN hardware devices) to encrypt messages sent to each other over an unsecure network. Since the data exchanged between the two computers is encrypted, the communication will be unintelligible to anyone else who might eavesdrop on that unsecure network. In essence, VPNs create a secure "tunnel" for data to travel between points on an otherwise unprotected network.

A popular use for VPN is securing remote access[9] to corporate networks, such as when business executives, salespersons, and engineers must do work while away from the facility site. By "tunneling" through to the company network via VPN, all communications between the employee's personal computer and the device(s) on the other end are unintelligible to eavesdroppers.

IP-based networks implement VPN tunneling by using an extension of the IP standard called *IPsec* (IP security), which works by encrypting the original IP packet payload and then encapsulating that encrypted data as the payload of a new (larger) IP packet. In its strongest form IPsec not only encrypts the original payload but also the original IP header which contains information on IP source and destination addresses. This means any eavesdropping on the IPsec packet will reveal nothing about the original message, where it came from, or where it's going.

Encryption may also be applied to non-broadcast networks such as telephone channels and serial data communication lines. Special cryptographic modems and serial data translators are manufactured specifically for this purpose, and may be applied to legacy SCADA and telemetry networks using on telephony or serial communication cables.

[9] Like all tools, VPN must be used with care. What follows is a cautionary tale. A controls engineer was hired to do PLC programming at an industrial facility, and the technical staff there insisted he connect his portable computer to the facility's PLC network via a VPN so that he could work via the internet. This limited his need to be on-site by ensuring he could securely upload, edit, and download code to PLC systems from any location. After completing the job and traveling to a different client to do more PLC programming work, this engineer accidently logged into the old client's VPN and placed one of their operating PLCs in Stop mode, causing a loss of control on a major process there, *hundreds of miles away from where he was*. Apart from the lesson of carefully checking login parameters when initiating a VPN connection, this example shows just how vulnerable some industrial control systems are and how over-confident some people are in tools such as VPN to protect their digital assets! Just because a VPN promises secure communication does not mean it is therefore safe to allow low-level access to control system components along public networks.

It should be noted that encryption does not necessarily protect against so-called *replay* attacks, where the attacker records a communicated message and later re-transmits that same message to the network. For example, if a control system uses an encrypted message to command a remotely-located motor to start up, an attacker might simply re-play that same message at any time in the future to force the motor to start up without having to decrypt the message. So long as the encryption key has not changed between the time of message interception and the time of message re-play, the re-played message should be interpreted by the receiving device the same as before, to the same effect. Key rotation therefore becomes an important element in fortifying simplex messages against replay attacks, because a new key will necessarily alter the message from what it was before and thereby render the old (intercepted) message meaningless.

An interesting form of encryption applicable to certain wireless (radio) data networks is *spread-spectrum* communication. This is where radio communication occurs over a range of different frequencies rather than on a single frequency. Various techniques exist for spreading digital data across a spectrum of radio frequencies, but they all comprise a form of data encryption because the spreading of that data is orchestrated by means of a cryptographic key. Perhaps the simplest spread-spectrum method to understand is *frequency-hopping* or *channel-hopping*, where the transmitters and receivers both switch frequencies on a keyed schedule. Any receiver uninformed by the same key will not "know" which channels will be used, or in what order, and therefore will be unable to intercept anything but isolated pieces of the communicated data. Spread-spectrum technology was invented during the second World War as a means for Allied forces to encrypt their radio transmissions such that Axis forces could not interpret them.

Spread-spectrum capability is built into several wireless data communication standards, including Bluetooth and *Wireless*HART.

Network communication is not the only form of data subject to encryption. Static files stored on computer drives may also be encrypted, such that only users possessing the proper key(s) may decrypt and use the files. This fortification is especially useful for securing data stored on portable media such as flash memory drives, which may easily fall into malevolent hands.

2.3.7 Read-only system access

One way to thwart so-called "active" attacks (where the attacker inserts or modifies data in a digital system to achieve malicious ends) is to engineer the system in such a way that all communicated data is *read-only* and therefore cannot be written or edited by anyone. This, of course, by itself will do nothing to guard against "passive" (read-only) attacks such as eavesdropping, but passive attacks are definitely the lesser of the two evils with regard to industrial control systems.

In systems where the digital data is communicated serially using protocols such as EIA/TIA-232, read-only access may be ensured by simply disconnecting one of the wires in the EIA/TIA-232 cable. By disconnecting the wire leading to the "receive data" pin of the critical system's EIA/TIA-232 serial port, that system cannot receive external data but may only transmit data. The same is true for EIA/TIA-485 serial communications where "transmit" and "receive" connection pairs are separate.

Certain serial communication schemes are inherently simplex (i.e. one-way communication) such as EIA/TIA-422. If this is an option supported by the digital system in question, the use of that option will be an easy way to ensure remote read-only access.

For communication standards such as Ethernet which are inherently duplex (bi-directional), devices called *data diodes* may be installed to ensure read-only access. The term "data diode" invokes the functionality of a semiconductor rectifying diode, which allows the passage of electric current in one direction only. Instead of blocking reverse current flow, however, a "data diode" blocks reverse *information* flow.

The principle of read-only protection applies to computing systems as well as communication networks. Some digital systems do not strictly require on-board data collection or modification of operating parameters, and in such cases it is possible to replace read/write magnetic data drives with read-only (e.g. optical disk) drives in order to create a system that cannot be compromised. Admittedly, applications of this strategy are limited, as there are few control systems which never store operational data nor require any editing of parameters. However, this strategy should be considered where it applies[10].

Many digital devices offer *write-protection* features in the form of a physical switch or key-lock preventing data editing. Just as some types of removable data drives have a "write-protect" tab or switch located on them, some "smart" field instruments also have write-protect switches inside their enclosures which may be toggled only by personnel with direct physical access to the device. Programmable Logic Controllers (PLCs) often have a front-panel write-protect switch allowing protection of the running program.

[10] An example of this strategy in action is an internet-connected personal computer system I once commissioned, running the Linux operating system from a DVD-ROM optical disk rather than a magnetic hard drive. The system would access the optical disk upon start-up to load the operating system kernel into its RAM memory, and then access the disk as needed for application executable files, shared library files, and other data. The principal use of this system was web browsing, and my intent was to make the computer as "hacker-proof" as I possibly could. Since the operating system files were stored on a read-only optical disk, it was impossible for an attacker to modify that data without having physical access to the machine. In order to thwart attacks on the data stored in the machine's RAM memory, I configured the system to automatically shut down and re-start every day at an hour when no one would be using it. Every time the computer re-booted, its memory would be a *tabula rasa* ("clean slate"). Of course, this meant no one could permanently store downloaded files or other data on this machine from the internet, but from a security perspective that was the very point.

Not only do write-protect switches guard against malicious attacks, but they also help prevent innocent mistakes from causing major problems in control systems. Consider the example of a PLC network where each PLC connected to a common data network has its own hardware write-protect switch. If a technician or engineer desires to edit the program in one of these PLCs from their remotely-located personal computer, that person must first go to the location of that PLC and disable its write protection. While this may be seen as an inconvenience, it ensures that the PLC programmer will not mistakenly access the wrong PLC from their office-located personal computer, which is especially easy to do if the PLCs are similarly labeled on the network.

Making regular use of such features is a policy measure, but ensuring the exclusive use of equipment with this feature is a system design measure.

2.3.8 Control platform diversity

In control and safety systems utilizing redundant controller platforms, an additional measure of security is to use different models of controller in the redundant array, or at the very least controllers running different software to achieve the same end. For example, a redundant control or safety system using two-out-of-three voting (2oo3) between three controllers might use controllers manufactured by three different vendors, each of those controllers running different operating systems and programmed using different editing software. This mitigates against device-specific attacks, since no two controllers in the array should have the exact same vulnerabilities.

A design strategy common in the nuclear and chemical-processing industries, and spreading to other industries as well, is a *layered* control system philosophy where one controller handles ordinary tasks and another entirely separate controller takes over the system in the event of an emergency. In the continuous process control industries, the "emergency" system is called a *Safety Instrumented System*, or *SIS*. An SIS system is designed to bring the process to a safe (shut-down) condition in the event that the regular control system is unable to maintain normal operating conditions. In order to avoid common-cause failures, the SIS must be implemented on a control platform independent from the regular control system. The SIS might even employ *analog* control technology (and/or discrete relay-based or gate-based control technology) in order to give it complete immunity from digital attacks.

In either case, improving security through the use of multiple, diverse control systems is another example of the *defense-in-depth* philosophy in action: building the system in such a way that no essential function depends on a single layer or single element, but rather multiple layers exist to ensure that essential function.

2.4 Policy-based fortifications

These fortifications focus on human behavior rather than system design or component selection. In some ways these are the simplest to implement, as they generally require little in the way of technical expertise. This is not to suggest, however, that policy-based fortifications are therefore the *easiest* to implement. On the contrary, changing human behavior is usually a very difficult feat. Policy-based fortifications are not necessarily cheap, either: although little capital is generally required, operational costs will likely rise as a result of these policies. This may take the form of monetary costs, additional staffing costs, and/or simply costs associated with impeding normal work flow (e.g. pulling personnel away from their routine tasks to do training, requiring personnel to spend more time doing things like inventing and tracking new passwords, slowing the pace of work by limiting authorization).

Again, a foundational design principle in digital system security is *defense-in-depth*, a design philosophy relying on multiple layers of protection to decrease risk. The goal of defense-in-depth is to mitigate risk by designing the system in such a way that multiple things would have to fail (or become compromised) rather than any single thing. The policy-based fortifications described in this section are not exclusive to each other, but always should be practiced together in order to embody a defense-in-depth strategy.

2.4.1 Foster awareness

Ensure all personnel tasked with using and maintaining the system are fully aware of security threats, and of best practices to mitigate those threats. Given the ever-evolving nature of cyber-attacks, this process of educating personnel must be continuous.

A prime mechanism of cyber-vulnerability is the casual sharing of information between employees, and with people outside the organization. Information such as passwords and network design should be considered "privileged" and should only be shared on a need-to-know basis. Critical security information such as passwords should never be communicated to others or stored electronically in plain ("cleartext") format. When necessary to communicate or store such information electronically, it should be encrypted so that only authorized personnel may access it.

In addition to the ongoing education of technical personnel, it is important to keep management personnel aware of cyber threat and threat potentials, so that the necessary resources will be granted toward cyber-security efforts.

2.4.2 Employ dedicated security personnel

Digital security is far too complex a discipline to expect people with other duties to master and implement. Although everyone has a role to play in maintaining security, it would be unwise for any organization to rely on everyone's lay knowledge of digital security to ensure safety. Instead, qualified personnel should also be dedicated to this one task, in order to orchestrate and maintain a strong security posture throughout the organization.

For any organization managing important processes and services, "important" being defined here as *threatening* if compromised by the right type of cyber-attack, it is imperative to employ qualified and diligent personnel tasked with the ongoing maintenance of digital security. These personnel must be capable of securing the control systems themselves and not just general data systems.

One of the routine tasks for these personnel should be evaluations of risks and vulnerabilities. This may take the form of security audits or even simulated attacks whereby the security of the system is tested with available tools.

2.4.3 Utilize effective authentication

Simply put, it is imperative to correctly identify all users accessing a system. This is what "authentication" means: correctly identifying the person (or device) attempting to use the digital system. *Passwords* are perhaps the most common authentication technique.

The first and foremost precaution to take with regard to authentication is to never use default (manufacturer) passwords, since these are public information. This precautionary measure may seem so obvious as to not require any elaboration, but sadly it remains a fact that too many password-protected devices and systems are found operating in industry with default passwords.

Another important precaution to take with passwords is to not use the same password for all systems. The reasoning behind this precaution is rather obvious: once a malicious party gains knowledge of that one password, they have access to all systems protected by it. The scenario is analogous to using the exact same key to unlock every door in the facility: all it takes now is one copied key and suddenly intruders have access to every room.

Passwords must also be changed on a regular basis. This provides some measure of protection even after a password becomes compromised, because the old password(s) no longer function.

Passwords chosen by system users should be "strong," meaning difficult for anyone else to guess. When attackers attempt to guess passwords, they do so in two different ways:

- Try using common words or phrases that are easy to memorize
- Try every possible combination of characters until one is found that works

The first style of password attack is called a *dictionary attack*, because it relies on a database of common words and phrases. The second style of password attack is called a *brute force attack* because it relies on a simple and tireless ("brute") algorithm, practical only if executed by a computer.

A password resistant to dictionary-style attacks is one not based on a common word or phrase. Ideally, that password will appear to be nonsense, not resembling any discernible word or simple pattern. The only way to "crack" such a password, since a database of common words will be useless against it, will be to attempt every possible character combination (i.e. a brute-force attack).

A password resistant to brute-force-style attacks is one belonging to a huge set of possible passwords. In other words, there must be a very large number of possible passwords limited to the same alphabet and number of characters. Calculating the brute-force strength of a password is a matter of applying a simple exponential function:

$$S = C^n$$

Where,

S = Password strength (i.e. the number of unique password combinations possible)

C = Number of available characters (i.e. the size of the alphabet)

n = Number of characters in the password

For example, a password consisting of four characters, each character being a letter of the English alphabet where lower- and upper-case characters are treated identically, would give the following strength:

$$S = 26^4 = 456976 \text{ possible password combinations}$$

If we allowed case-sensitivity (i.e. lower- and upper-case letters treated differently), this would double the value of C and yield more possible passwords:

$$S = 52^4 = 7311616 \text{ possible password combinations}$$

Obviously, then, passwords using larger alphabets are stronger than passwords with smaller alphabets.

2.4.4 Cautiously grant authorization

While *authentication* is the process of correctly identifying the user, *authorization* is the process of assigning rights to each user. The two concepts are obviously related, but not identical. Under any robust security policy, users are given only as much access as they need to perform their jobs efficiently. Too much access not only increases the probability of an attacker being able to cause maximum harm, but also increases the probability that benevolent users may accidently cause harm.

Perhaps the most basic implementation of this policy is for users to log in to their respective computers using the lowest-privilege account needed for the known task(s), rather than to log in at the highest level of privilege they *might* need. This is a good policy for all people to adopt when they use personal computers to do any sort of task, be it work- or leisure-related. Logging in with full ("administrator") privileges is certainly convenient because it allows you to do anything on the system (e.g. install new software, reconfigure any service, etc.) but it also means any malware accidently engaged[11] under that account now has the same unrestricted level of access to the system. Habitually logging in to a computer system with a low-privilege account helps mitigate this risk, for any accidental execution of malware will be similarly limited in its power to do harm.

Another implementation of this policy is called *application whitelisting*, where only trusted software applications are allowed to be executed on any computer system. This stands in contrast to "blacklisting" which is the philosophy behind anti-virus software: maintaining a list of software applications known to be harmful (malware) and prohibiting the execution of those pre-identified applications. Blacklisting (anti-virus) only protects against malware that has been identified and notified to that computer. Blacklisting cannot protect against "zero-day" malware known by no one except the attacker. In a whitelisting system, each computer is pre-loaded with a list of acceptable applications, and no other applications – benign or malicious – will be able to run on that machine.

Whitelisting is related to another concept in digital security called *denial-by-default*. This is where software applications and hardware alike start with default settings denying access rather than permitting access. While this means more work is necessary to get authorized users access to the system, it is a more cautious and therefore more secure posture.

2.4.5 Maintain good documentation

While this is important for effective maintenance in general, thorough and accurate documentation is especially important for digital security because it helps identify vulnerabilities. Details to document include:

- Network diagrams
- Software version numbers
- Device addresses

[11] Consider the very realistic scenario of logging in as administrator (or "root" in Unix systems) and then opening an email message which happens to carry an attached file infected with malware. Any file executed by a user is by default run at that user's level of privilege because the operating system assumes that is the user's intent.

2.4.6 Close unnecessary access pathways

All access points to the critical system must be limited to those necessary for system function. This means all other potential access points in the critical system must be closed so as to minimize the total number of access points available to attackers. Examples of access points which should be inventoried and minimized:

- Hardware communication ports (e.g. USB serial ports, Ethernet ports, wireless radio cards)
- Software TCP ports
- Shared network file storage ("network drives")
- "Back-door" accounts used for system development

That last category deserves some further explanation. When engineers are working to develop a new system, otherwise ordinary and sensible authentication/authorizations measures become a major nuisance. The process of software development always requires repeated logins, shutdowns, and tests forcing the user to re-authenticate themselves and negotiate security controls. It is therefore understandable when engineers create simpler, easier access routes to the system under development, to expedite their work and minimize frustration.

Such "back-door" access points become a problem when those same engineers forget (or simply neglect) to remove them after the developed system is released for others to use. An interesting example of this very point was the so-called `basisk` vulnerability discovered in some Siemens S7 PLC products. A security researcher named Dillon Beresford working for NSS Labs discovered a telnet[12] service running on certain models of Siemens S7 PLCs with a user account named "basisk" (the password for this account being the same as the user name). All one needed to do in order to gain privileged access to the PLC's operating system was connect to the PLC using a telnet client and enter "basisk" for the user name and "basisk" for the password! Clearly, this was a back-door account used by Siemens engineers during development of that PLC product line, but it was not closed prior to releasing the PLC for general use.

[12] Telnet is a legacy software utility used to remotely access command-line computer operating systems. Inherently unsecure, telnet exchanges login credentials (user name and password) unencrypted over the network connection. A modern replacement for telnet is SSH (Secure SHell).

2.4.7 Maintain operating system software

All operating system software manufacturers periodically release "patches" designed to improve the performance of their products. This includes patches for discovered security flaws. Therefore, it is essential for all computers belonging to a critical system to be regularly "patched" to ensure maximum resistance to attack.

This is a significant problem within industry because so much industrial control system software is built to run on consumer-grade operating systems such as Microsoft Windows. Popular operating systems are built with maximum convenience in mind, not maximum security or even maximum reliability. New features added to an operating system for the purpose of convenient access and/or new functionality often present new vulnerabilities[13].

Another facet to the consumer-grade operating system problem is that these operating systems have relatively short lifespans. Driven by consumer demand for more features, software manufacturers develop new operating systems and abandon older products at a much faster rate than industrial users upgrade their control systems. Upgrading the operating systems on computers used for an industrial control system is no small feat, because it usually means disruption of that system's function, not only in terms of the time required to install the new software but also (potentially) re-training required for employees. Upgrading may even be impossible in cases where the new operating system no longer supports features necessary for that control system[14]. This would not be a problem if operating system manufacturers provided the same long-term (multi-decade) support for their products as industrial hardware manufacturers typically do, but this is not the case for consumer-grade products such as Microsoft Windows[15].

[13] I am reminded of an example from the world of "smart" mobile telephones, commonly equipped with *accelerometer* sensors for detecting physical orientation. Accelerometers detect the force of acceleration and of gravity, and are useful for a variety of convenient "apps" having nothing to do with telephony. Smart phone manufacturers include such sensors in their mobile devices and link those sensors to the phone's operating system because doing so permits innovative applications, which in turn makes the product more desirable to application developers and ultimately consumers. It was discovered, though, that the signals generated by these accelerometers could be used to detect "keystrokes" made by the user, the sensors picking up vibrations made as the user taps their finger against the glass touch-screen of the smart phone. With the right signal processing, the accelerometers' signals could be combined in such a way to identify which characters the user was tapping on the virtual keyboard, and thereby eavesdrop on their text-based communications!

[14] An example of this is where a piece of obsolete industrial software runs on the computer's operating system, for example a data acquisition program or data-analysis program made by a company that no longer exists. If this specialized software was written to run on a particular operating system, and no others, future versions of that operating system might not permit proper function of that specialized software. I have seen such cases in industry, where industrial facilities continue to run obsolete (unsupported) operating systems in order to keep running some specialized industrial software (e.g. PLC programming editors), which is needed to operate or maintain some specialized piece of control hardware which itself is obsolete but still functions adequately for the task. In order to upgrade to a modern operating system on that computer (e.g. an obsolete version of Microsoft Windows), one must upgrade the specialized software (e.g. the PLC programming editor software), which in turn would mean upgrading the control hardware (e.g. the PLCs themselves). All of this requires time and money, much more than just what is required to upgrade the operating system software itself.

[15] As a case in point, there are still a great many industrial computers running Microsoft Windows XP at the time of this writing (2016), even though this operating system is no longer supported by Microsoft. This means no more Service Pack upgrades from Microsoft, security patches, or even research on vulnerabilities for this obsolete operating system. All users of Windows XP are "on their own" with regard to cyber-attacks.

2.4.8 Routinely archive critical data

The data input into and generated by digital control systems is a valuable commodity, and must be treated as such. Unlike material commodities, data is easily replicated, and this fact provides some measure of protection against loss from a cyber-attack. Routine "back-ups" of critical data, therefore, is an essential part of any cyber-security program. It should be noted that this includes not just operational data collected by the control system during operation, but also data such as:

- PID tuning parameters
- Control algorithms (e.g. function block programs, configuration data, etc.)
- Network configuration parameters
- Software installation files
- Software license (authorization) files
- Software drivers
- Firmware files
- User authentication files
- All system documentation (e.g. network cable diagrams, loop diagrams)

This archived data should be stored in a medium immune to cyber-attacks, such as read-only optical disks. It would be foolish, for example, to store this sort of critical data only as files on the operating drives of computers susceptible to attack along with the rest of the control system.

2.4.9 Create response plans

Just as no industrial facility would be safe without incident response plans to mitigate physical crises, no industrial facility using digital control systems is secure without response plans for cyber-attacks. This includes such details as:

- A chain of command for leading the response
- Instructions on how to restore critical data and system functions
- Work-arounds for minimal operation while critical systems are still unavailable

2.4.10 Limit mobile and personal device access

Mobile digital devices such as cell phones and even portable storage media (e.g. USB "flash" drives) pose digital security risks because they may be exploited as an attack vector bypassing air gaps and firewalls. It should be noted that version 0.5 of Stuxnet was likely inserted into the Iranian control system in this manner, through an infected USB flash drive.

A robust digital security policy will limit or entirely prohibit personal electronic devices into areas where they might connect to the facility's networks or equipment. Where mobile devices are essential for job functions, those devices should be owned by the organization and registered in such a way as to authenticate their use. Computers should be configured to automatically reject non-registered devices such as removable flash-memory storage drives. Portable computers not owned and controlled by the organization should be completely off-limits[16] from the process control system.

Above all, one should never underestimate the potential harm allowing uncontrolled devices to connect to critical, trusted portions of an industrial control system. The degree to which any portion of a digital system may be considered "trusted" is a function of *every* component of that system. Allowing connection to untrusted devices violates the confidence of that system.

2.4.11 Secure all toolkits

A special security consideration for industrial control systems is the existence of software designed to create and edit controller algorithms and configurations. The type of software used to write and edit Ladder Diagram (LD) code inside of programmable logic controllers (PLCs) is a good example of this, such as the Step7 software used to program Siemens PLCs in Iran's Natanz uranium enrichment facility. Instrumentation professionals use such software on a regular basis to do their work, and as such it is an essential tool of the trade. However, this very same software is a weapon in the hands of an attacker, or when hijacked by malicious code.

A common practice in industry is to leave computers equipped with such "toolkit" software connected to the control network for convenience. This is a poor policy, and one that is easily remedied by simply disconnecting the programming computer from the control network immediately after downloading the edited control code. An even more secure policy is to never connect such "toolkit" computers to a network at all, but only to controllers directly, so that the toolkit software cannot be hijacked.

Another layer of defense is to utilize robust password protection on the programmable control devices when available, rather than leaving password fields blank which then permits any user of the toolkit software full access to the controller's programming.

[16] This raises a potential problem from the perspective of outside technical support, since such support often entails contracted or manufacturer-employed personnel entering the site and using their work computers to perform system configuration tasks. For any organization implementing a strong security access policy, this point will need to be negotiated into every service contract to ensure all the necessary pieces of hardware and software exist "in-house" for the service personnel to use while on the job.

2.4.12 Close abandoned accounts

Given the fact that disgruntled technical employees constitute a significant security threat to organizations, it stands to reason that the user accounts of terminated employees should be closed as quickly as possible. Not only do terminated employees possess authentication knowledge in the form of user names and passwords, but they may also possess extensive knowledge of system design and vulnerabilities.

Chapter 3

Historical References

This chapter is where you will find references to historical texts and technologies related to the module's topic.

Readers may wonder why historical references might be included in any modern lesson on a subject. Why dwell on old ideas and obsolete technologies? One answer to this question is that the initial discoveries and early applications of scientific principles typically present those principles in forms that are unusually easy to grasp. Anyone who first discovers a new principle must necessarily do so from a perspective of ignorance (i.e. if you truly *discover* something yourself, it means you must have come to that discovery with no prior knowledge of it and no hints from others knowledgeable in it), and in so doing the discoverer lacks any hindsight or advantage that might have otherwise come from a more advanced perspective. Thus, discoverers are forced to think and express themselves in less-advanced terms, and this often makes their explanations more readily accessible to others who, like the discoverer, comes to this idea with no prior knowledge. Furthermore, early discoverers often faced the daunting challenge of explaining their new and complex ideas to a naturally skeptical scientific community, and this pressure incentivized clear and compelling communication. As James Clerk Maxwell eloquently stated in the Preface to his book *A Treatise on Electricity and Magnetism* written in 1873,

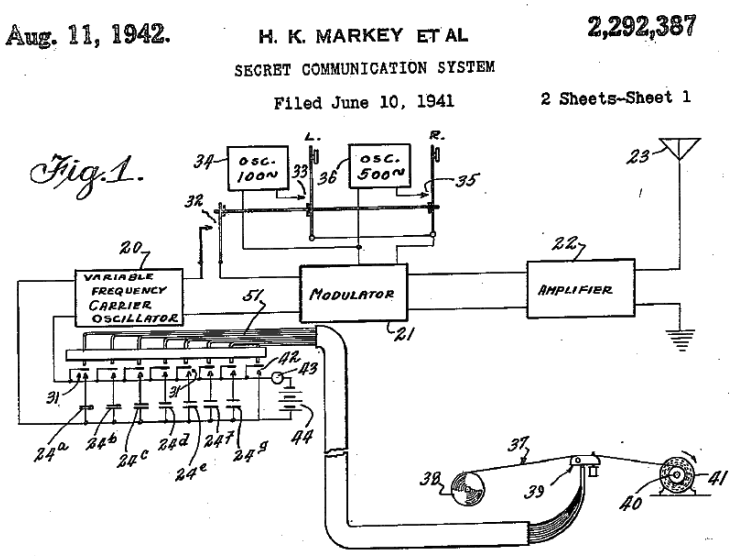
> It is of great advantage to the student of any subject to read the original memoirs on that subject, for science is always most completely assimilated when it is in its nascent state . . . `[page xi]`

Furthermore, grasping the historical context of technological discoveries is important for understanding how science intersects with culture and civilization, which is ever important because new discoveries and new applications of existing discoveries will always continue to impact our lives. One will often find themselves impressed by the ingenuity of previous generations, and by the high degree of refinement to which now-obsolete technologies were once raised. There is much to learn and much inspiration to be drawn from the technological past, and to the inquisitive mind these historical references are treasures waiting to be (re)-discovered.

3.1 Spread-spectrum radio patent

A popular method of fortifying the security of radio-based data communication is to broadcast the information over a range of different frequencies rather than on a single frequency where it may be more easily intercepted. The conceptually simplest method of spreading the information across a spectrum of frequencies ("spread-spectrum") is to have the radio transmitter (and receiver) units "hop" between defined frequencies in the same pattern at the same times. An early conception of this idea took the form of a patent filed by George Antheil and the famous actress Hedy Lamarr (born under the name Hedwig Eva Maria Kiesler, with Markey being the surname of her husband at the time of the patent's filing) in 1941.

In this patent – "Secret Communication System" (US patent number 2,292,387) filed 10 June 1941 and awarded 11 Aug 1942 – a radio transmitter circuit is shifted between different frequencies by a network of switched capacitors, these capacitors controlling the resonant frequency of the carrier oscillator. This radio-frequency oscillator signal is then modulated by one of two audio-frequency tones (100 Hz and 500 Hz), amplified, and sent to a broadcast antenna. Those two audio tones represent the information to be conveyed over the radio wave, while the "carrier" signal provides the high frequency necessary for effective electromagnetic-wave radiation at the antenna:

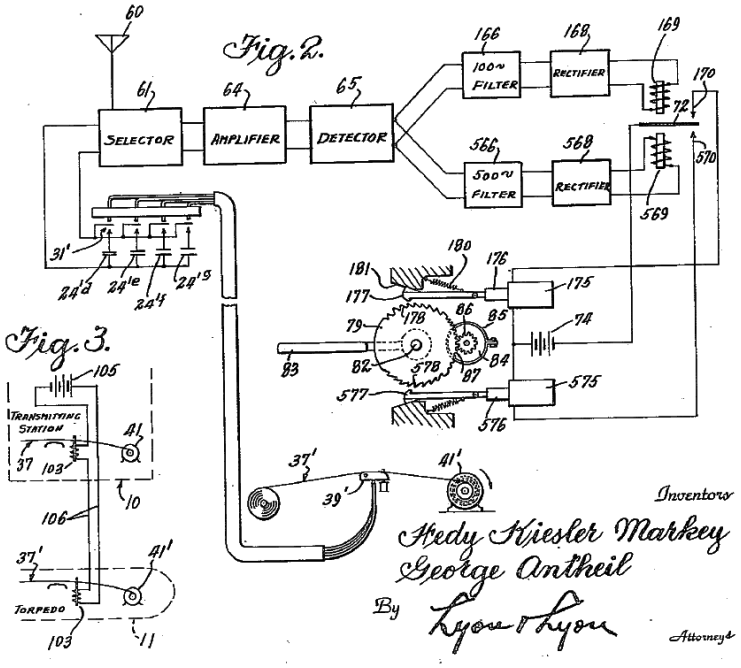


The oscillator's pattern of frequency-changes is controlled by a paper tape moved between two spools, with holes punched in the tape in a manner not unlike that of a player-piano mechanism which served as this invention's inspiration. Although "frequency-hopping" radio communication technology may be used for many purposes, these inventors intended it for the remote guidance of torpedoes to be used by the Allied forces during World War Two. In the inventors' own words:

> Briefly, our system as adapted for radio control of a remote craft, employs a pair of synchronous records, one at the transmitting station and one at the receiving station, which change the tuning of the transmitting and receiving apparatus from time to time, so that without knowledge of the records an enemy would be unable to determine at

> what frequency a controlling impulse would be sent. Furthermore, We contemplate employing records of the type used for many years in player pianos, and which consist of long rolls of paper having perforations variously positioned in a plurality of longitudinal rows along the records. In a conventional player piano record there may be 88 rows of perforations, and in our system such a record would permit the use of 88 different carrier frequencies, from one to another of which both the transmitting and receiving station would be changed at intervals. Furthermore, records of the type described can be made of substantial length and may be driven slow or fast. This makes it possible for a pair of records, one at the transmitting station and one at the receiving station, to run for a length of time ample for the remote control of a device such as a torpedo. [page 1]

The receiver is shown below, a matching paper tape controlling the "selector" which tunes into the broadcast signals. So long as the transmitter and receiver tapes had matching sequences and were synchronized to each other, they would form a radio communication system suitable for conveying steering signals to a torpedo that would be difficult to jam:



An interesting feature of this design is that the transmitter was capable of "hopping" between more frequencies than the receiver was built to receive, and this was done purposely to permit the occasional broadcast of false signals which would further confound any attempt for an enemy to intercept or jam the radio transmissions:

> It will be noted that whereas there are seven tuning condensers[1] 24 at the transmitting station, there are only four tuning condensers 24' at the receiving station. The extra three tuning condensers at the transmitting station provide three additional channels for the transmitter for which there are no corresponding channels at the receiver, to thereby permit the sending of false impulses to confuse the enemy. [page 3]

Any "frequency-hopping" spread-spectrum radio system must somehow manage to synchronize the frequency-hopping pattern between transmitter(s) and receiver(s) or else it simply will not function. To this end, the patent's authors suggest a method for synchronizing the starting times of the two units' tape-drive motors, illustrated in figure 3:

> It is of course necessary that the record strips 37 and 37' at the transmitting and receiving stations, respectively, be started at the same time and in proper phase relation with each other, so that corresponding perforations in the two record strips will move over their associated control heads at the same time. We therefore provide an apparatus for holding both record strips in a starting position until the torpedo is fired, and for then simultaneously releasing both strips so that they can be moved at the same speed by their associated motors 41 and 41'. [page 3]

The long-term synchronization of two paper strips driven by electric motors may at first seem to be a challenging problem to solve, but this is obviated by the fact that a running torpedo is a short-lived machine. The transmitter and receiver tapes need only be synchronized with each other for a short run time, so slight differences in motor speed would not be as debilitating as one might assume.

[1] "Condenser" is an obsolete term for *capacitor*.

3.2 Stuxnet

Although it may seem strange at the time of this writing (2010) to refer to an event in the early 21st century as being "historical", the release of the so-called *Stuxnet* virus certainly qualifies as an historical event. This is the first known nation-state attack using a digital computer virus, opening a new chapter in international warfare.

In November of 2007 a new computer virus was submitted to a virus scanning service. The purpose of this new virus was not understood at the time, but it was later determined to be an early version of the so-called *Stuxnet* virus which was designed to infiltrate and attack programmable logic controllers (PLCs) installed at the uranium enrichment facility in Iran, a critical part of that country's nuclear program located in the city of Natanz. Stuxnet stands as the world's first known computer virus ever designed to specifically attack an industrial control platform, in this case Siemens model S7 PLCs.

Later forensic analysis revealed the complexity and scope of Stuxnet for what it was: a digital weapon, directed against the Iranian nuclear program for the purpose of delaying that program's production of enriched uranium. Although the origins of Stuxnet are rather unique as viruses go, the lessons learned from Stuxnet help us as industrial control professionals to fortify our own control systems against similarly-styled digital attacks. The next such attack may not come from a nation-state like Stuxnet did, but you can be sure whoever attacks next will have gained from the lessons Stuxnet taught the world.

Since the Stuxnet attack was directed against a nuclear facility, it is worthwhile to know a little about what that facility did and how it functioned. The next subsection will delve into some of the details of modern uranium enrichment processes, while further subsections will outline how Stuxnet attacked those physical processes through the PLC control system.

3.2.1 A primer on uranium enrichment

Uranium is a naturally occurring metal with interesting properties lending themselves to applications of nuclear power and nuclear weaponry. Uranium is extremely dense, and also (mildly) radioactive. Of greater importance, though, is that some of the naturally occurring isotopes[2] of uranium are *fissile*, which means those atoms may be easily "split" by neutron particle bombardment, releasing huge amounts of energy as well as more neutrons which may then go on to split more uranium atoms in what is called a *chain reaction*. Such a chain-reaction, when controlled, constitutes the energy source of a fission reactor. Nuclear weapons employ violently uncontrolled chain reactions.

The most fissile isotope of uranium is uranium 235, that number being the total count of protons and neutrons within the nucleus of each atom. Unfortunately (or fortunately, depending on your view of nuclear fission), ^{235}U constitutes only 0.7% of all uranium found in the earth's crust. The vast majority of naturally occurring uranium is the isotope ^{238}U which has all the same chemical properties of ^{235}U but is non-fissile, that is to say an atom of ^{238}U will not be "split" by neutron particle bombardment[3].

Naturally-occurring uranium at a concentration of only 0.7% ^{235}U is too "dilute" for most[4] nuclear reactors to use as fuel, and certainly is not concentrated enough to construct a nuclear weapon. Most power reactors require uranium fuel at a ^{235}U concentration of at least 3% for practical operation, and a concentration of at least 20% is considered the low threshold for use in constructing a uranium-based nuclear weapon. Mildly concentrated uranium useful for reactor fuel is commonly referred to "low-enriched uranium" or *LEU*, while uranium concentrated enough to build a nuclear weapon is referred to as "highly enriched uranium" or *HEU*. Modern uranium-based nuclear bombs rely on the uranium being concentrated to at least 90% ^{235}U, as do military power reactors such as the extremely compact designs used to power nuclear submarines. All of this means that an industrial-scale process for concentrating (enriching) ^{235}U is a necessary condition for building and sustaining a nuclear program of any kind, whether its purpose be civilian or military.

Different technologies currently exist for uranium enrichment, and more are being developed. The technical details of uranium enrichment set the background for the Stuxnet story, the site of this cyber-attack being the Natanz uranium enrichment facility located in the middle-eastern nation of Iran.

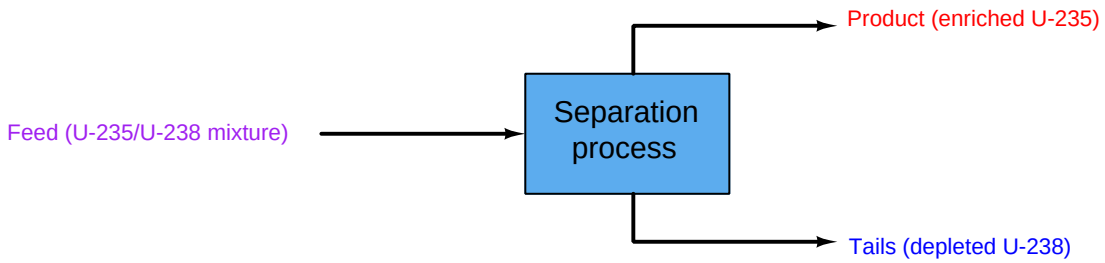
---

[2] The term *isotope* refers to differences in atomic mass for any chemical element. For example, the most common isotope of the element *carbon* (C) has six neutrons and six protons within each carbon atom nucleus, giving that isotope an atomic mass of twelve (^{12}C). A carbon atom having two more neutrons in its nucleus would be an example of the isotope ^{14}C, which just happens to be radioactive: the nucleus is unstable, and will over time *decay*, emitting energy and particles and in the process change into another element.

[3] It is noteworthy that ^{238}U can be converted into a different, fissile element called plutonium through the process of neutron bombardment. Likewise, naturally-occurring thorium 232 (^{232}Th) may be converted into ^{233}U which is fissile. However, converting non-fissile uranium into fissile plutonium, or converting non-fissile thorium into fissile uranium, requires intense neutron bombardment at a scale only seen within the core of a nuclear reactor running on some other fuel such as ^{235}U, which makes ^{235}U the critical ingredient for any independent nuclear program.

[4] Power reactors using "heavy" water as the moderator (such as the Canadian "CANDU" design) are in fact able to use uranium at natural ^{235}U concentration levels as fuel, but most of the power reactors in the world do not employ this design.

Like all two-phase separation processes, uranium enrichment breaks a single input "feed" stream into two out-going streams of differing composition. Since in the case of uranium enrichment only one stream is of strategic interest, the stream containing concentrated ^{235}U is called the *product*. The other stream coming exiting the separation process, having been largely depleted of valuable ^{235}U, is called the *tails*:



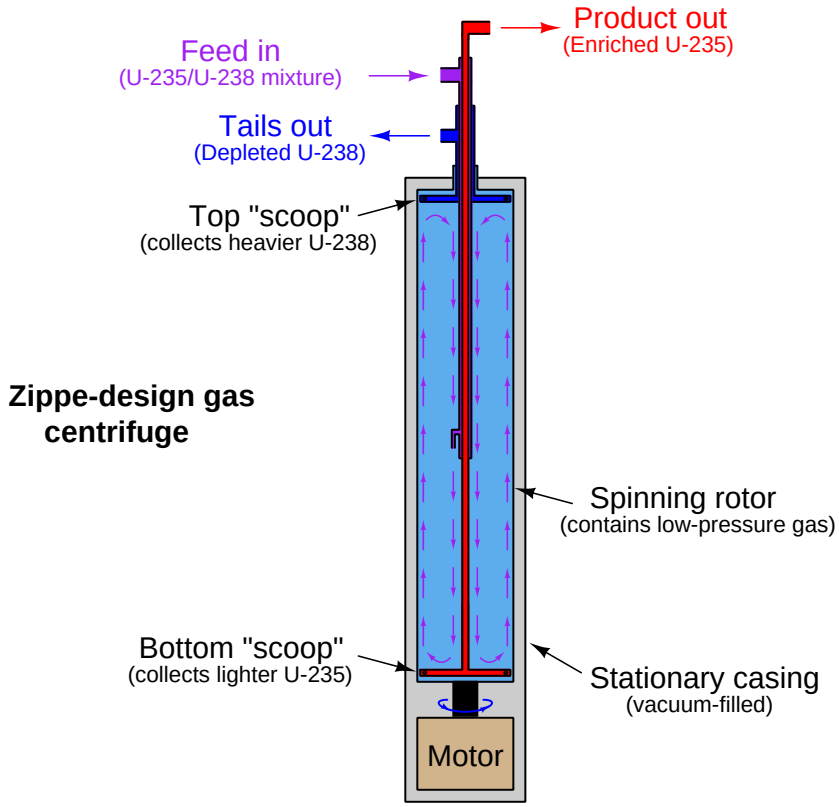
During the United States' Manhattan Project of World War Two, the main process chosen to enrich uranium for the first atomic weapons and industrial-scale reactors was *gaseous diffusion*. In this process, the uranium metal is first chemically converted into *uranium hexafluoride* (UF_6) gas so that it may be compressed, transported through pipes, processed in vessels, and controlled with valves. Then, the UF_6 gas is run through a long series of diffusion membranes (similar to fine-pore filters). At each membrane, those UF_6 molecules containing ^{235}U atoms will preferentially cross through the membranes because they are slightly less massive than the UF_6 molecules containing ^{238}U atoms. The mass difference between the two isotopes of uranium is so slight, though, that this membrane diffusion process must be repeated thousands of time in order to achieve any significant degree of enrichment. Gaseous diffusion is therefore an extremely inefficient process, but nevertheless one which may be scaled up to industrial size and used to enrich uranium at a pace sufficient for a military nuclear program. At the time of its construction, the world's first gaseous diffusion enrichment plant (built in Oak Ridge, Tennessee) also happened to be the world's largest industrial building.

An alternative uranium enrichment technology considered but later abandoned by the Manhattan Project scientists was *gas centrifuge* separation. A gas centrifuge is a machine with a hollow rotor spun at extremely high speed. Gas is introduced into the interior of the rotor, where centrifugal force causes the heavier molecules to migrate toward the walls of the rotor while keeping the lighter molecules toward the center. Centrifuges are commonly used for separating a variety of different liquids and solids dissolved in liquid (e.g. separating cells from plasma in blood, separating water from cream in milk), but gas centrifuges face a much more challenging task because the difference in density between various gas molecules is typically far less than the density differential in most liquid mixtures. This is especially true when the gas in question is uranium hexafluoride (UF_6), and the only difference in mass between the UF_6 molecules is that caused by the miniscule[5] difference in mass between the uranium isotopes ^{235}U and ^{238}U.

[5] The formula weight for UF_6 containing fissile ^{235}U is 349 grams per mole, while the formula weight for UF_6 containing non-fissile ^{238}U is only slightly higher: 352 grams per mole. Thus, the difference in mass between the two molecules is less than one percent.

Gas centrifuge development was continued in Germany, and then later within the Soviet Union. The head of the Soviet gas centrifuge effort – a captured Austrian scientist named Gernot Zippe – was eventually brought to the United States where he shared the refined centrifuge design with American scientists and engineers. As complex as this technology is, it is far[6] more energy-efficient than gas diffusion, making it the uranium enrichment technology of choice at the time of this writing (2016).

An illustration of Gernot Zippe's design is shown below. The unenriched UF_6 feed gas is introduced into the middle of the spinning rotor where it circulates in "counter-current" fashion both directions parallel to the rotor's axis. Lighter (^{235}U) gas tends to stay near the center of the rotor and is collected at the bottom by a stationary "scoop" tube where the inner gas current turns outward. Heavier (^{238}U) gas tends to stay near the rotor wall and is collected at the top by another stationary "scoop" where the outer current turns inward:

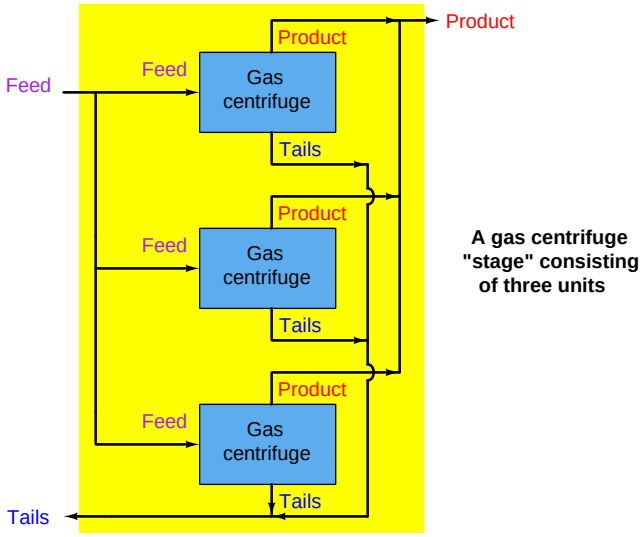


Like the separation membranes used in gaseous diffusion processes, each gas centrifuge is only able to enrich the UF^6 gas by a very slight amount. The modest enrichment factor of each centrifuge necessitates many be connected in series, with each successive centrifuge taking in the out-flow of the previous centrifuge in order to achieve any practical degree of enrichment. Furthermore, gas

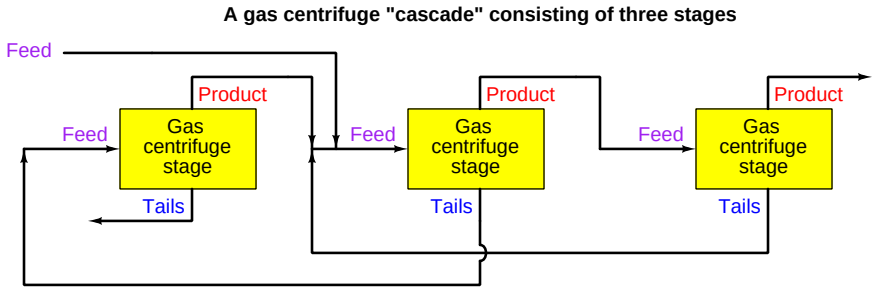
[6] By some estimates, gas centrifuge enrichment is 40 to 50 *times* more energy efficient than gaseous diffusion enrichment.

centrifuges are by their very nature rather limited in their flow capacity[7]. This low "throughput" necessitates parallel-connected gas centrifuges in order to achieve practical production rates for a national-scale nuclear program. A set of centrifuges connected in parallel for higher flow rates is called a *stage*, while a set of centrifuge stages connected in series for greater enrichment levels is called a *cascade*.

A gas centrifuge *stage* is very simple to understand, as each centrifuge's feed, product, and tails lines are simply paralleled for additional throughput:

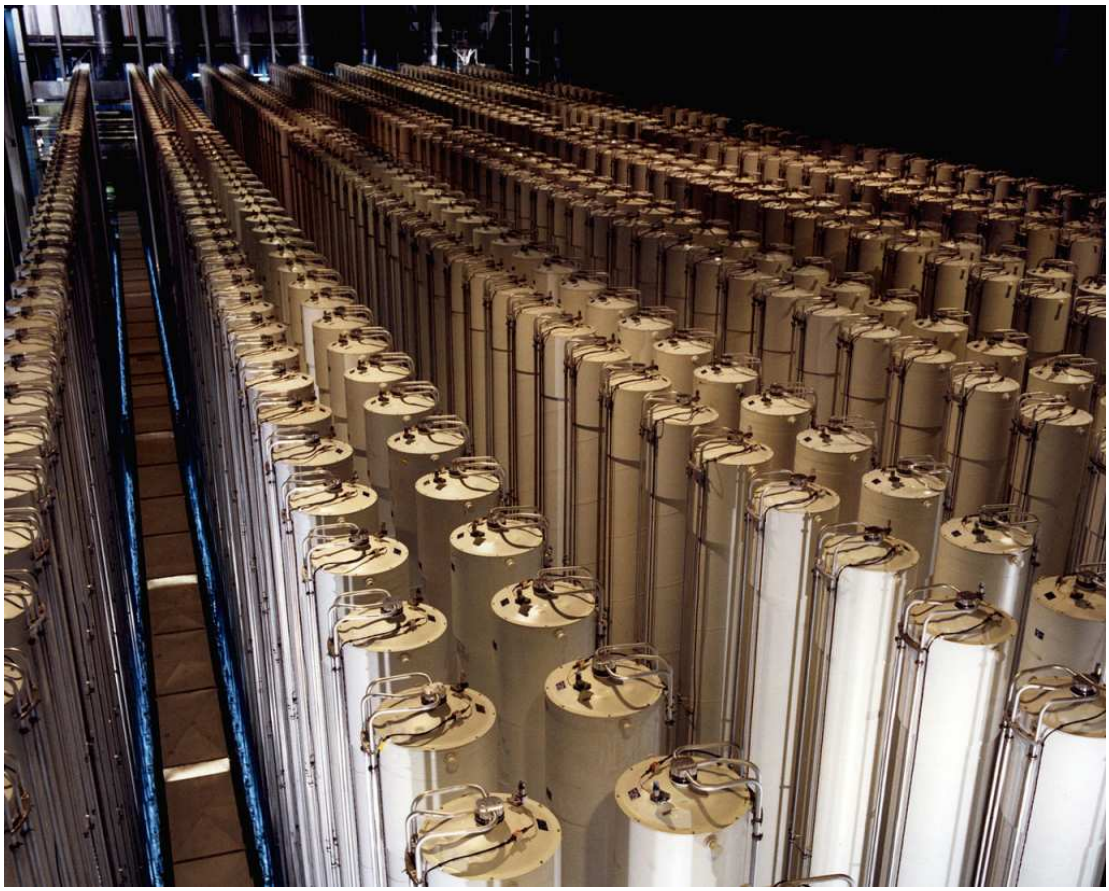


A gas centrifuge *cascade* is a bit more complex to grasp, as each centrifuge's product gets sent to the feed inlet of the next stage for further enrichment, and the tails gets sent to the feed inlet of the previous stage for further depletion. The main feed line enters the cascade at one of the middle stages, with the main product line located at one far end and the main tails line located at the other far end:



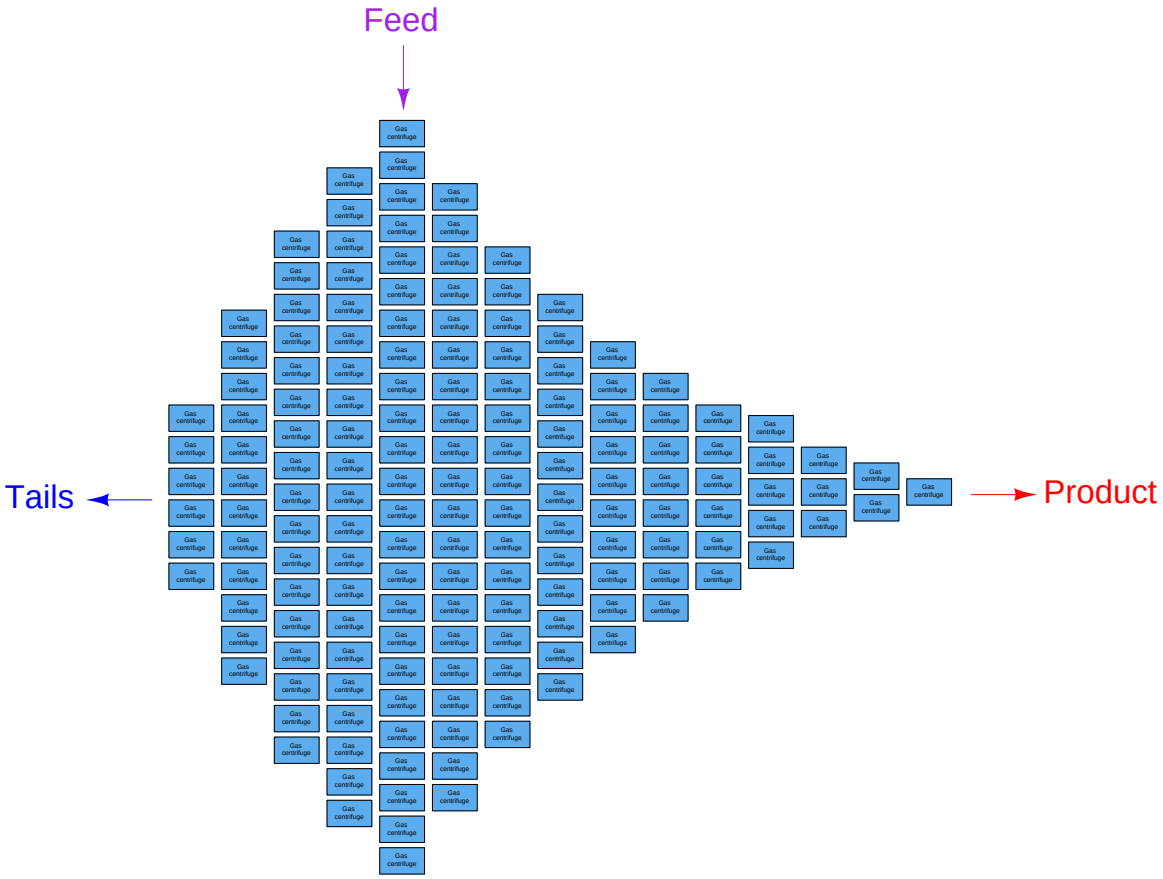
[7] A typical gas centrifuge's mass flow rating is on the order of milligrams per second. At their very low (vacuum) operating pressures, a typical centrifuge rotor will hold only a few grams of gas at any moment in time.

This US Department of Energy (DOE) photograph shows an array of 1980's-era American gas centrifuges located in Piketon, Ohio. Each of the tall cylinders is a single gas centrifuge machine, with the feed, product and tails tubing seen connecting to the spinning rotor at the top of the stationary casing:



Without any familiar objects in this photograph it is difficult to get an accurate sense of physical scale, but know that each of these centrifuge units measures 40 feet in height.

The size of each stage in a gas centrifuge cascade is proportional to its feed flow rate. The stage processing the highest feed rate must be the largest (i.e. contain the most centrifuges), while the stages at the far ends of the cascade contain the least centrifuges. A cascade similar to the one at the Natanz enrichment facility in Iran – the target of the Stuxnet cyber-attack – is shown here without piping for simplicity, consisting of 164 individual gas centrifuges arranged in 15 stages. The main feed enters in the middle of the cascade at the largest stage, while enriched product exits at the right-hand end and depleted tails at the left-hand end:



The sheer number of gas centrifuges employed at a large-scale uranium enrichment facility is impressive. At the Natanz facility, where just one cascade contained 164 centrifuges, cascades were paralleled together in *sub-units* of six cascades each (984 centrifuges per sub-unit), and three of these sub-units made one cascade *unit* (2952 centrifuges total).

3.2.2 Gas centrifuge vulnerabilities

It would be an understatement to say that a gas centrifuge is a delicate machine. In order to perform their task efficiently[8], gas centrifuge rotors must be long and made to rotate at extremely high rates of speed. Maintaining any rotating machine in a state of near-perfect balance is difficult, much more so when the rotating element is very long[9]. Furthermore, since the gas pressure inside each centrifuge rotor is sub-atmospheric, leak-free seals must be maintained between the spinning rotor and the stationary components (the casing and internal tubing). The extremely high rotational speeds of modern gas centrifuges (many tens of thousands of revolutions per minute!) necessitate advanced materials be used in rotor construction, optimizing light weight and high strength so that the rotors will not be torn to pieces by their own centrifugal force.

A peculiar problem faced by any high-speed rotating machine is a phenomenon called *critical speed*. Any object possessing both mass and resilience is capable of oscillating, which of course includes any and every rotating machine component. If the rotating component of a machine happens to spin at a rate equal to its own natural oscillating frequency, a condition of *mechanical resonance* occurs. *Any* amount of imbalance in the rotating component while spinning at this speed, however slight, will generate a force driving the assembly into continuous oscillation. The speed at which this resonance occurs is called the "critical speed" of the machine, and it should be avoided whenever possible.

Destructive resonance will be avoided so long as the machine is maintained at any speed significantly below or above its critical speed. Most modern gas centrifuges are classified as *supercritical* machines, because they are designed to operate at rotational speeds exceeding their critical speeds. The only time resonance becomes a problem in a supercritical machine is during start-up and shut-down, when the speed must momentarily pass through the critical value. So long as this moment is brief, however, oscillations will not have enough time to grow to destructive levels.

In addition to the problems faced by all high-speed rotating machines, a problem unique to gas centrifuges is gas pressure control. Since the rotor of a gas centrifuge spins inside of an evacuated[10] stationary casing, the existence of any gas pressure inside the rotor creates additional stress acting in the same outward direction as the rotor's own centrifugal force. This means rotor gas pressure must be maintained at a very low level in order to minimize rotor stress. Furthermore, if pressure and temperature conditions are not carefully controlled in a gas centrifuge, the gas may actually *sublimate* into a solid state which will deposit material on the inside wall of the rotor and surely throw it out of balance.

[8] Three major factors influence the efficiency of a gas centrifuge: rotor wall speed, rotor length, and gas temperature. Of these, rotor wall speed is the most influential. Higher speeds separate isotopes more effectively, because higher wall speeds result in greater amounts of radial acceleration, which increases the amount of centrifugal force experienced by the gas molecules. Longer rotors also separate isotopes more effectively because they provide more opportunity for the counter-flowing gas streams to separate lighter molecules toward the center and heavier molecules toward the wall. Higher temperatures reduce separation efficiency, because gas molecules at higher temperatures are more mobile and therefore diffuse (i.e. mix together) at higher rates. Therefore, the optimum gas centrifuge design will be long, spin as fast as possible, and operate as cool as possible.

[9] To give you an idea of just how long some gas centrifuge rotors are, the units built for the US Department of Energy facility in Ohio used rotors *40 feet in length*!

[10] This means the hollow casing exists in a state of vacuum, with no air or other gases present. This is done in order to help thermally insulate the rotor from ambient conditions, as well as avoid generating heat from air friction against the rotor's outside surface. Remember, elevated temperatures cause the gas to diffuse at a faster rate, which in turn causes the gas to randomly mix and therefore not separate into light and heavy isotopes as intended.

One could argue that the temperamental nature of gas centrifuges is a good thing, because it makes the manufacture of enriched uranium difficult to achieve, which in turn complicates the development of nuclear weapons. This fragility also makes gas centrifuges an ideal target for anyone interested in halting or delaying nuclear weapons development, which was precisely the aim of the Stuxnet computer virus.

3.2.3 The Natanz uranium enrichment facility

Iran used an obsolete gas centrifuge design, perhaps the best they could obtain at the time, as the uranium enrichment platform of choice for their Natanz facility. By modern standards, this design was inefficient and troublesome, but the Iranians were able to coax serviceable performance from this centrifuge design by means of extensive instrumentation and controls.

Simply put, the Iranian strategy was to manufacture centrifuges faster than they would break and equip the centrifuge cascades with enough piping and supervisory instrumentation that they could detect and isolate failed centrifuges without stopping production, rather than wait until they had perfected the design of the centrifuges themselves. The extensive network of sensors, valves, piping, and PLCs (Programmable Logic Controllers) installed at the Natanz facility facilitated this fault-tolerant design.

The key to the Natanz system's fault tolerance was a set of isolation ("block") valves installed at each gas centrifuge. Each machine was also equipped with a sufficient array of sensors to detect malfunctions. If a centrifuge experienced trouble, such as excessive vibration, the PLC control system would automatically shut all the isolation valves for that failed centrifuge and turn off its drive motor. Since most stages in each cascade contained multiple centrifuges in parallel, the isolation of a single centrifuge within a stage would not shut down the entire cascade. Instead, maintenance personnel could repair the failed centrifuge while production continued, and return it to service when ready.

One undesired consequence of shutting isolation valves on operating centrifuges, though, was increased gas pressure in portions of the cascade. With fewer centrifuges left to handle a constant feed flow, the pressure drop across that stage increases. All upstream stages therefore experience more gas pressure, which as described earlier increases the stress imparted on the spinning centrifuge rotors. In answer to this problem was another innovation at the Natanz facility: using the "dump system" (a standard feature in any gas centrifuge cascade, for evacuating gas from the centrifuges in the event of an emergency shut-down event) as a pressure relief in the event of overpressure resulting from too many isolated centrifuges. Of course, engaging this "dump" system as a means of pressure control would reduce production rates, but it was a better outcome for the system operators than a complete shut-down of the cascade.

In summary, sensors in the Natanz facility would automatically detect problems in each centrifuge, causing the PLCs to automatically isolate any failed centrifuges from the running cascade and open dump valves as necessary to reduce gas pressure on the remaining centrifuges. This so-called *Cascade Protection System* was implemented by Siemens model S7-417 PLCs, one per sub-unit (six cascades, each sub-unit containing 984 individual gas centrifuges). All-digital *Profibus* technology was used to communicate process data over network cables between the field instruments and the PLCs, as a means of reducing what would have otherwise been a huge amount of analog and discrete signal wiring.

Additional Siemens PLCs were used at the Natanz facility to control the gas centrifuges, notably the model S7-315 employed to issue commands to variable-frequency drive units sending power to the rotor drive motors. Like the larger S7-417 PLC units, one S7-315 PLC was used to control the motor drives of each cascade sub-unit (six cascades, 984 centrifuges). As subsequent portions of this chapter will detail, both of these Siemens PLC platforms were targets of the Stuxnet virus.

3.2.4 How Stuxnet worked

Stuxnet is a highly complex computer virus with many components, as well as multiple versions with different attack vectors, but its basic functionality may be summarized in simple terms. It consists of two major portions: the *dropper* and the *payload*. The payload is the malicious code intended to infect PLC control systems and the dropper is malicious code intended to distributed and deliver the payload onto computer systems capable of accessing the PLCs.

The dropper portion of Stuxnet was designed to infect personal computers running Siemens Step7 PLC programming software under Microsoft Windows operating system – the type of application used by technicians and engineers to edit PLC code. Once installed, Stuxnet corrupts the Step7 software in such a way that any PLC program downloaded to a PLC from that personal computer will differ significantly from the PLC code seen on the programming screen. In other words, any person using Step7 software infected by Stuxnet would unwittingly infect the Siemens PLC they were trying to program or maintain. In this capacity, Stuxnet represents a "man-in-the-middle" attack, the "man" in this case being the infected Step7 application which would alter whatever PLC code the user intended to transfer to the PLC.

The PLC code alterations were highly specific in their design, intended to attack the centrifuge systems by altering rotor speeds and manipulating control valves in an attempt to over-stress the centrifuge rotors and thereby cause premature failures. Moreover, the altered PLC code performed these manipulations in such a way that they would not be visible to the human operators or even to other portions of the control system: rotor speeds and valve positions would appear to be normal while in reality they were anything but.

A noteworthy aspect of the Stuxnet dropper code is that it was designed to be introduced via a removable USB-style flash memory data drive. This allowed Stuxnet to cross any "air gap" separating the control system network from the internet: all that was required for infection of the Natanz site was some person to carry an infected USB drive into the facility and plug it in to any personal computer there. While "air gaps" are a good security design practice for any industrial control network, Stuxnet serves as a sobering reminder that they are not enough to protect against external cyber-attacks.

Chapter 4

Derivations and Technical References

This chapter is where you will find mathematical derivations too detailed to include in the tutorial, and/or tables and other technical reference material.

4.1 The OSI Reference Model

Digital communication is a fairly complex topic, and there are a great many ways in which one digital device might communicate information to another. In order to help describe how different methods (protocols) function in forming a complete communication system, the *OSI Reference Model* was developed. This is not a standard itself, but rather a taxonomical system for *describing* and *classifying* various aspects of any communications system.

For example, one communication system might use discrete voltage pulses in a non-return-to-zero (NRZ) format to represent 0 and 1 bit states, while another might use analog audio tone frequency voltage signals to do the same. According to the OSI model, the choice over electrical signal types is a *Layer 1* matter. The patterns in which bit states make up larger serial data frames is another matter which must be standardized in any communication system, and according to the OSI model that is the domain of *Layer 2*. At the highest level of the OSI model we have *Layer 7* which regards the practical meaning of the bits communicated, whether those be alphanumeric codes representing an internet web page's contents, Modbus control functions, or anything else.

Layer 7 **Application**	This is where digital data takes on practical meaning in the context of some human or overall system function. *Examples: HTTP, FTP, HART, Modbus*
Layer 6 **Presentation**	This is where data gets converted between different formats. *Examples: ASCII, EBCDIC, MPEG, JPG, MP3*
Layer 5 **Session**	This is where "conversations" between digital devices are opened, closed, and otherwise managed for reliable data flow. *Examples: Sockets, NetBIOS*
Layer 4 **Transport**	This is where complete data transfer is handled, ensuring all data gets put together and error-checked before use. *Examples: TCP, UDP*
Layer 3 **Network**	This is where the system determines network-wide addresses, ensuring a means for data to get from one node to another. *Examples: IP, ARP*
Layer 2 **Data link**	This is where basic data transfer methods and sequences (frames) are defined within the smallest segment(s) of a network. *Examples: CSMA/CD, Token passing, Master/Slave*
Layer 1 **Physical**	This is where data bits are equated to electrical, optical, or other signals. Other physical details such as cable and connector types are also specified here. *Examples: EIA/TIA-232, 422, 485, Bell 202*

Students of philosophy might see a parallel between the OSI model's description of digital communication versus the ancient Greek philosopher Aristotle's classification of *causes*. Aristotle proposed four different ways to describe what causes a thing to exist or to occur. I have arranged these four causes in an order analogous to the OSI model's layers:

Aristotle's Four Causes

Layer 4 **Final**	This is the end-purpose for which something exists or has happened.
Layer 3 **Efficient**	This is the means by which something came into existence or occurred.
Layer 2 **Formal**	This is the structure or form taken by something.
Layer 1 **Material**	This is the fundamental substance something is made of.

For example, we might classify the four causes of a textbook as follows:

- **Final cause** – to educate and enlighten the reader
- **Efficient cause** – a human author wrote the book, and it came into existence through the operation of a printing press
- **Formal cause** – a series of characters and images arranged into words, paragraphs, diagrams, graphs, equations, etc. following one or more standards of language (e.g. English, international math notation, Cartesian-coordinate graphs)
- **Material cause** – paper and ink

All of these causes describe different aspects of what was required for a book to come into existence. Similarly, with the OSI model, each level of the model is like one of Aristotle's causes in that it describes something necessary for the communication to take place.

4.2 Lexicon of cyber-security terms

Cyber-security seems to have its own vocabulary, ranging from unwieldy technical acronyms to slang terms borrowed from amateur computer enthusiasts. What follows is a partial listing of some common terms and their definitions. This list is not only useful as a definitional reference when encountering such terms in cyber-security literature, but it also serves to outline a number of common attack strategies:

- **Active attack**: an attack involving data written to a network or to device. See *passive attack* for contrast.
- **AES**: Advanced Encyrption Standard, a cryptographic algorithm using symmetrical keys (i.e. the same key to encrypt and decrypt). Also known as *Rijndael*.
- **Asset**: the physical device and/or data you wish to protect.
- **Authentication**: to correctly identify a person or device requesting access to a system.
- **Authorization**: to correctly assign rights to a person or device requesting access to a system.
- **Backdoor**: an easy-to-access pathway into a system, typically used by system developers for convenience in their work. There is nothing wrong with a backdoor during development, but backdoors are very dangerous when left in place on commissioned systems.
- **Blacklist**: a database of prohibited messages or users or software applications.
- **Broadcast network**: a form of network where all transmissions are heard by all connected devices, even those devices the data is not intended for. Any communication network sharing a common physical channel is a broadcast network.
- **Brute-force attack**: attempting every combination of characters in an effort to forge a working password.
- **Checksum**: a rudimentary form of "hash" algorithm applied to any set of digital data, generating a fixed-width binary result designed to vary if any bits in the original data set become corrupted due to communication noise or read/write errors.
- **Ciphertext**: any digital message that has been encypted so as to be unintelligible to anyone but the holder of the proper decryption key/tool. Compare this with *cleartext*.
- **Cleartext**: ASCII text messages that are communicated over a network without any form of special encoding or encryption, but rather are "clear" for anyone to read. Also called *plaintext*.
- **Comsec**: shorthand for "communications security".
- **CRC**: a simple type of "hash" algorithm used to verify the integrity of data in certain communication protocols, notably Ethernet.
- **Crypto**: shorthand for "cryptography", which is the purposeful scrambling of data to render it unintelligible to all but the intended recipient.

- **Data diode**: a device permitting only one-way (simplex) data communication. Data diodes eliminate the possibility of active attacks, because they make writing data to the protected system impossible.

- **Defense-in-Depth**: a design philosophy utilizing multiple layers of protection, such that a failure in or compromise of any one layer will not result in the system failing or becoming compromised.

- **Denial-by-Default**: a policy whereby the default (factory) configuration settings for software and hardware are such that all access is denied unless explicitly granted by whomever sets up the system for use.

- **Denial-of-Service (DoS)**: a form of attack where the intended function of the system is either downgraded or entirely faulted. This may be achieved by "flooding" the targeted system with messages until it cannot process legitimate traffic, but it should be noted that flooding is not the only form of DoS attack.

- **Dictionary attack**: attempting common words and character combinations in an effort to forge a working password. This form of attack is based on the fact that most human beings choose words and phrases for their computer passwords that are easy for them to remember, and that these easy-to-remember words and phrases will likely resemble common speech.

- **Distributed Denial-of-Service (DDoS)**: a form of DoS based on flooding where the attack originates from multiple locations – for example, a large number of independent computers programmed to flood a single target with messages until that target can no longer perform its intended service(s).

- **DMZ**: an acronym standing for DeMilitarized Zone, referring to a network segment that stands between a private (trusted) network and some untrusted network, akin to a strip of land separating two nations at odds with each other. DMZs are created through the use of multiple firewalls, the intermediate network inhabited by *proxy* machines tasked with relaying messages safely between the separated networks.

- **Eavesdropping**: passively "listening" to the traffic on a network, for the purpose of gaining information.

- **Encryption**: any process by which a message may be converted into a form that is inscrutable to everyone but the intended recipient. **Decryption** is the reversal of that process, where the encrypted message becomes intelligible again.

- **Exploit**: when used as a noun, this term refers to a specific attack that takes advantage of a system vulnerability (or "vuln" for short).

- **Firewall**: a software or hardware application intended to limit connectivity between networked devices by permitting or denying specific messages along a network path.

- **Flooding**: an attack technique consisting of overloading a digital system with data or requests for data, generally the point of which being to achieve denial of service (DoS) when the target system becomes overloaded.

- **FTP**: an acronym standing for File Transfer Protocol, a protocol used for reading and writing files on one computer remotely from another computer. FTP is a predecessor to *SFTP* which includes public-private key encryption for much better security.
- **Hash**: either an algorithm or the output of that algorithm, depending on how the word is used, whereby a data set is reduced to a relatively fixed-width digital word (sometimes called a *digest*). Since the digest is generally much smaller than the original data set, digests may be more easily compared than data sets to check if any corruption occurred.
- **HTTP**: an acronym standing for Hyper Text Transfer Protocol, the method used for computers to exchange web page data (encoded in HTML files). HTTP is not encrypted.
- **HTTPS**: an acronym standing for Hyper Text Transfer Protocol Secure, the method used for computers to exchange web page data (encoded in HTML files) using encryption.
- **IP**: an acronym standing for Internet Protocol, the packaging of data into "packets" which may be routed independently of each other across a large network.
- **IT**: an acronym standing for Information Technology, used to broadly describe general-purpose digital data systems and communications.
- **Key**: a relatively small segment of digital data that serves to either encrypt or decrypt other digital data. The imagery here is that of a key used to engage or disengage a physical lock.
- **LAN**: an acronym standing for Local Area Network, a network connecting multiple devices over a limited distance, such as the span of an office building or campus. See *WAN* for contrast.
- **Logic bomb**: a form of malware designed to delay its malicious action until some time after infection.
- **LRC**: Longitudinal Redundancy Check, a form of checksum or hash applied in some network protocols for error detection, most notably Modbus.
- **Malware**: software written to fulfill some malicious purpose.
- **Man-in-the-Middle**: an attack where the attacker is positioned directly in between sender and receiver, in such a way as to be able to modify messages sent over the network without either sender or receiver being aware.
- **Multi-factor authentication**: when a digital system employs more than one method to authenticate access, for example a password *and* a token.
- **Operating system**: software installed on a computer for the purpose of directly managing that computer's hardware resources, functioning as an intermediate layer between the application and the hardware itself. The existence of operating system software vastly simplifies the design and development of application software. Popular consumer-grade operating systems at the time of this writing (2020) include Microsoft Windows, Apple OS X, Linux, and BSD.
- **Packet sniffing**: the act of passively monitoring data transmitted over an IP network, where individual packets of transmitted data are inspected for valuable information.

- **Passive attack**: an attack only involving the reading of data from a network or device. See *active attack* for contrast.

- **Passphrase**: an easily-memorized sentence which may be used to generate complex passwords. For example, one could take the first letter of every word in the passphrase "What we have here is a failure to communicate" to generate the password `wwhhiaftc`. Passphrases are useful because they make complex passwords easy to remember, and in fact may be used to generate multiple passwords from the same phrase (e.g. replacing words like "to" with numerals such as 2, and/or using the *last* letter of each word instead of the first, to create the password `teeesaeoe` from the same passphrase used previously).

- **Phishing**: an anonymous or strange invitation from an online source to either reveal sensitive information or download an infected file.

- **Ping**: a simple network utility used on IP networks to test connectivity, and part of the Internet Control Message Protocol (ICMP). The ping message is sent from one computer to another, with the receiving computer replying to declare successful receipt of the ping message.

- **Ping flood**: a crude denial-of-service attack that works by bombarding a device with ping "echo-request" messages in an attempt to keep that device so occupied with answering these ping requests that it cannot service other messages as it should.

- **Plaintext**: a synonym for *cleartext*.

- **Private key**: a cryptographic key that is part of an asymmetrical key algorithm useful for decrypting encrypted data. "Private" refers to the fact that this key must be held in confidence by authorized parties only, since it has the ability to unlock (i.e. decrypt) coded messages.

- **Public key**: a cryptographic key that is part of an asymmetrical key algorithm useful only for encrypting data. "Public" refers to the fact that this key may be shared openly, as it cannot be used to unlock a coded message, but instead is only useful for encoding messages sent to a party holding a *private key* which can decode the message.

- **Replay attack**: a form of attack where a message is intercepted, recorded, and later broadcast to the network in order to inflict damage. An interesting feature of replay attacks is that they may work on encrypted messages, and even when the purpose of the message is unknown to the attacker!

- **SCADA**: Supervisory Control And Data Acquisition, a common moniker in the network security realm for any industrial control system tasked with measuring and/or controlling real processes. Instrumentation professionals typically use the term "SCADA" more specifically in reference to control systems spanning large geographic distances.

- **SFTP**: an acronym standing for Secure File Transfer Protocol, a protocol used for reading and writing files on one computer remotely from another computer. SFTP is a successor to *FTP* which lacked encryption.

- **SHA**: a class of cryptographic "hash" algorithms designed as a means of integrity-checking for communicated or stored/retrieved messages. Passing a block of data through a hash function generates a "digest" that is fairly unique to that data block, in that most errors in that data will result in a different digest. Cryptographic hash algorithms are designed to make it less likely that anyone could derive any details of the source data by analyzing its digest. Many cryptographic hash algorithms exist, and SHA is a label applied to several of them.

- **Sniffing**: inspecting network communications for important data. So-called "packet sniffers" monitor data traffic on a broadcast network for certain information such as passwords, network addresses, and system data.

- **Spear phishing**: an invitation from a seemingly trusted online source (e.g. friend, colleague) to either reveal sensitive information or download an infected file.

- **Spoofing**: presenting a false identification to the receiver of digital data. This commonly takes the form of presenting fake network address information, to trick the receiver into thinking the source is from a legitimate location or device.

- **Spread spectrum**: a type of radio communications technology where the information is "spread" over multiple frequency channels rather than a single channel and is therefore more challenging to intercept or mimic.

- **SSH**: an acronym standing for Secure Shell, a remote-access utility commonly used in Unix operating systems allowing users to log into a computer from another computer connected to the same network. SSH is a successor to *telnet*, which lacked encryption.

- **Syn flood**: a specific form of denial-of-service (DoS) attack used on TCP connections, which works by flooding the target computer with TCP Synchronize (SYN) messages. TCP begins each connection with a three-way "handshake" between the two devices to ensure data integrity. This attack exploits the handshake by bombarding the target machine with only one portion of the handshake until it is no longer able to accept legitimate TCP connection requests.

- **TCP**: an acronym standing for Terminal Control Protocol, the protocol used to ensure segments of data make it to their intended destinations after being routed by IP (see *Internet Protocol*).

- **Telnet**: a remote-access utility commonly used in Unix operating systems allowing users to log into a computer from another computer connected to the same network. Telnet is a predecessor to *SSH* which includes public-private key encryption for much greater security.

- **Trojan horse**: malware masquerading as legitimate (useful) code.

- **Trusted**: a component or section of a digital system that is assumed to be safe from intrusion.

- **UDP**: an acronym standing for User Datagram Protocol, a protocol used to transport data packets after being routed by IP (see *Internet Protocol*).

- **Virus**: a form of malware designed to spread via human interactions with computers, for example by inserting an infected data storage device into a computer.

- **VPN**: an acronym standing for Virtual Private Network, which encrypts every aspect of a transaction between two computers connected on a network. The effect is to form a "virtual network" or "tunnel" between the machines, the privacy of which being ensured by the encryption algorithm and key(s) used to scramble the data.
- **Vuln**: shorthand for "vulnerability" or weakness in a system.
- **Walled garden**: a term used to describe an area of a digital system assumed to be safe from intrusion. See *trusted*.
- **WAN**: an acronym standing for Wide Area Network, a network connecting multiple devices over a long range, such as the span of a city. See *LAN* for contrast.
- **War dialing**: the exploratory practice of dialing random phone numbers in search of telephone modem connections, which may connect to computer systems.
- **Whitelist**: a database of permitted messages or users or software applications.
- **Worm**: a form of malware designed to propagate itself along a network with no human interaction necessary.
- **Zero-day**: a system vulnerability that is unknown to the designer(s). In other words, the designer(s) knew about this vulnerability for zero days when it was first exploited.

Chapter 5

Questions

This learning module, along with all others in the ModEL collection, is designed to be used in an inverted instructional environment where students independently read[1] the tutorials and attempt to answer questions on their own *prior* to the instructor's interaction with them. In place of lecture[2], the instructor engages with students in Socratic-style dialogue, probing and challenging their understanding of the subject matter through inquiry.

Answers are not provided for questions within this chapter, and this is by design. Solved problems may be found in the Tutorial and Derivation chapters, instead. The goal here is *independence*, and this requires students to be challenged in ways where others cannot think for them. Remember that you always have the tools of *experimentation* and *computer simulation* (e.g. SPICE) to explore concepts!

The following lists contain ideas for Socratic-style questions and challenges. Upon inspection, one will notice a strong theme of *metacognition* within these statements: they are designed to foster a regular habit of examining one's own thoughts as a means toward clearer thinking. As such these sample questions are useful both for instructor-led discussions as well as for self-study.

[1] Technical reading is an essential academic skill for any technical practitioner to possess for the simple reason that the most comprehensive, accurate, and useful information to be found for developing technical competence is in textual form. Technical careers in general are characterized by the need for continuous learning to remain current with standards and technology, and therefore any technical practitioner who cannot read well is handicapped in their professional development. An excellent resource for educators on improving students' reading prowess through intentional effort and strategy is the book textitReading For Understanding – How Reading Apprenticeship Improves Disciplinary Learning in Secondary and College Classrooms by Ruth Schoenbach, Cynthia Greenleaf, and Lynn Murphy.

[2] Lecture is popular as a teaching method because it is easy to implement: any reasonably articulate subject matter expert can talk to students, even with little preparation. However, it is also quite problematic. A good lecture always makes complicated concepts seem easier than they are, which is bad for students because it instills a false sense of confidence in their own understanding; reading and re-articulation requires more cognitive effort and serves to verify comprehension. A culture of teaching-by-lecture fosters a debilitating dependence upon direct personal instruction, whereas the challenges of modern life demand independent and critical thought made possible only by gathering information and perspectives from afar. Information presented in a lecture is ephemeral, easily lost to failures of memory and dictation; text is forever, and may be referenced at any time.

General challenges following tutorial reading

- Summarize as much of the text as you can in one paragraph of your own words. A helpful strategy is to explain ideas as you would for an intelligent child: as simple as you can without compromising too much accuracy.
- Simplify a particular section of the text, for example a paragraph or even a single sentence, so as to capture the same fundamental idea in fewer words.
- Where did the text make the most sense to you? What was it about the text's presentation that made it clear?
- Identify where it might be easy for someone to misunderstand the text, and explain why you think it could be confusing.
- Identify any new concept(s) presented in the text, and explain in your own words.
- Identify any familiar concept(s) such as physical laws or principles applied or referenced in the text.
- Devise a proof of concept experiment demonstrating an important principle, physical law, or technical innovation represented in the text.
- Devise an experiment to disprove a plausible misconception.
- Did the text reveal any misconceptions you might have harbored? If so, describe the misconception(s) and the reason(s) why you now know them to be incorrect.
- Describe any useful problem-solving strategies applied in the text.
- Devise a question of your own to challenge a reader's comprehension of the text.

General follow-up challenges for assigned problems

- Identify where any fundamental laws or principles apply to the solution of this problem, especially before applying any mathematical techniques.
- Devise a thought experiment to explore the characteristics of the problem scenario, applying known laws and principles to mentally model its behavior.
- Describe in detail your own strategy for solving this problem. How did you identify and organized the given information? Did you sketch any diagrams to help frame the problem?
- Is there more than one way to solve this problem? Which method seems best to you?
- Show the work you did in solving this problem, even if the solution is incomplete or incorrect.
- What would you say was the most challenging part of this problem, and why was it so?
- Was any important information missing from the problem which you had to research or recall?
- Was there any extraneous information presented within this problem? If so, what was it and why did it not matter?
- Examine someone else's solution to identify where they applied fundamental laws or principles.
- Simplify the problem from its given form and show how to solve this simpler version of it. Examples include eliminating certain variables or conditions, altering values to simpler (usually whole) numbers, applying a limiting case (i.e. altering a variable to some extreme or ultimate value).
- For quantitative problems, identify the real-world meaning of all intermediate calculations: their units of measurement, where they fit into the scenario at hand. Annotate any diagrams or illustrations with these calculated values.
- For quantitative problems, try approaching it qualitatively instead, thinking in terms of "increase" and "decrease" rather than definite values.
- For qualitative problems, try approaching it quantitatively instead, proposing simple numerical values for the variables.
- Were there any assumptions you made while solving this problem? Would your solution change if one of those assumptions were altered?
- Identify where it would be easy for someone to go astray in attempting to solve this problem.
- Formulate your own problem based on what you learned solving this one.

General follow-up challenges for experiments or projects

- In what way(s) was this experiment or project easy to complete?
- Identify some of the challenges you faced in completing this experiment or project.

- Show how thorough documentation assisted in the completion of this experiment or project.
- Which fundamental laws or principles are key to this system's function?
- Identify any way(s) in which one might obtain false or otherwise misleading measurements from test equipment in this system.
- What will happen if (component X) fails (open/shorted/etc.)?
- What would have to occur to make this system unsafe?

5.1 Conceptual reasoning

These questions are designed to stimulate your analytic and synthetic thinking[3]. In a Socratic discussion with your instructor, the goal is for these questions to prompt an extended dialogue where assumptions are revealed, conclusions are tested, and understanding is sharpened. Your instructor may also pose additional questions based on those assigned, in order to further probe and refine your conceptual understanding.

Questions that follow are presented to challenge and probe your understanding of various concepts presented in the tutorial. These questions are intended to serve as a guide for the Socratic dialogue between yourself and the instructor. Your instructor's task is to ensure you have a sound grasp of these concepts, and the questions contained in this document are merely a means to this end. Your instructor may, at his or her discretion, alter or substitute questions for the benefit of tailoring the discussion to each student's needs. The only absolute requirement is that each student is challenged and assessed at a level equal to or greater than that represented by the documented questions.

It is far more important that you convey your *reasoning* than it is to simply convey a correct answer. For this reason, you should refrain from researching other information sources to answer questions. What matters here is that *you* are doing the thinking. If the answer is incorrect, your instructor will work with you to correct it through proper reasoning. A correct answer without an adequate explanation of how you derived that answer is unacceptable, as it does not aid the learning or assessment process.

You will note a conspicuous lack of answers given for these conceptual questions. Unlike standard textbooks where answers to every other question are given somewhere toward the back of the book, here in these learning modules students must rely on other means to check their work. The best way by far is to debate the answers with fellow students and also with the instructor during the Socratic dialogue sessions intended to be used with these learning modules. Reasoning through challenging questions with other people is an excellent tool for developing strong reasoning skills.

Another means of checking your conceptual answers, where applicable, is to use circuit simulation software to explore the effects of changes made to circuits. For example, if one of these conceptual questions challenges you to predict the effects of altering some component parameter in a circuit, you may check the validity of your work by simulating that same parameter change within software and seeing if the results agree.

[3] *Analytical* thinking involves the "disassembly" of an idea into its constituent parts, analogous to dissection. *Synthetic* thinking involves the "assembly" of a new idea comprised of multiple concepts, analogous to construction. Both activities are high-level cognitive skills, extremely important for effective problem-solving, necessitating frequent challenge and regular practice to fully develop.

5.1.1 Reading outline and reflections

"Reading maketh a full man; conference a ready man; and writing an exact man" – Francis Bacon

Francis Bacon's advice is a blueprint for effective education: reading provides the learner with knowledge, writing focuses the learner's thoughts, and critical dialogue equips the learner to confidently communicate and apply their learning. Independent acquisition and application of knowledge is a powerful skill, well worth the effort to cultivate. To this end, students should read these educational resources closely, journal their own reflections on the reading, and discuss in detail their findings with classmates and instructor(s). You should be able to do all of the following after reading any instructional text:

☑ Briefly SUMMARIZE THE TEXT in the form of a journal entry documenting your learning as you progress through the course of study. Share this summary in dialogue with your classmates and instructor. Journaling is an excellent self-test of thorough reading because you cannot clearly express what you have not read or did not comprehend.

☑ Demonstrate ACTIVE READING STRATEGIES, including verbalizing your impressions as you read, simplifying long passages to convey the same ideas using fewer words, annotating text and illustrations with your own interpretations, working through mathematical examples shown in the text, cross-referencing passages with relevant illustrations and/or other passages, identifying problem-solving strategies applied by the author, etc. Technical reading is a special case of problem-solving, and so these strategies work precisely because they help solve any problem: paying attention to your own thoughts (metacognition), eliminating unnecessary complexities, identifying what makes sense, paying close attention to details, drawing connections between separated facts, and noting the successful strategies of others.

☑ Identify IMPORTANT THEMES, especially GENERAL LAWS and PRINCIPLES, expounded in the text and express them in the simplest of terms as though you were teaching an intelligent child. This emphasizes connections between related topics and develops your ability to communicate complex ideas to anyone.

☑ Form YOUR OWN QUESTIONS based on the reading, and then pose them to your instructor and classmates for their consideration. Anticipate both correct and incorrect answers, the incorrect answer(s) assuming one or more plausible misconceptions. This helps you view the subject from different perspectives to grasp it more fully.

☑ Devise EXPERIMENTS to test claims presented in the reading, or to disprove misconceptions. Predict possible outcomes of these experiments, and evaluate their meanings: what result(s) would confirm, and what would constitute disproof? Running mental simulations and evaluating results is essential to scientific and diagnostic reasoning.

☑ Specifically identify any points you found CONFUSING. The reason for doing this is to help diagnose misconceptions and overcome barriers to learning.

5.1.2 Foundational concepts

Correct analysis and diagnosis of electric circuits begins with a proper understanding of some basic concepts. The following is a list of some important concepts referenced in this module's full tutorial. Define each of them in your own words, and be prepared to illustrate each of these concepts with a description of a practical example and/or a live demonstration.

Authentication

Authorization

Firewall

Encryption

Defense-in-Depth

Server

Internet Protocol (IP)

Transmission Control Protocol (TCP)

OSI model

Hash

Programmable Logic Controller (PLC)

Variable Frequency Drive (VFD)

Ethernet

SCADA

Denial of Service

LAN

WAN

5.1.3 Personal computer security

A very good place to start exploring computer security is with your own personal computer. Regardless of manufacturer or operating system, your own personal computer has vulnerabilities and protections related to data security. Here we will explore some of them.

How secure (i.e. difficult to guess) is your password? When you must change your password, how do you choose and remember a new one?

When you typically log into your computer, how much privilege do you have as a user on the operating system? All modern operating systems have the ability to grant different levels of read and write and execute access to different users depending on how those user accounts are set up. The most privileged level of access is often called "Administrator" (or "root" in Unix-based operating systems) in which that user is allowed to do anything at all, which is convenient when doing tasks such as installing new software, but represents a vulnerability in that any malicious access made under that user session enjoys the same unrestricted power (to do harm). Inspect the user accounts on your computer's operating system, and identify the level of privilege you typically use when logged on.

Explain the difference between anti-virus software and firewall software.

Identify how to disable your computer's wireless network access, and explain why that might be a good policy in certain situations.

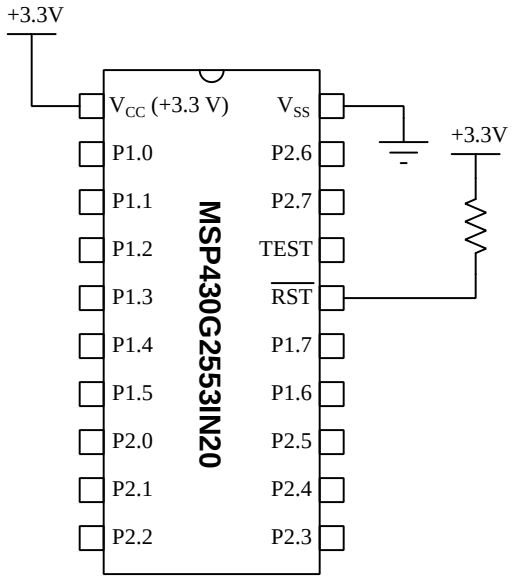
Do you store personally important files on your computer, such as financial or legal records? If so, is that entirely necessary? Identify ways to maintain the security of those critical files if they need to be maintained in digital form.

Challenges

- Identify strategies for inventing strong passwords.
- Identify a way that malicious code (malware) might enter your computer even if it never connects to the internet.

5.1.4 Microcontroller security fuse

The Texas Instruments model MSP430 microcontroller ICs contain a *security fuse* which may be intentionally blown by applying an abnormally high voltage between the `TEST` pin and ground (V_{SS}). An example of a microcontroller within this "family" of products is the 20-pin MSP430G2553IN20 shown below:



Identify the recommended fuse-blowing voltage to apply to this IC, and also what the consequence(s) is/are to blowing it.

Challenges

- Describe an application where you might wish to blow this fuse.

5.1.5 Vulnerability databases

Identify at least two different online databases of known cybersecurity vulnerabilities – especially industrial system vulnerabilities – and find within those databases some specific examples.

Challenges

- For each vulnerability you researched, what type digital platform (e.g. PLC, SCADA, personal computer) did it compromise?
- For each vulnerability you researched, what is the recommended mitigation?

5.2 Quantitative reasoning

These questions are designed to stimulate your computational thinking. In a Socratic discussion with your instructor, the goal is for these questions to reveal your mathematical approach(es) to problem-solving so that good technique and sound reasoning may be reinforced. Your instructor may also pose additional questions based on those assigned, in order to observe your problem-solving firsthand.

Mental arithmetic and estimations are strongly encouraged for all calculations, because without these abilities you will be unable to readily detect errors caused by calculator misuse (e.g. keystroke errors).

You will note a conspicuous lack of answers given for these quantitative questions. Unlike standard textbooks where answers to every other question are given somewhere toward the back of the book, here in these learning modules students must rely on other means to check their work. My advice is to use circuit simulation software such as SPICE to check the correctness of quantitative answers. Refer to those learning modules within this collection focusing on SPICE to see worked examples which you may use directly as practice problems for your own study, and/or as templates you may modify to run your own analyses and generate your own practice problems.

Completely worked example problems found in the Tutorial may also serve as "test cases[4]" for gaining proficiency in the use of circuit simulation software, and then once that proficiency is gained you will never need to rely[5] on an answer key!

[4] In other words, set up the circuit simulation software to analyze the same circuit examples found in the Tutorial. If the simulated results match the answers shown in the Tutorial, it confirms the simulation has properly run. If the simulated results disagree with the Tutorial's answers, something has been set up incorrectly in the simulation software. Using every Tutorial as practice in this way will quickly develop proficiency in the use of circuit simulation software.

[5] This approach is perfectly in keeping with the instructional philosophy of these learning modules: *teaching students to be self-sufficient thinkers*. Answer keys can be useful, but it is even more useful to your long-term success to have a set of tools on hand for checking your own work, because once you have left school and are on your own, there will no longer be "answer keys" available for the problems you will have to solve.

5.2.1 Miscellaneous physical constants

Note: constants shown in **bold** type are *exact*, not approximations. Values inside of parentheses show one standard deviation (σ) of uncertainty in the final digits: for example, the magnetic permeability of free space value given as $1.25663706212(19) \times 10^{-6}$ H/m represents a center value (i.e. the location parameter) of $1.25663706212 \times 10^{-6}$ Henrys per meter with one standard deviation of uncertainty equal to $0.0000000000019 \times 10^{-6}$ Henrys per meter.

Avogadro's number (N_A) = **6.02214076** $\times 10^{23}$ **per mole** (mol^{-1})

Boltzmann's constant (k) = **1.380649** $\times 10^{-23}$ **Joules per Kelvin** (J/K)

Electronic charge (e) = **1.602176634** $\times 10^{-19}$ **Coulomb** (C)

Faraday constant (F) = **96,485.33212...** $\times 10^{4}$ **Coulombs per mole** (C/mol)

Magnetic permeability of free space (μ_0) = $1.25663706212(19) \times 10^{-6}$ Henrys per meter (H/m)

Electric permittivity of free space (ϵ_0) = $8.8541878128(13) \times 10^{-12}$ Farads per meter (F/m)

Characteristic impedance of free space (Z_0) = 376.730313668(57) Ohms (Ω)

Gravitational constant (G) = $6.67430(15) \times 10^{-11}$ cubic meters per kilogram-seconds squared ($m^3/kg\text{-}s^2$)

Molar gas constant (R) = **8.314462618... Joules per mole-Kelvin** (J/mol-K) = 0.08205746(14) liters-atmospheres per mole-Kelvin

Planck constant (h) = **6.62607015** $\times 10^{-34}$ **joule-seconds** (J-s)

Stefan-Boltzmann constant (σ) = **5.670374419...** $\times 10^{-8}$ **Watts per square meter-Kelvin**^{4} ($W/m^2 \cdot K^4$)

Speed of light in a vacuum (c) = **299,792,458 meters per second** (m/s) = 186282.4 miles per second (mi/s)

Note: All constants taken from NIST data "Fundamental Physical Constants – Complete Listing", from `http://physics.nist.gov/constants`, National Institute of Standards and Technology (NIST), 2018 CODATA Adjustment.

5.2.2 Introduction to spreadsheets

A powerful computational tool you are encouraged to use in your work is a *spreadsheet*. Available on most personal computers (e.g. Microsoft Excel), *spreadsheet* software performs numerical calculations based on number values and formulae entered into cells of a grid. This grid is typically arranged as lettered columns and numbered rows, with each cell of the grid identified by its column/row coordinates (e.g. cell `B3`, cell `A8`). Each cell may contain a string of text, a number value, or a mathematical formula. The spreadsheet automatically updates the results of all mathematical formulae whenever the entered number values are changed. This means it is possible to set up a spreadsheet to perform a series of calculations on entered data, and those calculations will be re-done by the computer any time the data points are edited in any way.

For example, the following spreadsheet calculates average speed based on entered values of distance traveled and time elapsed:

	A	B	C	D
1	Distance traveled	46.9	Kilometers	
2	Time elapsed	1.18	Hours	
3	Average speed	= B1 / B2	km/h	
4				
5				

Text labels contained in cells `A1` through `A3` and cells `C1` through `C3` exist solely for readability and are not involved in any calculations. Cell `B1` contains a sample distance value while cell `B2` contains a sample time value. The formula for computing speed is contained in cell `B3`. Note how this formula begins with an "equals" symbol (`=`), references the values for distance and speed by lettered column and numbered row coordinates (`B1` and `B2`), and uses a forward slash symbol for division (`/`). The coordinates `B1` and `B2` function as *variables*[6] would in an algebraic formula.

When this spreadsheet is executed, the numerical value `39.74576` will appear in cell `B3` rather than the formula `= B1 / B2`, because 39.74576 is the computed speed value given 46.9 kilometers traveled over a period of 1.18 hours. If a different numerical value for distance is entered into cell `B1` or a different value for time is entered into cell `B2`, cell `B3`'s value will automatically update. All you need to do is set up the given values and any formulae into the spreadsheet, and the computer will do all the calculations for you.

Cell `B3` may be referenced by other formulae in the spreadsheet if desired, since it is a variable just like the given values contained in `B1` and `B2`. This means it is possible to set up an entire chain of calculations, one dependent on the result of another, in order to arrive at a final value. The arrangement of the given data and formulae need not follow any pattern on the grid, which means you may place them anywhere.

[6] Spreadsheets may also provide means to attach text labels to cells for use as variable names (Microsoft Excel simply calls these labels "names"), but for simple spreadsheets such as those shown here it's usually easier just to use the standard coordinate naming for each cell.

Common[7] arithmetic operations available for your use in a spreadsheet include the following:

- Addition (`+`)
- Subtraction (`-`)
- Multiplication (`*`)
- Division (`/`)
- Powers (`^`)
- Square roots (`sqrt()`)
- Logarithms (`ln()` , `log10()`)

Parentheses may be used to ensure[8] proper order of operations within a complex formula. Consider this example of a spreadsheet implementing the *quadratic formula*, used to solve for roots of a polynomial expression in the form of $ax^2 + bx + c$:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

	A	B
1	x_1	= (-B4 + sqrt((B4^2) - (4*B3*B5))) / (2*B3)
2	x_2	= (-B4 - sqrt((B4^2) - (4*B3*B5))) / (2*B3)
3	a =	9
4	b =	5
5	c =	-2

This example is configured to compute roots[9] of the polynomial $9x^2 + 5x - 2$ because the values of 9, 5, and -2 have been inserted into cells `B3`, `B4`, and `B5`, respectively. Once this spreadsheet has been built, though, it may be used to calculate the roots of *any* second-degree polynomial expression simply by entering the new a, b, and c coefficients into cells `B3` through `B5`. The numerical values appearing in cells `B1` and `B2` will be automatically updated by the computer immediately following any changes made to the coefficients.

[7] Modern spreadsheet software offers a bewildering array of mathematical functions you may use in your computations. I recommend you consult the documentation for your particular spreadsheet for information on operations other than those listed here.

[8] Spreadsheet programs, like text-based programming languages, are designed to follow standard order of operations by default. However, my personal preference is to use parentheses even where strictly unnecessary just to make it clear to any other person viewing the formula what the intended order of operations is.

[9] Reviewing some algebra here, a *root* is a value for x that yields an overall value of zero for the polynomial. For this polynomial ($9x^2 + 5x - 2$) the two roots happen to be $x = 0.269381$ and $x = -0.82494$, with these values displayed in cells `B1` and `B2`, respectively upon execution of the spreadsheet.

Alternatively, one could break up the long quadratic formula into smaller pieces like this:

$$y = \sqrt{b^2 - 4ac} \qquad z = 2a$$

$$x = \frac{-b \pm y}{z}$$

	A	B	C
1	x_1	= (-B4 + C1) / C2	= sqrt((B4^2) - (4*B3*B5))
2	x_2	= (-B4 - C1) / C2	= 2*B3
3	a =	9	
4	b =	5	
5	c =	-2	

Note how the square-root term (y) is calculated in cell C1, and the denominator term (z) in cell C2. This makes the two final formulae (in cells B1 and B2) simpler to interpret. The positioning of all these cells on the grid is completely arbitrary[10] – all that matters is that they properly reference each other in the formulae.

Spreadsheets are particularly useful for situations where the same set of calculations representing a circuit or other system must be repeated for different initial conditions. The power of a spreadsheet is that it automates what would otherwise be a tedious set of calculations. One specific application of this is to simulate the effects of various components within a circuit failing with abnormal values (e.g. a shorted resistor simulated by making its value nearly zero; an open resistor simulated by making its value extremely large). Another application is analyzing the behavior of a circuit design given new components that are out of specification, and/or aging components experiencing drift over time.

[10] My personal preference is to locate all the "given" data in the upper-left cells of the spreadsheet grid (each data point flanked by a sensible name in the cell to the left and units of measurement in the cell to the right as illustrated in the first distance/time spreadsheet example), sometimes coloring them in order to clearly distinguish which cells contain entered data versus which cells contain computed results from formulae. I like to place all formulae in cells below the given data, and try to arrange them in logical order so that anyone examining my spreadsheet will be able to figure out *how* I constructed a solution. This is a general principle I believe all computer programmers should follow: *document and arrange your code to make it easy for other people to learn from it.*

5.2.3 Password strength

Calculate the number of possible passwords available from a 5-character, case-insensitive password consisting only of alphabetical characters.

If the system is case-sensitive (i.e. it can distinguish between upper- and lower-case letters and not treat them as identical), how does this influence the number of possible passwords?

This raises an interesting question: which is more effective in "strengthening" a password against a brute-force attack, more symbols or more characters? Mathematically prove your conclusion.

Challenges

- Based on what you conclude here, devise a strategy for generating strong passwords for your own use.

5.2.4 Exclusive-OR encryption

A very simple method of encrypting a digital message is to perform a bitwise-XOR logical function with every byte of data in the message. A "bitwise-XOR" function is where two digital words of equal length are processed by an Exclusive-OR function on each pair of respective bits. Two different examples are shown below:

Original data	1	0	0	1	0	1	1	1		Original data	0	0	0	1	0	1	1	1
Key	0	1	1	1	1	1	0	0		Key	1	1	1	1	0	0	0	0
Encrypted data	1	1	1	0	1	0	1	1		Encrypted data	1	1	1	0	0	1	1	1

Each bit within the "key" word determines whether the respective bit in the original data will be inverted or not. This same key may then be used to decrypt the encrypted message to give us the original data:

Encrypted data	1	1	1	0	1	0	1	1		Encrypted data	1	1	1	0	0	1	1	1
Key	0	1	1	1	1	1	0	0		Key	1	1	1	1	0	0	0	0
Original data	1	0	0	1	0	1	1	1		Original data	0	0	0	1	0	1	1	1

Determine the encrypted version of the ASCII text string `Hello there!!` using a key of `0x01`.

Challenges

- Is this a form of *public* key encryption or *private* key encryption?

5.3 Diagnostic reasoning

These questions are designed to stimulate your deductive and inductive thinking, where you must apply general principles to specific scenarios (deductive) and also derive conclusions about the failed circuit from specific details (inductive). In a Socratic discussion with your instructor, the goal is for these questions to reinforce your recall and use of general circuit principles and also challenge your ability to integrate multiple symptoms into a sensible explanation of what's wrong in a circuit. Your instructor may also pose additional questions based on those assigned, in order to further challenge and sharpen your diagnostic abilities.

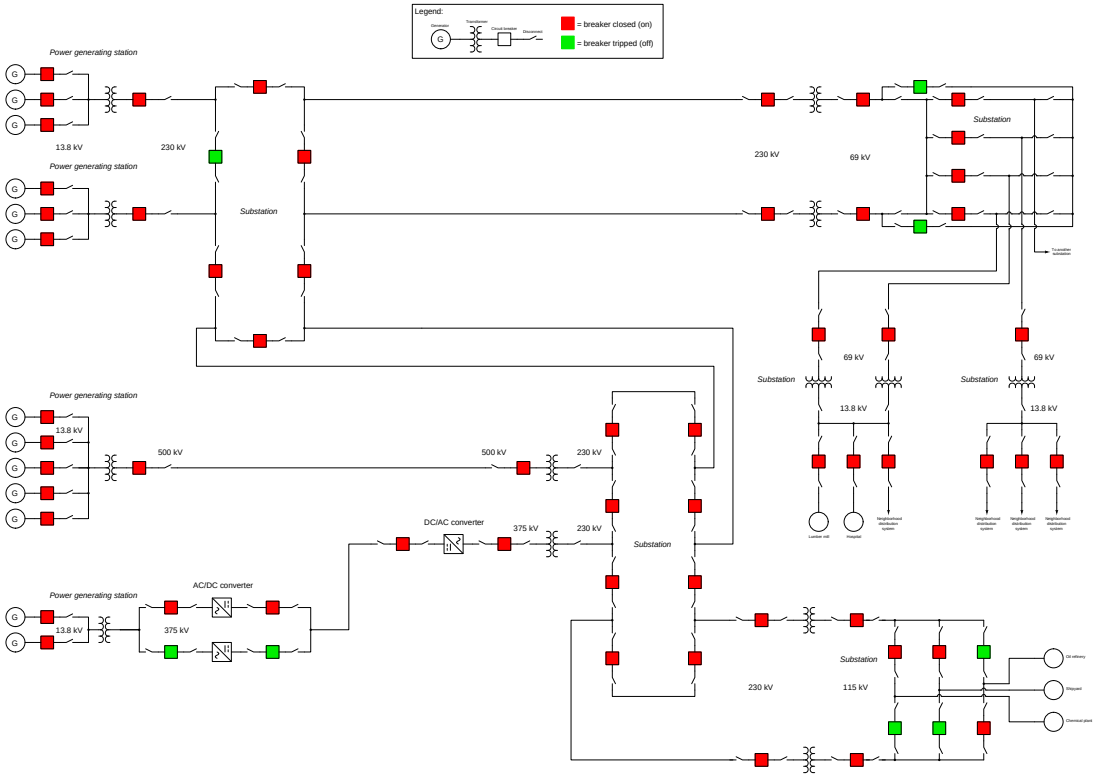
As always, your goal is to fully *explain* your analysis of each problem. Simply obtaining a correct answer is not good enough – you must also demonstrate sound reasoning in order to successfully complete the assignment. Your instructor's responsibility is to probe and challenge your understanding of the relevant principles and analytical processes in order to ensure you have a strong foundation upon which to build further understanding.

You will note a conspicuous lack of answers given for these diagnostic questions. Unlike standard textbooks where answers to every other question are given somewhere toward the back of the book, here in these learning modules students must rely on other means to check their work. The best way by far is to debate the answers with fellow students and also with the instructor during the Socratic dialogue sessions intended to be used with these learning modules. Reasoning through challenging questions with other people is an excellent tool for developing strong reasoning skills.

Another means of checking your diagnostic answers, where applicable, is to use circuit simulation software to explore the effects of faults placed in circuits. For example, if one of these diagnostic questions requires that you predict the effect of an open or a short in a circuit, you may check the validity of your work by simulating that same fault (substituting a very high resistance in place of that component for an open, and substituting a very low resistance for a short) within software and seeing if the results agree.

5.3.1 Power grid vulnerabilities

Examine the following single-line diagram of a wide-area electric power system ("grid") and analyze it for points of vulnerability, assuming all circuit breakers in this system are equipped with digital protective relays, monitored and controlled via networked SCADA system:



Identify any "high-value targets" for cyber-attack in this system. These would be components in the power system most likely to result in extended outages and/or maximum effect from an outage of any duration.

Do you see any differences in the value of these targets as viewed from different attackers' perspectives? Consider the following categories of attackers:

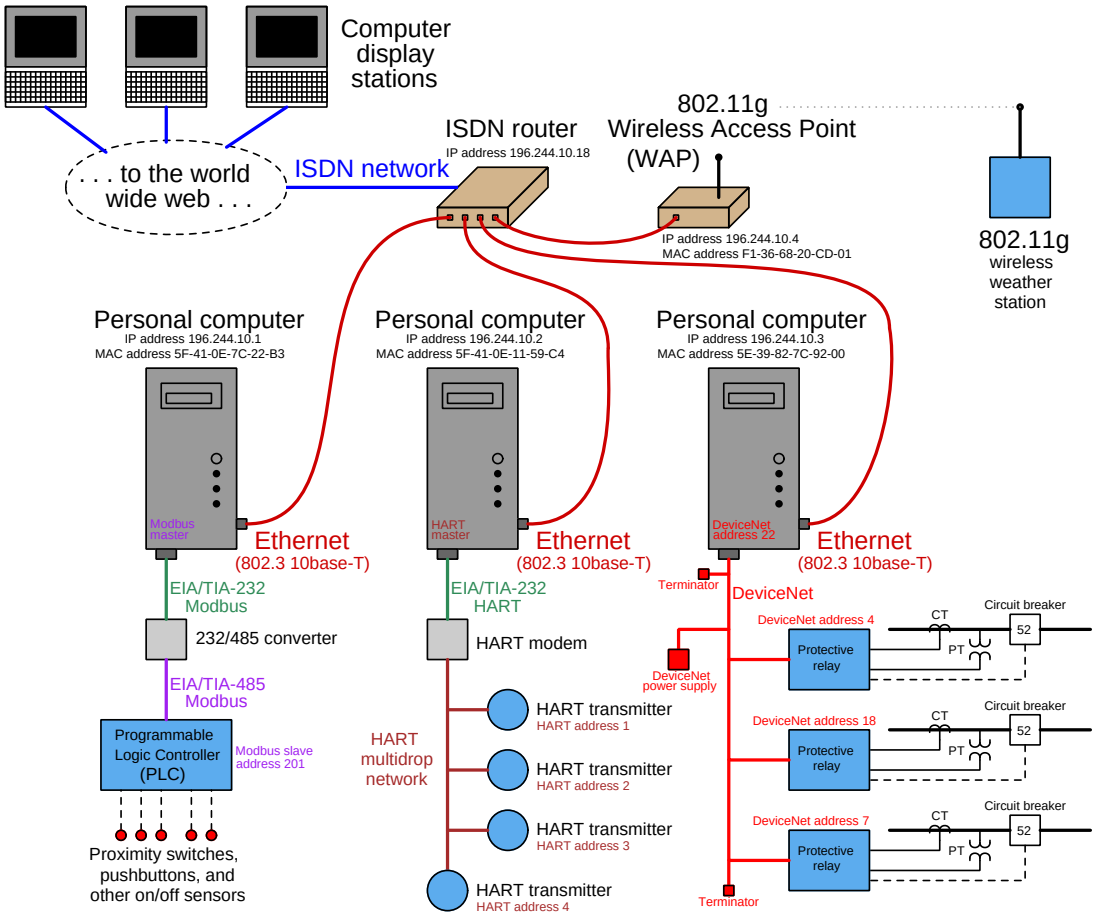
- Military (i.e. a nation-state attack)
- Terrorist
- Individual hacker

Challenges

- Identify within one of the substations shown in this system the presense of redundant circuit breakers.

5.3.2 Fortifying a generic SCADA system

Examine this illustration of an industrial SCADA (Supervisory Control And Data Acquisition) system to identify any potential security vulnerabilities, and then recommend ways to fortify it from attack:



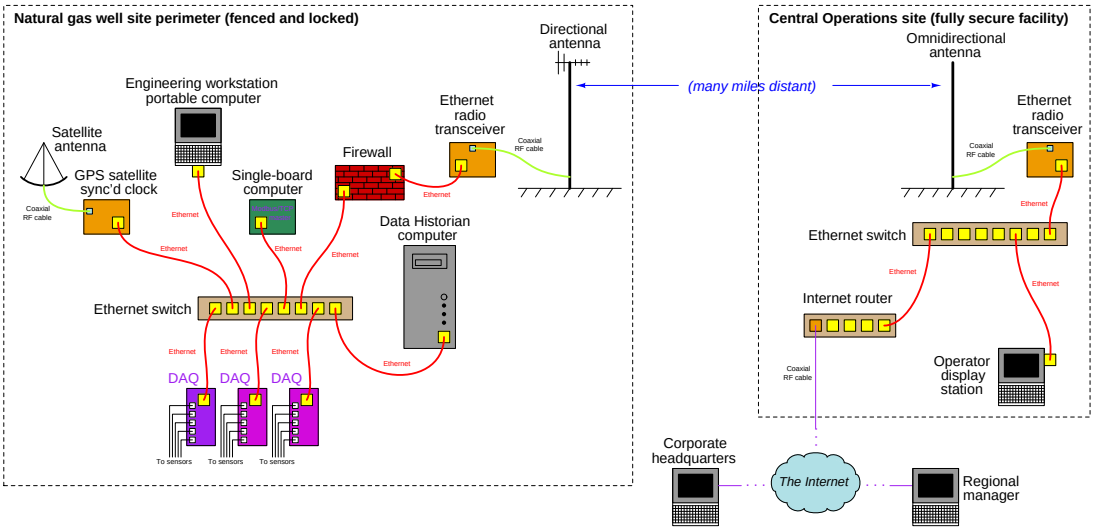
Challenges

- Do you see any application for a *firewall* in this system? If so, where should that firewall be located?
- Do you see any application for a *VPN* in this system? If so, how should that VPN be implemented?
- A good strategy for formulating any security policy is to first identify all functions provided by the system, then identify all authorized users of the system, delineate which users get to

access which functions, decide on security techniques for limiting access accordingly, and finally forbid all other uses of the system. Apply this strategy to the analysis of the system shown, and try to either identify those elements or formulate questions you would ask of the system's owner to identify those elements.

5.3.3 Fortifying a natural gas SCADA system

The following diagram shows the components of a SCADA (Supervisory Control And Data Acquisition) system, used to collect process data from a remote natural gas well, archive that data over time, and make that archived data available to personnel at both a centralized operations site as well as to corporate and regional headquarters via the internet.



Basic system operation is as follows: the DAQ units receive analog and discrete inputs from a variety of sensors on the natural gas wellhead. The single-board computer uses Modbus/TCP to poll these DAQ units and read their input states, assigning tag names to all data, scaling the data (where applicable) to engineering units from the raw signal voltages (e.g. a 3 Volt analog input signal becomes 2400 kPa of gas pressure), and time-stamping each signal with a time value polled from the GPS clock (also via Modbus/TCP). A portable Engineering Workstation computer is used to edit and update software in the single-board computer as necessary, as well as perform diagnostic tests on the system. The Data Historian computer periodically reads the single-board computer's collated process data using SFTP (Secure File Transfer Protocol) and archives it in a database holding records for up to several years. Any other operations computer connected to this system (e.g. the Operator Display Station at the Central site, the Regional Manager's computer, Corporate Headquarters) has access to this archived data by reading database files off of the Historian computer.

Multiple natural gas wells are interconnected in the same way to the centralized operations site – the diagram only shows one gas well's communication network for simplicity.

Identify as many potential cyber-vulnerabilities evident in this system diagram as you can, explaining how hackers might gain unauthorized access to critical system functions.

Modify this system to include a DMZ for added security. Be as specific as you can in your modifications.

Suggest alterations (other than a DMZ) which could enhance the cyber-security of this SCADA system.

Challenges

- The directional antenna at the gas well site not only concentrates RF energy in one direction for greater range, but it also serves as an aid to security. Explain how this is.

Appendix A

Problem-Solving Strategies

The ability to solve complex problems is arguably one of the most valuable skills one can possess, and this skill is particularly important in any science-based discipline.

- Study principles, not procedures. Don't be satisfied with merely knowing how to compute solutions – learn *why* those solutions work.
- Identify what it is you need to solve, identify all relevant data, identify all units of measurement, identify any general principles or formulae linking the given information to the solution, and then identify any "missing pieces" to a solution. Annotate all diagrams with this data.
- Sketch a diagram to help visualize the problem. When building a real system, always devise a plan for that system and analyze its function *before* constructing it.
- Follow the units of measurement and meaning of every calculation. If you are ever performing mathematical calculations as part of a problem-solving procedure, and you find yourself unable to apply each and every intermediate result to some aspect of the problem, it means you don't understand what you are doing. Properly done, every mathematical result should have practical meaning for the problem, and not just be an abstract number. You should be able to identify the proper units of measurement for each and every calculated result, and show where that result fits into the problem.
- Perform "thought experiments" to explore the effects of different conditions for theoretical problems. When troubleshooting real systems, perform *diagnostic tests* rather than visually inspecting for faults, the best diagnostic test being the one giving you the most information about the nature and/or location of the fault with the fewest steps.
- Simplify the problem until the solution becomes obvious, and then use that obvious case as a model to follow in solving the more complex version of the problem.
- Check for exceptions to see if your solution is incorrect or incomplete. A good solution will work for *all* known conditions and criteria. A good example of this is the process of testing scientific hypotheses: the task of a scientist is not to find support for a new idea, but rather to *challenge* that new idea to see if it holds up under a battery of tests. The philosophical

principle of *reductio ad absurdum* (i.e. disproving a general idea by finding a specific case where it fails) is useful here.

- Work "backward" from a hypothetical solution to a new set of given conditions.
- Add quantities to problems that are qualitative in nature, because sometimes a little math helps illuminate the scenario.
- Sketch graphs illustrating how variables relate to each other. These may be quantitative (i.e. with realistic number values) or qualitative (i.e. simply showing increases and decreases).
- Treat quantitative problems as qualitative in order to discern the relative magnitudes and/or directions of change of the relevant variables. For example, try determining what happens if a certain variable were to increase or decrease before attempting to precisely calculate quantities: how will each of the dependent variables respond, by increasing, decreasing, or remaining the same as before?
- Consider limiting cases. This works especially well for qualitative problems where you need to determine which direction a variable will change. Take the given condition and magnify that condition to an extreme degree as a way of simplifying the direction of the system's response.
- Check your work. This means regularly testing your conclusions to see if they make sense. This does *not* mean repeating the same steps originally used to obtain the conclusion(s), but rather to use some other means to check validity. Simply repeating procedures often leads to *repeating the same errors* if any were made, which is why alternative paths are better.

Appendix B

Instructional philosophy

B.1 First principles of learning

- **Anyone can learn anything** given appropriate time, effort, resources, challenges, encouragement, and expectations. Dedicating time and investing effort are the student's responsibility; providing resources, challenges, and encouragement are the teacher's responsibility; high expectations are a responsibility shared by both student and teacher.

- **Transfer is not automatic.** The human mind has a natural tendency to compartmentalize information, which means the process of taking knowledge learned in one context and applying it to another usually does not come easy and therefore should never be taken for granted.

- **Learning is iterative.** The human mind rarely learns anything perfectly on the first attempt. Anticipate mistakes and plan for multiple tries to achieve full understanding, using the lessons of those mistakes as feedback to guide future attempts.

- **Information is absorbed, but understanding is created.** Facts and procedures may be memorized easily enough by repeated exposure, but the ability to reliably apply principles to novel scenarios only comes through intense personal effort. This effort is fundamentally creative in nature: casting new concepts in one's own words, running mental experiments to perceive patterns and trends, etc. These acts of making knowledge "one's own" need not be perfect in order to be effective, as the value lies in the activity and not necessarily the finished product. Depth of understanding and permanence of memory are both directly related to mental effort during the learning process – i.e. *easy learning is shallow and fleeting.*

- **Education trumps training.** There is no such thing as an entirely isolated subject, as all fields of knowledge are connected. Training is narrowly-focused and task-oriented. Education is broad-based and principle-oriented. When preparing for a life-long technical career, education beats training every time.

- **Character matters.** Poor habits are more destructive than deficits of knowledge or skill. This is especially true in collective endeavors, where a team's ability to function depends on trust between its members. Simply put, no one wants an untrustworthy person on their team. An essential component of education then, is character development.

- **People learn to be responsible by bearing responsibility.** An irresponsible person is someone who has never *had* to be responsible for anything that mattered enough to them. Just as anyone can learn anything, anyone can become responsible if the personal cost of irresponsibility becomes high enough.

- **What gets measured, gets done.** Accurate and relevant assessment of learning is key to ensuring all students learn. Therefore, it is imperative to measure what matters.

- **Failure is nothing to fear.** Every human being fails, and fails in multiple ways at multiple times. Eventual success only happens when we don't stop trying.

B.2 Proven strategies for instructors

- Assume every student is capable of learning anything they desire given the proper conditions. Treat them as capable adults by granting real responsibility and avoiding artificial incentives such as merit or demerit points.
- Create a consistent culture of high expectations across the entire program of study. Demonstrate and encourage patience, persistence, and a healthy sense of self-skepticism. Anticipate and de-stigmatize error. Teach respect for the capabilities of others as well as respect for one's own fallibility.
- Replace lecture with "inverted" instruction, where students first encounter new concepts through reading and then spend class time in Socratic dialogue with the instructor exploring those concepts and solving problems individually. There is a world of difference between observing someone solve a problem versus actually solving a problem yourself, and so the point of this form of instruction is to place students in a position where they *cannot* passively observe.
- Require students to read extensively, write about what they learn, and dialogue with you and their peers to sharpen their understanding. Apply Francis Bacon's advice that "reading maketh a full man; conference a ready man; and writing an exact man". These are complementary activities helping students expand their confidence and abilities.
- Use artificial intelligence (AI) to challenge student understanding rather than merely provide information. Find productive ways for AI to critique students' clarity of thought and of expression, for example by employing AI as a Socratic-style interlocutor or as a reviewer of students' journals. Properly applied, AI has the ability to expand student access to critical review well outside the bounds of their instructor's reach.
- Build frequent and rapid feedback into the learning process so that students know at all times how well they are learning, to identify problems early and fix them before they grow. Model the intellectual habit of self-assessing and self-correcting your own understanding (i.e. a cognitive *feedback loop*), encouraging students to do the same.
- Use "mastery" as the standard for every assessment, which means the exam or experiment or project must be done with 100% competence in order to pass. Provide students with multiple opportunity for re-tries (different versions of the assessment every time).
- Require students to devise their own hypotheses and procedures on all experiments, so that the process is truly a scientific one. Have students assess their proposed experimental procedures for risk and devise mitigations for those risks. Let nothing be pre-designed about students' experiments other than a stated task (i.e. what principle the experiment shall test) at the start and a set of demonstrable knowledge and skill objectives at the end.
- Have students build as much of their lab equipment as possible: building power sources, building test assemblies[1], and building complete working systems (no kits!). In order to provide

[1] In the program I teach, every student builds their own "Development Board" consisting of a metal chassis with DIN rail, terminal blocks, and an AC-DC power supply of their own making which functions as a portable lab environment they can use at school as well as take home.

this same "ground-up" experience for every new student, this means either previous students take their creations with them, or the systems get disassembled in preparation for the new students, or the systems grow and evolve with each new student group.

- Incorporate external accountability for you and for your students, continuously improving the curriculum and your instructional methods based on proven results. Have students regularly network with active professionals through participation in advisory committee meetings, service projects, tours, jobshadows, internships, etc. Practical suggestions include requiring students to design and build projects for external clients (e.g. community groups, businesses, different departments within the institution), and also requiring students attend all technical advisory committee meetings and to dialogue with the industry representatives at those meetings.
- Repeatedly explore difficult-to-learn concepts across multiple courses, so that students have multiple opportunities to build their understanding.
- Relate all new concepts, whenever possible, to previous concepts and to relevant physical laws. Challenge each and every student, every day, to *reason* from concept to concept and to explain the logical connections between. Challenge students to verify their conclusions by multiple approaches (e.g. double-checking their work using different methods). Ask *"Why?"* often.
- Maintain detailed records on each student's performance and share these records privately with them. These records should include academic performance as well as professionally relevant behavioral tendencies.
- Hold mandatory "check-in" meetings between all program faculty and each new student during their first term. Offer these to all other students as an option, except for any students continuing to manifest unprofessional behaviors, poor academic performance, or who have some other need for a face-to-face meeting with faculty.
- Address problems while they are small, before they grow larger. This is equally true for tutoring technical concepts as it is for helping students build professional habits.
- Build rigorous quality control into the curriculum to ensure every student masters every important concept, and that the mastery is retained over time. This includes (1) review questions added to every exam to re-assess knowledge taught in previous terms, (2) cumulative exams at the end of every term to re-assess all important concepts back to the very beginning of the program, and (3) review assessments in practical (hands-on) coursework to ensure critically-important skills were indeed taught and are still retained. What you will find by doing this is that it actually boosts retention of students by ensuring that important knowledge gets taught and is retained over long spans of time. In the absence of such quality control, student learning and retention tends to be spotty and this contributes to drop-out and failure rates later in their education.
- Finally, *never rush learning.* Education is not a race. Give your students ample time to digest complex ideas, as you continually remind yourself of just how long it took you to achieve mastery! Long-term retention and the consistently correct application of concepts are always the result of *focused effort over long periods of time* which means there are no shortcuts to learning.

B.3 Proven strategies for students

The single most important piece of advice I have for any student of any subject is to take responsibility for your own development in all areas of life including mental development. Expecting others in your life to entirely guide your own development is a recipe for disappointment. This is just as true for students enrolled in formal learning institutions as it is for auto-didacts pursuing learning entirely on their own. Learning to think in new ways is key to being able to gainfully use information, to make informed decisions about your life, and to best serve those you care about. With this in mind, I offer the following advice to students:

- **Approach all learning as valuable**. No matter what course you take, no matter who you learn from, no matter the subject, there is something useful in every learning experience. If you don't see the value of every new experience, you are not looking closely enough!
- **Continually challenge yourself**. Let other people take shortcuts and find easy answers to easy problems. The purpose of education is to stretch your mind, in order to shape it into a more powerful tool. This doesn't come by taking the path of least resistance. An excellent analogy for an empowering education is productive physical exercise: becoming stronger, more flexible, and more persistent only comes through intense personal effort.
- **Master the use of language**. This includes reading extensively, writing every day, listening closely, and speaking articulately. To a great extent language channels and empowers thought, so the better you are at wielding language the better you will be at grasping abstract concepts and articulating them not only for your benefit but for others as well.
- **Do not limit yourself to the resources given to you**. Read books that are not on the reading list. Run experiments that aren't assigned to you. Form study groups outside of class. Take an entrepreneurial approach to your own education, as though it were a business you were building for your future benefit.
- **Express and share what you learn**. Take every opportunity to teach what you have learned to others, as this will not only help them but will also strengthen your own understanding[2].
- Realize that **no one can give you understanding**, just as no one can give you physical fitness. These both must be *built*.
- **Above all, recognize that learning is hard work, and that a certain level of frustration is unavoidable**. There are times when you will struggle to grasp some of these concepts, and that struggle is a natural thing. Take heart that it will yield with persistent and varied[3] effort, and never give up! That concepts don't immediately come to you is not a sign of something wrong, but rather of something right: that you have found a worthy challenge!

[2] On a personal note, I was surprised to learn just how much my own understanding of electronics and related subjects was strengthened by becoming a teacher. When you are tasked every day with helping other people grasp complex topics, it catalyzes your own learning by giving you powerful incentives to study, to articulate your thoughts, and to reflect deeply on the process of learning.

[3] As the old saying goes, "Insanity is trying the same thing over and over again, expecting different results." If you find yourself stumped by something in the text, you should attempt a different approach. Alter the thought experiment, change the mathematical parameters, do whatever you can to see the problem in a slightly different light, and then the solution will often present itself more readily.

B.4 Design of these learning modules

"The unexamined circuit is not worth energizing" – Socrates (if he had taught electricity)

These learning modules, although useful for self-study, were designed to be used in a formal learning environment where a subject-matter expert challenges students to digest the content and exercise their critical thinking abilities in the answering of questions and in the construction and testing of working circuits. Every effort has been made to embed the following instructional and assessment philosophies within:

- The first goal of education is to enhance clear and independent thought, in order that every student reach their fullest potential in a highly complex and inter-dependent world. Robust reasoning is *always* more important than particulars of any subject matter, because its application is universal.
- Literacy is fundamental to independent learning and thought because text continues to be the most efficient way to communicate complex ideas over space and time. Those who cannot read with ease are limited in their ability to acquire knowledge and perspective.
- Articulate communication is fundamental to work that is complex and interdisciplinary.
- Faulty assumptions and poor reasoning are best corrected through challenge, not presentation. The rhetorical technique of *reductio ad absurdum* (disproving an assertion by exposing an absurdity) works well to discipline student's minds, not only to correct the problem at hand but also to learn how to detect and correct future errors.
- Important principles should be repeatedly explored and widely applied throughout a course of study, not only to reinforce their importance and help ensure their mastery, but also to showcase the interconnectedness and utility of knowledge.

These learning modules were expressly designed to be used in an "inverted" teaching environment[4] where students first read the introductory and tutorial chapters on their own, then individually attempt to answer the questions and construct working circuits according to the experiment and project guidelines. The instructor never lectures, but instead meets regularly with each individual student to review their progress, answer questions, identify misconceptions, and challenge the student to new depths of understanding through further questioning. Regular meetings between instructor and student should resemble a Socratic[5] dialogue, where questions serve as scalpels to dissect topics and expose assumptions. The student passes each module only after consistently demonstrating their ability to logically analyze and correctly apply all major concepts in each question or project/experiment. The instructor must be vigilant in probing each student's understanding to ensure they are truly *reasoning* and not just *memorizing*. This is why "Challenge" points appear throughout, as prompts for students to think deeper about topics and as starting points for instructor queries. Sometimes these challenge points require additional knowledge that hasn't been covered in the series to answer in full. This is okay, as the major purpose of the Challenges is to stimulate analysis and synthesis on the part of each student.

The instructor must possess enough mastery of the subject matter and awareness of students' reasoning to generate their own follow-up questions to practically any student response. Even completely correct answers given by the student should be challenged by the instructor for the purpose of having students practice articulating their thoughts and defending their reasoning. Conceptual errors committed by the student should be exposed and corrected not by direct instruction, but rather by reducing the errors to an absurdity[6] through well-chosen questions and thought experiments posed by the instructor. Becoming proficient at this style of instruction requires time and dedication, but the positive effects on critical thinking for both student and instructor are spectacular.

An inspection of these learning modules reveals certain unique characteristics. One of these is a bias toward thorough explanations in the tutorial chapters. Without a live instructor to explain concepts and applications to students, the text itself must fulfill this role. This philosophy results in lengthier explanations than what you might typically find in a textbook, each step of the reasoning process fully explained, including footnotes addressing common questions and concerns students raise while learning these concepts. Each tutorial seeks to not only explain each major concept in sufficient detail, but also to explain the logic of each concept and how each may be developed

[4] In a traditional teaching environment, students first encounter new information via *lecture* from an expert, and then independently apply that information via *homework*. In an "inverted" course of study, students first encounter new information via *homework*, and then independently apply that information under the scrutiny of an expert. The expert's role in lecture is to simply *explain*, but the expert's role in an inverted session is to *challenge*, *critique*, and if necessary *explain* where gaps in understanding still exist.

[5] Socrates is a figure in ancient Greek philosophy famous for his unflinching style of questioning. Although he authored no texts, he appears as a character in Plato's many writings. The essence of Socratic philosophy is to leave no question unexamined and no point of view unchallenged. While purists may argue a topic such as electric circuits is too narrow for a true Socratic-style dialogue, I would argue that the essential thought processes involved with scientific reasoning on *any* topic are not far removed from the Socratic ideal, and that students of electricity and electronics would do very well to challenge assumptions, pose thought experiments, identify fallacies, and otherwise employ the arsenal of critical thinking skills modeled by Socrates.

[6] This rhetorical technique is known by the Latin phrase *reductio ad absurdum*. The concept is to expose errors by counter-example, since only one solid counter-example is necessary to disprove a universal claim. As an example of this, consider the common misconception among beginning students of electricity that voltage cannot exist without current. One way to apply *reductio ad absurdum* to this statement is to ask how much current passes through a fully-charged battery connected to nothing (i.e. a clear example of voltage existing without current).

from "first principles". Again, this reflects the goal of developing clear and independent thought in students' minds, by showing how clear and logical thought was used to forge each concept. Students benefit from witnessing a model of clear thinking in action, and these tutorials strive to be just that.

Another feature of these learning modules is that they do not treat topics in isolation. Rather, important concepts are introduced early in the series, and appear repeatedly as stepping-stones toward other concepts in subsequent modules. This helps to avoid the "compartmentalization" of knowledge, demonstrating the inter-connectedness of concepts and simultaneously reinforcing them. Each module is fairly complete in itself, reserving the beginning of its tutorial to a review of foundational concepts.

To high standards of education,

Tony R. Kuphaldt

Appendix C

Tools used

I am indebted to the developers of many open-source software applications in the creation of these learning modules. The following is a list of these applications with some commentary on each.

You will notice a theme common to many of these applications: a bias toward *code*. Although I am by no means an expert programmer in any computer language, I understand and appreciate the flexibility offered by code-based applications where the user (you) enters commands into a plain ASCII text file, which the software then reads and processes to create the final output. Code-based computer applications are by their very nature *extensible*, while WYSIWYG (What You See Is What You Get) applications are generally limited to whatever user interface the developer makes for you.

The `GNU/Linux` computer operating system

> There is so much to be said about Linus Torvalds' `Linux` and Richard Stallman's `GNU` project. First, to credit just these two individuals is to fail to do justice to the *mob* of passionate volunteers who contributed to make this amazing software a reality. I first learned of `Linux` back in 1996, and have been using this operating system on my personal computers almost exclusively since then. It is *free*, it is completely *configurable*, and it permits the continued use of highly efficient `Unix` applications and scripting languages (e.g. shell scripts, Makefiles, `sed`, `awk`) developed over many decades. `Linux` not only provided me with a powerful computing platform, but its open design served to inspire my life's work of creating open-source educational resources.

Bram Moolenaar's `Vim` text editor

> Writing code for any code-based computer application requires a *text editor*, which may be thought of as a word processor strictly limited to outputting plain-ASCII text files. Many good text editors exist, and one's choice of text editor seems to be a deeply personal matter within the programming world. I prefer `Vim` because it operates very similarly to `vi` which is ubiquitous on `Unix`/`Linux` operating systems, and because it may be entirely operated via keyboard (i.e. no mouse required) which makes it fast to use.

Donald Knuth's TEX typesetting system

Developed in the late 1970's and early 1980's by computer scientist extraordinaire Donald Knuth to typeset his multi-volume magnum opus *The Art of Computer Programming*, this software allows the production of formatted text for screen-viewing or paper printing, all by writing plain-text code to describe how the formatted text is supposed to appear. TEX is not just a markup language for documents, but it is also a Turing-complete programming language in and of itself, allowing useful algorithms to be created to control the production of documents. Simply put, TEX *is a programmer's approach to word processing*. Since TEX is controlled by code written in a plain-text file, this means anyone may read that plain-text file to see exactly how the document was created. This openness afforded by the code-based nature of TEX makes it relatively easy to learn how other people have created their own TEX documents. By contrast, examining a beautiful document created in a conventional WYSIWYG word processor such as Microsoft `Word` suggests nothing to the reader about *how* that document was created, or what the user might do to create something similar. As Mr. Knuth himself once quipped, conventional word processing applications should be called WYSIAYG (What You See Is *All* You Get).

Leslie Lamport's LATEX extensions to TEX

Like all true programming languages, TEX is inherently extensible. So, years after the release of TEX to the public, Leslie Lamport decided to create a massive extension allowing easier compilation of book-length documents. The result was LATEX, which is the markup language used to create all ModEL module documents. You could say that TEX is to LATEX as `C` is to `C++`. This means it is permissible to use any and all TEX commands within LATEX source code, and it all still works. Some of the features offered by LATEX that would be challenging to implement in TEX include automatic index and table-of-content creation.

Tim Edwards' `Xcircuit` drafting program

This wonderful program is what I use to create all the schematic diagrams and illustrations (but not photographic images or mathematical plots) throughout the ModEL project. It natively outputs PostScript format which is a true vector graphic format (this is why the images do not pixellate when you zoom in for a closer view), and it is so simple to use that I have never had to read the manual! Object libraries are easy to create for `Xcircuit`, being plain-text files using PostScript programming conventions. Over the years I have collected a large set of object libraries useful for drawing electrical and electronic schematics, pictorial diagrams, and other technical illustrations.

`Gimp` graphic image manipulation program

Essentially an open-source clone of Adobe's `PhotoShop`, I use `Gimp` to resize, crop, and convert file formats for all of the photographic images appearing in the ModEL modules. Although `Gimp` does offer its own scripting language (called `Script-Fu`), I have never had occasion to use it. Thus, my utilization of `Gimp` to merely crop, resize, and convert graphic images is akin to using a sword to slice bread.

`SPICE` circuit simulation program

`SPICE` is to circuit analysis as TEX is to document creation: it is a form of markup language designed to describe a certain object to be processed in plain-ASCII text. When the plain-text "source file" is compiled by the software, it outputs the final result. More modern circuit analysis tools certainly exist, but I prefer `SPICE` for the following reasons: it is *free*, it is *fast*, it is *reliable*, and it is a fantastic tool for *teaching* students of electricity and electronics how to write simple code. I happen to use rather old versions of `SPICE`, version 2g6 being my "go to" application when I only require text-based output. `NGSPICE` (version 26), which is based on Berkeley `SPICE` version 3f5, is used when I require graphical output for such things as time-domain waveforms and Bode plots. In all `SPICE` example netlists I strive to use coding conventions compatible with all `SPICE` versions.

Andrew D. Hwang's `ePiX` mathematical visualization programming library

This amazing project is a `C++` library you may link to any `C/C++` code for the purpose of generating PostScript graphic images of mathematical functions. As a completely free and open-source project, it does all the plotting I would otherwise use a Computer Algebra System (CAS) such as `Mathematica` or `Maple` to do. It should be said that `ePiX` is *not* a Computer Algebra System like `Mathematica` or `Maple`, but merely a mathematical *visualization* tool. In other words, it won't determine integrals for you (you'll have to implement that in your own `C/C++` code!), but it can graph the results, and it does so beautifully. What I really admire about `ePiX` is that it is a `C++` programming library, which means it builds on the existing power and toolset available with that programming language. Mr. Hwang could have probably developed his own stand-alone application for mathematical plotting, but by creating a `C++` library to do the same thing he accomplished something much greater.

`gnuplot` mathematical visualization software

Another open-source tool for mathematical visualization is `gnuplot`. Interestingly, this tool is *not* part of Richard Stallman's GNU project, its name being a coincidence. For this reason the authors prefer "gnu" *not* be capitalized at all to avoid confusion. This is a much "lighter-weight" alternative to a spreadsheet for plotting tabular data, and the fact that it easily outputs directly to an X11 console or a file in a number of different graphical formats (including PostScript) is very helpful. I typically set my `gnuplot` output format to default (X11 on my Linux PC) for quick viewing while I'm developing a visualization, then switch to PostScript file export once the visual is ready to include in the document(s) I'm writing. As with my use of `Gimp` to do rudimentary image editing, my use of `gnuplot` only scratches the surface of its capabilities, but the important points are that it's *free* and that it *works well*.

`Python` programming language

Both Python and C++ find extensive use in these modules as instructional aids and exercises, but I'm listing Python here as a *tool* for myself because I use it almost daily as a *calculator*. If you open a Python interpreter console and type `from math import *` you can type mathematical expressions and have it return results just as you would on a hand calculator. Complex-number (i.e. *phasor*) arithmetic is similarly supported if you include the complex-math library (`from cmath import *`). Examples of this are shown in the Programming References chapter (if included) in each module. Of course, being a fully-featured programming language, Python also supports conditionals, loops, and other structures useful for calculation of quantities. Also, running in a console environment where all entries and returned values show as text in a chronologically-ordered list makes it easy to copy-and-paste those calculations to document exactly how they were performed.

Appendix D

Creative Commons License

Creative Commons Attribution 4.0 International Public License

By exercising the Licensed Rights (defined below), You accept and agree to be bound by the terms and conditions of this Creative Commons Attribution 4.0 International Public License ("Public License"). To the extent this Public License may be interpreted as a contract, You are granted the Licensed Rights in consideration of Your acceptance of these terms and conditions, and the Licensor grants You such rights in consideration of benefits the Licensor receives from making the Licensed Material available under these terms and conditions.

Section 1 – Definitions.

a. **Adapted Material** means material subject to Copyright and Similar Rights that is derived from or based upon the Licensed Material and in which the Licensed Material is translated, altered, arranged, transformed, or otherwise modified in a manner requiring permission under the Copyright and Similar Rights held by the Licensor. For purposes of this Public License, where the Licensed Material is a musical work, performance, or sound recording, Adapted Material is always produced where the Licensed Material is synched in timed relation with a moving image.

b. **Adapter's License** means the license You apply to Your Copyright and Similar Rights in Your contributions to Adapted Material in accordance with the terms and conditions of this Public License.

c. **Copyright and Similar Rights** means copyright and/or similar rights closely related to copyright including, without limitation, performance, broadcast, sound recording, and Sui Generis Database Rights, without regard to how the rights are labeled or categorized. For purposes of this Public License, the rights specified in Section 2(b)(1)-(2) are not Copyright and Similar Rights.

d. **Effective Technological Measures** means those measures that, in the absence of proper authority, may not be circumvented under laws fulfilling obligations under Article 11 of the WIPO Copyright Treaty adopted on December 20, 1996, and/or similar international agreements.

e. **Exceptions and Limitations** means fair use, fair dealing, and/or any other exception or

limitation to Copyright and Similar Rights that applies to Your use of the Licensed Material.

f. **Licensed Material** means the artistic or literary work, database, or other material to which the Licensor applied this Public License.

g. **Licensed Rights** means the rights granted to You subject to the terms and conditions of this Public License, which are limited to all Copyright and Similar Rights that apply to Your use of the Licensed Material and that the Licensor has authority to license.

h. **Licensor** means the individual(s) or entity(ies) granting rights under this Public License.

i. **Share** means to provide material to the public by any means or process that requires permission under the Licensed Rights, such as reproduction, public display, public performance, distribution, dissemination, communication, or importation, and to make material available to the public including in ways that members of the public may access the material from a place and at a time individually chosen by them.

j. **Sui Generis Database Rights** means rights other than copyright resulting from Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases, as amended and/or succeeded, as well as other essentially equivalent rights anywhere in the world.

k. **You** means the individual or entity exercising the Licensed Rights under this Public License. **Your** has a corresponding meaning.

Section 2 – Scope.

a. License grant.

1. Subject to the terms and conditions of this Public License, the Licensor hereby grants You a worldwide, royalty-free, non-sublicensable, non-exclusive, irrevocable license to exercise the Licensed Rights in the Licensed Material to:

A. reproduce and Share the Licensed Material, in whole or in part; and

B. produce, reproduce, and Share Adapted Material.

2. Exceptions and Limitations. For the avoidance of doubt, where Exceptions and Limitations apply to Your use, this Public License does not apply, and You do not need to comply with its terms and conditions.

3. Term. The term of this Public License is specified in Section 6(a).

4. Media and formats; technical modifications allowed. The Licensor authorizes You to exercise the Licensed Rights in all media and formats whether now known or hereafter created, and to make technical modifications necessary to do so. The Licensor waives and/or agrees not to assert any right or authority to forbid You from making technical modifications necessary to exercise the Licensed Rights, including technical modifications necessary to circumvent Effective Technological Measures.

For purposes of this Public License, simply making modifications authorized by this Section 2(a)(4) never produces Adapted Material.

5. Downstream recipients.

A. Offer from the Licensor – Licensed Material. Every recipient of the Licensed Material automatically receives an offer from the Licensor to exercise the Licensed Rights under the terms and conditions of this Public License.

B. No downstream restrictions. You may not offer or impose any additional or different terms or conditions on, or apply any Effective Technological Measures to, the Licensed Material if doing so restricts exercise of the Licensed Rights by any recipient of the Licensed Material.

6. No endorsement. Nothing in this Public License constitutes or may be construed as permission to assert or imply that You are, or that Your use of the Licensed Material is, connected with, or sponsored, endorsed, or granted official status by, the Licensor or others designated to receive attribution as provided in Section 3(a)(1)(A)(i).

b. Other rights.

1. Moral rights, such as the right of integrity, are not licensed under this Public License, nor are publicity, privacy, and/or other similar personality rights; however, to the extent possible, the Licensor waives and/or agrees not to assert any such rights held by the Licensor to the limited extent necessary to allow You to exercise the Licensed Rights, but not otherwise.

2. Patent and trademark rights are not licensed under this Public License.

3. To the extent possible, the Licensor waives any right to collect royalties from You for the exercise of the Licensed Rights, whether directly or through a collecting society under any voluntary or waivable statutory or compulsory licensing scheme. In all other cases the Licensor expressly reserves any right to collect such royalties.

Section 3 – License Conditions.

Your exercise of the Licensed Rights is expressly made subject to the following conditions.

a. Attribution.

1. If You Share the Licensed Material (including in modified form), You must:

A. retain the following if it is supplied by the Licensor with the Licensed Material:

i. identification of the creator(s) of the Licensed Material and any others designated to receive attribution, in any reasonable manner requested by the Licensor (including by pseudonym if designated);

ii. a copyright notice;

iii. a notice that refers to this Public License;

iv. a notice that refers to the disclaimer of warranties;

v. a URI or hyperlink to the Licensed Material to the extent reasonably practicable;

B. indicate if You modified the Licensed Material and retain an indication of any previous modifications; and

C. indicate the Licensed Material is licensed under this Public License, and include the text of, or the URI or hyperlink to, this Public License.

2. You may satisfy the conditions in Section 3(a)(1) in any reasonable manner based on the medium, means, and context in which You Share the Licensed Material. For example, it may be reasonable to satisfy the conditions by providing a URI or hyperlink to a resource that includes the required information.

3. If requested by the Licensor, You must remove any of the information required by Section 3(a)(1)(A) to the extent reasonably practicable.

4. If You Share Adapted Material You produce, the Adapter's License You apply must not prevent recipients of the Adapted Material from complying with this Public License.

Section 4 – Sui Generis Database Rights.

Where the Licensed Rights include Sui Generis Database Rights that apply to Your use of the Licensed Material:

a. for the avoidance of doubt, Section 2(a)(1) grants You the right to extract, reuse, reproduce, and Share all or a substantial portion of the contents of the database;

b. if You include all or a substantial portion of the database contents in a database in which You have Sui Generis Database Rights, then the database in which You have Sui Generis Database Rights (but not its individual contents) is Adapted Material; and

c. You must comply with the conditions in Section 3(a) if You Share all or a substantial portion of the contents of the database.

For the avoidance of doubt, this Section 4 supplements and does not replace Your obligations under this Public License where the Licensed Rights include other Copyright and Similar Rights.

Section 5 – Disclaimer of Warranties and Limitation of Liability.

a. Unless otherwise separately undertaken by the Licensor, to the extent possible, the Licensor offers the Licensed Material as-is and as-available, and makes no representations or warranties of any kind concerning the Licensed Material, whether express, implied, statutory, or other. This includes, without limitation, warranties of title, merchantability, fitness for a particular purpose, non-infringement, absence of latent or other defects, accuracy, or the presence or absence of errors,

whether or not known or discoverable. Where disclaimers of warranties are not allowed in full or in part, this disclaimer may not apply to You.

b. To the extent possible, in no event will the Licensor be liable to You on any legal theory (including, without limitation, negligence) or otherwise for any direct, special, indirect, incidental, consequential, punitive, exemplary, or other losses, costs, expenses, or damages arising out of this Public License or use of the Licensed Material, even if the Licensor has been advised of the possibility of such losses, costs, expenses, or damages. Where a limitation of liability is not allowed in full or in part, this limitation may not apply to You.

c. The disclaimer of warranties and limitation of liability provided above shall be interpreted in a manner that, to the extent possible, most closely approximates an absolute disclaimer and waiver of all liability.

Section 6 – Term and Termination.

a. This Public License applies for the term of the Copyright and Similar Rights licensed here. However, if You fail to comply with this Public License, then Your rights under this Public License terminate automatically.

b. Where Your right to use the Licensed Material has terminated under Section 6(a), it reinstates:

1. automatically as of the date the violation is cured, provided it is cured within 30 days of Your discovery of the violation; or

2. upon express reinstatement by the Licensor.

For the avoidance of doubt, this Section 6(b) does not affect any right the Licensor may have to seek remedies for Your violations of this Public License.

c. For the avoidance of doubt, the Licensor may also offer the Licensed Material under separate terms or conditions or stop distributing the Licensed Material at any time; however, doing so will not terminate this Public License.

d. Sections 1, 5, 6, 7, and 8 survive termination of this Public License.

Section 7 – Other Terms and Conditions.

a. The Licensor shall not be bound by any additional or different terms or conditions communicated by You unless expressly agreed.

b. Any arrangements, understandings, or agreements regarding the Licensed Material not stated herein are separate from and independent of the terms and conditions of this Public License.

Section 8 – Interpretation.

a. For the avoidance of doubt, this Public License does not, and shall not be interpreted to, reduce, limit, restrict, or impose conditions on any use of the Licensed Material that could lawfully

be made without permission under this Public License.

b. To the extent possible, if any provision of this Public License is deemed unenforceable, it shall be automatically reformed to the minimum extent necessary to make it enforceable. If the provision cannot be reformed, it shall be severed from this Public License without affecting the enforceability of the remaining terms and conditions.

c. No term or condition of this Public License will be waived and no failure to comply consented to unless expressly agreed to by the Licensor.

d. Nothing in this Public License constitutes or may be interpreted as a limitation upon, or waiver of, any privileges and immunities that apply to the Licensor or You, including from the legal processes of any jurisdiction or authority.

Creative Commons is not a party to its public licenses. Notwithstanding, Creative Commons may elect to apply one of its public licenses to material it publishes and in those instances will be considered the "Licensor." Except for the limited purpose of indicating that material is shared under a Creative Commons public license or as otherwise permitted by the Creative Commons policies published at `creativecommons.org/policies`, Creative Commons does not authorize the use of the trademark "Creative Commons" or any other trademark or logo of Creative Commons without its prior written consent including, without limitation, in connection with any unauthorized modifications to any of its public licenses or any other arrangements, understandings, or agreements concerning use of licensed material. For the avoidance of doubt, this paragraph does not form part of the public licenses.

Creative Commons may be contacted at `creativecommons.org`.

Appendix E

References

"21 Steps to Improve Cyber Security of SCADA Networks", Department of Energy, USA, May 2011.

"Announcing the Advanced Encryption Standard (AES)", Federal Information Processing Standards Publication 197, 26 November 2001.

Bartman, Tom and Carson, Kevin, "Securing Communications for SCADA and Critical Industrial Systems", Technical Paper 6678-01, Schweitzer Engineering Laboratories, Inc., Pullman, WA, January 22, 2015.

Beresford, Dillon, "Siemens Simatic S7 PLC Exploitation", technical presentation at Black Hat USA conference, 2011.

Byres, Eric, "Building Intrinsically Secure Control and Safety Systems – Using ANSI/ISA-99 Security Standards for Improved Security and Reliability", Byres Security Inc., May 2009.

Byres, Eric, "Understanding Deep Packet Inspection (DPI) for SCADA Security", document WP_INDS_TOF_514_A_AG, Belden, Inc., 2014.

Ciampa, Mark, *Security+ Guide to Network Security Fundamentals*, Course Technology (a division of Thompson Learning), Boston, MA, 2005.

"Common Cybersecurity Vulnerabilities in Industrial Control Systems", Department of Homeland Security, Control Systems Security Program, National Cyber Security Division, USA, May 2011.

Dang, Bruce, "Adventures in Analyzing Stuxnet", presentation on Day 1 of the 27th Chaos Communication Congress (27C3), Berlin, Germany, 27 December 2010.

Falliere, Nicolas; Murchu, Liam O.; Chien, Eric; "W32.Stuxnet Dossier", version 1.4, Symantec Corporation, Mountain View, CA, February 11, 2011.

Fischer, Ted, "Private and Public Key Cryptography and Ransomware", Center for Internet Security, Inc., Pullman, WA, December 2014.

Grennan, Mark, "Firewall and Proxy Server HOWTO", version 0.8, February 26, 2000.

Horak, Ray, *Webster's New World Telecom Dictionary*, Wiley Publishing, Inc., Indianapolis, IN, 2008.

Kemp, R. Scott, "Gas Centrifuge Theory and Development: A Review of US Programs", Program on Science and Global Security, Princeton University, Princeton, NJ, Taylor & Francis Group, LLC, 2009.

Langner, Ralph, "To Kill A Centrifuge – A Technical Analysis of What Stuxnet's Creators Tried to Achieve", The Langner Group, Arlington, MA, November 2013.

Lee, Jin-Shyan; Su, Yu-Wei; Shen, Chung-Chou, "A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi", Industrial Technology Research Institute, Hsinchu, Taiwan, November 2007.

Leidigh, Christopher, "Fundamental Principles of Network Security", White Paper #101, American Power Conversion (APC), 2005.

Leischner, Garrett and Whitehead, David, "A View Through the Hacker's Looking Glass", Technical Paper 6237-01, Schweitzer Engineering Laboratories, Inc., Pullman, WA, April 2006.

Makhijani, Arjun Ph.D.; Chalmers, Lois; Smith, Brice Ph.D.; "Uranium Enrichment – Just Plain Facts to Fuel an Informed Debate on Nuclear Proliferation and Nuclear Power", Institute for Energy and Environmental Research, October 15, 2004.

Markey, Hedy Keisler, and Antheil, George, *US Patent 2,292,387*, "Secret Communication System", application 10 June 1941, patent granted 11 August 1942.

McDonald, Geoff; Murchu, Liam O.; Doherty, Stephen; Chien, Eric; "Stuxnet 0.5: The Missing Link", version 1.0, Symantec Corporation, Mountain View, CA, February 26, 2013.

Oman, Paul W.; Risley, Allen D.; Roberts, Jeff; Schweitzer, Edmund O. III, "Attack and Defend Tools for Remotely Accessible Control and Protection Equipment in Electric Power Systems", Schweitzer Engineering Laboratories, Inc., Pullman, WA, March 12, 2002.

Postel, John, *Internet Protocol – DARPA Internet Program Protocol Specification*, RFC 791, Information Sciences Institute, University of Southern California, Marina Del Ray, CA, September 1981.

Rescorla, E. and Korver, B.; "Guidelines for Writing RFC Text on Security Considerations" (RFC 3552), The Internet Society, July 2003.

Risley, Allen; Marlow, Chad; Oman, Paul; Dolezilek, Dave, "Securing SEL Ethernet Products With VPN Technology", Application Guide 2002-05, Schweitzer Engineering Laboratories, Inc., Pullman, WA, July 11, 2002.

"Seven Strategies to Effectively Defend Industrial Control Systems", National Cybersecurity and

Communications Integration Center (NCCIC), Department of Homeland Security (DHS), USA.

"Tofino Xenon Security Appliance" data sheet, document DS-TSA-XENON version 6.0, Tofino Security, 2014.

"W32.DuQu – The Precursor to the next Stuxnet", version 1.4, Symantec Corporation, Mountain View, CA, November 23, 2011.

Whitehead, David and Smith, Rhett, "Cryptography: A Tutorial for Power Engineers", Technical Paper 6345-01, Schweitzer Engineering Laboratories, Inc., Pullman, WA, October 20, 2008.

Zippe, Gernot, "A Progress Report: Development of Short Bowl Centrifuges", Department of Physics, University of Virginia, July 1, 1959.

Appendix F

Version history

This is a list showing all significant additions, corrections, and other edits made to this learning module. Each entry is referenced by calendar date in reverse chronological order (newest version first), which appears on the front cover of every learning module for easy reference. Any contributors to this open-source document are listed here as well.

29 November 2022 – placed questions at the top of the itemized list in the Introduction chapter prompting students to devise experiments related to the tutorial content.

29-30 September 2021 – added a new "Anti-virus" section to the Tutorial, and also a new Historical References section on early spread-spectrum radio technology which is shared amongst other modules. Also updated reference to the Mitre cyber-vulnerability database.

10 May 2021 – commented out or deleted empty chapters.

23-24 February 2021 – added some content to the Introduction chapter, and made minor edits to the Tutorial and Historical Reference chapters.

28 December 2020 – corrected a minor typographical error in the Introduction chapter.

30 September 2020 – minor edits to the Historical References chapter.

29 September 2020 – added more content to the Introduction chapter.

28 September 2020 – deleted redundant (and empty) section in Tutorial, thanks to Tyshe Eisele for spotting this problem.

22 September 2020 – significantly edited the Introduction chapter to make it more suitable as a pre-study guide and to provide cues useful to instructors leading "inverted" teaching sessions.

1 July 2020 – added some lexicon entries and one Reference.

28-29 June 2020 – added Tutorial content as well as some questions.

27 June 2020 – document first created.

Index

Accelerometer, 31
Access Control List, 14
ACL, firewall, 14
Adding quantities to a qualitative problem, 86
Administrator privilege, computer, 29
Air gap, network, 12
Annotating diagrams, 85
Anti-virus blacklisting, 20
Anti-virus whitelisting, 20
Anti-virus, computer, 20, 29
Application blacklisting, computer, 29
Application whitelisting, computer, 29

basisk Siemens PLC exploit, 30
Beresford, Dillon, 30
Blacklist, 16
Blacklisting, anti-virus, 20
Blacklisting, application, 29
Bluetooth, 22
Brute force attack, password, 27

CANDU nuclear reactor, 40
Centrifuge, 41
Chain reaction, nuclear, 40
Checking for exceptions, 86
Checking your work, 86
Client, 19
Code, computer, 95
Critical speed, rotating machine, 46

Data diode, 23
Data Historian, 19
DCS, 10
Decryption, 21, 55
Deep Packet Inspection, 17
Defense-in-depth, 9, 24, 25
Demilitarized Zone, 18
Denial of service attack, 11
Denial-of-service attack, 55
Deny by default, 16
Department of Energy, 44
Dictionary attack, password, 27
Dimensional analysis, 85
Distributed control system, 10
Distributed denial-of-service attack, 55
DMZ, 18
DOE, 44
DPI firewall, 17

Echo Request, ICMP, 17, 57
Edwards, Tim, 96
EIA/TIA-232 serial communication, 23
EIA/TIA-422 serial communication, 23
EIA/TIA-485 serial communication, 23
Encryption, 21, 55
Enriched uranium, 40
Ethernet, 23, 30

Firewall, computer, 14
Fission, nuclear, 40

Gas centrifuge, 41
Gas centrifuge cascade, 43
Gas centrifuge stage, 43
Graph values to solve a problem, 86
Greenleaf, Cynthia, 61

HEU, 40
How to teach with these modules, 93
HTTP, 16
Hwang, Andrew D., 97

ICANN, 16
ICMP, 17, 57
Identify given data, 85

Identify relevant principles, 85
Information Technology (IT), 3, 5
Intermediate results, 85
Internet Control Message Protocol, 17, 57
Inverted instruction, 93
IPsec, 21
iptables, 16
Isotope, 40

Key rotation, 21
Key, cryptographic, 21
Knuth, Donald, 96

Lamport, Leslie, 96
LAN, computer, 14
LEU, 40
Limiting cases, 86

Malware, 6, 20, 29
Man-in-the-middle, 19
Man-in-the-middle attack, 49, 56
Maxwell, James Clerk, 35
Metacognition, 66
Moolenaar, Bram, 95
Murphy, Lynn, 61

NSS Labs, 30

Open-source, 95
OSI Reference Model, 16, 17

Password lockout, 11
Password timeout, 11
Passwords, 27, 30, 33, 57
PLC, 10, 39
Problem-solving: annotate diagrams, 85
Problem-solving: check for exceptions, 86
Problem-solving: checking work, 86
Problem-solving: dimensional analysis, 85
Problem-solving: graph values, 86
Problem-solving: identify given data, 85
Problem-solving: identify relevant principles, 85
Problem-solving: interpret intermediate results, 85
Problem-solving: limiting cases, 86
Problem-solving: qualitative to quantitative, 86
Problem-solving: quantitative to qualitative, 86
Problem-solving: reductio ad absurdum, 86
Problem-solving: simplify the system, 85
Problem-solving: thought experiment, 85
Problem-solving: track units of measurement, 85
Problem-solving: visually represent the system, 85
Problem-solving: work in reverse, 86
Profibus, 48
Programmable logic controller, 10, 39

Qualitatively approaching a quantitative problem, 86

Ransomware, 6
Reading Apprenticeship, 61
Reductio ad absurdum, 86, 92, 93
Resonance, mechanical, 46
Root privilege, computer, 29
RS-232 serial communication, 23
RS-422 serial communication, 23
RS-485 serial communication, 23
RTU, 10

SCADA, 10, 21, 80
Schoenbach, Ruth, 61
Scientific method, 66
Server, 19
Siemens S7-300 PLC, 30
Simplifying a system, 85
Socrates, 92
Socratic dialogue, 93
SPICE, 61
Spread-spectrum radio, 22
SSH, 16, 30
Stallman, Richard, 95
Stuxnet, 39
Stuxnet computer virus, 39
Sublimation, phase change, 46

TCP, 16
TCP SYN flood attack, 17
TELNET, 30
Thought experiment, 85
Tofino industrial network firewall, 17
Torvalds, Linus, 95

UDP, 17

Units of measurement, 85

Virtual Private Network, 21
Virus, digital, 20
Visualizing a system, 85
VPN, 21

Whitelist, 16
Whitelisting, anti-virus, 20
Whitelisting, application, 29
WirelessHART, 22
Work in reverse to solve a problem, 86
WYSIWYG, 95, 96