

Information security and privacy

Pierre Colson

Monday 03 January

Contents

Vocabulary	2
Basic properties	2
Security	2
Privacy	2
Cyber Threats	3
Web application vulnerabilities	3
Software vulnerabilities	3
Crypto	4
TLS and HTTPS	5
Dtatabase Security	5
Password Storage	5
Access Control	6
Authentication	6
Network and Operational Security Practices	7
Trusted Computing	8
Privacy	9
Definition and classification	9
Crypto based solution	10
Non crypto based solutions	11
PETs for data anonymization	11
Machine learning	13
Blockchain	15
Data protection for personalized health	17

Markdown version on [github](#)
Compiled using [pandoc](#)

Vocabulary

- **Virus** : a malware that infects a file and replicates by infecting other files
- **Worm** : a piece of malware that propagate automatically
- **Trojan** :
 - a malware hidden in a usefull software or file
 - a malware that stays on the victim's computer and communicate with a control center to carry out malicious activity
- **Rootkit** : hides the precense of a malware on a computer
- **Ransomware** : encrypts the files and request payment for decryption
- **Vulnerability** : weekness in the logic, the software or hardware of a system (bugs)
- **Exploit** : method/tool to make advantage of a vulnerability
- Vulnerability can be fixed by **patching** a system
- **Zero day** exploit : exploit for which no patch exists yet

Basic properties

Security

- Protects the data of data owners against attacks
- **Confidentiality** :
 - keep informations secret
 - give read access only to those who need to know
 - tools : access control, isolation, encryption
- **Integrity** :
 - keep information correct
 - prevent modification of the data
 - detect modification
 - tools add a hash, a MAC or a signature, make public
- **Availability** :
 - keep information available/systems running
 - tools : make copies, duplicate/distribute systems, prevent intrusions
- **Authenticity** :
 - demonstrate the authenticity of information
 - prevent fake information
 - detect modification
 - tools : add keyed hash (MAC) or a signature
- **Non repudiation** :
 - prevent denial of a statement
 - tool : add a signature as proof of origin

Privacy

- Protects the data *subject* against abuse
- **Confidentiality** :
 - keep information of *the data subject* secret
 - give access only to those who need to know
 - tools : access control, encryption, absence of data
- **Anonymity** :
 - prevent a link between data and a subject
 - reduce/modify information until no correlation is possible
 - tools : k-anonymity, defferential privacy
- **Absence of information** :
 - prevent revealing information

- do not request, or delete information that is no longer needed
- work on encrypted information
- tools : homomorphic encryption, private information retrieval, zero knowledge proofs

Cyber Threats

- A **threat** is a potential unwanted action that creates impact
- **Cyber attack lifecycle** :
 - Preparation
 - Gain access
 - Maintain access
 - Complete mission
 - Cover tracks
- **Commodity threats** :
 - Non targeted
 - Fully automated
 - Low risk to attackers
 - Short term financial gains
- **Hacktivism** :
 - Politically motivated hacking
 - Variant of (anarchic) civil disobedience

Web application vulnerabilities

- **OWASP** : Open Web Application Security Project
 - Documentation on the top 10 critical security risk of web application
- **Injection** :
 - Context can be : HTML, JavaScript, JSON, SQL
 - Special character sequences in user inputs can trigger an action in the context
- **Injection protection** :
 - Refuse characters you do not want
 - Escape (encode) special characters when you use them
- **Direct object reference** : When a user-submitted parameter is a direct reference to a resource, a user may try to change it to access other resources

Software vulnerabilities

- **Buffers overflows** : while writing data to a buffer, overruns the buffer's boundary and overwrites adjacent memory location
- **Buffers overflows protection** :
 - **Stack canaries** :
 - * Push a random value on the top of the stack at the beginning of a function
 - * Before returning, verify that the value has not been modified
 - **Non executable memory** :
 - * Do not want to set execution permission on a page that can be written while the program is running
 - **Address space randomization (ASLR)** :
 - * Every time the program is started, it is loaded at a random address
 - * Every time the system boots, the OS is loaded at a random address

Crypto

- **Symmetric Crypto** : Encryption and decryption is done with the same key
 - Solve the problem of transferring large amount of confidential data
 - Creates the problem of transferring a symmetric key
- **Stream cypher** : Use the key and a pseudo random generator to generate a stream of random bits
- **Block cypher** : Encrypt fixed blocks of data
 - a *padding scheme* is used to fill the last block
 - a *mode of operation* is used to combine multiple blocks
 - DES (collisions and brute force) \Rightarrow AES
- **Mode of operation** :
 - ECB :
 - * Encrypt each block separately with the same key
 - * Same cleartext block results in same ciphertext block
 - CBC :
 - * Introduces the use of an *initialization vector* (IV) for the first block
 - * Each ciphertext block acts as a IV of the next block
 - * Decryption is the opposite of encryption
 - * Does not reveal any structure
 - * Malleability : flipping one bit in a ciphertext block flips the same bit in the next cleartext block and mangles the current block
 - * The last block must be padded to obtain the correct block size, if not carefully implemented, validation of padding can lead to leakage of the cleartext
- **Hash** function takes an arbitrary length input and generates a fixed length output
 - *Pre-image resistance* : Given a hash h , it is difficult to find a message m for which $h = \text{hash}(m)$
 - *Second pre-image resistance* : Given a message m_1 it is difficult to find a second message m_2 such that $\text{hash}(m_1) = \text{hash}(m_2)$
 - *Collision resistance* : It is difficult to find two arbitrary messages that have the same hash
 - SHA-3 : no weakness known
- **Messages authentication codes** (MAC) :
 - Like a hash function, but involves a symmetric key
 - The same key is used to generate the MAC and to validate it
 - If the key is known only to the two parties of an exchange, a correct MAC proves
 - * that the message was not created by a third party (authentication)
 - * that the message was not been modified (integrity)
- **Public-key Crypto** : Uses a pair of public (encryption) and private key (decryption)
 - Solves the problem of having to agree on a pre-shared symmetric key
 - No need to keep the public key secret (as the name suggests ^^)
- Asymmetric is powerful but orders of magnitude slower than symmetric crypto
- Asymmetric is typically used to exchange a symmetric key
- All these algorithms are only safe if you use keys that are long enough
 - symmetric : 128 to 256 bits
 - asymmetric : RSA 2048 bits, ECC 256 bits
 - hash function : 256 bits
- With public key crypto the public key does not have to be secret but it still has to be authentic (e.g. man in the middle attack)
 - We need a trusted third party to distribute the public keys
 - The Certification Authority certifies the keys by signing them
 - * If we trust the key of the CA, we can trust all keys signed by the CA
 - A signed key is a certificate. It contains at least :
 - * The identity of the holder
 - * The validity date of the certificate
 - * The public key of the subject
 - * The signature by the CA

TLS and HTTPS

- **TLS** Transport layer Security : provide a secure channel between two communicating peers
 - The server is authenticated with a certificate
 - It proves its identity by signing some information received from the client with its private key
 - Client and server create a symmetric key using asymmetric crypto
 - They use a symmetric cipher to encrypt data:
 - They use HMAC to guarantee integrity
- Let's Encrypt \Rightarrow free certificates
- A **Public key infrastructure** (PKI) distributes public keys using certificates
- HSTS and Certificate transparency protect against MITM and fraudulent CAs

Database Security

- **Access control** : Least privilege
- Granularity at the row level can be achieved by defining *views*
- SQL databases also support *role based* access control
- To limit the impact of SQL injection, use different DB users for different accesses
- **Hardware/OS**
 - *Functions* : Data is encrypted when read/write to disk
 - *Protect against* : Stealing/cloning virtual machines
- **Database**
 - *Functions* : DB encrypts when read/write to file
 - *Protect against* : Access by OS users/admins
- **Network**
 - *Functions* : DB encrypts when read/write to network (e.g. TLS)
 - *Protect against* : Hackers cannot sniff data in transit
- **Application**
 - *Functions* : Application encrypts when read/write to the DB
 - *Protect against* : Access by admins, memory dumps by OS admins
- If the data is encrypted in the database then the DB cannot
 - search with wildcards (e.g. WHERE name='Pete%')
 - sort, compare or aggregate data
 - \Rightarrow the DB is pretty useless

Password Storage

- Classic way : use **salt** and **iterations**
- Modern way : use a **memory hard function**
- **Time-memory trade-offs** :
 - We create a *reduction* function r : it takes a hash as input and produces a password from our set
 - We build chains : $p_1 \xrightarrow{\text{hash}} h_1 \xrightarrow{\text{reduce}} p_7 \xrightarrow{\text{hash}} h_7 \xrightarrow{\text{reduce}} \dots \xrightarrow{\text{hash}} h_3$
 - We only keep the first and last element of each chain
 - * this is where we save memory
 - * we pay for this with more time to crack the password
 - We build a table with several chains
 - Hellman's original trade-off becomes inefficient when there are too many chains in a single table
 - * For each collision of the reduction function, we end up with two identical chains
 - **Rainbow table** solve the collision problem by using a different reduction function in each column
- If you search through all columns of all tables :
 - Hellman : t^2 memory look-ups, t^2 hash operations
 - Rainbow : t memory look-ups, $\frac{1}{2}t^2$ hash operations
- Adding a random value (**hash**) to the hash function prevents :

- cracking multiple hashes with a single hash calculation
- calculating the hashes in advance
- Another simple way to slow the attacker is to apply the hash functions multiple times
- **Memory hard function**
 - the function run through many steps
 - intermediate steps results are stored in memory
 - each step depends on results from previous steps

Access Control

- **Access control** defines and enforces the operations that can be done on objects
- Principle of **least privilege**
 - subjects should have the minimum rights necessary to their job
- Multiple level of access control :
 - Network
 - Operating system
 - Application
 - Within an enterprise
- Multiple approaches to access control
 - **Role-based Access Control (RBAC)**
 - * Simplifies the specification of permission by grouping users into roles
 - * A role can contain multiple permissions
 - **Discretionary Access Control (DAC)**
 - * Access control is at the discretion of the object owner
 - * Owner specifies policies to access resources it owns
 - **Mandatory Access Control (MAC)**
 - * Tries to ensure that even someone with access cannot leak the data
 - * Depends on the trusted software and admins
 - * no write down
- **ACL vs Capabilities**
 - Think of a door protected by a bouncer vs a door protected by a lock
 - ACL :
 - * The bouncer knows exactly who can get in
 - * People don't know where they can get in and where they cannot
 - Capabilities :
 - * Doors do not know who will show up with a key
 - * People know exactly for which doors they have a key
- Modern OSes make use of all of these types :
 - DAC with ACLs for file and most objects
 - DAC and capabilities for privileged operations
 - Using groups to implement RBAC
 - Mac for protection of the integrity of a system

Authentication

- Access control only makes sense if we can *authenticate* subjects
- **Password**
- **Something you own** : hardware/software token
- **OATH** is a standard that describes
 - How **OTPs** are generated from a seed
 - An XML format for importing the seeds into an authentication server
- **Biometrics**
 - no hashing is possible

- it is impossible to change a stolen finger
- **Challenge-response** : Rather than sending the password to the server
 - The server sends a random challenge to the client
 - The client uses the hash of the password to create a response
- **Kerberos** uses a three steps approach
 - An authentication server (AS) authenticates the client and delivers a ticket granting ticket (TGT)
 - The client can then present the TGT to the ticket granting server (TGS) to get a ticket for the service he wants to use
 - The client can access the service
- **OAuth2** is a protocol used for delegated authentication on the internet
 - Facebook, Google, Twitter etc. can be used to authenticate and access other application

Network and Operational Security Practices

- Secure communication, outside of our network
 - TLS, IPsec, VPNs
- **Network segmentation** : Break down the network based on system and data classification or into functional zones
 - Access from zone to zone can be managed by access control list (ACLs) in router or firewalls
 - Prevents all-at-once compromise of facilities
 - Protects the data center from external threats
 - Containment zones aims at stopping attacks from spreading between zones
- **Demilitarized Zone (DMZ)**
 - A physical or logical subnet that contains and exposes an organization's external-facing services to an untrusted network
 - An external network node can access only what is exposed in the DMZ
- **Zero trust network**
 - Do not trust anybody, not even internal machines
 - More work for configuring machines
 - Less work on configuring the network
 - Greatly reduces the impact if one machine is compromised
- **Virtual private network**
 - Encryption and encapsulation keep the network private
 - Before a packet is sent over the public network, it is encrypted and encapsulated with an IP header with the public address
 - Let remote workers access the internal company network
 - Interconnecting remote sites for a company
- **Firewalls**
 - Enforce network level access control
 - Firewalls operate at the network layer
 - Firewalls should also be present within the network
 - Principle of default deny
- **Proxies**
 - They operate at the application level
 - **(Direct) proxies** : between the client and internet
 - * Protect our users when they access servers on the internet
 - **Reverse proxies** : between internet and the server
 - * Protect our servers when accessed by users from the Internet
 - **Web proxies** protect users by
 - * Analyzing all data downloaded from the web with anti-virus software
 - * Blocking access to dangerous sites
- **Web application firewall (WAF)**
 - It stands in front of your web server and receives the requests from the internet.
 - It analyses the request, and if it deems them safe, it forwards them to the real server

- **Instructions detection systems**
 - Inspects traffic for all application to detect potential intrusions
 - **Signatures based** system
 - * Network traffic is compared to signature form a pattern database
 - * *Snort* is an example of signature based IDS
 - **Anomaly based** system
 - * IDS creates traffic profile during normal operation to callibrate
 - * Looks for unusual packets
 - Possible issues
 - * False posititves (too many alarms)
 - * False negatives (too many sucessful attacks)
- Keeping **audit trails** (logs) is an important part of network security
- Good way to protect data : **Backups**
 - We also need restorations tests, to check if we are actually able to restore data from backups
 - We also need a **Disaster Recovery Plan** (DRP) that explains in details how to rebuild each system in case of a major failure

Trusted Computing

- **Trusted hardware** : A piece of hardware can be trusted if it always behaves in the expected manned for the intended purpose
- **Attestation** : It can be prove that it does what you think it does
 - Attest there is secure hardware
 - Attest the state of the OS
 - Attest state of the code
 - **Secure boot**
- **Sealing** : It can store secrets in unprotected memory
 - The device derives a key that is tied to its current status and stores the encrypted data
 - Data can only be decrypted by a device with the same status
- **Isolation** : It is not possible to *peek* inside
 - Requires protection against side channel attacks
 - Trusted hardware offers one well identified entry-point to interact with the software
 - **Tamper resistance** : hard to open
 - **Tamper evident** : You can see if it has been opened
 - **Tamper responsive** : Delete keys when attacked
 - **Resitance to side channel attacks** and physical probing
- **Trusted Execution Environments** (TPM) : Isolated processing environment in which applications can be securely executed irrespective of the rest of the application
 - Dedicated devices : Strong physical protections
 - Secure enclaves : Prtected regions of memory
 - Enable processes to run while being protected from attacks perpetrated by the OS, hypervisor, firmware, drivers, or remote attackers
- **Hardware Secure Module** (HMS)
 - The user need to know the the public key of the HMS
- **Non-volatile Storage**
 - **Endorsement key** (EK)
 - * Created at manufacturing time
 - * Signed by manufacturer
 - * Cannot be changed
 - * Used for *attestation*
 - **Storage Root Key** (SRK)
 - * Used for encrypted storage
 - **OwnerPassword**
 - They are private and never leave the TPM

- **Platform Configuration registers** (PCR lol)
- **Side channels** : Determine the secret key of a cryptographic device by measuring its execution time, its power consumption, or its electromagnetic field
 - Learn how the system's secret by observing how different computations are
 - Difficult to create trusted hardware resistant to side channel
- For trusted hardware we need to **trust the manufacturer**

Privacy

- **Multi-disciplinary** : computer science, law, ethics, economics, sociology, politics
- **Personal Data** Any kind of information (a single piece of information or a set of information) that can personally identify an individual
- Privacy is **not** hiding the wrong
- Lack of privacy is equivalent to **loss of freedom**
- Main risk : People's mind manipulation
- **Function creep** : expansion of a process or system where data collected for one specific purpose is subsequently used for another unintended or unauthorized purpose
 - Aadhaar - India's "optional" unique Identity identification number scheme
 - Eurodac - Fingerprint database for asylum seekers
- Security and privacy are **not opposite ends** of a seesaw
 - There is no security without privacy (and vice-versa)
 - Liberty requires both security and privacy
- **Privacy by design**
 - **Proactive** not reactive : preventive not remedial
 - Privacy as the **Default**
 - Privacy **Embedded** into Design
 - Full functionality : **Positive sum**
 - End to end **Security** : Full Lifecycle protection
 - Visibility and transparency : **Keep it open**
 - Respect for user privacy : Keep us **User-Centric**
- Technical approaches to privacy : **privacy enhancing technologies** (PETs)

Definition and classification

- Privacy paradigms : privacy as
 - **Confidentiality**
 - * Minimize data disclosure : every bit counts
 - * Distribute trust : avoid single point of failure
 - * Rely / require open source : million eyes help security
 - **Control**
 - * User participation : let the user decide how data will be shared
 - * Transparency and Accountability : let the user know how data is used, and if against his will, point to who is responsible
 - * Organizational compliance : General Data Protection Regulation (GDPR), Fair Information Practice Principles (FIPPs)
 - **Practice**
 - * Improve user agency : help them negotiate privacy
 - * Aid decision making and transparency impact : helps user understand the consequences of their actions
 - * Privacy as a collective practice : help identify best practices for collectives
- Pets for **Social privacy**
 - *Concerns* : The privacy problem is defined by Users
 - *Goals* : Do not surprise the user
 - * Support decision making

- * Help identify actions impact
 - *Limitations* : Only protects from other users : **trusted service provider**
 - * Limited by user’s capabilities to understand policies
- Pets for **institutional privacy**
 - *Concerns* : The privacy problem is defined by **Legislation**
 - * Data should not be collected without user consent or processed for illegitimate uses
 - * Data should be secured : correct, integrity, deletion
 - *Goals* : Compliance with data protection principles
 - * Informed consent : Valid, freely given, specific, informed and active consent
 - * Purpose limitation : Data can only be used for the purpose it was collected
 - * Data minimization : One should only collect the data necessary for the purpose of the service
 - * Subject access rights : One should be able to know what information is stored/processed and how. Also right to modification, deletion, etc.
 - *Limitations* :
 - * Assumes collection and processing by organizations is necessary, organizations are (semi)-trusted and honest
 - * Focuses on limiting misuses, not collection
- Pets for **Anti surveillance privacy**
 - *Concerns* : How to evade / Fool a global adversary
 - *Goals* : Minimize the need to trust other and the amount of revealed information
 - *Limitations* :
 - * Privacy-preserving designs are narrow - difficult to create “general purpose privacy”
 - * Usability problems both for developers and users
 - * Lack of incentives

Crypto based solution

- **Anonymous communication** : Anonymity of participants is usually achieved by special **routing overlay network** that hide the physical location of each node from other participants
- **Anonymous Credentials** : Allow users to authenticate themselves in a privacy preserving manner
- **Blind signature**
 - Content of a message is blinded before it is signed
 - Resulting blind signature can be publicly verified against the original message
 - Cryptographic voting systems
 - * Authority checks the credentials of the voter to ensure that he is allowed to vote, and that he is not submitting more than one vote
 - * Authority does not learn the voter’s selection
- **Secure Multiparty Computation**
- **Garbled Circuits**
- **Deterministic Encryption**
 - Always produces the same ciphertext for a given plaintext and key
- **Homomorphic encryption**
 - Allows specific types of computations to be carried out on ciphertext
 - Paillier cryptosystem
- **Private Information Retrieval**
 - Allows a user to retrieve an item from a server in possession of a database without revealing which item is retrieved
- **Oblivious RAM**
 - Same as PIR, but with R/W
 - * Client outsources the storage of his data to a cloud
 - * Client stores only a small amount of data locally
 - * Client accesses (read/write) his data while hiding the identities of the items being accessed

Non crypto based solutions

- **Confidentiality**
- **Pseudonymity**
 - User persistent (random) identifiers
 - Use of hashed identifiers
 - Different email addresses for the same user
 - Nicknames
 - Pseudo identifiers, Quase-identifiers
- **De identification**
 - Removing or obscuring information from traces that would allow direct identification of a person
 - Allows research that would otherwise not be possible due to privacy
- **Anonymity** : The state of being not identifiable within a set of object
- **Unlinkability** : Two or more items within a system, are no more and no less related than they are related based on the a-priori knowledge
- **Unobservability** : An item of interest being indistinguishable from any item of interest at all. Sender unobservability means that it is not noticeable whether any sender within the unobservability set sends
- **Plausible Deniability**
 - Not possible to prove user knows, has done or has said something
- **PETs** depend on :
 - The privacy paradigm : Confidentiality, control and practice
 - The adversary model, other users, semi-trusted service provider, everyone

PETs for data anonymization

- *Scenario* : You have a set of data that contains personal data and you like to anonymize it to
 - not be subject to data protection while processing
 - make it public for profit
 - make it public for researchers
- *Goal* : Produce a dataset that **preserves the utility** of the original dataset **without leaking informations** about individuals. This process is known as **database sanitization**
- ***k*-anonymity**
 - Key Attribute / **Identifier**
 - **Quasi identifier**
 - **Sensitive attribute**
 - Each person contained in the database cannot be distinguished from at least $k - 1$ other individuals whose information also appears in the released database
 - **Generalization** : Replace attribute with less specific, but semantically consistent values (e.g. zip-code)
 - To improve anonymity identifying attributes can be suppressed
 - Does not provide privacy when sensitive values lack of diversity
 - Limitation if the adversary has background knowledge
- ***l*-diversity** : An equivalence class has *l*-diversity if there are at least *l* well represented values for the sensitive attribute
- ***t*-closeness** : An equivalence class has *t*-closeness if the distance between the distribution of a sensitive attribute in this class and the distribution of the attribute in the whole table is no more than a threshold *t*
- Anonymizing a dataset via generalization and suppression is extremely hard
 - The *k*-anonymity idea focuses on transformation of the dataset not its semantics
 - Achieving *k*-anonymity, *l*-diversity, *t*-closeness is hard and still does not guarantee privacy
- **Modifying outputs** :
 - **Subsampling** : A subset of the rows is chosen at random and released and statistics are computed on the subsample
 - **Input perturbation** : Data or queries are modified before a response is generated
 - **Adding random noise to the output**

- **Randomized response**
 - * Respondents a query flip a coin and, based on the outcome, they either honestly respond or respond randomly
 - * Privacy comes from the uncertainty of how to interpret a reported individual value
 - * Yet, data can be useful because randomness can be average out
 - * Not usable for every case, or combined with other techniques
- **Differential privacy**
 - To have any utility we must allow the leakage of some information, but we can set a bound to the extent of leakage
 - Output is similar whether any single individual's record is included in the database or not
 - Instead of the real answer to a query, output a random answer such that by a small change in the database, the distribution of the answer does not change much
- To ensure differential privacy either
 - **Input perturbation** : Add noise to the database
 - * Independent of the algorithm and easy to reproduce
 - * determining the amount of required noise is difficult
 - **Output perturbation** : Add noise to the function output
 - * Easier to control privacy and better guaranttes than input perturbation
 - * Results cannot be reproduced
 - **Algorithm perturbation** : Inherently add noise to the algo
 - * Algorithm can be optimized with the noise addition
 - * Difficult to generelaze and depends on the input
- **Traditional Encryption**
 - Protects data at rest and in transit
 - Cannot protect computation
- **Homomorphic Ecnryption**
 - Protects computations on untrusted environments
 - Limited versatility vs efficiency
- **Secure Multiparty Computation**
 - Protects computation in distributed environments
 - High communication overhead
- **Trusted Execution Environments**
 - Protects computation with Hardware trusted element
 - Requires trust in the manufacturer, vulnerable to side-channels
- **Differential Privacy**
 - Protects released data from inferences
 - Degrades data utility
- **Distributed ledger technologies (Blockchain)**
 - Strong accountability and traceability in distributed environments
 - Usually no data privacy
- **Attribute based credentials**
 - Digital variant of passport, drivers's license etc
 - Also known as anonymous credentials
 - Attributes are encoded as number, may represent
 - * Membership status
 - * Name
 - * Age
 - * Social security number
 - * Random identifiers
 - * Application specific identifiers
 - **Unforgeability** : Only the issuer should be able to produce valid credentials
 - **Selective disclosure** : The uer can hide irrelevant attributes
 - **Issuer unlinkability** : The issuer should not be able to recognize a credential that it previously issued

- **Verifier unlinkability** : The verifier should not be able to link two consecutive showings of the same credentials
- **Zero knowledge proof** : allows a *prover* to convince a *verifier* of some fact on a private input without revealing this input
 - **Completeness** : If the statement is true, an honest prover can convince an honest verifier that the statement is true
 - **Soundness** : If the statement is false, a cheating prover cannot convince an honest verifier with very high probability
 - **Zero-knowledge** : If the statement is true, no verifier learns anything other than the fact that the statement is true

Machine learning

- **Supervised**
 - Labeled data
 - Direct feedback
 - Predict outcome/future
- **Unsupervised**
 - No labels
 - NO feedback
 - “Find the structure”
- **Reinforcement**
 - Decision process
 - Reward system
 - Learn series of actions
- **Confidentiality** of the model itself (e.g. intellectual property)
- **Privacy** of the training or test data (e.g. medical records)
- **Integrity** of the predications
- **Availability** of the system deploying machine learning
- **BLack box attack**
 - Model architecture and parameters unknown
 - Can only interact blindly with the model
- **Grey box attack**
 - Model architecture known, parameters unknown
 - Can only interact with the model, but has information about the type of model
- **White box attack**
 - Known architecture and parameters
 - Can replicate the model and use the model’s internal parameters in the attack
- If a linear model uses d features, the adversary needs $d + 1$ different queries to steal by solving the linear system for w, b

$$w \cdot x^{(i)} + b = f(x^{(i)})$$

- **Retraining attack** : Observe many queries, and fit the model on it like any other training data. Takes many queries
- Preserving model stealing
 - **Output perturbations** : Add noise to the probabilities output by the model to hinder reconstruction, but not accuracy
 - **Detect suspicious queries** : Identify deviations from expected on distribution of successive queries from a client
- Privacy and utility are not in conflict
 - Overfitted models leak training data
 - Overfitted models lack predictive power
- **ML needs data to learn**
 - Machine learning is based on data to find features and train the model

- Data is highly unique ! Allow many inferences
 - * Anonymizing may not work
 - * Aggregation affects utility and requires careful evaluation)
- Hide data
 - * Noise \Rightarrow Differential privacy
 - * Encryption \Rightarrow Homomorphic encryption, secure multiparty computation
- **To obtain value you must give data**
 - To use the model you need to provide a sample
 - * If the model is remote you are giving this sample out
 - We can do *privacy preserving model evaluation*
 - * Predict on noisy data : utility hit
 - * Use advanced cryptography : performance hit
- **The output reveals information**
 - Membership inference
 - * Given the answer of the classifier, infer whether the queried example was used in training
 - * Attribute inference
 - * Given the answer of the classifier, infer whether a training sample had a particular attribute
- **Machine learning is very good at inferring**
 - Use new learning to classify/predict on new data
 - * The ML model can be used to breach privacy of that new data
- **Adversarial Examples** : Inputs to a model that an attacker has designed to cause the model to make a mistake
- **Transferability property** : Samples crafted to mislead a model A are likely to mislead a model B
- Defending in general is very hard. Can only defend against a particular threat model, and normally no guarantee
- Standard way is **adversarial training** (based on robust optimization). I means training on simulated adversarial example
- If the adversary controls the inputs
 - In deployed models, the adversary can always win
 - In training, the adversary learn other's input
- **Bias Reinforcement** : Action \Rightarrow Effect \Rightarrow Action
- **Statistical bias** : Difference between an estimator expected value and the true value
- **Group fairness** Outcome should not differ between demographic groups
 - Predictive parity : Same prediction regardless of group
 - Equal false positive
 - Equal false negative
- **Individual fairness** : similar ? individuals should be treated similarly ?
- **Bias detection and mitigation**
 - What if approach : play with the value until something changes, associate with bias
 - Explainability : Try to understand why the prediction happen, associate with bias
 - Mitigation
- Instead of sending their data directly, clients send data with differentially private noise
 - Given the sample one cannot learn the value
 - Challenge : add enough noise to hide the data but still provide a good model
- **Federated learning** : combine many small dataset to get large dataset
 - Clients need to reveal their models, that “reveal” their data. Solution ? :
 - * Homomorphic encryption
 - * Multi party computation
 - * Before sending models, add noise (tradeoff privacy vs functionality)
- **Fully centralized**
 - Transfer raw data to a central database
 - Data protection : security of the central database
 - Need to trust the central server
- **Meta-analysis**

- For each study, aggregated data provided by each site
- Still need to trust the central server
- **Decentralized**
 - Send the algorithm to the data
- **Privacy preserving distributed machine learning**
 - The querier defines the query, e.g. training of an ML model
 - Each data provider performs several training iterations on its data
 - The DP's collectively and iteratively combine their encrypted local model in a global model
 - After the training and based on a pre-agreement, the model is either :
 - * kept secret for oblivious predications
 - * revealed to the querier
- **Gradient Descent**
 - Non polynomial activations functions
 - * Sol : Least square **approximation** of activation function
 - Heavy homomorphic operations
 - * Sol : Problem specific **packing schemes** to enable Single instruction, Multiple Data
 - Model specific functions
 - * Sol : Introduce several functions **distributed bootstrapping**
- **Bootstrapping** : The model is persistent among multiple iterations → large multiplicative depth → ciphertext need to be bootstrapped
 - Sol : Efficient and collaborative distributed bootstrapping and minimizing the number of bootstraps via parametrization
- **Parametrization** : Tight link between learning parameters and cryptographic parameters
 - Sol : **A constrained optimization problem** for choosing the cryptographic parameters

Blockchain

- **Data structure** : *linked list* with specific properties
- It is a *distributed* database of *record* of all events that have been executed and shared among participating *parties*
- Each **block** except the first one contains the hash of the previous block
- Blocks store cryptographically secure information (**validated by nodes**)
- Each block contains :
 - A **Cryptographic hash** of the previous block
 - A **timestamp**
 - **Data**
- Purpose of blockchain : **Removing the trusted third party**
- **Transparency** : Each participant has a copy of the current blockchain data
- **Consensus** : All network participants must agree that an event added to the chain is valid
- **Transaction Content**
 - **Assets**
 - * The currency of the chain
 - * Blockchain imposes sum of all assets to be constant
 - **Smart contract**
 - * Small programs that work on the data in the ledger
 - * Allow to extend the functionality of the blockchain
 - * Are enforced by the consensus of the nodes
 - **Tokens**
 - * Digital representation of (physical world) objects
 - * Smart contracts define how tokens can be exchanged
- **Node governance**
 - **Proof of Authority (PoA)**
 - * A fixed set of nodes decide on consensus
 - * Updating this set often follows off-chain rules

- **Proof of Work (PoW)**
 - * Lottery - the first node to solve a cryptographic puzzle proposes the next block and gets a reward
 - * Everybody can joined
 - * Huge waste of energy
- **Proof of stake (PoS)**
 - * Nodes invest a stake to be allowed to propose blocks and gets rewards
 - * The stake can be lost if the node misbehaves
 - * Concentration of stake
- **Proof of Personhood (PoP)**
 - * Special case of PoS, where each person has the same power to stake and get rewards
 - * Experimental, socialist - universal basic income
- **Scaling out**
 - **Sharding**
 - * Create groups of nodes that each handle a part of the transactions
 - * Increase speed, but potential security problems
 - * Shared Security, shards are verified by a central chain
 - **Side Chains**
 - * Independant chain that is loosely tied to the main chain
 - * Increase speed, but decrease security
- **Permissioned Ledgers**
 - Also called **Permissioned blockchains**
 - Just decide *administratively* who participates; Fixed or manually changed group of trustees
 - *Liability clearly defined*
 - No proof of work → low energy cost
 - More mature consensus protocols applicable
 - Higher human organizational costs
 - No onger open for anyone to participate
 - Strong potential for regulated sectors such as finance and health
- **Public (permissioned) vs Private (permissioned) blockchains**
 - Who is able to *write* the data in the blockchain
- **Open vs Closed blockchains**
 - Who is able to *read* the data in the blockchain
- **Blockchain abstraction**
 - **Strict orderign of messages**
 - **Rule based write, global read**
 - **No message modification**
- **Consensus properties**
 - **Termination** : Every correct process will eventually decide on some output
 - **Integrity** : If all correct processes proposed the same value, then any correct process must decide this value
 - **Agreement** : Every process must agree on the same value
- **Conflict resolutiona** : bitcoin solves this problem by having a *leader* elected every 10 minutes that states which transactions are valid
 - Elected via *proof of work*
- **Eclipse attack**
 - Adversary targets a **specific node** to cut off all of its communication with the other peers and thus isolate this specific node
 - A sucessful Eclipse Attack enables isolating the victim node and prevent the victim from attaining true pricture of the real network activity and the current ledger state
 - By isolating a lot of nodes the attacker can remove significant of *hash power* from the system
 - How to mitigate :
 - * Random node selection
 - * Fewer nodes per IP address or machine

- * information storage (storing informations about nodes)
 - * Larger number of connections
- **Sybil attack**
 - Type of attack seen in peer to peer network in which a node in the network operates under multiple identities
 - How to prevent it
 - * Bitcoin uses Proof of Work consensus algorithm to prove the authenticity of any block that added to the blockchain
- **Double spending attack**
 - A miner or a group of miners controls 51% or more of the mining power of the blockchain network
- Bitcoin : issues with scalability
- **Smart contract**
 - Contract that formalizes a relationship between parties and contains a set of promises made between them
 - Provides new way to formalize and secure digital relationship
- Ethereum and **Gas**
 - A unit that measures the amount of **computational effort** that will take to execute certain operations
 - Each operation is tagged with an explicit cost
 - Each transaction incurs a cost to the sender
 - Gas is the unit of all computational tasks in ethereum

Data protection for personalized health

- **Homer's attack**
 - Adversary has access to a known participant's genome
 - Goal : determine if the target individual is in the case group
 - Uses simple correlation in the genome
- **GA4GH Beacon project**
 - Allows researcher to quickly query multiple database to find the sample they need
 - Encourages cross-border collaboration among researchers
 - Only provides minimal response back in order to mitigate privacy concerns
- **Surname interface attack**
 - Goals :
 - * Recover the surname of sequence donors from 1000 Genome Project
 - * Triangulate the identity of a sequence donor using his surname, age and state
 - Using :
 - * Surnames are paternally inherited in most human societies
 - * Y-chromosome haplotypes in male individuals are directly inherited from the father
 - Surname interference
 - * Profile short tandem repeats on the Y chromosome
 - * Query recreational generic genealogy database
 - * Obtain a list of possible surnames for the sequence in question
 - Identity Triangulation
 - * Combine surnames with age and state
 - * Triangulate the identity of the target
- **Genomic data** pose special privacy problems
 - They are inherently identifying
 - They can't be changed
 - They have unique statistical regularities
 - They contain sensitive and personal information
 - Their leakage can expose individuals to generic discrimination
 - Relatives can also be affected
- **Genome Privacy**

- Require duration of protection » 1 century
- Data size around 300 GBytes / person
- Need sometimes to carry out computations on millions of patient records
- Noisy data
- Correlations
- Several “semi trusted” stakeholders
- Diversity of applications
- Pragmatic approach, **gradual** introduction of new protections tools
- Different **sensitive levels** of the data
- Different **access right**
- Exploit **existing** data
- Be **future proof**
- Awareness and enforcement of **patient consent**
- **Deterministic encryption**
 - Preserves and leaks equality if the plaintext
- **Probabilistic encryption**
 - Random salt added to each encryption to achieve semantic security