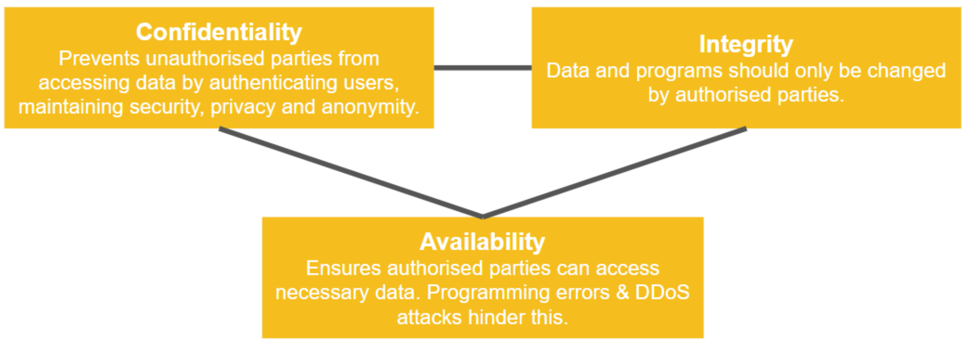
Introduction to Cybersecurity

Cybersecurity protects electronic systems, like servers and networks, and data from **malicious actors and cyberattacks** by ensuring information privacy and providing guidelines.

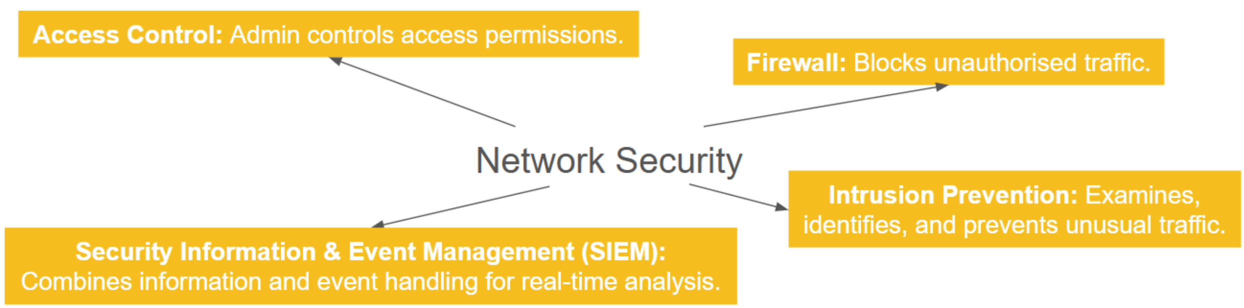
The CIA Triad

A model set of guidelines for security system vulnerabilities to protect data from unauthorised access, deletion and modification.



It's difficult to achieve all three at once and they're all linked!

Elements of Network Security



Types of Cyber Attack

Phishing

Pretends to be someone else to get confidential information, like login details. Usually through emails, messages, and phone calls.

- **Spear Phishing:** Targets an individual within an organisation.
- **Whale Phishing:** Targets high-level execs.

Ransomware

Blocks access to a computer system or threatens to delete data until money is paid.

- **Double Extortion:** Encrypting, stealing and threatening to leak, sell or delete it if the ransom isn't paid.

Malware

Disrupts, damages, or gains unauthorised access to a system.

Trojan Horse
Malware which **breaks into systems** disguised as another program.

Adware
Pop-up advertisements which harm or slow devices, hack into browsers, and install viruses or spyware.

Spyware
Software which **secretly collects and transmits information** about computer and user activities.

Man-in-the-Middle Attack
A hacker places themselves **between a client and owner** (users and server) to steal or trick them into giving confidential data. Eavesdroppers can only steal.

SQL Injection
Exploits poor design by **inserting queries** that damage a database or make it output confidential information.

Denial of Service (DoS) & Distributed Denial of Service Attacks (DDoS)
A device **sends fake requests** to overwhelm a server, reducing efficiency and crashing it. DDoS is with a network of devices called a botnet.

Drive-by Attacks
Malicious scripts embedded into websites which **collect visitors' data**.

Jobs in Cyber

- Incident reporter
- Auditor
- Penetration tester
- Consultant
- Forensic expert
- **Security Analyst:** Pinpoints vulnerabilities, assesses risks, and deploys security protocols to ensure data security.

The Importance of Security

Demand for **DevSecOps**, which integrates security into DevOps, is increasing! Security is often far down on a developer's to-do list, until something goes wrong! Business leaders often don't even view mobile apps as a security threat! It should be a **whole system priority**.

Many vulnerabilities are design-based and can't be fixed easily, requiring updates to **firmware and software**, a risky process.

e.g. the MG5 hashing algorithm can be brute-forced. If developers don't notice, low-skill attackers can get access to data with no special privileges or user interaction!

Types of Attackers

- Financially Motivated: Ransomware.
- Ideologically Motivated: Russian-Ukraine war.
- Nation States
- Curious People
- Show-offs
- Grudges

Information Security: The preservation of confidentiality, integrity, availability but also authenticity, accountability, non-repudiation and reliability of information.

Methods to help maintain them include:

- **Authenticity:** Address information and use signatures.
- **Accountability:** Keeping logs.
- **Non-repudiation:** Confidentiality of signature information.
- **Reliability:** Backup procedures.

The Payment Card System

Originally, imprinters copied card embossing to create purchase records. The magnetic stripe was created to log transactions electronically. Then, Chip & Pin or EMV followed by contactless.

Terminology

Retailer/Merchant: Sells goods to customers.

Primary Account Number (PAN): Card number.

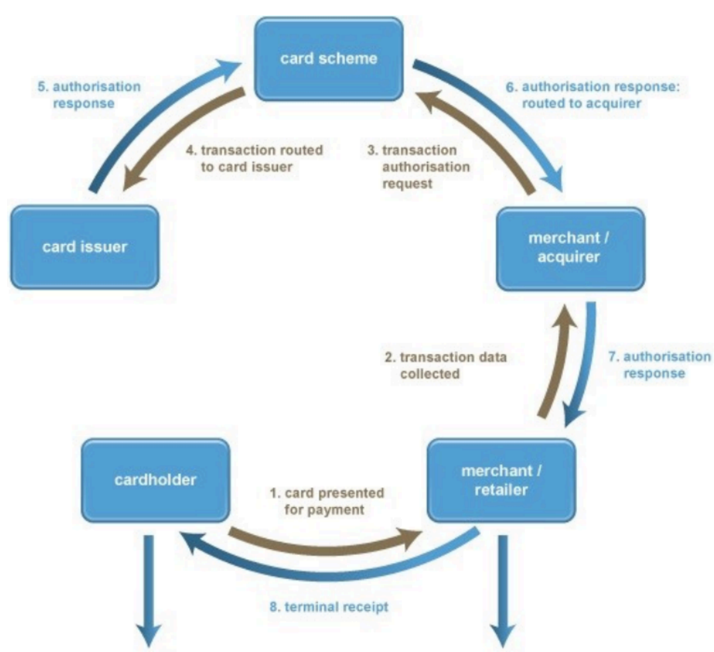
Acquirer: Holds a service agreement with merchants to **process transactions on their behalf**.

Issuer: Bank or organisation that provides cards.

Card Schemes: Organisations (*e.g. Visa*) that control the **operation** and **clearing of transactions** according to **card scheme rules.**

The Transaction Cycle

Card schemes and acquirers charge a small fee which is passed onto customers. Some card schemes act as issuers too.



Attractiveness to Attackers

The massive potential earnings attract attackers!

e.g. Visa's 2019 turnover was $1,970bn.

Famous Exploits

The Adverline Attack

Compromised JavaScript in an advertising service's code used to inject card skimming code into any e-commerce site using it.

The British Airways Hack

JavaScript code for adding an extra bag was changed to make payment details global in the code, allowing them to be stolen using an API.

JavaScript

A multi-paradigm, event-driven language for browser interactivity. Problems far away from confidential info, like a payment system, can be exploited to steal that info, making security a **whole system priority**. It's often loaded from unreliable sources for convenience!

Internet Security

As the internet has evolved, so have the security threats and protocols but most **systems are running behind**!
e.g. IPv4 was invented in 1983, IPv6 in 1997 but 55% of the internet still uses the IPv4.

Internet Protocol Version 4 (IPv4)

Facilitates communication by **assigning 32-bit IP addresses** to identify devices connected to the internet, enabling address and routing data packets across networks.

An IPv4/IPv6 **header is sent with every network transmission**. These contains essential information such as:

- The source & destination IP addresses.
- Flags.
- Version & protocol.
- Checksum: The sum of the 16-bit words to protect against accidents. **Not cryptographically secure!**

<table>
<tr><th>Offsets</th><th>Octet</th><th colspan="8">0</th><th colspan="8">1</th><th colspan="8">2</th><th colspan="8">3</th></tr>
<tr><th>Octet</th><th>Bit</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td>0</td><td>0</td><td colspan="4">Version</td><td colspan="4">IHL</td><td colspan="6">DSCP</td><td colspan="2">ECN</td><td colspan="16">Total Length</td></tr>
<tr><td>4</td><td>32</td><td colspan="16">Identification</td><td colspan="3">Flags</td><td colspan="13">Fragment Offset</td></tr>
<tr><td>8</td><td>64</td><td colspan="8">Time To Live</td><td colspan="8">Protocol</td><td colspan="16">Header Checksum</td></tr>
<tr><td>12</td><td>96</td><td colspan="32">Source IP Address</td></tr>
<tr><td>16</td><td>128</td><td colspan="32">Destination IP Address</td></tr>
<tr><td>20</td><td>160</td><td colspan="32" rowspan="5">Options (if IHL > 5)</td></tr>
<tr><td>24</td><td>192</td></tr>
<tr><td>28</td><td>224</td></tr>
<tr><td>32</td><td>256</td></tr>
</table>

IP Routing

The internet is made of **Autonomous Systems (ASs)** which route their own IP traffic and communicate through **Border Gateway Protocol (BGP)** *e.g. "I can reach AS 123 in 7 hops".* There are three types:

Stub
Connect to one another.
E.g. UoB to Joint Academic Network.

Multihomed
Connected to multiple ASs, but doesn't carry traffic for them. *E.g. IBM connected to every country it does business in but only carries IBM traffic.*

Transit
Provides connections to other networks through itself.
E.g. Joint Academic Network

Challenge: **BGP lacks validation** which enables attackers to snoop on traffic, route hijacking, route injection attacks, IP spoofing, etc. Hence, the **need for monitoring network traffic**.

Definitions

- **Secure Sockets Layer (SSL):** Precursor to TLS, provides encrypted network traffic.
- **Transport Layer Security (TLS):** Encrypts, authenticates, data to ensure integrity. Provides forward-secrecy.
- **Hypertext Transfer Protocol (HTTP):** Transmits web pages and resources between servers and browsers in plaintext.
- **HTTP Secure (HTTPS):** Pairing with SSL/TLS for encryption & security.
- **HTTP Strict Transport Security (HSTS):** Allows a site to force browsers to use HTTPS in future, protecting against man-in-the-middle attacks like downgrading and cookie hijacking.
- **SSL Certificate:** A file verifying a website's identity and used to encrypt communication between a server and browser. Usually provided by a trusted Certificate Authority.
- **Root Certificate:** A file owned by a Certificate Authority used to validate other certificates. The foundation of trust in a certificate hierarchy.

Don't memorise:

- **Transmission Control Protocol (TCP):** Ensures reliable, ordered and error-checked transmission.
- **User Datagram Protocol (UDP):** Transmits data without establishing a connection first. Useful in streaming and real-time applications.
- **Session Key:** A temporary key used for encryption during a single communication.

How Does TLS Work?

1. Client connects to a TLS-enabled server requesting a secure connection and presents a list of supported cipher sites.
2. Server picks a cipher and a hash function that it also supports and tells the client.
3. Usually the server provides an SSL certificate with its public encryption key.

4. Client confirms certificate validity.
5. To generate the session keys, the client either:
    - Encrypts a random number with the server's public key and sends it to the server through asymmetric encryption. Both parties use it to generate a session key for efficient symmetric encryption.
    - **Diffie-Hellman key exchange** used to securely generate a random, unique session key with **forward secrecy.**

Domain Name Systems (DNSs)

Translates **human-readable domain names** (*e.g. www.example.com*) into **machine-readable IP addresses** (*e.g. 192.0.2.1*) through **DNS resolution**, making it **easier to access** websites **without remembering** numerical addresses.

When a domain name is typed into a browser, it sends a request to a DNS server to find the associated IP address. The system consists of components like **DNS servers**, **records** and **domain names**.

- IP addresses are 32-bit (IPv4) or 128-bit (IPv6), and a system can handle both.
- **Fast & Universal:** Attractive for alternative uses like checking if emails from an IP address are spam.
- **Usually Uses UDP:** Responses can be received without a request, which might be believed.

Registrar: An accredited organisation which manages and registers domain names within a DNS for users to purchase.

Problems

- **Confusing:** Unstructured nature leads to duplicate domain names.
- **Automated Registrars:** Often don't make simple checks when assigning domains. *e.g. nhs.gov.co.uk*
- **Outdated & Poor Security:** Predates the web.

The Domain Name Structure

DNS Over HTTPS is much more secure.

Top-Level Domains

Found at the end. There are a few categories:

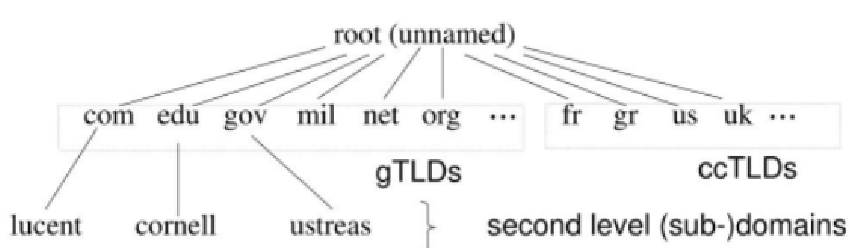
- **gTLDs:** Generic Top-Level Domains *e.g. .com, .edu*
- **ccTLDs:** Country Code Top-Level Domains *e.g. .uk, .nz*

Second-Level Domains

The customisable middle bit. *e.g. google in google.com*

Subdomains

The first part which extends from second-level. Used to create unique pages within a site. *e.g. mail in mail.google.com*



Effective Top-Level Domains (eTLDs)

A top-level domain **usable by any organisation**. It's difficult to tell where the publicly registerable part ends and the organisation begins. Hence, Mozilla maintains a list of eTLDs.

A/DNS Records: Map domains to IP addresses.

Functionality

- **Nodes:** Represent domains. *e.g. .ac.uk and bath.ac.uk*
- **Leaves:** Nodes without further subdivisions. *e.g. mail.google.com*
- **Authoritative Nameservers:** Each non-leaf, top-level node has a name server which responds to DNS requests for their children. For example, an authoritative server for *.ac.uk* can provides nameservers for *bath.ac.uk*.

Resolvers

a.k.a. DNS server. They query nameservers to translate domain names into IP addresses. There are three types:

- **Stub:** Relies on another resolver to perform queries for it, often a browser.
- **Recursive:** Queries multiple DNS servers & caches results.
- **Root:** Responds to queries by directing them to the authoritative server for the top-level domain.

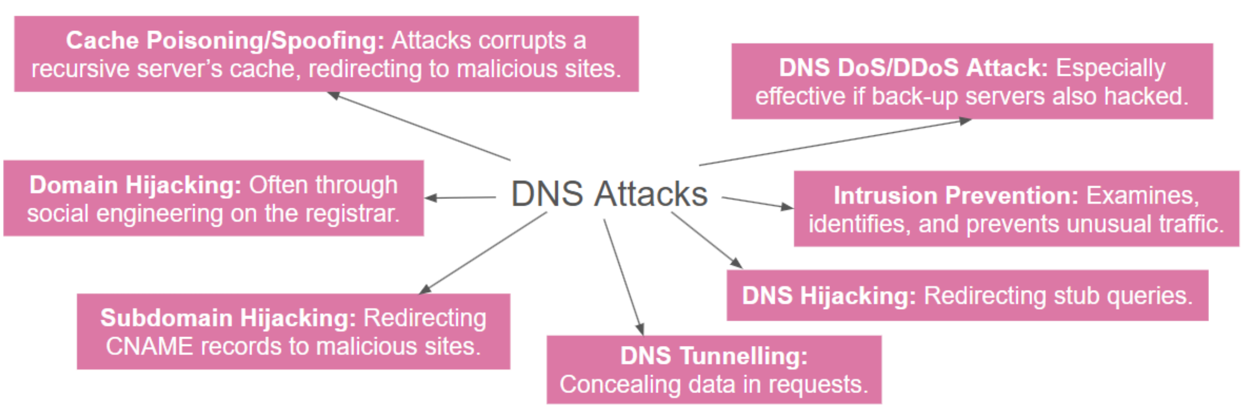
Don't memorise!

When someone searches for "www.example.com":

1. Stub asks recursive for "www.example.com".
2. Recursive checks cache and asks root nameserver if A record not present.
3. Root provides the nameserver for ".com".
4. Recursive asks ".com" server.
5. ".com" provide the nameserver for "example.com".
6. Recursive asks "example.com" server.
7. "example.com" provides the A record.
8. Recursive caches it and tells stub.

Queries to non-existent domains return NXDOMAIN.

Canonical Names (CNAME Records): Map a domain to another, allowing multiple to point to one server.



Subdomain Hijacking

1. Company uses a service and points a subdomain to it. *e.g. support.domain.com*
2. They stop using the service but don't remove the redirection to the service.
3. Attacker signs up for the service and claims the domain.
4. They clone the site, now redirected to by the company's site.

Don't memorise!

Enterprise Systems

In enterprise networks, all traffic goes through an enterprise resolver which blocks queries to malicious sites and inspects all traffic.

Linux DNS Tools

- **ping:** Sends packets & asks for a response.
- **traceroute:** Sends messages that time out at intermediate routers.
- **nslookup:** Asks DNS queries.

HTTP Strict Transport Security (HSTS)

Allows a site to **force future connections into HTTPS**. Browsers use **pre-loaded lists** to enforce it straight away without the "in-future" and must **terminate connections where the TLS certificate can't be trusted**.

HTTP Cookies

HTTP/S is **stateless**, so you can't tell if requests were sent by the same person. **Cookies enable states** and hence websites to:

- Remember information.
- Record browser activity.

They store that a user has been authenticated, not the password. This authentication key should be session-specific.

Types

- **Session Cookies:** Last until browser closes.
- **Persistent Cookies:** Last until expiry date.
- **Secure Cookies:** Only transferred over HTTPS.
- **HTTP-Only:** Can't be accessed by client-side APIs.

The SameSite Attribute

Set by websites to control how the cookies they create are sent across sites.

- **Strict:** Only send to the same domain.
- **Lax:** Target domain must be the same as origin, preventing third-party cookies.
- **None:** Sent in all cases, enabling third-party.

Ad companies will persuade sites to add a CNAME in their DNS which points to their sites, so SameSite is satisfied but they still receive the data.

Cross-site Request Forgery (CSRF)

Tricks the victim into submitting a malicious request, allowing the attacker to inherit their identity and privilege to perform actions on their behalf.

Can be prevented using **tokens** and **frameworks** like ruby-on-rails, but XSS defeats these and it's still a threat.

Cross-Site Scripting (XSS)

Attackers **inject client-side scripts into webpage**s. Composed of multiple attacks on the **same-origin policy**. Prevented using **sanitisation** and XSS-defensive programming.

- **Reflected:** Unsanitised user input included in HTML output.
- **Stored:** Unsanitised input stored.
- **DOM:** The document object model is the API to the stored model of the page in-browser.

Same-Origin Policy

Prevents scripts from accessing resources from different origins. Origins are the scheme (http/https), domain and port together.
e.g. prevents a page loaded from a malicious site from reading the cookies for a banking site.

Subverted with **cross-origin resource sharing**, document.domain and cross-document messaging.

The Content-Type HTTP Header

Tells us:

- **Download:** Filetype of the content being received. Sometimes set to *unknown* where sniffing is needed to determine it by checking the response.
- **Upload:** Filetype of uploaded files: use the content-type provided, the file extension or sniff for it.

Cryptography

Cryptography is vital for information security, but doesn't ensure it!

- **Forward Secrecy:** Protects data by generating a unique session key for each communication. Party A has forward secrecy with B if breaching B's key doesn't reveal the communication. **Provided by TLS 1.3.**
- **Semantic Security:** An attacker has negligible advantage in guessing any of the plaintext, because the chance of being successful is so low.
- **Basic Encryption:** Same-size messages don't help to get the plaintext.

Symmetric Encryption	***Asymmetric Encryption***
Uses the same key for encryption & decryption. ● **Faster:** More efficient. ● **Key Distribution:** Requires a secure method to share it between parties.	Uses a pair of keys: public for encryption and private for decryption. ● **Slower:** Complex maths. ● **More Secure.**

Attack Types

- **Chosen Plaintext Attack (CPA):** Getting ciphertext from plaintext.
- **Chosen Ciphertext Attack (CCA):** Attacker getting plaintext from ciphertext.

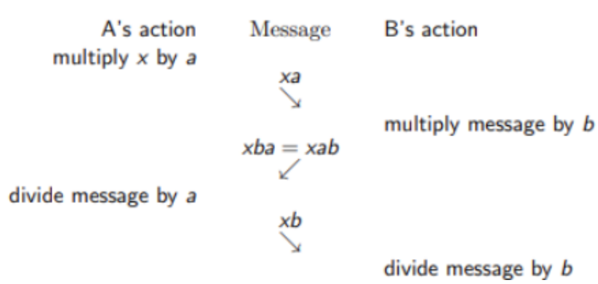
Secure Communication

Two individuals can communicate securely without a secret key:

1. A locks data with their algorithm A, and sends to B.
2. B locks data with their algorithm B, and sends to A.
3. A unlocks data with their algorithm A, and sends to B.
4. B unlocks the data with their algorithm B, giving them the original.

Multiplication: A Poor Implementation

All data is stored as integers modulo a large prime P. After every calculation you should do the result mod P.



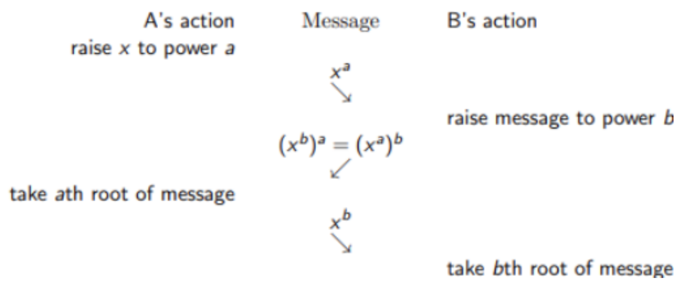
Don't memorise!

Using multiplication, as above, lets users calculate the message if they see all three!

Diffie-Hellman: A Secure Implementation

A **secure, asymmetric** key exchange algorithm used in **TLS/SSL** to **establish a shared session key for symmetric encryption**.

- A variant called ECDLP which uses elliptic curves for extra security is used in practice.
- **Discrete Logarithm Problem:** For large primes, it's impossible to get the message.

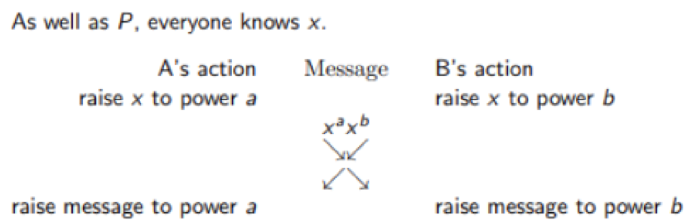


Don't memorise!

Secure Key Agreement

Agreeing a secure key beforehand **reduces the number of messages** required.

Now, they're both in possession of the key $x^{ab} = x^{ba}$.



Don't memorise!

The RSA Algorithm

A widely-used asymmetric algorithm for **encryption, digital signatures** and **key exchange**. It allows one party to encrypt messages and a receiving party to decrypt them. Based on the challenge of **factorising large numbers**.

Digital Signatures

Ensure the authenticity and integrity of a message by dividing it up, using RSA to encrypt each section, and the recipient checks them once they've been decrypted to ensure they're sensible.

Ciphers

A **block symmetric key cipher** on n-bit blocks is a pair of functions e_k and d_k to encrypt and decrypt s.t.:

1. Encrypting the message returns the original.
2. It's hard to find the plaintext without the key.
3. You can't find the key from the plain and ciphertext.
4. It's hard to encode another message from the plaintext and ciphertext.

Cryptography in Attacks

Man-in-the-Middle	***Meet-in-the-Middle***
In Diffie-Hellman, a middle-man can **intercept and relay all messages between parties**, reading them while tricking them into thinking they have a secure channel.	Encrypting the plaintext and decrypting the ciphertext for **different combinations until a match** is found.

Symmetric-key Cryptosystems

Data Encryption Standard (DES): Can be brute-forced.

2DES: Secure but a Meet-in-the-Middle attack is theoretically possible.

3DES: Backwards compatible but outdated.

Advanced Encryption Standards (AES): Modern approach.

Hash Functions

Convert inputs to unique, fixed-size strings of bytes called a **hash value**, enabling **data verification** and integrity. Fast hashes are useful but **make brute-force attacks easy**.

An n-bit hash function reduces a message to an n-bit string, $n \in [0, 2^{n-1}]$, with:

- **Collision Resistance:** Hard to find two messages $m_1 \neq m_2$ for $h(m_1) = h(m_2)$.
- **Pre-Image Resistance**
  1. Given v, it's hard to find m with $h(m) = v$.
  2. Given m_2, it's hard to find $m_1 \neq m_2$with $h(m_1) = h(m_2)$.

Actual Hash Functions

- **MD4 & 5:** Made in the 90s, now broken.
- **SHA-0 & 1:** Now broken.
- **SHA 2:** Current recommendation.
- **DES -> Triple DES -> AES:** AES block size is constant but key size varies.

Cryptographically Secure Random Number Generators (CSPRNGs)

Require a **seed** and have two key requirements:

1. **Next Bit:** Impossible to predict the next bit with more than 50% accuracy.
2. **State Compromise:** Can't reconstruct previous output from the revealed state.

Passwords

Verifiers store passwords, and should never do so in plaintext because if someone gets them all is lost!

Passwords should be hashed with a **cryptographic** hash function H to ensure the plaintext can't be recovered (**pre-image resistance)**. When a password is entered, H(Input) is compared with the H(Password) stored.

If H is leaked, a **system-wide brute-force** can be used to repeatedly hash different inputs and check if they match the stored passwords. Assuming the password is eight characters:

- **a-z:** 20 seconds.
- **a-z & 0-9:** 5 minutes.
- **a-z, 0-9 & A-Z:** 6 hours.

Dictionary Attacks, which test common passwords, are even quicker!

Rate limiting, guess limiting and using **computationally expensive hash functions** helps prevent brute-forcing!

Salts

Also prevent brute-forcing by **adding a random value to the plaintext to ensure identical inputs produce different hash values**. With passwords, the salt is **unique** to the user!

Always use a different hash for each user.

Can use a CSPRNG for the salt.

Salt should be the length of the hash.

Can encrypt hashes too.

Hash on the server, not the client.

Reuse & Changing

- Reusing passwords for **low-value** accounts is okay.
- Regularly changing passwords only leads to people writing them down!
- Password managers struggle to implement **authentication** and autocomplete-attributes correctly.
- Websites don't implement clean and well-structured authentication forms.
- **Good Server-side Practise:** Make users add MFA.

Migrating Password Management

Upgrading from method H_1 to H_2:

1. Add a flag to the database for the hashing type each password uses.
2. When a H_1 user logs in, compute the H_2 of their input, store and change flag.
3. After a period, email all H_1 users to say they must login within a time period.
4. After this period, lock their accounts and make them reset their password.

Multi-Factor Authentication (MFA)

Factor: Something you know, have or are.

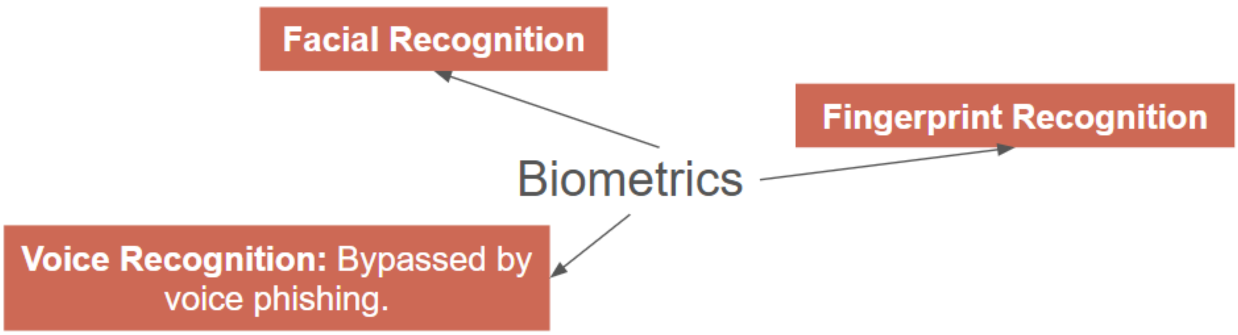
MFA relies on more than one factor, but **not the same twice!** *e.g. passwords and fingerprints*

Phones as 2FA

There are two types:

Code: Entering a code from an SMS to you. **App:** Entering a code from an authentication app.

Apps are more secure because they rely on you **owning the phone**, not the phone number, preventing **SIM-swapping**. With mobile banking, SMS is just the same factor twice.



Hardware Authentication

Provides more secure authentication.

Security within Computers

Programs need **access controls tailored to their content and role**. For example, the following may need different permissions:

- Programs in different tasks.
- The same program in different computations.
- Different programs in the same computation.
- The same program in the same computation for different conditions.

The hardware and OS ensure processes only execute permitted instructions, which are **application-dependent**. *e.g. government/military systems have strong requirements: unclassified, restricted, confidential, etc.*

Bell-LaPadula *a.k.a multi-level security*

A model for **mandatory**, **system-enforced access controls**. It assume apps are buggy and users make mistakes, with the requirements:

- **Simple Security:** Processes can't read data at a higher level. *e.g. restricted can't read confidential*
- ***Property:** Processes can't write data at the lower level.

High Water Mark Principle: Output confidentiality should be at least as high as the maximal input confidentiality.

Evolution of Security Requirements

- **Multics:** Started as an MIT project. One of the first OSs with data protection and user privacy.
- **Unix:** A simpler, uni-tasking response to Multics.
- **Orange Book:** US DoD criteria for secure systems.
- **Common Criteria:** An ISO standard for evaluating security and reliability, evolved from Orange Book.
- **Biba:** A low water mark model.

Levels of Access Control

Simple

Access controls can be modelled as an **access control matrix**, a matrix of permissions. The description of a user's right to access a file/program is called a **cell**.

- r - reading
- w - writing
- x - executing

	Operating System	Accounts Program	Accounting Data	Audit Trail
Sam	rwx	rwx	rw	r
Alice	x	x	rw	–
Bob	rx	r	r	r

User	Operating System	Accounts Program	Accounting Data	Audit Trail
Sam	rwx	rwx	r	r
Alice	rx	x	–	–
Accounts program	rx	rx	rw	w
Bob	rx	r	r	r

Complex

Accounts have rules enforced by the accounting program, a **pseudo-user**. Alice is an employee running the OS who can make only changes to accounts via the program.

Storing Access Control Matrices

The cost to store and access all cells is massive! *e.g.linux.bath.ac.uk has 10^4 users and 10^7 files.*

The Unix Architecture

Keeps **Access Control Lists (ACLs)** for each file, specifying which users can access it. **Runs efficiently** but **groups can be clumsy**!

- **UserIDs & GroupIDs:** Stored as 32 bits.
- **List of Permissions:** Specifies user, group and others' permissions.

Each process has a **file descriptor table** which tracks the resources it has open, a weak form of capability. They contain **too much state** - passing it on to another process allows it to change the state.

Capability Architectures

A **capability** is an unforgeable right to perform an action, like reading & writing memory.

Real-world Use Cases

- Military/Classified Systems.
- Mobiles & Browsers.
- Kernels: OKL4 & seL4.

Cloud Security

The **Chief Security Office (CSO)** is responsible for **physical** security (*e.g. locks, ID cards*). The **Chief Information Security Officer (CISO)** is responsible for **electronic security** (*e.g. passwords, MFA*).

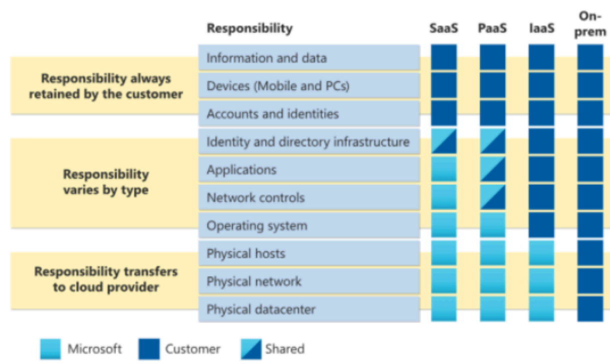
Cloud computing enables networks access to scalable resources. It has: **on-demand and measured self-service, broad network access, resource pooling and rapid elasticity.**

- **Cloud Supplier:** Ensures physical security and that people unauthorised by the customer's system can't act.
- **Customer:** Authorises people and manages inter-user interactions.

Models

- **Software as a Service (SaaS):** Email, zoom, etc.
- **Infrastructure as a Service (IaaS):** Provides a virtual machine.
- **Platform as a Service (PaaS):** Provides a physical machine.

The supplier owns the physical machines with two user types: their own staff and customers.



Cloud Access Security Broker (CASB)

A **security policy enforcement point** between enterprise users and cloud service providers. It enforces **visibility** (helps IT know what's going on)**, data security, threat protection** and **compliance**.

Cloud Security Posture Management (CSPM)

Assesses and responds to threats to secure cloud infrastructure and ensure regulation compliance.

Versioning

Cloud backup services have **ransomware mitigation tools** that need to be managed by users, but can often be **worked around by attackers**.

Cloud Provider Tools

- **Write-Once-Read-Many (WORM) Model:** Ensures data, once written, cannot be modified or deleted, for regulatory compliance.
- **Immutable Containers/Objects:** Cannot be modified or deleted for a period.

- **Server-side Encryption:** Often provide multiple algorithms, where the provider or customer manages the key. Ransomware can just encrypt encrypted files again.
- **Key Management Systems (KMS):** Manage keys and carry out server-side encryption. Can be abused.
- **Lifecycle Policies:** Reduce storage costs by reducing file storage tiers.
- **Control Policies:** Blocks users depending on their connection location.

Hardware Security

Moving beyond stand-alone devices, we need:

- **Protected OS:** No uncontrolled method for user programs to take admin.
- **Memory Protection:** Write and read protection.

Zero-day vulnerabilities are security flaws **unknown to developers** until they're discovered!

Vulnerability Exploits

A device being off doesn't prevent malware injection!

Rowhammer

Exploits the proximity of DDR3 memory cells to cause **charge degradation in adjacent rows** through repeated exercise of a row.

Spectre

Exploits **speculative execution** to access inaccessible data. Misguided speculations leave exploitable side effects: malicious programs cause cache hits or misses and the timing difference can connote information about restricted data.

- The *cflush* function clears cache for consistency and a high-resolution timer is needed.
- Retbleed is a variant.

Firmware Underpins Everything

Firmware is critical but often overlooked.

"Pantsdown" allows access to specific circuit boards, giving unauthenticated access to memory allowing attackers to dump firmware, read/write to RAM and brick the board.

Hardware for Secure Computing

Hardware Security Modules (HSMs)

A **trusted network computer** built on lab-certified hardware with a security focused OS.

- Limited access via network interface with rules.
- Hides and protects cryptographic material.
- Generally has anti-tamper resistance and destroys keys on tampering.

Uses

- Secure documents. (*e.g. passport printers & gates*)
- Digitally signed games.
- Smart metering systems.
- The aircraft ecosystem.

Trusted Platform Modules (TPMs)

An **international standard for secure cryptoprocessors**, allowing people to:

- Create, store and limit cryptographic keys.
- Authenticate devices and encrypt data, ensuring integrity.
- Verify hardware and software hasn't been changed.

Trusted Execution Environments (TEEs)

A **secure processor area** guaranteeing confidentiality and integrity of its contents.

The Common Vulnerabilities and Exposures System

A standardised way to categorise and track all sorts of vulnerabilities, not just hardware. Major security databases store CVE numbers alongside exploit details.

e.g. CVE-2023-48788

SQL Injection

Sends **untrusted SQL command**s to an interpreter, allowing it to execute untrusted queries. Prevented by **sanitising** inputs in the backend. Front-end validation can be disabled!

- **Boulder attacks** exploit **web app** vulnerabilities to manipulate databases.
- **Second order attacks** save data to the database which will manipulate it when its output.

e.g. DROP TABLE was used to erase the FBI's 2016 election fraud records.

The MOVEit Hack

Exploited a zero-day vulnerability in the MOVEit file transfer software using SQLi to access customer databases, including the BBC and British Airways.

Prepared Statements / Parameterisation

Parameterise queries if you can, involves using **precompiled placeholders** for user inputs.

Mitigation

Developers should train in SQLi, we should audit all open-source code, closed-source should have an SQLi-focussed penetration testing report and databases should enforce least privilege!

Supply Chain Risks

Supply chains are the **resources, processes and systems involved to create, distribute and deliver hardware and software**. They present plenty of weaknesses for exploitation.

Compilers

It is possible to create a compiler with a completely innocent source, which compiles the source identically but can add code to it.

SolarWinds

An supplychain attack on the SolarWinds network infrastructure management and monitoring software which provided attackers with a backdoor to remotely access and steal information from systems using the software.

Security, Forensics & Frameworks

Forensics is the application of science to criminal and civil law during a criminal investigation. The aim of digital forensics is to **examine digital media in a sound manner** to identify, preserve, recover, analyse and present facts about information.

Locations of Evidence

- User devices.
- Public & private remote resources.
- Remote file storage.

Implications

The prosecution must prove to a court that their evidence is no more or less than it was when it was first seized. Sadly, it's often easy for first responders to destroy the trail.

Attribution

Cyber attribution is the process of tracking and identifying the perpetrator of a cyberattack or hacking exploit. Attributing attacks to real-world entities is difficult and risks misidentification.

Information Management Systems

ISO standards **outline information security management** which large organisations like to comply with. Medium-sized organisations look at them when managing incidents.

Import Cryptographic Controls

Some governments impose **controls on the importation of cryptographic tools** to address national security, privacy and economic concerns, like:

- Decoding tools must be provided.
- Banning it.
- Must request government permission.

Popular Cyber Defence Frameworks

- PCI DSS.
- CIS.
- NIST.

Mobile & Pervasive Systems

Mobile/pervasive/IoT systems and operational technology are easily attackable because they're often **designed by people with little cybersecurity knowledge**, like engineers, and the chance of vulnerabilities is massive!

- **Pervasive Systems:** Everyday systems with computational components. *e.g. toothbrushes*
- **Internet of Things (IoT):** The concept that most devices are part of a network of other devices.
- **Operational Technology (OT):** Hardware and software that detects or causes changes by monitoring and controlling industrial equipment and processes.

Updating every IoT devices OS is expensive, so they're usually protected by a separate firewalled framework.

OWASP Top 10 Mobile Threats

1. **Improper Credential Usage:** Hardcoded, insecure transmission, storage & authentication.
2. **Poor Supply Chain Security:** Third-party components should be vetted and updated.
3. **Insecure Authentication**
4. **Poor Input/Output Validation**
5. **Insecure Communication:** Not checking the TLS certificate.
6. **Poor Privacy Controls**
7. **Poor Binary Protection**
8. **Security Misconfiguration:** Least privilege enforced, debugging features disabled, etc.
9. **Insecure Data Storage:** Data stolen after devices fall into hostile hands.
10. **Insufficient Cryptography**

Email

There are two different *FROM* addresses which can be different, in the envelope and content.

Envelope
Contains metadata, like labels and headers added by handlers, used by email servers.

Content
The actual message.

Mail eXchanger Records in the DNS

Sending an email to "user@example.com", your email server **queries the DNS** for example.com's MX records, and it will usually respond with a **prioritised** list of **mail servers**

e.g. for cl.cam.ac.uk, it responds with:

```
cl.cam.ac.uk     mail exchanger = 20 mx2.forwardemail.net.
cl.cam.ac.uk     mail exchanger = 10 mx1.forwardemail.net.
```

Simple Mail Transfer Protocol (SMTP)

One of the original internet protocols!

Allows anyone to pretend to be anybody! But there are some solutions:

Sender Policy Framework (SPF)

Allows domain owners to **specify which computers can send mail with their domain** in the *envelope-from* address, by creating an SPF record in their DNS.

DomainKeys Identified Mail (DKIM)

Allows receivers to check email form a domain was authorised, by affixing a digital signature, linked to the *envelope-from* domain name, to each outgoing email. It **confirms message integrity** even if the sender's address has been spoofed.

But, the *From* section visible to users isn't validated, so attackers could send an email from their server with a spoofed *From* to trick users.

Domain-based Message Authentication, Reporting and Conformance (DMARC)

Allows domain owners to publish a policy in their DNS records for how receivers should **check the user-visible *From* field** and handle failed validation.

If SPF or DKIM pass, then the DMARC **alignment check** passes, because it definitely came from the authenticated site. If not, it **checks alignment with the *envelope-from* or DKIM signature.**

The options on failing are: **none**, **quarantine** & **reject**.

Disadvantages of DMARC
● **Complicated setup.** ● **Insufficient Staff:** Smaller companies don't have the staff to run it. ● **Machine-Readable Files:** Difficult for humans to read. ● **Doesn't Prevent Phishing:** Prevents mail pretending to be *x.com* but not *xx.com*.

Penetration Testing

An **ethical hacker** simulates a real-world attack to identify **vulnerabilities**. Similar to vulnerability scanning but more thorough, requires more expertise and human interaction.

Ethical Hacker: Trained to identify and fix system vulnerabilities before hackers can exploit them.

Red Team Exercise: Simulates the latest attack types and methods, with the incident response.

Organisations should document their PenTesting methodology including:

- Approaches.
- Coverage.
- Network & application layer testing.
- Plan to resolve problems.
- Review of threats in the last 12 months.

Steps/Standard

1. Pre-engagement interaction.
2. Intelligence gathering.
3. Threat modelling.
4. Vulnerability analysis.
5. Exploitation.
6. Post exploitation.
7. Reporting - must be understood by non-specialists.

Types

Network

Web App

Wireless

Social Engineering

Software have requirements to employ techniques which mitigate common attacks: injection, cryptography, access control, etc.

Vulnerability Scanning

Automatically scans to detect defects in an organisation's security program and generates reports

- Can be run on a schedule, on-demand or in response to an event.
- Can check compliance with standards.
- Fast, cost-effective, scalable and accurate!

Internal

Inside the organisation.

- At least once every 3 months.
- Resolves high-risk vulnerabilities and rescans confirm resolution.
- Scan tool kept up-to-date with latest vulnerability info.
- Performed by qualified and independent testers.

External

Outside the organisation.

- At least once every 3 months.
- BY a PCI SCC Approved Scanning Vendor (ASV).
- Resolves vulnerabilities and rescans confirm resolution.
- ASV Program Guide outlines requirements.

Vulnerability Management Systems

Monitor applications for vulnerabilities using industry-recognised sources and assign a risk based on the potential impact. Often keep a Software BIll of Materials to track third-party components. Can be used with standard, bespoke and third-party software.

Firewalls

Used in public-facing web apps to detect and prevent web-based attacks by blocking them or setting off an alert. They generate audit logs and must be kept up-to-date.

- Frequently bypassed.
- Costly.

Third-party Risks

X is the first party (often the supplier), engaged with second party Y (often the customer), and the **third party** (Z) **is everyone else**.

Outsourcing

X contracts Z to process Y's data.

- Expanded by cloud: SaaS.
- GDPR controls liability for privacy breaches.
- X and Y don't know where data is or what it's with!

e.g. SITA stored data passenger data alongside extremely confidential values, leading them to getting stolen.

Insourcing

X uses Z's software/libraries to process $Y's$ data, but Z often **poorly documents security weaknesses**. *e.g. SQL injection*

Development

- Traditionally, components were decided at writing. Libraries and software were **static** and well-controlled so only changed when the program was compiled.
- Shared libraries/dynamic link libraries meant they running programs **connected** to them instead using a major/minor version system. The code comes from the machine through software repositories, allowing it to be **updated without program recompilation**.
    - Attackers can create a public package of the same name and interface as a private one but with a higher version number.
- **Systems are now assembled dynamically**. *e.g. websites retrieve JS from other sites.*

If X outsources or insources, it's still the **data controller** and has to **ensure data privacy** by choosing trustworthy processors.

Mixsourcing

Z's software is somehow involved while X processes Y's data.

Case Study: Target

Customer's card details, including CVV, were leaked by infecting their card terminals with an executable file, likely automatically downloaded from a hacked server. Infected devices stored and forwarded mag-stripe data.

Attackers broke into Target's network using login credentials stolen from a mechanical service they were using. This led to PCI requiring network separation.

Case Study: Log4j

A common application monitoring/logging software. A bug allowed remote execution by injecting prepared strings into the logging library. Impacted loads of apps!