Computer Security (CM2025)

Course Notes

Felipe Balbi

January 12, 2021

Contents

Week 1

Key Concepts

- Understand the central goals and aspects of computer security.
- Understand and explain the differences between a range of malware types.
- Identify key examples of malware and their historical significance.

Reading: Ethics in information security

Read the following article:
Ethics in information security, IEEE Security & Privacy, vol.15 May/June 2017, pp.3–4.
You should also familiarise yourselves with the ACM ethics guidelines.

Defining computer security

With the advent of the Internet, computer security became a very important field of research.

Computer Security is the *protection of computer-related assets against danger, loss or loss of control of something valuable.*

Security has three main goals:

Prevention The safeguarding of assets from threats;

Detection Systems that alert if malicious activity is, or is about to, take place;

Reactiion Definition of procedures that enables us to deal with an attack.

Security can also be split into three main components:

Policy deals with confidentiality, integrity and availability of data;

Threat Model assumptions about those involved with malicious activity;

Mechanism The SW/HW used to make sure the policies are enforced using the assumptions made during Threat Modelling.

There are five important terms that need to be defined:

Attack activities harmful to computer systems, data, software, and hardware;

Risk the possibility of damage or loss of digital assets in case of an attack;

Zero-day (vulnerability) a vulnerability used by an attacker before its discovery by the developer of the SW;

Exploit SW used to take advantage of a bug or vulnerability;

Hacker Subdivided into three groups

White Hats Find vulnerabilities with a goal of fixing them before its discovery by an attacker. They work under the permission of the owner of the computer system being attacked;

Black Hats Try to penetrate the system to gain unauthorized access. Often, their motivation is harm operations or steal sensitive operations;

Gray Hats They fall somewhere in-between White Hats and Black Hats and work with varying combinations of good and bad intentions.

Types of malicious software 1

Malware is a piece of softwarte designed to disrupt, damange and destroy an information system. There are many types of malwares, some of which are discussed in the following subsections

Viruses

They self-replicate by inserting themselves into other files, programs, documents, etc. Can spread through emails, USB sticks and downloads from unknown sources.

The *Creeper* is considered to be one of the first computer viruses. Developed in 1971, it infected computers connected to the ARPANET, the internet prototype.

Another early virus was the *Elk Cloner*. Written by a high-school student to infect Apple II computers using floppy disks in 1982. Every 50th time the computer booted, it would display a poem written by the hacker.

Not all viruses are harmless. The *I Love You* virus in 2000 caused around $10 billion worth of damages by affecting nearly 10% of all computers around the globe.

Worms

They can replicate without attaching themselves to existing software. The *Stuxnet* is a well-known worm, considered to be one of the most destructive worms ever created. It was designed to attack Programmable Logic Controllers by Siemes.

PLC devices are used for the automation of processes in machinery. In this case, it targetted centrifuges in Iranian nuclear power plants and altered the speed of the machine, causing it to tear itself apart.

It's estimated that *Stuxnet* destroyed 20% of Iranian's nuclear power plant centrifuges.

Adware

These display advertisements on your screen during browsing or online shopping. Possibly the most visible form of malware one can encounter. Its main purpose is to collect user data.

Trojans

Trojans are named after the famous ancient Greek tale of the invasion of the city of Troy by the Greek during the Trojan War.

After trying and failing several times to get access to the city, the Greeks came up witha plan where squatter soldiers would hide in the joint wooden status of a horse presented as a gift.

During the night, after entiring the city within the horse, the Greeks broke out of the horse to attack the city.

Trojans have a similar way of working: they hide themselves inside an application or program data and spread based on specific user action.

Spyware

Designed to spy on the target machine or the user, it collects information and sends it back to the hacker for further use or for sale on the dark web.

The *Dark Hotel* spyware is one famous case which used Hotel Wi-Fi to target the personal systems of government officials, business tycoons and political leaders to extract sensitive information.

Types of malicious software 2

We have a look at *Keyloggers*, *Ramsonware*, *Botnets*, and *Rootkits*.

A *Keylogger* records every keystroke from the user. This may include messages being typed, emails, confidential information such banking credentials, users, passwords, etc.

Olympic Vision is a keylogger used for Business Email Compromise (BEC) attacks. It also uses several other pieces of malicious software to steal sensitive information and spy on business transactions. Nowadays it's very easy to get a hold of a keylogger.

In a *Ramsonware* attack, the victim's data is encrypted, backup files are deleted and the people responsible demand money in exchange for decryption of this data. In other words, the victims are held to *ramsom* for renewed access to their data. The *WannaCry* attack is a recent example that took place in May 2017.

In a *Botnet* attack, computers connected to the internet are taken over by an attacker which remotely controls the computers using a Command And Control (CNC) server to carry out Distributed Denial Of Service (DDOS) attacks. A DDOS attack is when a given server is flooded with so much traffic at one moment, that it collapses.

EchoBot is a botnet used to exploit over 59 known vulnerabilities and launch a number of attacks, such as DDOS attack, steal sensitive information, conduct corporate espi-

onage, and infects a wide range of Internet Of Things connected devices. Furthermore, it also scans for old vulnerabilities in legacy systems for future exploitation.

Finally, we have *Rootkits*. These remain hidden in a target computer and activate in secret. They can perform several activities rangin from giving attackers remote access to a computer all the way to stealing sensitive information such as a password or credit card details. They can also use a compromised computer to launch any of the other attacks described before.

Week 2

Key Concepts

- Explain the key differences between static and dynamic analysis.
- Explain the usage of sandboxes in malware analysis.
- Understand the need for a variety of methods of malware analysis.

Malware analysis and techniques

Static Malware Analysis is one of the techniques used to analyze and combat the types of malware discussed previously.

More generally, Malware Analysis is a set of processes and techniques that help a Security Analyst understand the functionality, origin, impact, and intent of malicious software.

The goal of this activity is find the *Indication Of Compromise* (IOC) that depicts the behavior of malicious software. IOCs are also used to develop signatures of malware. The two techniques used for such analysis are Static Malware Analysis and Dynamic Malware Analysis.

In Static Malware Analysis, the executable files are examined without being executed. We can determine if a file is clean or malicious and also discover information about its functionality.

With the information collected during Static Malware Analysis, allows us to determine signatures, which are a collection of distinguishing features that can be used to recognize malware.

Two essential techniques used in static analysis are antivirus scanning and hashing. Antivirus scanning is the traditional method of running a file through an antivirus scanner to try to detect whether the file contains malware. Hashing is an algorithm that produces a value referred to as a *Hash* which is a unique fingerprint for a given file. Any modification to a file will result in a different hash fingerprint, including infection by a malware.

While static analysis methods are useful as a starting point for some more basic types of attack, it can be powerless against more recent and advanced types of malware. These have found ways to circumvent the detection methods used during static analysis.

The solution to this is Dynamic Malware Analysis.

Malware analysis 2

Dynamic Analysis or Behavioral Analysis is where we execute the malware in a controlled environment known as a sandbox. This allows a Security Professional to observe the behavior of malicious software, help to understand its functionality and, hopefully, find the Indicator Of Compromise.

This method overcomes some of the limitations of Static Analysis with regards to catching the more advanced and adaptive forms of malware.

Dynamic Analysis is an efficient method for analyzing malware because it helps uncover the functionality of the malware, which is not entirely possible with static analysis.

A Sandbox is a virtualized environment that contains a virtual network, services, drives, etc, to ensure that the malware behaves exactly as it would in a real environment.

There are two main types of sandboxes: Agent-based and Agent-less. Agent-based sandboxes require software to be installed on every computer that needs to be monitored. Well known examples are cuckoo, threat expert, bit blase, and Comodo. Conversely, an Agent-less sandbox monitors computers on the network from afar without needing to be installed on every device. Popular examples are VMRay, Analyzer, and SNDBOX.

Security Researchers use both types of sandboxes, but some research suggests that agent-less sandboxes are more efficient.

Another common tool for dynamic analysis on Windows machines is process monitor (Procmon). It's used to monitor the registry, file system, network, running processes, etc.

Ethics

Ethics is really important in the field of Computer Security.

Because we frequently work with computers, we may be exposed to security issues and vulnerabilities. When that occurs, we may be able to fix the issue ourselves by setting a rules in our *iptables* or blocking traffic from a certain port for instance.

There may be, however, wider ramifications to the problem we have discovered. It could be something worth mentioning to the maker of the faulty software.

That's where **Responsible Disclosure** comes into the picture. The term itself is somewhat subjective, but there may be legal ramifications related to the disclosure of a security issue.

In a situation where we find an issue in a popular software package, e.g. a popular Operating System, it may be the case that many other users are affected. This means that we have some responsibility to the Company or Service Provider in terms of disclosing the issue.

It is common practice to identify the bug to the provider and offer them enough time to fix the issue before disclosing the problem publicly. There are considerations about transparency.

Some companies may refuse to patch issues or even seek legal action against those exploiting or even simply highlighting the issue.

We should also think about security in a distributed way. For example, cloud computing services offer on-demand computing. These services may not be hosted in the locality of the developer or the client. As such, we must consider conforming to the law in **all** of these locations.

As a final thought, actions have consequences. Identity theft and distributed denial of service attacks have real world consequences. People's lives can be destroyed if we do something that can cause harm.

Passwords

Designing truly secure systems is very hard, only made harder by data leaks happening periodically as can be seen in the news. Data is becoming more and more valuable and there are places in the *Dark Web* where one can buy leaked data.

One of the most prolific leaks utilised a simple encryption method where the same encryption key was utilized. This resulted in a one-to-one mapping between plain-text and encrypted passwords. In other words, there was a situation where different users with the same password would end up with the same encrypted string stored in the database.

What this means is that if one password is cracked, all other users who happen to be using the same password were also compromised. To make matters worse, some users used password hints, which were also stored in the database.

Reusing passwords is also a common problem. If a user's password for one service is leaked, there is a probability that the same user employed the exact same password on multiple services which renders all of such services compromised.

There are ways to design more secure systems, however that also has implications. We could require longer passwords and encourage two-factor authentication, but a user may loose their phone or forget that longer, more complex password.

A good system design balances accessibility with security and usability.

Social Engineering

Insecure designs can have far reaching ramifications, however a system is only as secure as its safest link.

Social Engineering is one of the most common attack vectors and it does not rely on technical subtleties of attacks. These attacks rely on the fact that not all staff are properly trained in security and, as such, attackers may exploit gaps in their knowledge.

A company may have a robust security policy in place to handle access control, however **compliance** is a different matter.

Phishing emails are a good example of Social Engineering attacks. They try to trick you into thinking the email comes from a reliable source and convince you to give them the information they're after. Some of these emails may look fairly authentic and we may have to look deeper to determine their authenticity.

Week 3

Key Concepts

- Describe the CIA objectives of network security.
- Use real examples to describe how DoS attacks and DDoS attacks work including those using botnets.
- Describe the levels of security in wireless networks and common attack vectors.

The objectives of network security: confidentiality, integrity and accessibility

Network security has three main objectives:

Confidentiality Read access control, i.e. who can **read** which piece of information

Integrity Write access control, i.e. who can **write** which piece of information

Availability Maintaining function, i.e. guaranteeing that the information will be available to those who can access it

It's composed of a set of policies and practices to protect the network. One example of a policy may be:

Every access to the network is unauthorized unless the user is authenticated with username and password

A practice related to the policy may be the fact that any user must be given a username and password and an authentication server needs to be maintained.

Paper about CIA

Reading about the conflicting aspects of confidentiality, integrity and availability:

K. S. Wilson, Conflicts Among the Pillars of Information Assurance, in IT Professional, vol. 15, no. 4, pp. 44-49, July-Aug. 2013, doi: 10.1109/MITP.2012.24.

The attack surface and the denial of service attack

A *Denial of Service*, or *DoS*, attack is when we flood a server with so many requests that it collapses under the load, therefore **denying service** to the users.

This attack can happen in many of the 7 layers of OSI model. As a quick summary, here are what each of 7 layers represents:

1. Physical Layer

   The hardware pieces. Cables, networking cards, etc. Responsible for the transmission of unstructured raw data. Converts digital bits into electrical, radio, or optical signals.

2. Data Link Layer

   Provides node-to-node transfers. Can detect and maybe correct errors.

3. Network Layer

   Provides the infrastructure for transmission of variable-length packets from one node to another connected in different networks.

4. Transport Layer

   Provides the infrastructure for the transmission of variable-length data sequences from a source to a destination.

5. Session Layer

   Controls the dialogues between computers. Establishes, manages and terminates connections between local and remote application.

6. Presentation Layer

   Establishes context between application-layer entities.

7. Application Layer

   The layer closest to the user. User and OSI interacts directly with the application.

With these in mind, we can look at some example attacks for some of these layers.

ARP Flood Attack is a layer 2 attack where one would keep broadcasting the network with ARP requests consuming a lot of the available processing power of the target machine. ARP requests are broadcast messages used to ask the network which MAC address corresponds to an IP address.

ICMP Ping Flood Attack is a layer 3 attack relying on the *ping* diagnostic message. Hosts are required to respond to ICMP Ping requests. By sending a flood of ping requests, one can keep a server busy processing such ping requests.

TCP-SYN Flood Attack is a layer 4 attack which tries to open several TCP connections by sending a flood of TPC-SYN packets.

The anatomy of a DDOS, botnets and Mirai

A *Distributed Denial of Service* Attack, or *DDoS*, is similar to a *DoS* attack however it's accomplished with many different machines targetting a service, hence the "distributed" in the name.

A BotNet is slightly different. An attacker will take over the control of several machines from regular users on the internet, and make those machines target a single service with a flood attack of some kind.

Many *Internet of Things*, or *IoT*, devices on the market have poor security features, which makes them a target for BotNets and carry out DDoS attacks on other servers.

One such case is the Mirai BotNet, which targets certain IoT devices. The most prominent Mirai DDoS attack, took down the DNS provider Dyn resulting in Netflix, Github, Twitter, Reddit and many other major services being rendered inaccessible. After analysis, Dyn claimed that there were up to 100,000 malicious endpoints involved in the attack.

Mirai GitHub and Research Paper

- Analysis of the Mirai botnet:

  H. Sinanović and S. Mrdovic, Analysis of Mirai malicious software, 2017 25th International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Split, 2017, pp. 1-5, doi: 10.23919/SOFTCOM.2017.8115504.

- Mirai Source Code

  `https://github.com/jgamblin/Mirai-Source-Code`

- Recent Paper About the Threat of Botnets

  A. Woodiss-Field and M. N. Johnstone, Assessing the Suitability of Traditional Botnet Detection against Contemporary Threats, 2020 Workshop on Emerging Technologies for Security in IoT (ETSecIoT), Sydney, Australia, 2020, pp. 18-21, doi: 10.1109/ETSecIoT50046.2020.00008.

Wireless attacks: WiFi attack vectors

The Wi-Fi Alliance is a network of companies working on wifi technology and standards. The idea being that device manufacturers want their devices to interoperate, therefore a standard is created which describes the method of communication in the wireless network.

A timeline of the different standards are as follows:

- 802.11a: 1999
- 802.11b: 1999
- 802.11g: 2003

- 802.11n: 2009
- 802.11ac (wifi 5): 2012
- 802.11ax (wifi 6): 2020

Paired with the wireless standards, there a set of wireless security protocols:

- WEP (802.11a/b): 1997
- WPA (802.11g): 2003
- WPA2 (802.11i): 2004
- WPA3: due in 2020

WEP is notorious for having weak encryption in several aspects, which allowed attackers to exploit it and gain access to the network.

There are different attacks that can be carried out on Wireless networks. Some examples:

Dictionary Attacks short keys allow brute force encryption breaking. In other words, if the key is small, it's feasible to try all options

Fluhrer, Mantin and Shamir Attack famous encryption breaking hack for WEP

Replay Attacks replay a sequence of packets, with edits. It may allow an attacker to get a valid session key

PRGA / Packet Tampering Attack allows an attacker to masquerade as another device

Wireless Networking Security

- Classic paper reporting on WEP's vulnerabilities

  Scott R. Fluhrer, Itsik Mantin, and Adi Shamir. 2001. Weaknesses in the Key Scheduling Algorithm of RC4. In Revised Papers from the 8th Annual International Workshop on Selected Areas in Cryptography (SAC '01). Springer-Verlag, Berlin, Heidelberg, 1–24.

- Article about attack vectors for different Wifi protocols such as WEP, WPA and LEAP

  Hal Berghel and Jacob Uecker. 2005. WiFi attack vectors. Commun. ACM 48, 8 (August 2005), 21–28. DOI: `https://doi.org/10.1145/1076211.1076229`

- Very thorough review of security in different wireless technologies used in IoT devices

  K. Lounis and M. Zulkernine, Attacks and Defenses in Short-Range Wireless Technologies for IoT, in IEEE Access, vol. 8, pp. 88892-88932, 2020, DOI: 10.1109/ACCESS.2020.2993553.

Week 4

Key Concepts

- Describe three types of firewall and reason about the appropriate type of firewall to use for a given situation.
- Explain how intrusion detection systems work and give examples of historical and contemporary systems.

Firewalls – our first line of defence

Stateless Firewalls A type of Access Control Lists (or ACL). It checks all traffic against a set of rules;

stateful Firewalls More efficient than Stateless Firewalls. Once a packet session is allowed, no filters need to be applied

Proxy Firewall Carries out external network access.

Intrusion detection systems (IDS)

An Intrusion Detection System, or IDS, is a system which runs on a network and aims at detecting when a intruder has compromised the network. The reason such systems are used is because it's virtually impossible to make any network perfectly secure.

/"Most security experts agree that a completely secure system is impossible to achieve, so we must stay alert for attacks."/. (Kemmerer and Vigna, 2002.)

The IDS is about staying alert for attacks and detecting them as soon as possible.

When it comes to implementation of IDSs, Dorothy Denning state, in 1987, that "the model is based on the hypothesis that exploitation of a system's vulnerabilities involves abnormal use of the system; therefore, security violations could be detected from abnormal patterns of system usage". What this means is that an intruder is likely to exhibit abnormal usage patterns on the network.

If one can detect such abnormal usage patterns, then we can detect intruders.

While looking for intruders, we want to avoid false positives (a regular user classified as an intruder) or false negatives (an intruder classified as a regular user).

We also want our IDS to be fast and come to a conclusion using minimal resources. All this while analyzing traffic over the entire network. Modern systems using Docker containers can have a complex network structure and our IDS still needs to be fast while analyzing traffic of complex networking schemes.

Recent IDSs employ state-of-the-art Machine Learning and AI algorithms for improved pattern recognition. Deep-learning is used in many recent research papers in computer security related to IDS.

When developing state-of-the-art IDS using Deep Learning, we need a dataset to train the neural network in order to verify correctness of the system. A common dataset to use for this case is the KDD99 Dataset.

Intrusion Detection Systems

- D. E. Denning, An Intrusion-Detection Model, in IEEE Transactions on Software Engineering, vol. SE-13, no. 2, pp. 222-232, Feb. 1987, doi: 10.1109/TSE.1987.232894.
- R. A. Kemmerer and G. Vigna, Intrusion detection: a brief history and overview, in Computer, vol. 35, no. 4, pp. supl27-supl30, April 2002, doi: 10.1109/MC.2002.1012428.
- S. Wang, C. Xia and T. Wang, A Novel Intrusion Detector Based on Deep Learning Hybrid Methods, 2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS), Washington, DC, USA, 2019, pp. 300-305, doi: 10.1109/BigDataSecurity-HPSC-IDS.2019.00062.

Week 5

Key Concepts

- Describe the features of a typical operating system, with specific details about the Windows system.
- Explain the core functionality of a file system and why it needs to be secure.
- Identify security features and flaws of contemporary and historical versions of Windows operating systems.

Operating systems: windows and OS hardening

The Operating System (OS) is a piece of Software that manages hardware resources and provides services to competing programs.

A few important definitions:

OS Security The processes and methods involved in guaranteeing the the integrity, confidentiality and availability of the OS

OS Protection Refers to methods and procedures to protect the OS from intruders and attacks

OS security includes all precautionary control techniques to help protect any computer resources that might be removed or modified if the OS is compromised.

Some key components of an OS are:

Kernal Executes services of the OS at the lowest level

Security Kernal Manages all OS's security processes

Reference Monitor Manages access to the device

The Security Kernal and Reference Monitor, together, for the Trusted Computing Base or TCB, which has everything necessary to enforce OS security policies.

In the Windows OS, the basic security blocks are:

Security Reference Monitor executes access checks

Local Security Authority executes windows local security policies

Security Account Manager database that stores credentials

Active Directory a directory service for Windows domain networks

WinLogon and NetLogon WinLogon manages local input logins; NetLogon manages network-wide logins

Some of the most commonly found client-side vulnerabilities are found in web browsers, office suites and the like. While any software could be attacked, attacks are, in practice, concentrated on the most popular software titles.

In order to overcome attacks, we must *harden* our systems before deployments. Hardening refers to the process of making an operating system more secure. For example, disabling automatic logins, enabling screen lock during screen saver, switching system security on, ensuring wireless connection is disabled if not needed, and so on.

File system and directory structure

A file is a set of linked data stored in non-volatile media. Each file has a logical location for storage and retrieval. In the operating system the *File System* is the data structure used to store, retrieve, and keep track of the files stored on the disk.

Some commonly used filesystems are:

- Windows
  - FAT32
  - exFAT
  - NTFS
- macOS
  - HFS+
  - APFS
- Linux
  - ext3
  - ext4
  - btrfs
  - ffs
- Unix
  - UFS
  - ZFS

There are three main types of files available for use in the OS:

Text File sequence of letters arranged in lines

Object File collection of bits stored in blocks

Source File refers to a variety of operations and activities in the operating system

The file system also maintains a set of metadata for each file, including creation time, update time, the actual volume, etc. All of this information is referred to as the file attributes.

Important attributes are:

File type the type of the file

Permissions who can read, write, and execute the file

Timestamps time and date of creation and last update

A file system also has a directory structure.

Windows Security

- Brief History of Windows Security
- Windows Consumer Security circa Oct 2020:
- Windows Enterprise Security circa Oct 2020

Week 6

Key Concepts

- Use a security auditing tool to identify and solve security problems in a Linux operating system.
- Describe the attack surface presented by a typical Android device.
- Discuss the issues with containerised systems and explain why this is relevant in contemporary internet architectures.

Why is Linux important for security & hardening Linux with Harbian

Linux is, nowadays, used everywhere. It runs on baby monitors, Azure Sphere devices, Smartphones (Android), the Financial Market, many of the Top 500 supercomputers, and so on. Securing and hardening Linux is a very important endeavor.

To that end, we look at Harbian – A Debian-based distribution focused at security – and how it can help us hardening a Linux system.

Harbian on Github

- `https://github.com/hardenedlinux/harbian-audit`

The Android attack surface

An average android smartphone has a very large attack surface.

Wi-Fi connects to outside world (internet) and is a possible attack vector. A badly written Firmware or Driver for this could allow remote code execution and data leaks

Bluetooth used mostly to connect with headsets, fitness devices, input devices, etc. One could design a malicious Bluetooth device, perhaps something claiming to be a headset to record private conversations

Cellular 5G, 4G, or any other mobile network goes here. It relies on a large software stack running in a modem.

Sensors Any of the sensors in a phone (camera, microphone, touchscreen, accelerometer, gyroscope, GPS, etc) have a software stack required to make them function. Any of these pieces may be susceptible to attacks

Android Stack The Linux kernel with a bunch of middleware and vendor-specific extensions, any of which may be susceptible to attacks

Apps Any of the apps installed on a phone may to prone to contain security flaws

Smartphone security: an overview of emerging threats.

- S. Grzonkowski, A. Mosquera, L. Aouad and D. Morss Smartphone security: an overview of emerging threats, IEEE Consumer Electronics Magazine 3(4) Oct 2014, pp.40–44.
- Wang, Y., W. Cai, T. Gu and W. Shao Your eyes reveal your secrets: an eye movement based password inference on smartphone, IEEE Transactions on Mobile Computing 19(11) 2020, pp.2714–2730.

Virtualised and containerised operating systems

In a virtual machine we spin up and entire model of a computer and, with it, an entire new operating system. Because of that, it ends up having a slow startup time.

Containers, on the other hand, runs the same kernel as the host OS. Because of that, it's smaller and starts up pretty fast.

There are four common attack routes to containers:

Host to Container The Host OS could be compromised and used to attack its Guest containers

Container to Host A security flaw may allow a container to leave its sandbox and access the Host's resources directly. Another possibility would be hundreds of containers running a DDoS attack against the Host

Container to Container A malicious container may attack another container

Application to Container A malicious application running on a container attacks it

Container Security

- S. Sultan, I. Ahmad and T. Dimitriou, Container Security: Issues, Challenges, and the Road Ahead, in IEEE Access, vol. 7, pp. 52976-52996, 2019, doi: 10.1109/ACCESS.2019.2911732

Week 7

Key Concepts

- Familiarise yourself with the basic components that underpin cryptography.
- Consider the application of cryptographic techniques in different settings.
- Explore practical concerns around cryptographic techniques and their associated security implications.

What is Cryptography?

Cryptography is all about encoding messages in a manner that can't be understood if intercepted by an eavesdropper. To motivate the discussion, let's introduce a scenario:

Alice wants to send messages to *Bob* in a secure manner. While this is happening, *Eve* wants to eavesdrop the communication, something like shown in figure 1 below.

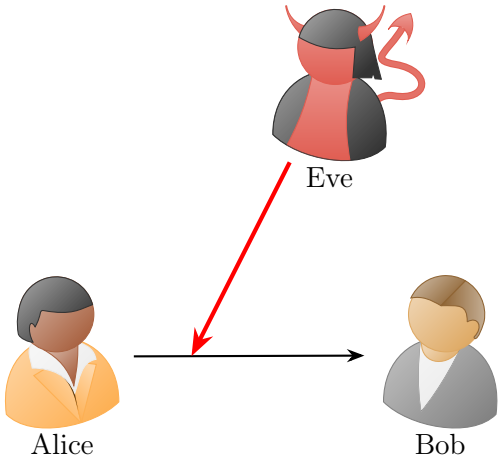

Figure 1: Alice sends messages to Bob while Eve listens

The goal is, therefore, to allow *Alice* to send messages to *Bob* in a way that even if *Eve* manages to intercept messages, they won't be able to understand the contents of said messages.

We can think about this as if *Alice* puts the message inside a box and locks it with a key. *Bob* will, then, proceed to unlock the box upon its reception. For this to work, *Bob* must have a working key, provided by *Alice*.

In a more real scenario, *Alice* takes her message (e.g. *The moon is full*) and proceeds to *encrypt* the message. Encrypting the message, modifies it in a way that it looks like random gibberish. One cannot *decrypt* the message without access to a *Decryption Key*. More technically, *Alice* converts *plaintext* into *ciphertext*.

A new question arises: seeing that *Alice* converts *plaintext* into *ciphertext* and sends the *ciphertext* to *Bob*, we can conclude that *Bob* **must** be able to convert the *ciphertext* back into *plaintext*, otherwise the message can't be read. We need to come up with ways to agree on this method without telling *Eve*.

History of Cryptography

Cryptography is all around us:

- Military Applications
- Web browsing
- Email
- Social Media
- Online Transactions
- Pay-per-view Television
- Cloud Services (including storage)
- etc

Brief history of cryptography:

- First used around 1900 BC at the tomb of Khnumhotep II
- Most early uses are Military
- Julius Caesar credited for the technique standard for centuries
- Al Kindi's book on cryptography involved the process of decoding cryptographic messages
  - Technique used: frequency analysis
- In the 16^{th} century, Vigenère created a cipher which used and encryption key (the first of its kind)
- Modern Times: Public Key Cryptography

  RSA Most of the internet

  Elliptic Curves bitcoin et al

The goal of cryptography are:

Confidentiality noone can read the message except authorized receiver

Integrity assurance that message hasn't been altered, modified, or damaged accidentally by an attacker

Authentication assurance that messages comes from expected source

Non-repudiation sender cannot deny a comment or action

Symmetric cryptography

Using our Alice and Bob scenario again, reproduced in figure 2 below:

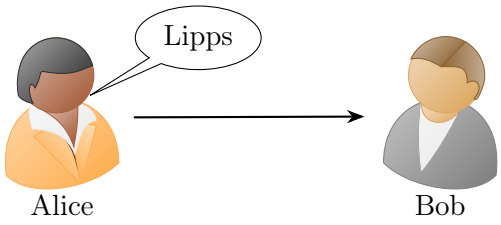

Figure 2: Alice sends locked messages to Bob

In a Symmetric Cryptography scenario, the key used to lock and unlock the box with a message is the same exact key. All encryption worked like this for millennia.

A famous example is the Caesar Cipher, where, given a message, we take all letter and slide down the alphabet by 4 (wrapping around to the beginning when necessary):

Plaintext	**Ciphertext**
A	E
B	F
C	G
...	...
V	Z
W	A
X	B
Y	C
Z	D

For example, if Alice wants to send the message *Hello*, she would convert each letter using the table above, therefore the ciphertext would be: *Lipps*.

This is called a *Substitution Cipher* because we **substitute** each letter by another letter using a rule. Unfortunately, this is very easy to break, as there are only 26 letters in the alphabet, therefore each letter can only move by at most 25 places. We can easily test all possibilities.

In case we have a large body of ciphertext, we can rely on the frequency of letters. In the English language, **a**, **s**, and **e** are very frequently used letters, therefore we can look at the most frequent letter in the ciphertext and assume it to be **a**, or **s**, or **e**, calculate the offset from **a**, or **s**, or **e** to that letter in the ciphertext and try that as a key.

Asymmetric cryptography

In Asymmetric Cryptography, the keys used for locking and unlocking (or encrypting and decrypting) are different. Not only that, but it **must** be very difficult to determine one of the keys from the other in at least one direction.

The main idea is that each player has two keys:

Private Key nobody, except for the owner, knows about it. Let's denote it **G**

Public Key everybody has access it. Let's denote **P**

These keys undo each other:

1. text -> F(text) -> G(F(text)) -> text
2. text2 -> G(text2) -> F(G(text2)) -> text2

The public key has to be easy to compute, however, those with access to the public key, should be able to compute its inverse (the private key).

When Alice wants to send a message to Bob, she should encrypt the message with Bob's public key. In thise situation only Bob's private key is able to decrypt the message.

When Alice wants to **sign** a message, she uses her private key since she's the only one with access to that key.

Some basic cyptographic techniques

- Buchanan, W.J. Cryptography. (Denmark: River Publishers, 2017) pp.1–11.

Week 8

Key Concepts

- Explore practical concerns around cryptographic techniques and their associated security implications.
- Consider the application of cryptographic techniques in different settings.
- Familiarise yourself with the basic components that underpin cryptography.

Cryptography techniques transposition methods

Substitution ciphers, will substitute a symbol for another symbol, while Transposition ciphers will permute the symbols.

For example, the following is an example of a Transposition cipher:

```
Plaintext: meetmeatthecorner
Ciphertext: renrocehttaemteem
```

Writing letters in reverse order is far too simple. A slightly better approach is to use a *Transposition Grid*. When employing a Transposition Grid, our message comes acompanied by a number which denotes the number of columns in the grid.

Assuming we're sending the message **THEEAGLELANDSTONIGHT** and the number 4, we will get the following grid:

T	H	E	E
A	G	L	E
L	A	N	D
S	T	O	N
I	G	H	T

The ciphertext is a read of the columns, therefore, the ciphertext is **TALSIHGATGELNOHEEDNT**. If we wanted to send the same message, but the number of columns was 3, instead of 4, we would get the following grid:

T	H	E
E	A	G
L	E	L
A	N	D
S	T	O
N	I	G
H	T	X

Which results in the ciphertext **TELASNHHAENTITEGLDOGX**. Another variation of the same technique is that instead of transmitting the number of columns, we use a keyword. Let's say the keword is **BASE**. The length of the keywords tells us the number of columns in the grid, so we get the same grid as before:

B	**A**	**S**	**E**
T	H	E	E
A	G	L	E
L	A	N	D
S	T	O	N
I	G	H	T

After doing that, we sort the columns by alphabetising the keyword, which gives us the following grid:

A	**B**	**E**	**S**
H	T	E	E
G	A	E	L
A	L	D	N
T	S	N	O
G	I	T	H

The ciphertext is, again, the read of the columns which results in **HGATGTALSIEEDNTELNOH**. One advantage of this method over the previous is that we also move the location of the first and last characters.

The basics of transposition cipher

- `https://en.wikipedia.org/wiki/Transposition_cipher`

Cryptographic techniques substitution 1: Playfair

The Playfair Cipher is named after Lord Playfair for promoting its use. The cipher itself is grid-based and has four simple rules to be memorized. Because there are 26 letters in the alphabet, making a grid would be awkward as the only possibility would be 13×2 grid (no, we're not considering 26×1). To circumvent this, we treat I and J as the same letter which gives us a 5×5 grid. Now we fill a grid with a keyword.

To give an example of how this works, let's assume our message is **THISISMYMESSAGE** and the keyword is **MONARCHY**. To make the grid we will:

1. Put in the keyword in the grid, one letter at a time
2. Put the rest of the alphabet in alphabetic order

M	**O**	**N**	**A**	**R**
C	**H**	**Y**	B	D
E	F	G	I	K
L	P	Q	S	T
U	V	W	X	Z

Given this grid, the rules for encryption are:

1. Break up the plaintext into pairs

   TH IS IS MY ME SS AG E

2. If the letters in a pair is the same, put an **X** between them:

   TH IS IS MY ME SX SA GE

3. If you end up with and odd number of letters, put an **X** in the end.

   TH IS IS MY ME SX SA GE

4. For each pair, look at where the letters in the pair are

   a) If they are in the different rows and columns, look at the rectangle formed by the letters, replace with the letters in the other corner (e.g. TH -> PD)

M	O	N	A	R
C	**H**	**Y**	**B**	**D**
E	**F**	**G**	**I**	**K**
L	**P**	**Q**	**S**	**T**
U	V	W	X	Z

   b) If the letters are in the same column, replace each with the letter below it (e.g. IS -> SX)

M	O	N	A	R
C	H	Y	B	D
E	F	G	**I**	K
L	P	Q	**S**	T
U	V	W	X	Z

   c) If the letters are in the same row, please each letter with the letter to its right (e.g. GE -> IF)

M	O	N	A	R
C	H	Y	B	D
E	**F**	**G**	I	K
L	P	Q	S	T
U	V	W	X	Z

After all these conversions, our result is the ciphertext **PDSXSXNCCLXAXBJF**.

Playfair

- Buchanan, pp.11–16

Cryptographic techniques substitution 2: Vigenere

The Vigenère cipher is a form os polyalphabetic substitution. It's essentially a generalization of the Caeser Cipher which makes it more difficult to break. The amount of shifting a letters gets, is determined by a keyword.

For example, let's say we're using the keyword **EASY** and our plaintext is **MEETMEATNINE**. Each letter of the alphabet gets mapped to a number that dictates how far to move a character in the plaintext. Basically, A maps to 0, B maps to 1, C to 2, and so on.

For our keyword **EASY** we have E=4, A=0, S=18, Y=24.

M	E	E	T	M	E	A	T	N	I	N	E
E	A	S	Y	E	A	S	Y	E	A	S	Y
Q	E	W	R	Q	E	S	R	R	I	F	C

Vigenère

- Buchanan, pp.17-19

Frequency analysis

- Buchanan, pp.44-48

Week 9

Key Concepts

- Consider RSA for practical settings.
- Identify the strengths and weaknesses of RSA in applied cryptography.
- Practise basic implementations of the RSA algorithm to encrypt and decrypt messages.

Intro to RSA: Public key cryptography

RSA is one of the most important cryptographic algorithms around. It underpins most of today's secure communications.

It's an implementation of asymmetric cryptography, or public key encryption system. The basis of the system is that if the private key is used to encrypt, only the public key can decrypt and *vice versa*.

To send a message, Alice encrypts it with Bob's public key. In this scenario, only Bob can decrypt it with his private key. The process is shown in figure 3 below.

In practice, the public key is a function that's easy to compute, while the private key is hard to compute. This is referred to as a *One-way function*.

One example of such function is Prime Factorization. Given two primes, multiplying them is easy. Given a number, factorizing into two primes is hard. More specifically, RSA works with two **large** primes.

Primes and Euler's Phi Function

A prime number is a number with exactly two factors: 1 and itself. 1 is not a prime number as it has a single factor.

Euclid stated that *every whole number is made up as a product of prime number in exactly one way*. For example $150 = 2 \cdot 3 \cdot 5^2$.

Two numbers `n` and `m` are said to be co-prime if the following holds:

- Their Greatest Common Divisor is 1
- Their only common factor is 1
- Their prime factorizations have no numbers in common

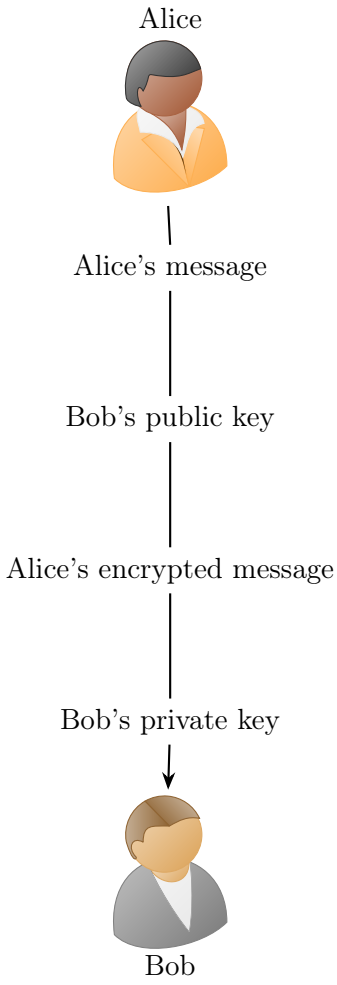

Figure 3: Alice Encrypts Message for Bob

Euler's Φ Function (or Totient Function) of a number `n` is given by $\Phi(n)$ and is defined as the amount of numbers co-prime with `n` that are also less than `n`. In other words, it's the number of integers k within the range $1 \leq k \leq n$ for which the greatest common divisor $gdc(n, k)$ is equal to 1.

For every prime p, we have that $\Phi(p) = p - 1$.

The Φ function is also a multiplicative function, which means that $\Phi(p \cdot q) = \Phi(p) \cdot \Phi(q)$. Moreover, if p and q are both prime numbers, then $\Phi(p \cdot q) = (p - 1) \cdot (q - 1)$.

Further reading

- Buchanan, W. J. Cryptography, River Publishers, 2017, pp.143-152.

Modular arithmetic

$13 \mod 6 = 1$ is the same as saying that the remainder of $\frac{13}{6}$ is 1. Also $(ab) \mod c = a \mod c \cdot b \mod c$

RSA example in practice

Given a message M, Alice encrypts it using Bob's public key, which is a pair (e, N). The encryption process is given by $E = M^e \mod N$. Therefore, Alice sends the ciphertext E to Bob.

Bob, consequently, decrypts the ciphertext using the process $D = E^d \mod N = (M^e)^d \mod N = M^{de} \mod N$.

Security of RSA

Alice is sending messages to Bob in a way that only Bob can read it. The way to achieve this is by using Bob's public key.

Bob, takes two large primes, p and q, and multiplies them, resulting $N = pq$. Then Bob chooses another number e to be coprime with $\Phi(N)$. The pair (N, e) is our public key.

Alice uses the pair (N, e) to encode message M to Bob using $E = M^e \mod (N)$ and sends the encoded message E to Bob. Bob uses private key, a number d such that $d \equiv e \mod \Phi(N)$.

Given number d, Bob decrypts the message with $D = E^d \mod N$ and that gives back M.

The reason this is safe, is that $\Phi(N)$ is very hard to compute.

Further RSA revision and practice

- `https://brilliant.org/wiki/rsa-encryption/`
- `https://www.geometer.org/mathcircles/RSA.pdf`

Week 10

Key Concepts

- Consider RSA for practical settings.
- Identify the strengths and weaknesses of RSA in applied cryptography.
- Practise basic implementations of the RSA algorithm to encrypt and decrypt messages.

5.6 Beginning the proof, first steps and an introduction to Fermat

Given a message M, Alice encrypts it using Bob's public key, which is a pair (e, N). The encryption process is given by $E = M^e \mod N$. Therefore, Alice sends the ciphertext E to Bob.

Bob, consequently, decrypts the ciphertext using the process $D = E^d \mod N = (M^e)^d \mod N = M^{de} \mod N$. We want to show that $D = M$.

We know that e is co-prime with $\Phi(N)$ and $de \equiv 1 \mod \Phi(N)$.

5.7 Fermat's little theorem, playing with proofs example

Fermat's Little Theorem states that given two numbers, a and p, where p is prime, $p \mid a^p - a$. Another way to state this is that $a^p - a \equiv 0 \mod p$. Note that by adding a to both sides, we get $a^p \equiv a \mod p$. Yet another to state this is by dividing both sides by a, which gives us $a^{p-1} \equiv 1 \mod p$. For this last version, a and p must be co-primes.

Lemma: Let a be a number coprime with p which is prime. The numbers $a \mod p, 2a \mod p, \ldots, (p-1) \mod p$ are all different.

Suppose $am = an \mod p$, dividing both sides by a, gives $m = n \mod p$. If $m, n < p$, then $m = n$.

We can use this result to proof Fermat's Little Theorem.

Considering the numbers $1 \cdot 2 \cdot \ldots \cdot (p-1) = (p-1)!$, multiplying by a we get $a \cdot 2a \cdot \ldots \cdot (p-1)a$, which is the same as $a^{p-1}(1 \cdot 2 \cdot \ldots \cdot (p-1)) = a^{p-1}(p-1)!$. We have, then:

$$a^{p-1}(p-1)! = (p-1)! \mod p$$
$$a^{p-1} = 1 \mod p$$

5.8 Finalising our proof

Given the result from Fermat's Little Theorem, we still need to show that $M = M^{de} \mod N$.

Lemma 1. *If p and q are prime, then if:*

1. $a = b \mod p$, *and*
2. $a = b \mod q$, *then*
3. $a = b \mod pq$

Proof. (1) implies that p is a divisor of $a - b$. (2) implies that q is a divisor of $a - b$. In other words, when we consider the prime factorization of $a - b$, p is one such factor and q is another factor.

Since both p and q are divisors of $a - b$, then so is pq, and hence $a - b = 0 \mod pq$ and therefore $a = b \mod pq$. □

Theorem 2. *Given a message M, an encryption key e and a decryption key d, $M^{de} \mod N = M$.*

Proof. By the lemma, it suffices to show the following two things:
i. $M = M^{de} \mod p$ ii. $M = M^{de} \mod q$

We know that d and e are chosen so that $de = 1 \mod \Phi(N)$, therefore $de = K \mod \Phi(N) + 1$ for some K. We also know that $\Phi(N) = (p-1)(q-1)$. Putting the last two together, gives us:

$$\begin{aligned} de &= K(p-1)(q-1) + 1 \\ M^{de} \mod p &= M^{K(p-1)(q-1)+1} \mod p \\ &= M \cdot M^{K(p-1)(q-1)} \mod p \\ &= M \cdot (M^{K(q-1)})^{p-1} \mod p \end{aligned}$$

There are two possible cases here. If M and p are coprime, then:

$$\begin{aligned} M^{de} &= M \cdot (M^{K(q-1)})^{p-1} \mod p \\ &= M \cdot 1 \mod p && \text{By Fermat's Little Theorem} \\ &= M \mod p \end{aligned}$$

If M and p are not coprime, then p divides M and $M \mod p = 0$:

$$\begin{aligned} M^{de} &= M \cdot (M^{K(q-1)})^{p-1} \mod p \\ &= 0 \end{aligned}$$

The same process is applied for $(q-1)$. □

Guidelines on cryptography

- Chief Security Office 'Security standard - use of cryptography (SS-007)', Department for Work & Pensions (March 2020).

Week 11

Key Concepts

- Be able to describe parity checking and hash functions.
- Be able to understand the differences between parity checking and hash functions.
- Be able to explain the advantages and disadvantages of hash functions and parity checking relating to data security.

Parity checking

Parity Checking is a simple method for ensuring a given message hasn't been tampered with.

If we count the number of a *1s* in a binary string, there are two possible outcomes:

Even Parity The number *1s* is even

Odd Parity The number of *1s* is odd

A simple error detection method can be constructed by assuming every message of a given length (e.g. 8-bits) gets one extra bit for parity. We can put a message rule stating that every message **must** be of even parity. Therefore, the final 9^{th} bit[1] will be initialized in such a way to guarantee even parity on the message.

In other words, if our message is the bit-string 11010011, which contains 5 *1s*, our parity bit must be set to *1*, so the number the *1s* in the message is even. To calculate the value for the parity bit, the sender can rely on the bitwise Exclusive-Or logical function, i.e. $1 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \oplus 1 = 1$.

Hash functions

A Hash function is a function that has a fixed size output, i.e. it takes any string as input and always produces another string with a given length n, or $H : \Sigma^* \to \Sigma^n$ where Σ is the binary alphabet. A Hash function must also be effectively computable, easy to compute.

[1] Note that this adds overhead on transmission. Every byte, in this protocol, is 9 bits long. In other words, for every 8-bits, we have 1-bit of overhead. This extra bit, allows for 1-bit error detection, so the overhead is acceptable.

The output of a hash function is referred to as a hash value or simply a hash. The simplest hash function is the 1-bit hash function:

$$H(x) = \sum_{i=0}^{n} x_i \mod 2$$

Where x_i is the i$^{\text{th}}$ bit of the input string x.

By increasing the amount of bits in the hash function, we get longer strings as output. The main application here is error detection. In practice, when publishing a certain content, we also publish the hash value for the content. Anyone downloading our content can regenerate the hash value and compare with the canonical value we published[2].

Hashing

- Buchanan, W.J. Cryptography. (Denmark: River Publishers, 2017) pp.87-112.

[2] A simple example is with the tool is the `sha256sum` tool available on UNIX-like systems.

Week 12

Key Concepts

- Be able to describe hash tables and how they are used.
- Understand how hash functions can be used for authentication purposes.
- Understand the relevance of bitcoin and why its central design goals are so important.

Cryptographic hash functions

We want a cryptographic hash to serve as proof of authenticity for a document. Our hash must be:

Public anyone should be able to use it

Effectively computable and deterministic there must be a polynomial-time algorithm for computing it and for the same input it must produce the same value

One way given $H(x)$ it is computationally infeasible to find x.

Target collision resistant given x it's infeasible to find x' that causes a collision

Random-like output output looks random

The first two requirements together, mean that if we publish a hash, anyone can authenticate it.

Given an input x, it is computationally infeasible to find x' such that $x \neq x' \wedge H(x) = H(x')$. Note that x' may very well exist, but a computer can't calculate it in a reasonable amount of time. In fact, given that the input space in infinite (i.e. $\{0,1\}^*$, or all possible strings over the binary alphabet) and output space is finite (i.e. the algorithm maps input to output of a finite length of e.g. 256 bits).

A small perturbation to input, should match a largely different output.

Introduction to Bitcoin

- Buchanan, W.J. Cryptography. (Denmark: River Publishers, 2017) pp.303-311.

Week 13

Key Concepts

- Understand how the concepts of hash functions, peer-to-peer networks and blockchain are combined in the design of the bitcoin network.
- Be able to describe the 'identity problem' and explain how bitcoin solves this problem.
- Be able to describe the 'solvency problem' and explain how bitcoin solves this problem.

The origins of Bitcoin

In 2008 Satoshi Yakimoto published a paper entitled *Bitcoin: A Peer-to-Peer Electronic Cash System*[1] which establishes a protocol for decentralized currency transactions which solves the double-spending problem[2] using a peer-to-peer network.

Network and transactions

In a Bitcoin network there are no Trusted Authorities, whenever Eve wants to send Bob a transaction such as *"I, Eve, give Bob 2.3 Bitcoins"*, concerns are raised:

Identity Question How can we confirm that Eve initiated the transaction?

Affording Question Can Eve actually afford/fullfill this transaction?

Double-spending Question How do we know Eve hasn't already spent the money?

The identity problem

The identity problem is two-fold:

1. How do we know that Eve wrote the message?
2. How do we stop Eve from later denying she sent the message?

[1] `https://bitcoin.org/bitcoin.pdf`
[2] `https://en.wikipedia.org/wiki/Double-spending`

Both of these questions can be answered by the application of Asymmetric key cryptography. In summary, Eve signs the message with her private key. In fact, each cryptowallet has its own pair of Private and Public keys. To generate a transaction Eve follows the process below:

1. Sign the message with secret key

```
sig = sign(sk, msg);
```

   where:

   - `sig` is a string containing a hash
   - `sk` is the secret key
   - `msg` is the message or transaction

   Note that the signature is a function of both the secret key `sk` and the message `msg`. This is because if we were to use the same signature for every message, it would be easy to generate new messages with the same signature.

2. Verify the signature

```
verify(pk, msg, sig);
```

   where:

   - `pk` is the public key
   - `msg` is the message or transaction
   - `sig` is the signature produced by `sign()`

   This verification process will check that the signature is valid and indeed matches with Eve's public key. The function `verify()` will produce a boolean value (i.e. True or False) to encode the validity of the signature.

Both `sign()` and `verify()` are mathematical functions. Unlike RSA, however, these are not based on modular arithmetic of large primes but on Elliptic Curves[3].

The solvency problem

The question of whether Eve has the money to fullfill the transaction can be answered by making sure that every transaction has a reference to where the money came from for that transaction.

Figure 4 below shows a depiction of this:

When transferring currency from wallet A to wallet B, we use the hash of the recipient's public key (the wallet's public key, that is) as the destination address.

The inputs to a transaction are:

[3] `https://en.wikipedia.org/wiki/Elliptic-curve_cryptography`

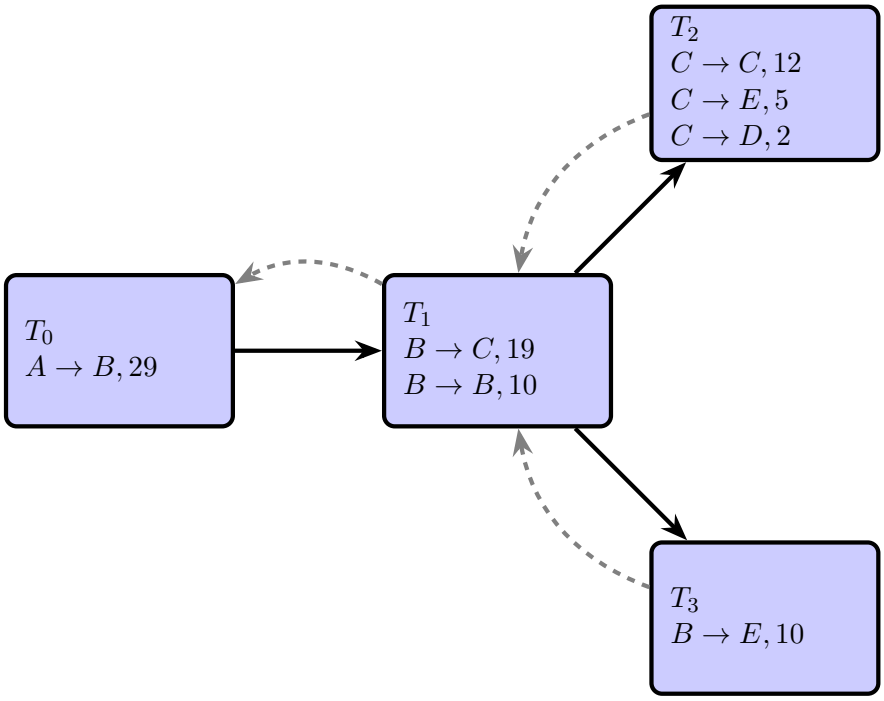

Figure 4: Transactions and References

1. Amount
2. Signature of the account sending the money
3. Transaction ID: the hash of the transaction

Reading Activity

- Blockchain technologies for the Internet of Things: Research issues and challenges

Week 14

Key Concepts

- Be able to describe the 'double spending problem' and explain how bitcoin solves this problem.
- Be able to explain the concept of 'immutability' and why it's a critically important characteristic for data on the blockchain.
- Be able to explain the historical importance of 'The White Paper', and how blockchain technology can be used for more than just financial transactions.

Double spending

How do we know Eve hasn't already spent the money? The way to solve this is by having a public ledger available to all, meaning that anyone in the network would be able to spot a double spending attempt.

This *Public Ledger* **is** the blockchain!

On top of this, there needs to be a consensus to ensure that one and only one transaction gets accepted and added to the blockchain. This consensus is based on *Proof of Work*.

Proof of Work is an idea adapted from Hashcash. It's essentially a method for one party, known as the prover, to prove to other parties, known as the verifiers, that a certain amount of computational effort has been carried out. One requirement is that the effort must be high (but doable) from the prover perspective, but cheap from the verifier perspective.

The way this was adapted to bitcoin was that a node must do work to earn the right to put things in the blockchain. This is commonly referred to as *mining*.

Once a node has the right to add transactions to the blockchain, that allows the node to put a number of transactions into the chain – roughly around 2000 transactions. Such transactions come together in a structured known as a *block*.

The block contains the list of transactions and also a hash of the block before it, which makes the chain behave similarly to a linked list. This is the *chain* part.

The final question is what happens if Even tries to spend the same money with two different miners and they happen to mine a new bitcoin at the same time. In this situation, the conflicting blocks will have to wait until another block is created on top of one of them. Any following block will always choose the longest chain.

Immutability of the blockchain

The blockchain is the immutable. This means that once something is added to the public ledger, it can't ever be change, this is because it would break the hashes of the blocks.

While hash collisions are possible, they are very hard to come by, to the point that it's pointless to try.

Even if an attacker were to re-do the proof of work for an older block, it would still not help, because newer blocks will always choose the longest chain.

When it come so the proof of work, the problem involved is as follows:

Given a cryptographic hash function $H(x)$ and a block B, find a value n, referred to as the *nonce*, such that $H(B \cdot n)$ results in a hash starting with a least T zeroes. Where T is referred to as the *target*.

In practice, $H(x)$ is the double application of a standard hash functions called H256. I.e. $H(x) = H256(H256(x))$. As T grows, the problem has exponentially more difficult. If $T = 1$, then $\frac{1}{2}$ hashes will start with 1 zero. If $T = n$, then $\frac{1}{2^{n+1}}$ hashes will solve the problem.

Week 15

Key Concepts

- Explain how arbitrary data can be securely stored on the bitcoin blockchain.
- Define the term 'smart contract' and state the limitations of smart contracts on the bitcoin blockchain.
- Identify differences between the bitcoin and ethereum systems.

Data on the blockchain

Data can be stored in the blockchain, usually using the COINBASE field.

Programs on the blockchain

Programs can be stored on the blockchain. Bitcoin scripts dictate how the transaction can be verified. These programs are called *Smart Contracts*. The *P2PKH* (Pay to Public Key Hash) is a commonly used program for this purpose.

Introduction to ethereum

Ethereum was created focussed on data and programs, rather than currency.
Ethereum consists of the following major concepts:

Blockchain Very similar to Bitcoin

Nodes Powerful computers that maintain the state of the blockchain

Currency Nodes gain ETH (the currency) for mining blocks and processing transactions

Ethereum Virtual Machine (EVM) Embedded in the blockchain. Transactions can execute tasks on the EVM, which changes its state

Accounts Hold ETH balance

Transactions Transfer ETH from one account to another

Smart Contracts Execute tasks on the EVM

Week 16

Key Concepts

- Describe a real world system that uses blockchain technology for a computer security-related application.
- State the purpose and motivation for the IPFS distributed filesystem and make connections between this and computer security concepts.
- Explain how blockchain technology can be used to incentivise different types of activity.

Data on the blockchain case study

Information on the web can be updated at any time and there's little to no *paper trail* of what has been changed and when. In some cases, it may be necessary to know that a certain piece of information has (or has not) been modified. Moreover, if the database system of a major news outlet or financial institution has been tampered with, there's no way of knowing without a costly audit.

Timestamped Hashes can help solve this problem. In summary, this means storying in a blockchain a pair consisting of the cryptographic hash of the contents of the e.g. webpage in question and a timestamp of when the hash was generated. If the webpage is modified somehow, the hash won't match anymore and we know it has been tampered with.