Introduction To Blockchain – Francesco Grossetti

Introduction – Class 01

What this course is about

- It's about blockchain technology and related applications
- Provides a very general introduction
- We will discuss the main components of a blockchain
- We will deal with some technological aspects
- We will have some guest speakers who will discuss real-cases
- Develop critical sense.

What this course is not about

- Cryptocurrencies
- Hacking systems
- Make money through cryptocurrencies
- Mathematical aspects of cryptography

The Ledger

A ledger is a container which gathers every business transaction occurring for a given account. It generally contains the date and amount of the transaction. There are several ledgers, most notably, the Accounts Receivable Ledger and the Accounts Payable Ledger. The concept of ledger goes back to 1494 when mathematician Luca Pacioli first invented the double-entry bookkeeping accounting system. Plenty of accounts: assets, liabilities, capital, income, expenses…
This system still functions today. The accounts are reported in the so called *balance sheet* and *income statement* (cash flows are reported in the CFS). There are several ledger formats, most notably the T-account and the three- or four-column ledger.

Two concepts are fundamental to understand the Blockchain:

1. Thinking in Layers
2. Thinking about Blockchain as **software architecture**.

1. The metaphor: do you have a mobile phone?
    - How much do you know about the different wireless communication protocols that are used to send and receive data?
    - How much do you know about electromagnetic waves that are the foundation of mobile communication?
    - How can use your phone then?

    We use this approach all the time when we learn how to use a new technology. There is a problem though: these mental partitions are **highly individual**. This typically leads to **problems in communication**. Unifying the way of partitioning a system is the key point when discussing technology.

Application vs. Implementation Layers

Separating the user's needs from the technical internals of a system leads to a clear separation of the **application layer** from the **implementation layer**. Everything that belongs to the application layer is concerned with the user's needs (e.g., listening to music, taking photos, or booking hotel rooms). Everything that belongs to the implementation layer is concerned with making these things happen (e.g., converting digital information into acoustic signals, recognizing the color of a pixel in a digital camera, or sending messages over the Internet to a booking system). Elements of the implementation layer are technical by nature and are considered a mean to an end.

Functional vs. Nonfunctional Layers
Distinguishing between what a system does and how it does what is does leads to the separation of **functional** and **nonfunctional** aspects. Examples of functional aspects are sending data over a network, playing music, taking photos, and manipulating individual pixels of a picture. Examples of nonfunctional aspects are a beautiful graphical user interface, fast-running software, and an ability to keep user data private and safe. Other important nonfunctional aspects of a system are **security** and **integrity**. Integrity means that a **system behaves as intended**, and it involves many aspects such as security and correctness.

Let's layer the mobile phone…

Layer	Functional Aspects	Nonfunctional Aspects
Application	Taking photos	The graphical user interface looks beautiful
	Making phone calls	Easy to use
	Sending e-mails	Messages are sent fast
	Browsing the Internet	
	Sending chat messages	
Implementation	Saving user data internally	Store data efficiently
	Making a connection to the nearest mobile connector	Saving energy
		Maintaining integrity
	Accessing pixels in the digital camera	Ensure user privacy

Functional aspects of the application layer serve obvious needs of the users. These elements are typically the ones users learn about. On the other hand, the nonfunctional aspects of the implementation layer are rarely seen as major elements of the system.

Integrity
Three components:

- **Data Integrity**: The data used and maintained by the system are complete, correct, and free of contradictions.
- **Behavioral Integrity**: The system behaves as intended and it is free of logical errors.
- **Security**: The system is able to restrict access to its data and functionality to authorized users only.

Everybody uses software everyday with great success. Everybody is pretty happy about it. We may change our feelings quite drastically the minute our interaction with the software fails or the software itself fails. On these occasions, we begin to realize that software integrity is a highly valuable commodity. Hence, it should not come as a surprise that software professionals spend a lot of their time working on this seemingly tiny nonfunctional aspect of their implementation layer.

Blockchain as a Software Architecture
The metaphor: Have you ever bought a car? Cars are equipped with different types of engines (e.g., diesel, gasoline, or electric engine). This is an example of the process of **modularization**, which is the result of applying the idea of layering to cars. Two cars that look identical from the outside can differ dramatically with respect to the power of their engines and hence have very different driving performance. Additionally, your choice of the engine will have an impact on other characteristics of the car, like its price, its operational costs, the type of fuel consumed, the exhaust system, and the dimensions of the brakes.

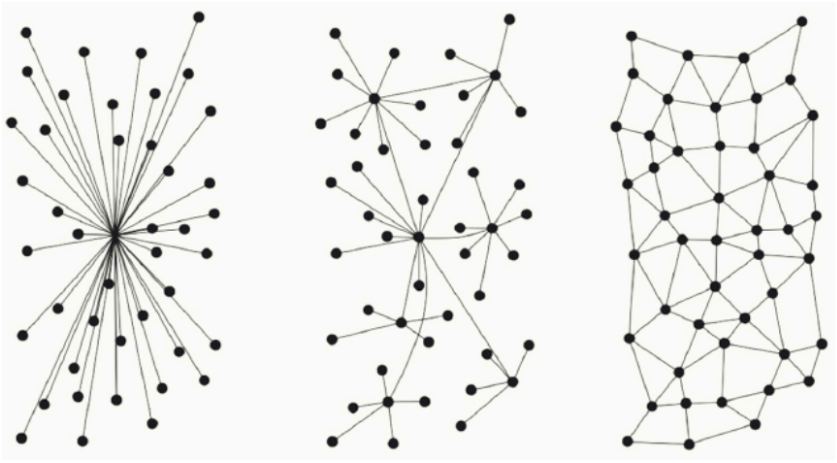
Layering a Payment System

Layer	Functional Aspects	Nonfunctional Aspects
Application	Deposit money	The graphical user interface looks beautiful
	Withdraw money	Easy to use
	Transfer money	Transfer of money is done fast
	Monitor account balance	System has many participants
Implementation	?	Available 24 hours a day
		Fraud resistant
		Maintaining integrity
		Ensure user privacy

Why do we have a question mark? That's the "engine" of our system. In other words, it's the architecture of the system.

Hints on Software Architectures

There exist dozens of ways to implement software systems. One of the fundamental decisions we must take regards its architecture. An architecture is the way components are organized and related to one another. There are three major approaches:



- **Centralized**
  - Components are located around and connected with one central component.
  - Circles represent node while lines represent connections between nodes.
- **Decentralized**
  - In decentralized systems, there are multiple central components.
  - Each central component is connected to another central component.
  - In this regards, a decentralized network is composed by multiple centralized networks.
- **Distributed**
  - Components are connected with one another without having a central element.
  - None of the components is directly connected with all other components.
  - Though, all components are connected with one another at least with one indirect connection.

Pros of a Distributed System

- **Higher computing power**:
  The computing power of a distributed system is the result of combining the computing power of all connected computers. Hence, distributed systems typically have more computing power than each individual computer. This has been proven true even when comparing distributed systems comprised of computers of relatively low computing power with isolated super computers.
- **Cost reduction**:
  The price of mainstream computers, memory, disk space, and networking equipment has fallen dramatically during the past 20 years. Since distributed systems consist of many computers, the initial costs of distributed systems are higher than the initial costs of individual computers. However, the costs of creating, maintaining, and operating a super-computer are still much higher than the costs of creating, maintaining, and operating a distributed system. This is particularly true since replacing individual computers of a distributed system can be done with no significant overall system impact.
- **Higher reliability**:
  The increased reliability of a distributed system is based on the fact that the whole network of computers can continue operating even when individual machines crash. A distributed system does not have a single point of failure. If one element fails, the remaining elements can take over. Hence, a single super-computer typically has a lower reliability than a distributed system.
- **Ability to grow naturally**:
  The computing power of a distributed system is the result of the aggregated computing power of its constituents. One can increase the computing power of the whole system by connecting additional computers with the system. As a result, the computing power of the whole system can be increased incrementally on a fine-grained scale. This supports the way in which the demand for computing power increases in many organizations. The incremental growth of distributed systems is in contrast to the growth of the computing power of individual computers. Individual computers provide identical power until they are replaced by a more powerful computer. This results in a discontinuous growth of computing power, which is only rarely appreciated by the consumers of computing services.

Cons of a Distributed System

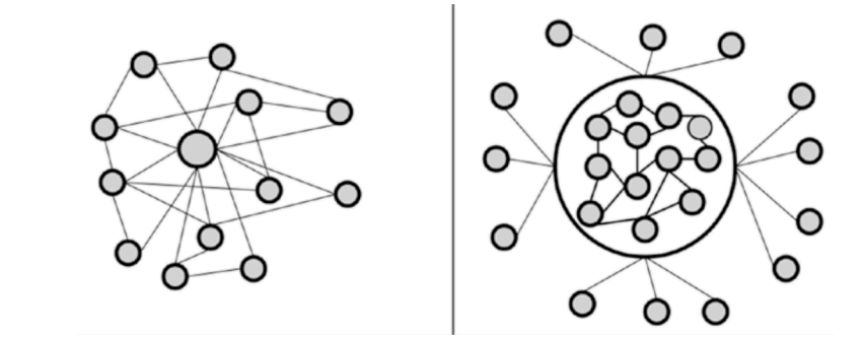
- **Coordination overhead**:
  Distributed systems do not have central entities that coordinate their members. Hence, the coordination must be done by the members of the system themselves. Coordinating work among coworkers in a distributed system is challenging and costs effort and computing power that cannot be spent on the genuine computing task, hence, the term coordination overhead.
- **Communication overhead**:
  Coordination requires communication. Hence, the computers that form a distributed system have to communicate with one another. This requires the existence of a communication protocol and the sending, receiving, and processing of messages, which in turn costs effort and computing power that cannot be spend on the genuine computing task, hence, the term communication overhead.
- **Dependency on networks**:
  Any kind of communication requires a medium. The medium is responsible for transferring information between the entities communicating with one another. Computers in distributed systems communicate by means of messages passed

through a network. Networks have their own challenges and adversities, which in turn impact the communication and coordination among computers that form a distributed system. However, without any network, there will be no distributed system, no communication, and therefore no coordination among the nodes, thus the dependency on networks.

- **Higher program complexity**:
  Solving a computation problem involves writing programs and software. Due to the disadvantages mentioned previously, any software in a distributed system has to solve additional problems such as coordination, communication, and utilizing of networks. This increases the complexity of the software.
- **Security issues**:
  Communication over a network means sending and sharing data that are critical for the genuine computing task. However, sending information through a network implies security concerns as untrustworthy entities may misuse the network in order to access and exploit information. Hence, any distributed system has to address security concerns. The less restricted the access to the network over which the distributed nodes communicate is, the higher the security concerns are for the distributed system.

Distributed Peer-to-Peer Systems (P2P)
P2P networks are a special kind of distributed systems. Each node shares its computing power over the network so that other nodes can exploit it. Each and every node has the exact same rights and roles in the system. All the nodes are both suppliers and consumers of resources. Lots of applications: file sharing, content distribution, privacy protection.

What about hybrid architectures?
There are pros and cons both for centralized and distributed networks. What if we combine them in an hybrid shape? These are just two examples of typical blockchain systems.



How can we identify a distributed system?
The increasing diffusion of hybrid systems makes it hard to clearly and uniquely identify distributed systems. It is really hard to come up with a generally accepted definition of distributed system. Here is a trick: If you are in doubt whether or not a system is distributed, look for a single component (e.g., a database, a name or user registry, a login or logoff component, or an emergency switch-off button) that could terminate the whole system. If you find such a component, the system under consideration is not distributed.

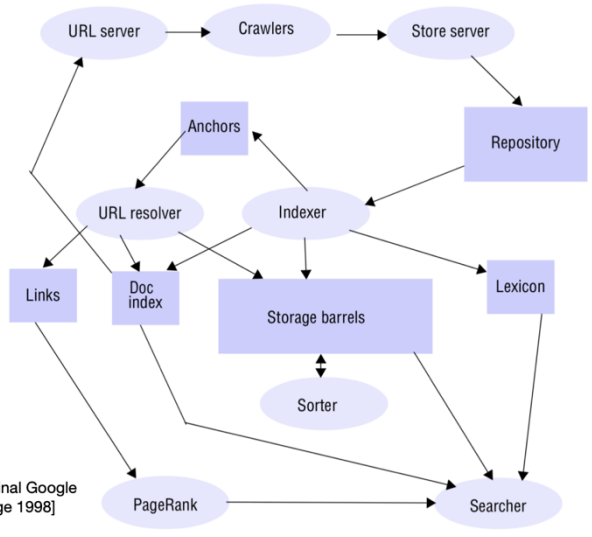
Designing Distributed Systems - Class 02

Designing Distributed Systems – Google Case Study

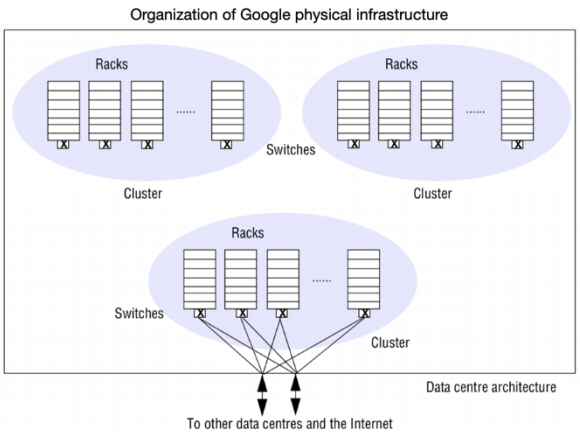
Google is a US-based internet company. Born as a research project at Stanford in 1998. Now it has a dominant share of the Internet search market. Capable of diversifying itself: cloud computing. Amazing case study from distributed systems perspective: extremely demanding requirements, particularly in terms of scalability, reliability, performance and openness.

Google Hetereogeneous Business

Gmail	Mail system with messages hosted by Google but desktop-like message management.
Google Docs	Web-based office suite supporting shared editing of documents held on Google servers.
Google Sites	Wiki-like web sites with shared editing facilities.
Google Talk	Supports instant text messaging and Voice over IP.
Google Calendar	Web-based calendar with all data hosted on Google servers.
Google Wave	Collaboration tool integrating email, instant messaging, wikis and social networks.
Google News	Fully automated news aggregator site.
Google Maps	Scalable web-based world map including high-resolution imagery and unlimited user-generated overlays.
Google Earth	Scalable near-3D view of the globe with unlimited user-generated overlays.
Google App Engine	Google distributed infrastructure made available to outside parties as a service (platform as a service).



Outline architecture of the original Google search engine [Brin and Page 1998]



1 PC x 2 TB
80 PC x Rack = 160 TB
1 Rack x 30 = 4.8 PB
Google has around 200 clusters —> 960 PB ~ 1 Exabyte / 1 billion GB (2010)
In 2013 it's been estimated a whopping 5 Exabytes

Overall System Architecture

- **Scalability**, need to scale up to Ultra-Large Scale distributed system. Google views the problem in 3D:
  - Being able to deal with more data
  - Being able to deal with more queries
  - Seeking better results (more accurate)

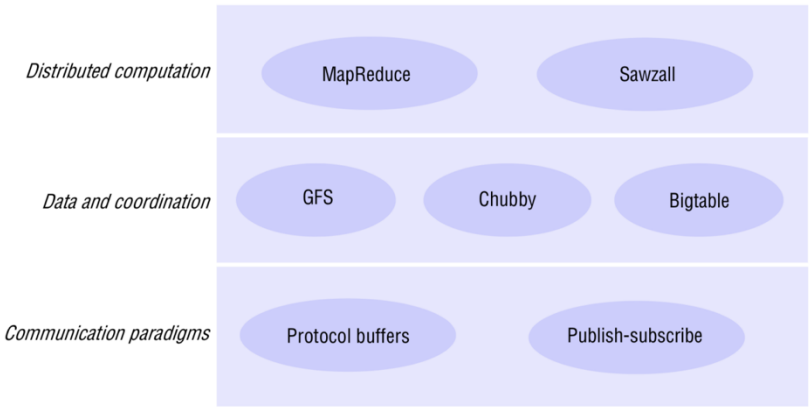
- **Reliability**, Google has stringent reliability requirements, especially with regard to availability of services. This demands both detecting failures and adopting strategies to

mask or tolerate such failures. Such strategies rely heavily on the redundancy in the underlying physical architecture.

- **Performance**, keen on achieving low latency of user interactions.
  - The importance of performance is exemplified by the target of completing web search operations in 0.2 seconds.
  - This applies to a wide range of functions associated with the operation of Google, including web crawling, indexing and sorting.
  - Top performance requires resources to work together, including network, storage and computational resources.

- **Openness**, strong requirement for openness, particularly to support further development in the range of web applications on offer.
  - It is well known that Google as an organization encourages and nurtures innovation, and this is most evident in the development of new web applications.
  - This is only possible with an infrastructure that is extensible and provides support for the development of new applications.

Google Infrastructure

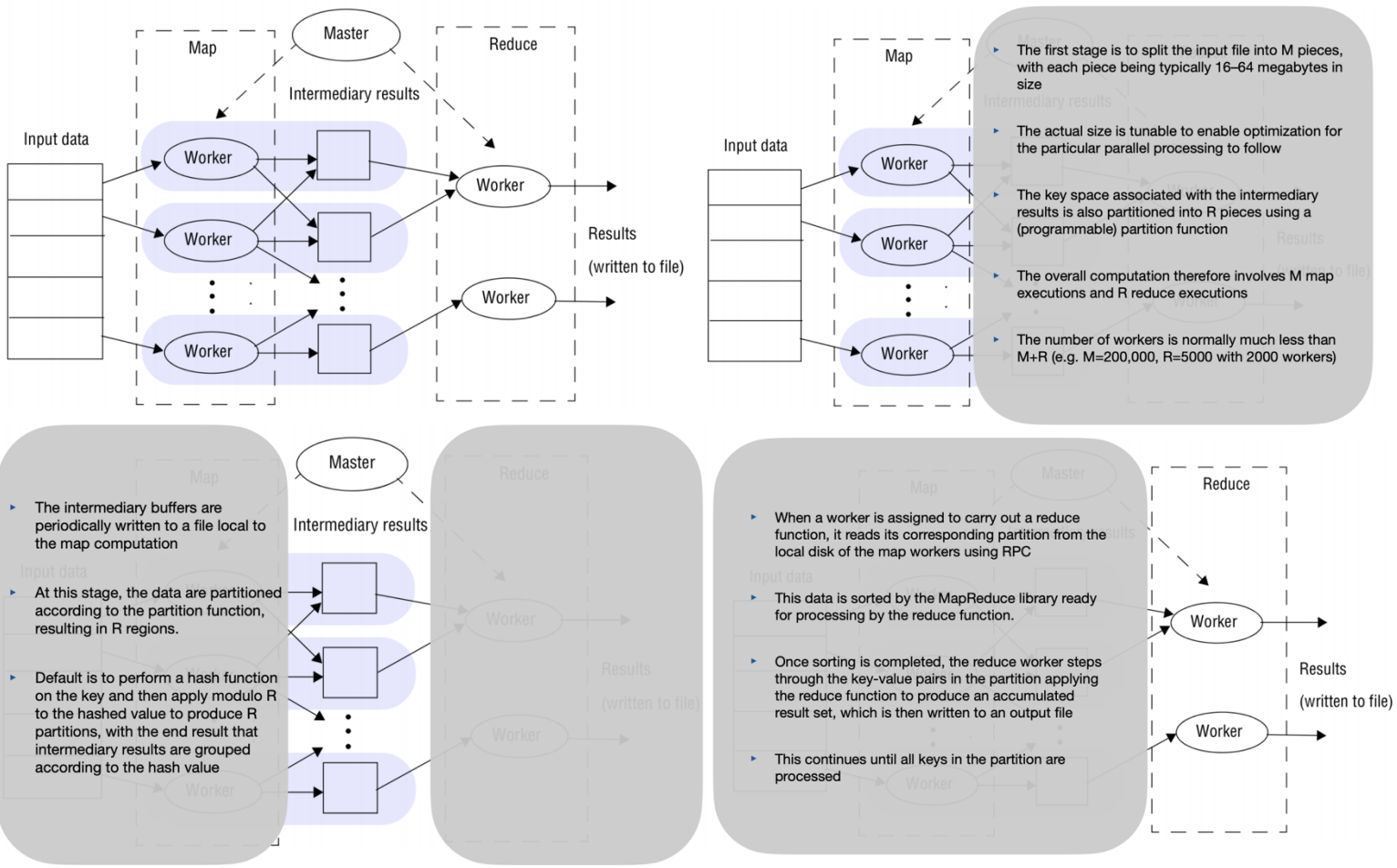


Hints on distributed computing: MapReduce

To complement the storage and coordination services, it is also important to support high-performance distributed computation over the large datasets. This is the main goal of the paradigm MapReduce. There are three key principles:

- Break the input data into a number of chunks. (Map)
- Carry out initial processing on these chunks of data to produce intermediary results. (Map)
- Combine the intermediary results to produce the final output. (Reduce)

Function	*Initial step*	*Map phase*	*Intermediate step*	*Reduce phase*
Word count	*Partition data into fixed-size chunks for processing*	For each occurrence of word in data partition, emit <*word*, 1>	*Merge/sort all key-value keys according to their intermediary key*	For each word in the intermediary set, count the number of 1s
Grep		Output a line if it matches a given pattern		Null
Sort *N.B. This relies heavily on the intermediate step*		For each entry in the input data, output the key-value pairs to be sorted		Null
Inverted index		Parse the associated documents and output a <*word*, document ID> pair wherever that word exists		For each word, produce a list of (sorted) document IDs



The Purpose of the Blockchain

The metaphor: Can you remember the last time you bought a CD for yourself in a music store? Well, maybe not so much since nowadays people tend to download music or simply stream it (e.g., Spotify, Apple Music, …). The real game changer was just a piece of software called Napster (released in 1999) which allowed to share music files among peers. "This system, what's most interesting about it is, you're interacting with peers, you're exchanging information with a person down the street". (Shawn Fanning, cofounder of Napster).

A Revolutionary Change – the Case of Music Industry

The music industry has worked for a long time in the following way:

- Musicians made contracts with studios.
- The studios record the songs, produce and market them on a variety of media (e.g., vinyl, tape, or CD).
- These were sold to the customers via a variety of distribution channels, including department stores and specialized shops.
- The studios actually worked as **intermediaries** between musicians and people who enjoy listening to music.
- The studios could maintain their **role as intermediaries due to their exclusive knowledge and skills** in producing, marketing, and distributing records.
- This is true till the first decade of the 2000s.

In early 2000s, lots of things changed:

- The digitalization of music, the availability of recording equipment at affordable prices, the growing spread of privately used PCs, and the wide spread of the Internet made music studio dispensable.

- The three functions of music studios – producing, marketing, and distributing records – could be done by the artists and the consumers themselves.
- Napster played a major role in the replacement of the music studios as intermediaries.
- The P2P approach of Napster gave consumers access to a wider range of music than ever before, making the music studios far less relevant and causing them significant losses.

Say bye bye to intermediaries
The idea here is simple: a P2P system has the power to completely replace the middlemen with P2P interactions. In the case of the music industry, the studios and their marketing and distribution channels that acted as the middlemen between artists and consumers have been replaced by P2P file sharing systems. The major characteristics that made the music industry so vulnerable to being replaced by P2P systems are the immaterial nature of music and the low costs of copying and transferring data.

What is the potential of a P2P system?
The power of peer-to-peer systems is not restricted to the music industry. Each industry that mainly acts as a middleman between producers and customers of immaterial or digital goods and services is vulnerable to being replaced by a peer-to-peer system. An example: the financial industry.

Towards the Full Digitalization?
What is it that you have in your bank account or on your credit card?
The concept of digitalization has been around for a long time. Only a small amount of actual money and assets do exist as physical entities like banknotes and coins. All the rest is in the form of immaterial information, namely bits and bytes, in the centralized information technology systems. Actors like banks are just middlemen between procedures and consumers of those bits and bytes. Each transaction being borrowing, lending or transferring money from one account to another is the **transfer of an immaterial good**. This is operated by middlemen or intermediaries. The world is full of intermediaries: a simple money transfer across countries involves several intermediaries with a long processing time. Of course, this increases the transaction costs.

So why choosing a P2P system?
In a P2P system, the same transfer is much simpler, way faster and would cost less. The transaction is just a **transfer of bits and bytes between two nodes**, respectively. Again, no middlemen required: interactions occur between contractual partners. The replacement of intermediaries is called **disintermediation**. **Note: disintermediation is considered a serious threat to many business**.

Wrapping up the concept of P2P system
P2P systems are distributed software systems which consist of nodes which make their computational resources directly available to other nodes. Each node has equal rights and roles even if nodes have different resources. Each node is both a supplier and a consumer of resources. P2P systems are generally distributed, but can come in the form of a hybrid architecture.

Linking P2P system with Blockchain
Remember the concept of integrity? A system behaves as intended. Purely distributed P2P systems may use blockchain to achieve and to maintain system integrity. Here is the take out:
"Blockchain is the technology used to achieve and maintain integrity in purely distributed systems".

So why all this hype about blockchain?

Purely distributed P2P systems have commercial potential (e.g., think about the music industry). Purely distribute **P2P systems use blockchain to achieve and maintain integrity**, which is a fundamental requirement for such systems. Plus: **blockchain enables the disintermediation**.

Integrity is not the only word…
The main purpose of the blockchain is to maintain integrity. Why maintaining integrity in distributed systems and purely distributed P2P systems in particular is such a challenge? Here comes another keyword: **trust**. There exists a subtle relation between integrity and trust.
The metaphor: have you ever heard the expression "herding cats"? This illustrates the challenges of herding a group of obstinate and intractable animals (hums?) that do not accept or recognize a central authority. This is what happens in purely distributed P2P systems in which individual and independent nodes have no central control or coordination.

Trust and Integrity in P2P systems
We know what integrity is: a nonfunctional aspect of a system to be safe, consistent, correct and free of corruption and errors. **Trust is the human belief in the reliability and truth of someone or something without a proof or a further investigation**. In a Bayesian flavor, trust is given a priori and then gets updated based on the results of interactions. In a P2P system, **integrity is a necessary condition** to fulfill the expectations of the users and thus reinforce their trust in the system. **Whenever trust is not reinforced due to a lack of integrity, users will abandon the system**. How do we achieve and maintain integrity in such systems? This depends on several factors like:

- Knowledge about the number of nodes
- Knowledge about the trustworthiness of the nodes.

If both of theme are known, the chances of achieving integrity are higher. Note: running a P2P system over the Internet that is open and public is challenging because neither the number of nodes nor the trustworthiness of them are known.

Integrity Threats in P2P systems
Two main categories:

- Technical failures (easy to fix, just replace the broken hardware/software).
- Deliberate attacks by malicious peers (way more complex to deal with).

There exist plenty of malicious attacks:

- Distributed Denial-of-Service (DoS): massive connections to a server making it inoperable.
- Network poisoning: injection of massive amount of useless data in the network. Also used as a fodder to DoS.
- Privacy breaches and Identity steal.

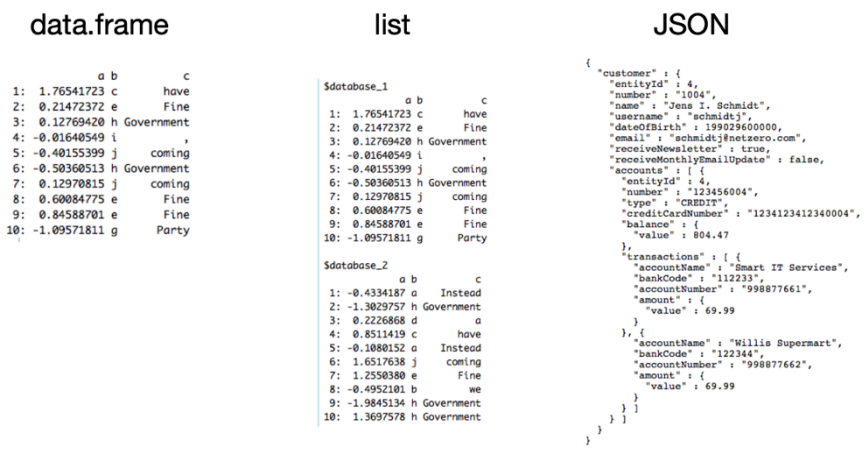
Blockchain as a problem solver
When all the conditions are met, reaching integrity as well as trust and maintain them is easy. What if none of those conditions are met? What if we have to face the worst-case scenario?
The blockchain comes into play in this kind of situation. The blockchain plays a major role when we have an **unknown number of peers with unknown reliability and trustworthiness**. This is a well known problem in computer science called the **Byzantine Generals' Problem**.

Ok then, but what is a blockchain?
At this point, it should be clear what is the purpose of a blockchain. Though, we still miss a formal definition of it. We can think of a blockchain in four different ways:

- As a Data Structure

- As an Algorithm
- As a Technology
- As a general term to identify purely distributed P2P systems within a common application area.

Blockchain as a Data Structure
In computer science and software engineering, a data structure is a way to organize data regardless of their concrete informational content.



Blockchain refers to data put together into units called **blocks**. Think of these blocks much like pages in a book: they are connected to one another like a **chain**. In relation to a book, the words and sentences are the information to be stored. They are written on different pages instead of being written on a large spool. The pages are connected with one another via their position in the book and via the page numbers. You can determine if someone removed a page from the book by checking whether the page numbers continue without leaving out a number. The information on the pages as well as the pages within the book are ordered. The ordering is an important detail, which will be used extensively. The chaining of the data blocks in the data structure is achieved by using a very special numbering system (more on that later).

Blockchain as an Algorithm
An algorithm is just a collection of information put together into a sequence which a computer can understand and execute. Instruction often involves data structures. When used as a name for an algorithm, blockchain refers to a sequence of instructions that negotiates the informational content of many blockchain-data-structures in a purely distributed peer-to-peer system.

Blockchain as a Technology and More
Seeing the blockchain as a technology means to view it in the big picture. The technology involve data structures that contain information as well as the algorithms needed to make something with that information. The technology also involves **cryptography** and **security tools**. Combined together they can be used to achieve integrity and trust in a purely distributed P2P system.

Combining everything we know
Let's try to come up with an intermediary definition of blockchain: **"The blockchain is a purely distributed peer-to-peer system of ledgers that utilizes a software unit that consist of an algorithm, which negotiates the informational content of ordered and connected blocks of data together with cryptographic and security technologies in order to achieve and maintain its integrity"**. First proposed in 2008 under the pseudonym of Satoshi Nakamoto with the paper: Bitcoin: a peer-topper electronic cash system. The definition does not talk about Bitcoin or any cryptocurrency. The blockchain has a wide and diverse range of applications.

Ownership - Class 03

How do you know what you really own?

The metaphor:

- You are going to work out to work and packing your bag lunch with an apple.
- On your way, you stop to a supermarket to buy a sandwich and cookies.
- At the checkout, the employee sees you have the exact same apple they are selling in your bag.
- How could you prove that you did not steal the apple?

Who Records Ownership?

- Imagine your cast gets stolen: what would you do?
- Go to the police department to log your complaint.
- If your car gets noticed with someone else, the police will verify the ownership via Automobile Registry.
- The thief will be put to trial.
- That's way too easy!

- What if someone captures your drone?
- How would you prove you own it?
- Did any authority record that you bought the drone?
- Your best bet would be the paper receipt that you received from the shopkeeper.
- Will they prove your ownership of the drone?

The old paper is not enough… Paper receipts are not unique!

- The thief can forge one too claiming that he owned the drone.
- Right now, there's no easy way to prove you own what you own.
- Even if there's a way, it'd be probably a slow and tedious process.
- You own something because others believe you own it. The thief doesn't.

A new hope?

- According to Economist, "estimates for the total value of fakes sold worldwide each year go as high as $1.8 trillion." – Economist, July 30th, 2015.
- Last year was about 3.3% of world trade.
- This leads to losses for brands.
- This also leads to **consumers losing their trust** in those brands.
- Imagine a world in which when someone goes out to buy something that must be authentic to have value, the shopper could simply pull out their phones, type in some number and verify if the shopkeeper is trying to dupe them.
- Just for a moment, imagine a world where each and every valuable thing could be assigned a number and ownership.

Ownership and Witnesses

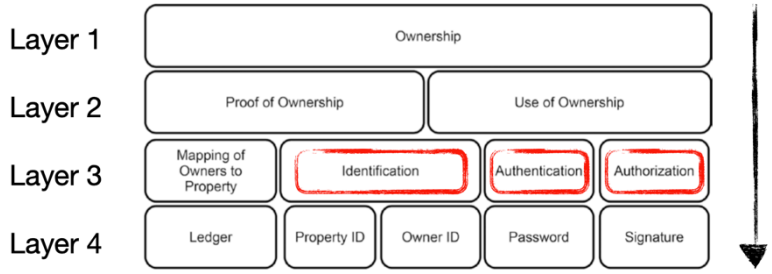
- You are in court disputing your alleged apple-theft case: how do you prove your innocence?
- Best case: you **find someone that could testify** you had your apple before the event.
- The employee who sold you the apple is having hard time with cross-examination: can you identify the exact apple you sold to the accused?
- It could even be the case you bribed the witness to remember all fancy details about the apple transaction.
- Here is the take out: **be sure to have many independent witnesses testifying the exact same thing**.
- To prove the ownership three elements are keys:

    - An identification of the owner.
    - An identification of the object being owned.
    - A mapping of the owner to the object.

- You can accomplish these through the testimony of witnesses.
- Though this is **time consuming** that's why we replaced lot of things with documents issued by trustworthy entities (namely governmental).
- A birth certificate does not change with time.

The mapping between owners and objects

- The mapping step can be achieved through a ledger or register.
- A ledger does not stay constant over time: it gets updated.
- Having a robust and fast updating process is a key requirement.
- Typically, the higher the value of certain objects (i.e. diamonds), the higher are the chances that there is a regulated ledger documenting the ownership of those objects.



Hints on Security

We have introduced three major security concepts. Since we are in the context of software systems, let's provide definitions. The concepts are:

- **Identification**
- **Authentication**
- **Authorization**

What is the link?

- You want to buy a bottle of wine in a liquor shop. That are not allowed to sell alcoholic to underage.
- How can the liquor shop employee be sure that he is selling wine to the right people?

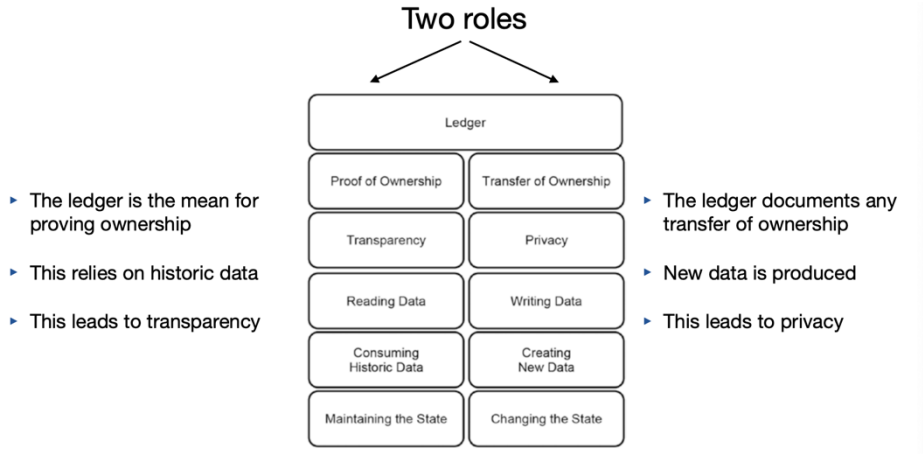
Identify and Authenticate

The identification step involves a simple **statement that can be used as an identifier**. The **identification** step **does not prove your real identity** though: it is just a claim. But we want to prove you are not underage: so **we require an authentication**. In our simple example, an ID card will do the job since it is directly connected to a single individual (e.g., with a photograph). The employee now compares the face in the shop with the one on the ID card and accomplishes the authentication. He might request a two-steps verification by asking the driver license or another type of document.

Authorize

Once the employee is convinced you are not underage, he grants access to specific resources or services (you get your bottle of wine). The authorization changes with the characteristics of the individual's identity (think about a Starfleet's captain vs. lieutenant). Remember: if you are too young but has shown a correct ID, both the identification and authentication processes went good. What failed was the authorization step because it did not comply with certain rules.

The proof of ownership and the ledger

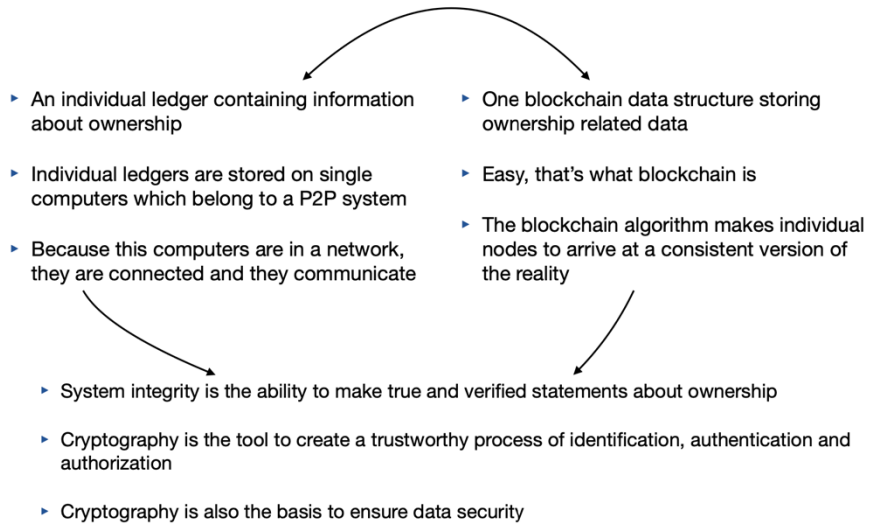


- Transparency vs. Privacy
- Proving ownership vs. transferring ownership → All relates to the blockchain.
- Reading vs. writing the ledger.

Ownership and the Blockchain

Assume you have a very good and trustworthy witness like a government ledger. What if this ledger is damaged or destroyed? What if the guys responsible for updating the ledger make an error or just throws it in on purpose? **This is a disaster! The ledger does not reflect the reality anymore. It does not represent the truth**.

How do you solve the issue when you have just one ledger? Well, you just increase the number of independent witnesses. Having many witnesses who independently make their own observations free of mutual influences is the key for this approach to finding the truth. Here is the link: **Through the use of a purely distributed P2P system of ledgers you get the proof of ownership based on that version of the reality on which the majority of peers agree on**.



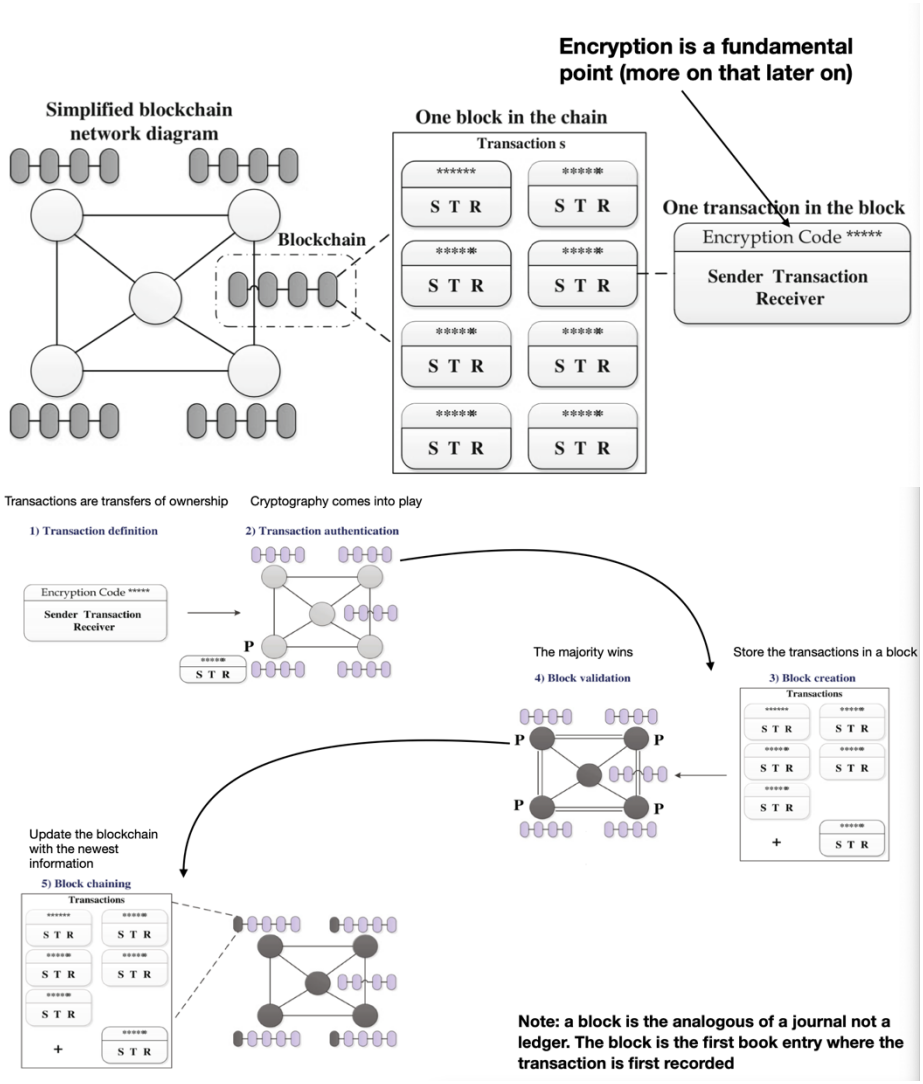
Let's build a blockchain

We now know everything about the relation between trust, integrity, purely distributed P2P systems and the blockchain. We know what a blockchain is, why we need it, and what kind of problems it can solve. **That's fancy, but seriously … How does it work internally?**

Key Points

- We consider a purely distributed P2P system.
- Its users with their computational resource contribute to the overall system's computational power.
- The system uses the Internet as a network. **The network is public and open**.
- We have **no a priori information** neither on the number of nodes nor on their trustworthiness and reliability.
- The system aims at **managing the ownership** of a digital asset, whatever it is.
- The system is therefore a **completely open and untrustworthy environment**.

Blockchain Structure



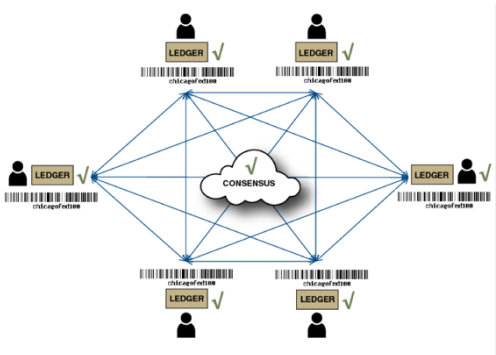
Path to Blockchain

There are **seven major tasks** which are mandatory to design a blockchain system:

1. **Describing ownership – defining the transaction**
   - What do you want to do with your fancy blockchain?
   - We are building a system that manages ownership so we **need to describe what the ownership is**.
   - Analogy: **consider transactions as transfers of ownership**.
   - The **full history of transactions** is the way to identify the current owners.
   - A transaction generally contains sender's and receiver's information as well as the value of the transaction itself.

- Plus: there is a **cryptographic digital signature**.

2. **Protecting ownership – transaction authentication**
   - Fundamental to find a **way to prevent people from accessing the property of others**.
   - **Cryptography** is the way to go.
   - It protects transactions at individual level.
   - **Once again: identify, authenticate and then authorize**.
   - First, each node will validate the incoming transaction by decrypting its digital signature.
   - The transaction is then temporarily held until finalized into a block.
3. **Storing transaction data – block creation**
   - We described a transaction and we protected it. Now we have to store it.
   - In particular, we need a way to **store the whole history of transactions: ownership clarification**.
   - We use the **blockchain as a data structure** to securely store all the transactions.
   - Now we need to distribute it…
4. **Distributing ledgers in an untrustworthy environment – Block validation**
   - We have copies of the ledger on untrustworthy nodes in a untrustworthy network.
   - There is no central control or coordination.
   - How can you prevent the transaction history from being changed?
   - You make it **unchangeable: the blockchain is immutable.**
   - Still, the blockchain needs to accept new transaction.
   - To achieve immutability, **computer science plays a major role**.
   - The blockchain data structure is an **append-only structure**.
   - Simply distributing copies of the ledger across nodes does not fulfill the goals.
   - What we need to understand is **how nodes interact with each other** and what information is exchanged.
5. **Adding new transaction to the ledger – Block chaining**
   - So far, each member has a copy of the ledger.
   - We verify that new transactions are valid and unauthorized.
   - Since it is a purely P2P system, **we turn each member into the supervisor of all its peers**.
   - If no errors are met, a new block with a given list of transactions is then added, permanently, to the blockchain.
6. **Deciding which ledgers represent the truth – Achieve consensus.**
   - It's cool that anybody can be a supervisor… But who is right?
   - Typical problem of P2P systems: different peers may have received different transactions.
   - This implies different versions of the history which threats the system integrity.
   - Also typical: you can't prevent different versions in a P2P system.
   - You have to come up with a criterion to decide what the truth is.
   - But how? There is no central authority!
   - Here is the trick: **distributed consensus**. You let each and every node decides on its own what's true. This way, when the majority independently agrees for a transaction history, the system considers it as the truth.



Source: Federal Reserve Bank of Chicago

Potential Challenges

- The blockchain system introduces a unified system for conducting financial transactions over a network (e.g. the Internet)
- Organizations need an agreement which governs the rules of the network…
- And this is huge! Companies have heterogeneous policies and protocols to perform their operations which affect their best practices.
- Don't forget privacy and security issues.
- We need standards: the focus of blockchain implementation is more on the standardization of data flows and the intermediate language used to communicate within blockchain rather than the technology that supports its platform.

Advantages and Limitations

- The blockchain technology is based on the idea of distributing information over connected nodes represented by computers.
- These nodes work together as one giant system which stores encrypted sequences of transactions into blocks.
- One of the main advantage is that there is no dependence anymore on middlemen or third party to provide trust and authentication.

Advantages

- **Empowered users:** each user has the ability to control its information and the transaction it is in.
- **Durability, Reliability and Longevity:** there is no centralized computing architecture: no global failure because of a single one.
- **Integrity, Transparency, Immutability**: transactions are public and cannot be changed.
- **Faster and Lower Costs**: no intermediaries so fast interactions and less management costs.

Limitations

- **Regulations:** currencies used in financial transactions are ruled by national governments. Governments need to reach an agreement to regulate the status of blockchain.
- **Security and Privacy:** yes, we have strong encryption algorithms, though cybersecurity remains an open problem. Think about sharing personal information over a public network…
- **Software Vulnerability:** you have a software, you have bugs! This open the door to malicious activities so no more integrity. Plus, what if there is a hack on a global technology?
- **Integration:** organizations will face problems and costs to integrate the technology into their operations.
- **Understanding the Technology:** few people just get it. Simpler for coders and hackers, harder for business professionals.

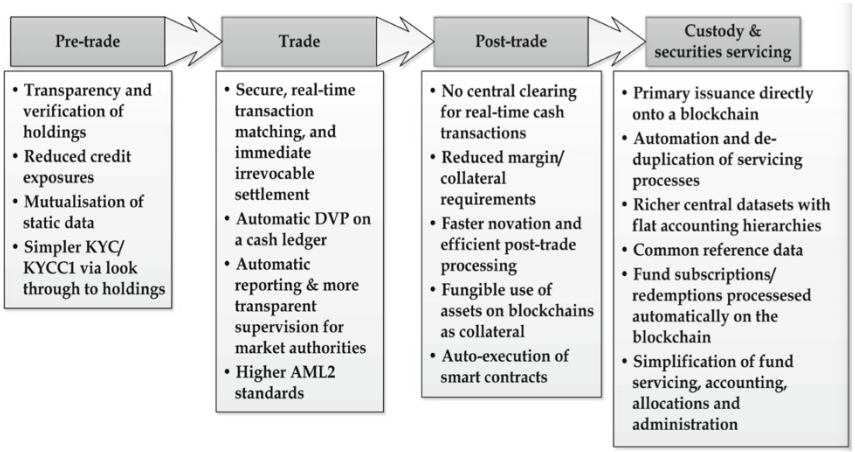
Potential Applications: Financial Services

There is an increasing interest by financial services sector in blockchain. Pretty much as an alternative to the current transactional system. That's because of inefficiencies cause by third party organizations, processing time, costs. J.P. Morgan and Goldman Sachs created a partnership to invest in the technology. Santander bank estimated a saving of approx. $20bn by eliminating centralized trust agencies. Each financial institution maintains its own ledger. Reconciling ledgers is a costly process, particularly in big banks with hundreds of ledgers. Even worse when this process is carried out through primitive and unsecured tools such as VBA.

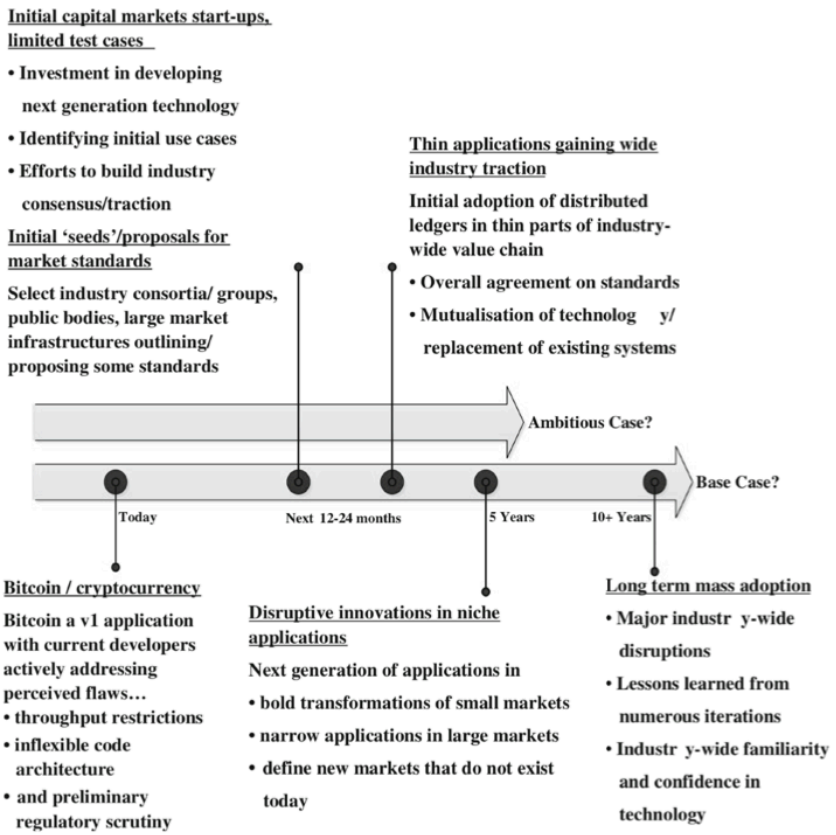
- **Cost Reduction:** no more ledger duplication as well as reduction of post-trade processing time.

- **Smart Contract:** automation of existing logic where all financial assets are already in electronic format.
- **Risk Management:** increase speed of settlement with an increase in liquidity and decreasing of balance sheet risk.
- **Improved Regularity Compliance:** authorized regulator to view a transparent ledger distributed among financial organizations (i.e., better anti-money laundering).

Adoption by Organizations
Potential benefits across the different trading stages within the financial market.



Whenever a new technology comes up, developing real-life applications is vital to ensure investments. Though, developing the technology requires time.



Blockchain Governance and Contracts

Using a blockchain provides services in a more efficient and decentralized way. Less dependence on state or government bureaucracy. You get a more distributed diffusion of authority. Contracts definition and management will improve. The ongoing legal system requires different statements to improve the enforceability of contracts. Some contracts need to be notarized to prove each party real intentions. Contracts must be registered in order for the transaction to be stored in the public record. With a blockchain, no need for human intermediation and easy way to provide provenance. Lawyers would just prepare self-executing legal documents. The ownership of intellectual property rights could be checked by referencing to time-stamped locations on blocks. We would move towards **Automated Contractual Negotiation**.

Preview on Smart Contracts.
Blockchain has the potential to decrease the costs of contracting. Smart contracts would drastically reduce the friction in commerce and society by providing greater precision to transactions. A smart contract is just a source code which can be executed like any other programming language. In addition, smart contracts offer a significant advantage to existing contractual drafting practices by eliminating the inherent ambiguity of natural language. Legal parties could use vagueness and poor language to step back from contractual conditions they no longer want to honor. A smart contract offers an effective solution by incorporating legal provisions into the code. It comes with a **zero-tolerance policy**: parties are obliged to fulfill the contract. There is freedom to breach rules because legal enforcement takes place after the act. Judicial enforcement is less needed in a system controlled by self-executing smart contracts as the manner in which the rules have been defined in the code matches exactly the manner by which they are enforced. If you want to violate the rules, you have to break the code … not that easy.

Towards a P2P economy?
We all, as individuals, interact with the internet. Developers are trying to integrate Blockchain into web browsers. Websites would employ distributed data centers. Back to the music industry: authors and musicians could use this technology to collect royalties right after there's been a purchase. Think about piracy: self-executing contracts can track duplicates and related distribution of unoriginal work.

Synergy with banking sector
Banks started to systematically become more active with blockchain in recent years. Main reason: since blockchain eliminates middlemen and is faster and more secure, banks are going to save billions. They explore the concept of decentralized systems as well as systems where only authorized users are accepted. They created innovation labs. Possible business uses: settling trades and issuing bonds, payments and settlements, securities issuance, transfers, clearing, anti-money laundering, asset registries.

Decentralization and some questions
"There is an increasing risk that we will end up with a patchwork quilt of inconsistent privacy"- Leonard Cali, Senior VP of Global Public Policy, AT&T.

- **Can Blockchain systems comply with privacy regulations?**
  - The French Data Protection Authority (DPA), members of the EU parliament and the EU Blockchain Observatory and Forum, are among the few governmental actors that have publicly acknowledged the tensions between blockchain and the GDPR.
  - In particular, the rules around the right to erasure, right to rectification and the principle of data minimization.

- A number of proposed solutions to GDPR compliance exist, such as zero-knowledge proofs and destruction of private keys, but it remains unclear whether they constitute methods of erasure or anonymization.
- Will the EU Data Protection Board issue guidelines and recommendations to "ensure that blockchain technology is compliant with EU law?".

- **Will international regulators work together?**
  - As Blockchain projects become more geographically decentralized, anonymous and/or censorship resistant, domestic regulators must tackle breaches of their laws by facilitating global coordination or, perhaps, harmonization of their securities, commodities, money transmitter, and tax laws.

- **How can the wide range of regulatory responses from different nations within these international organizations be reconciled?**
- **Will (and can) privacy coins be banned?**
  - Easier to track cash or legally accepted transactions by banks and financial institutions.
  - Harder with transactions in privacy coins like zcash or monero.
  - Perhaps the most practical way to regulate privacy coins today is to allow them to be traded on regulated crypto exchanges, which could encourage trading under the watchful eye of regulators and create an initial auditable trail.
  - For example, two regulated crypto exchanges, Gemini and Coinbase, recently began offering the trading of zcash. Both exchanges now allow withdrawals of zcash to be made to transparent addresses only.

- **Will we be able to regulate decentralized exchanges?**
  - Prior to 2018, many believed that DEXs were unstoppable.
  - Now, many DEXs implement know-your-customer procedure (KYC).
  - In 2018, the SEC published guidance on online platforms for trading digital assets.
  - ShapeShift reluctancy introduced KYC in the form of compulsory membership.
  - The SEC fined EtherDelta's creator for causing software to violate the law requiring registration of securities exchanges.

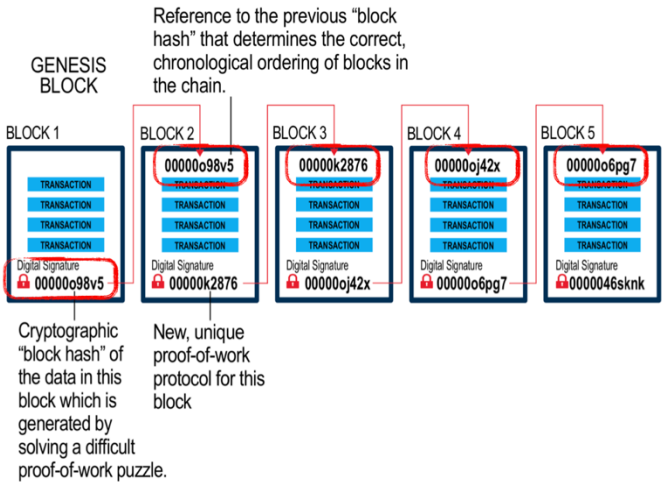
- **Will developers be held responsible for violations of law?**
  - In corporate law, the "corporate veil" allows a corporation to be treated as a separate legal entity, insulating the company's owners, in most cases, from personal liability for the company's violations.
  - A "tech veil" has helped code developers escape liability from state and federal regulation and civil lawsuits arising from bugs in, or third parties' malicious use of, their code.
  - Sometimes, the "tech veil" can be pierced.
  - In 2018, *Commodity Future Trading Commission* (CFTC) Commissioner Brian Quintenz suggested that smart contract code developers could be prosecuted for wrong doing.
  - First: reasonably foreseeable the code would likely be used by U.S. persons in a manner violative of CFTC regualtions.
  - Second: when the SEC charged Zachary Coburn (founder of EtherDelta and writer/deployer of the EtherDelta smart contract) with operating an unregistered national securities exchange.

- **What is or is not reasonably foreseeable in an age of constant innovation?**

- **How, if at all, will courts and regulators distinguish between the role of the code writer, deployer of the code, and platform operator?**
- **Will the "tech veil" be pierced further in criminal or civil cases?**
- **If so, how will enforcement be affected by decentralized networks, unstoppable smart contracts and anonymous code developers?**

Cryptography - Class 04

Sketching a Blockchain



Cryptography

Can you read this document at all?

luoxbMklxsc0K1U+qrL/UMafTV5zrDG7DLGFkMSyxgj8FNOgEQpLAErzPtLpqEZMKQIC5mHFoZyVW/b9KaaG3ll/0Z0LXm8
fpEug1whAQdDRmyLhCOOxg0p/5aOQv7OGt76xsda2ucxWnWf7fySayRAhmCU6b+SL1VQwyFjYdQIz45F1SbXmpUwm5V
oDdH/8ITmXHhoMoBjxH/qiF5augDUJGff/IUKD2xfHfyPtz/+7n+fUu0uFOfFgaMxtM5C0GcznvY6kFO1C87GW9Qqc9Lh/+I
kuH4PilySdFQvE8/YirYfkbEHX1nJpPgU+Uf3YfU6lvZn2T7tbwtwrqssUEdX7gHOJ9JJ8AAmpulmXp0g4Xm/PQSas/KK4oK
h4zOtpnsuh2gYTza6pEBsx9JQh0puOMD8OPMTfYNwq50kvPLP9dwI4pRUxeed16lvOlNM8cmyMtClU6ccoPCgJmR7Cp
gcEvdvF+P52Jla3J9stb6mhSRVWRIBsvR0bGGM6Q+jn

Well, it's…

To be, or not to be: that is the question:
Whether 'tis nobler in the mind to suffer
The slings and arrows of outrageous fortune,
Or to take arms against a sea of troubles,
And by opposing end them? To die: to sleep;
No more; and by a sleep to say we end
The heart-ache and the thousand natural shocks
That flesh is heir to, 'tis a consummation
Devoutly to be wish'd.

Cryptography or **Cryptology** come from Ancient Greek: kryptós (hidden, secret) and gaphein (to write) or logia (study). The idea is to protect data from being accessed by unauthorized people. It provides a mechanism for securely encoding the set of rules in the system. Cryptography is a deep academic research field utilizing many advanced mathematical techniques. Computer science also has its own branch focusing on solving cryptography problems.

Symmetric vs Asymmetric Cryptography

Definition: a key is random string consisting of hundreds or thousands of ones and zeroes (i.e. binary digits). The key is used by a cryptographic algorithm to transform plain text into cipher text or vice versa.

Symmetric vs Asymmetric keys:

- Symmetric algorithms use a single key for both encryption and decryption.
- Asymmetric algorithms use two different keys: a private and a public one, hence the name.

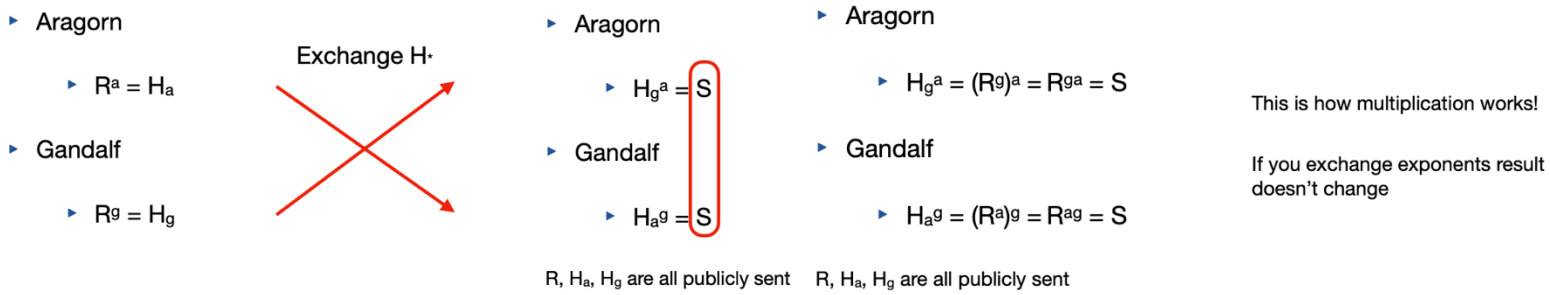
The example

Let's assume that two friends want to communicate securely over the Internet. How do you get shared keys? The framework: Diffie-Hellman key exchange protocol.

Aragorn, Legolas, Gimli and other friends are facing the Uruks at the Helm's Deep. They are in desperate need of assistance in order to defeat Sauron's army. They decide to use cryptography to

send a message to Gandalf the White who can arrive with the Rohirrim. Aragorn and Gandalf came up with this protocol:

- Share an arbitrary number R (public key) (don't care if intercepted)
- Each think of their own secret keys without sharing them:
  - Aragorn picks "a"
  - Gandalf picks "g".



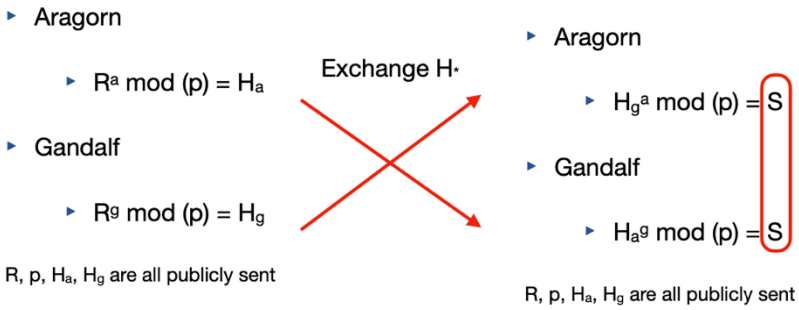
We have the same secret key S, so problem solved! R and H* have been exchanged! They can be intercepted! And if so, it would be trivial to know what a and g are.

The Log and Discrete Log problems

Log Problem	$2^n = 16$	Compute n?
Discrete Log Problem	$2^n \bmod 17 = 16$	Compute n?

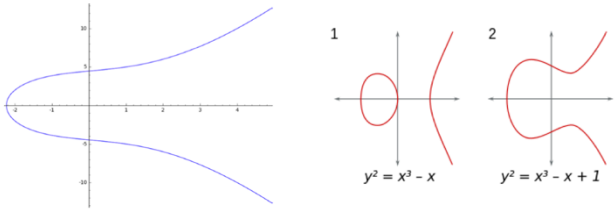
With Discrete Log Problem, guessing is the way-to-go. Basis for Diffie-Hellman schemes.
Exponentiation is easy, but it's very hard to know secret keys.

Moving to the discrete
Modulus operation doesn't screw up exponentiation



So, what's the problem here?
The problem is that with increasing computer power and new algorithms it's getting easier to solve the problem! We have a conflict: keys have to be larger, but keys have to be smaller at the same time.

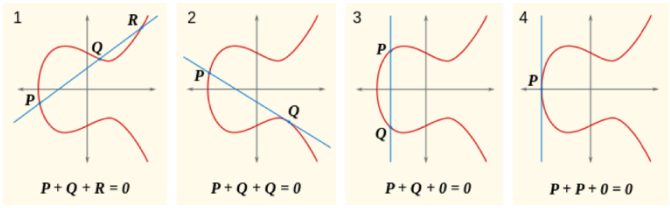
Elliptic Curves for the good!
Elliptic curves pop up when solving elliptic functions over a given space. An elliptic curve is a curve that's also naturally a group. The group law is constructed geometrically. Elliptic curves have (almost) nothing to do with ellipses, so put



ellipses and conic sections out of your thoughts. Elliptic curves appear in many diverse areas of mathematics, ranging from:

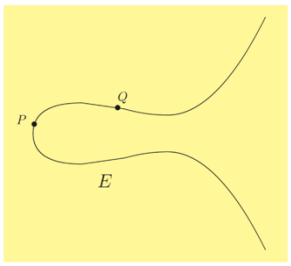
- Number theory
- Complex analysis
- Cryptography
- Mathematical Physics

Property 1: Elliptic curves are symmetric over the X-axis (aka, known as horizontal symmetry).
Property 2: Given two points A and B, if we draw a straight line between A and B, that line will intercept the elliptic curve in at most one more point.



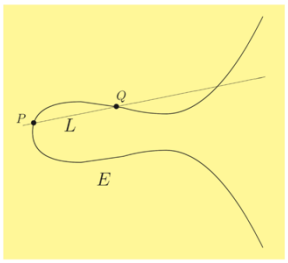
$E: y^2 = x^3 - 5x + 8$

Let's start with points P and Q defined on curve E



$E: y^2 = x^3 - 5x + 8$

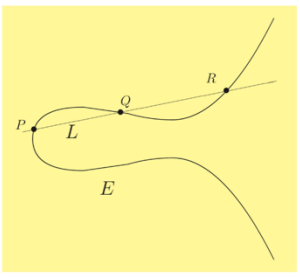
Draw a line L through P and Q



$E: y^2 = x^3 - 5x + 8$

The line L intersects the cubic curve E in a third point

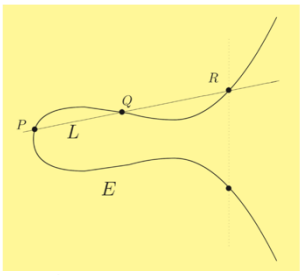
Call that third point R



$E: y^2 = x^3 - 5x + 8$

Draw a vertical line from R.

This line intersects the curve E in another point

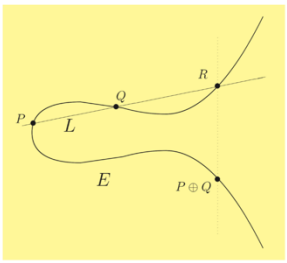


$E: y^2 = x^3 - 5x + 8$

We define the sum of *P* and *Q* on *E* to be the reflected point

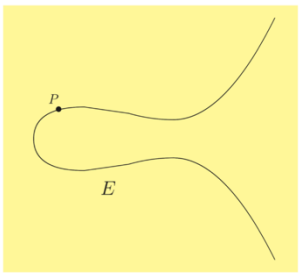
$P \oplus Q$

$P + Q$



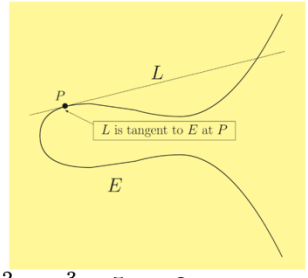
$E: y^2 = x^3 - 5x + 8$

How do we add a point *P* to itself, since there are many different lines that go through *P*?



$E: y^2 = x^3 - 5x + 8$

If we think of adding *P* to *Q* and let *Q* approach *P*, then the line *L* becomes the tangent line to *E* at *P*



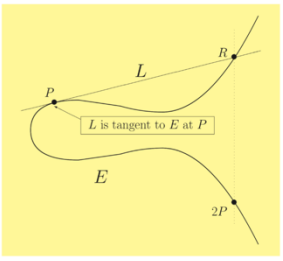
$E: y^2 = x^3 - 5x + 8$

Then we take the third intersection point *R*, reflect across the x-axis, and call the resulting point

$P \oplus P$

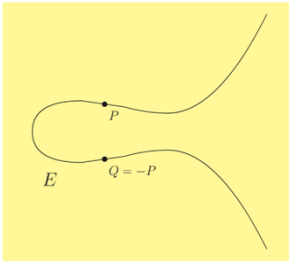
or

$2P$



$E: y^2 = x^3 - 5x + 8$

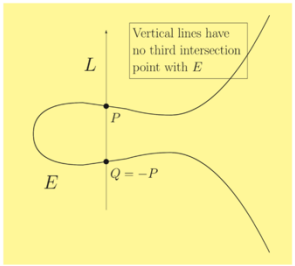
We denote the reflected point by -*P*



The vertical line L through *P* and -*P* does not intersect *E* in a third point!

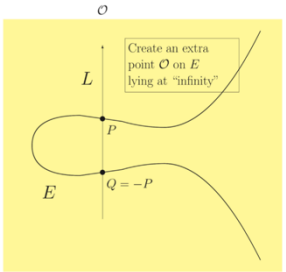
But we actually need a third point to be able to define

$P \oplus (-P)$

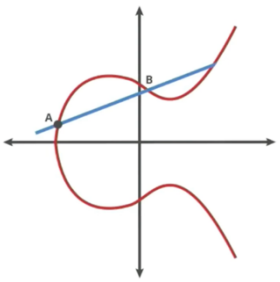


Since there is no point in the plane that works, we create an extra point *O* "**at infinity**"

O is a point on every vertical line



Property 2: given two points A and B, if we draw a straight line between A and B that line will intercept the elliptic curve in at most one more point



Some Properties of Addition

Theorem, the addition law on E has the following properties:

$$\begin{aligned}
&\text{(a)}\ \ P+\mathcal{O} && \text{for all } P\in E\\
&\text{(b)}\ \ P+(-P)=\mathcal{O} && \text{for all } P\in E\\
&\text{(c)}\ \ P+(Q+R)=(P+Q)+R && \text{for all } P,Q,R\in E\\
&\text{(d)}\ \ P+Q=Q+P && \text{for all } P,Q,R\in E
\end{aligned}$$

Elliptic Curve Discrete Log Problem

Let's represent this operation in: ***nA = E***. Where *nA* does not represent standard multiplication. It's more of a … "dot" operation which eventually yields to the generation of other points.

- A dot A dot A …

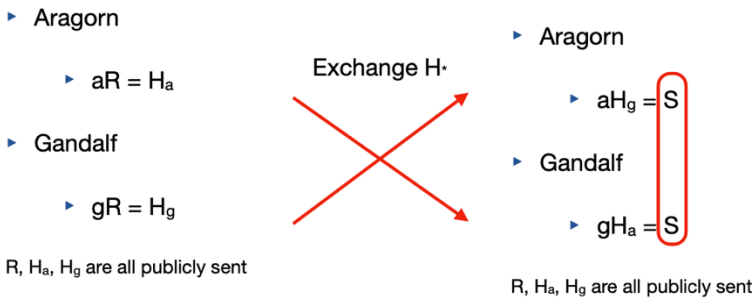
Turns out, it's super hard to find n event if we had the starting point A and the arriving point E. In other words, it's hard to find how many times the "dot" operation has been done.

Order Independence is preserved

Diffie-Hellman	Elliptic Curve
$(R^a)^g = (R^g)^a$	$g(aR) = a(gR)$

Elliptic Curve Diffie-Hellman (ECDH)

As we did with mod(p), with EC we have to limit the set of possible values.



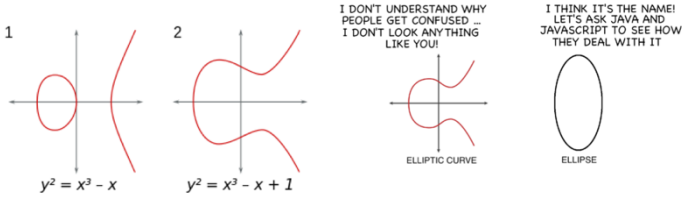
Elliptic Curves solve the issue

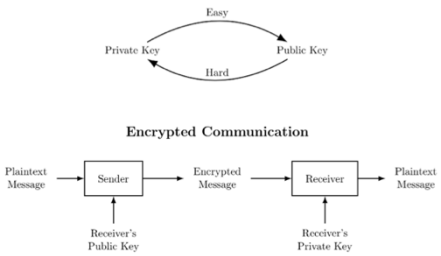
Diffie-Hellman	Elliptic Curve
$R^a \bmod(p) = H_a$ p * p * p … 2380 bit	$aR = H_a$ R dot R dot R … 228 bit

- Math involved in ECs is not vulnerable to the same algorithms
- It's as easy to break a 228 bit EC as a 2380 bit DH
- It seems we solved our problem: resilient to attacks and small in size.

Elliptic Curve Cryptography

One method for implementing public key cryptography is the Elliptic Curve Cryptography (ECC).





Downsides of Elliptic Curves

- Difficult mathematical representation
- Many are patented
- Fail without sufficient randomness → Bad RNGs lead to successful attacks.
- Lack of theoretical foundation. Are they really secure?
- Built-in trap doors (i.e., backdoors)?

Hash Functions

The metaphor: What is a fingerprint?

- Impressions of the friction ridges of all or any part of the fingers of the human hand.
- Highly considered since they identify human beings uniquely.
- Used to investigate crimes, identify offenders, etc.

We want to be able to identify the data by using its digital fingerprints. One way to achieve data identification is to use **hash functions**.

Has functions are small computer programs which transform any kind of data into a string of fixed length, regardless the size of input data. Remember: the number of possible inputs is larger than the number of possible outputs. **A hash function is a many-to-one function**. We focus on special cases called **cryptographic hash functions**.

```
> hello = digest( object = "Hello World!", algo = "sha256" )
> going = digest( object = "Hi there, how is going?", algo = "sha256" )
> hello
[1] "9374f2c6f404965fb4ef8299642a8ede7b1cbe5a5e58f65a05d7a52e4e20ed91"
> going
[1] "44cbaea6d3534a1beece82cbc37c394270b26c3bb99a3ac17a425f0e0d69c114"
> nchar( hello )
[1] 64
> nchar( going )
[1] 64
```

Cryptographic Hash Functions

A cryptographic hash function is defined as a hash function which has the following 3 main properties:

- **Preimage Resistance**: let y be the output of H for some unknown input. An input x which satisfies H(x) = y is called a preimage of y under H. By the many-to-one nature of H, preimages are not necessarily unique. The function H is said to be preimage resistant if it is computationally infeasible to calculate any preimage x of a given y.
- **Second Presage Resistance**: given an input string x, it is computationally infeasible to find a different input string x' such that H(x) = H(x').
- **Collision Resistance**: it is computationally infeasible to find a pair of distinct input strings x, x' such that H(x) = H(x').

Let's add two more properties:

- **Deterministic**: a hash function yields identical hash values for identical input data.
- **Pseudorandom**: the hash value returned by the hash function changes (pseudo) randomly when the input data are changed. It should not be possible to predict the hash value based on the input data.

What are the requirements to be a "good" hash function?

How large does n (output length in bits) need to be?

Having a large output length is not sufficient for a hash function to be considered cryptographic. The relation between the input bit string x and the output bit string y should be complicated to prevent easy recovery of x from y.

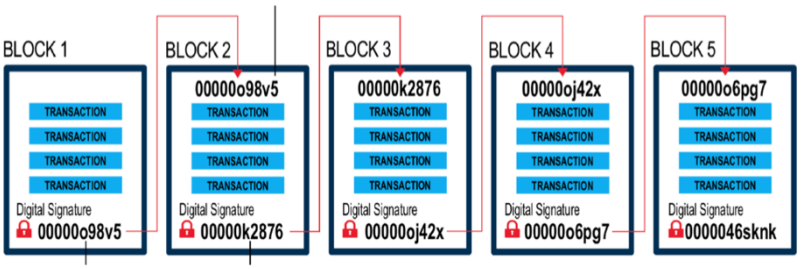
Example with `sha256`

```
> hello = digest( object = "Hello World!", algo = "sha256" )
> going = digest( object = "Hi there, how is going?", algo = "sha256" )
> hello
[1] "9374f2c6f404965fb4ef8299642a8ede7b1cbe5a5e58f65a05d7a52e4e20ed91"
> going
[1] "44cbaea6d3534a1beece82cbc37c394270b26c3bb99a3ac17a425f0e0d69c114"
> nchar( hello )
[1] 64
> nchar( going )
[1] 64
> # let's check that they are 256 bit long
> # in hexadecimal notation, each char is 4 bits so it's simple!
> bits_hello = nchar( hello ) * 4
> bits_going = nchar( going ) * 4
>
> if ( bits_hello == bits_going ) {
+   cat( "They are really the same and the result is:", bits_hello )
+ }
They are really the same and the result is: 256
```

When can an input-output relationship be considered complicated? Really hard to answer! An example could be SHA-256.

Secure Hash Algorithm
The SHA-256 hash function was announced in 2001 by the National Institute of Standards and Technology (NIST). The number 256, we already know, indicates the output length in bits. The SHA-256 function specification restricts the input to be at most $2^{64} - 1$ bits long. Up to now, there is no formal proof that SHA-256 is in fact a cryptographic hash function. But it has no known weakness which make finding preimages, second presages and collisions computationally feasible. In other words, it would take an infinite amount of computational time to break it.

One hash value to rule them all…
What if we have to provide one unique hash value for several independent data? Simply, you can't use a hash function directly. Blockchain data structure has to deal with multiple transactions at once and requires one single hash value.



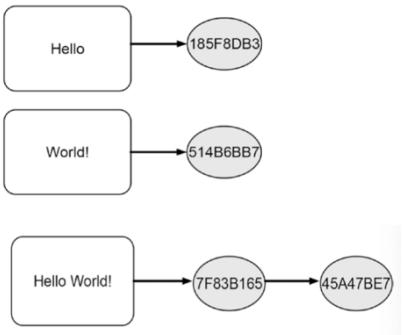
So, how do we solve this issue?

Hashing Patterns
The idea is to build structures made by hash functions. There exist different patterns:

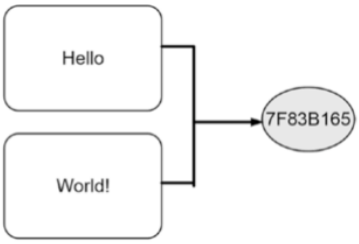
- Independent Hashing
- Repeated Hashing
- Combined Hashing
- Sequential Hashing
- Hierarchical Hashing

Independent and Repeated Hashing

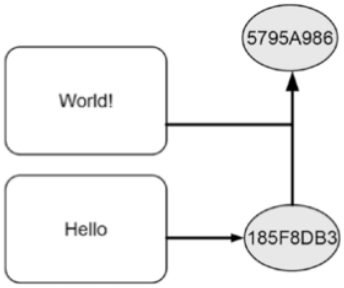
- **Independent Hashing**, just apply the hash function to each and every independent piece of data.
- **Repeated Hashing**
  - Just apply the hash function repeatedly to the output of the step before.



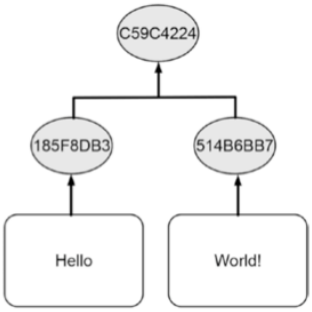
- You can indeed hashing an hash value.



Combined Hashing
The goal is to compute a single hash value for more than one piece of data in one single step. The idea is to combine all the independent data into one and then apply the hash function. Combining data is computationally intensive: it costs time, memory allocation, storage. Be sure to apply this method when individual chunks of data are small. Also, we have no hash value associated with each chunk.



Sequential Hashing
With sequential hashing we incrementally update of a hash value as new data arrive. It is a mixture of the combined and repeated hashing methods in this order. We first combined the chunks and compute a single hash value (combined hashing), then we compute another hash value on the previous one (repeated hashing). This comes in handy if we want to have a single hash value over time with a well tracked updating process.



Hierarchical Hashing
Here we just want to have a single hash value which represents a given structure. It combines the independent hashing and the combined hashing. It's more efficient then combined hashing since here the chunks are simple hash values.

Digital Signatures
A digital signature is supposed to be the digital analog to a handwritten signature on paper. We desire **two properties** from digital signatures that correspond well to the handwritten signature technology:

- Only you can make your signature, but anyone who sees it can verify that it's valid.
- We want the signature to be tied to a particular document so that the signature cannot be used to indicate your agreement or endorsement of a different document.

How can we build this in a digital form using cryptography?
A digital signature scheme consists of 3 algorithms:

- **(sk, pk) := *generateKeys*(keysize):** The *generateKeys()* method takes a key size and generates a key pair. The secret key sk is kept privately and used to sign messages. pk is the public verification key that you give to everybody. Anyone with this key can verify your signature
- **sig := *sign*(sk, message):** The *sign()* method takes a message and a secret key, sk, as input and outputs a signature for message under sk
- **isValid := *verify*(pk, message, sig):** The *verify()* method takes a message, a signature, and a public key as input. It returns a boolean value, isValid, that will be true if sig is a valid signature for message under public key pk, and false otherwise

Two properties:

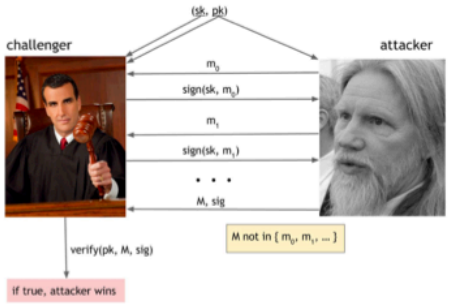
- Valid signature must verify that: verify(pk, message, sign(sk, message)) == true.
  - It is straightforward that valid signatures must verify.
  - If someone signs a message with the secret key, and someone else later tries to validate that signature over that same message using the public key, the signature must validate correctly.
  - This is a basic requirement for signatures to be useful at all.

- Signatures are **existentially unforgeable**.
  - It states that it's computationally infeasible to forge signatures.
  - That is, an adversary who knows your public key and gets to see your signatures on some other messages can't forge your signature on some message for which he has not seen your signature.
  - This unforgeability property is generally formalized in terms of a game that we play with an adversary.

Digital Signatures: the unforgeability game

In the unforgeability game, there is an adversary who claims that he can forge signatures and a challenger that will test this claim. The first thing we do is use *generateKeys()* to generate a secret signing key and a corresponding public verification key. We give the secret key to the challenger so he can make signatures. We give the public key both to the challenger and to the adversary. So the adversary only knows information that's public, and his mission is to try to forge a message. Intuitively, the setup of this game matches real world conditions. A real-life attacker would likely be able to see valid signatures from their would-be victim on a number of different documents. And maybe the attacker could even manipulate the victim into signing innocuous-looking documents if that's useful to the attacker.

- To model the game, we allow the attacker to get signatures on some documents of his choice, for as long as he wants, as long as the number of guesses is plausible
- Example: we would allow the attacker to try 1 million (2^{20}) guesses but no 2^{80}
- Once the attacker is satisfied that he's seen enough signatures, then the attacker picks some message, M, that will attempt to forge a signature on
- The only restriction on M is that it must be a message for which the attacker has not previously seen a signature
- The challenger runs the *verify()* algorithm to determine if the signature produced by the attacker is a valid signature on M under the public verification key
- If it successfully verifies, the attacker wins the game



The signature scheme is unforgeable if and only if, no matter what algorithm the adversary is using, his chance of successfully forging a message is extremely small

Hash Structures and Consensus Protocols – Class 05

Applying Hash Functions… for real

Different applications:

1. Comparing data
2. Detecting changes in data
3. Referring to data in a change-sensitive way (**very important!**)
4. Storing data in a change-sensitive way
5. Causing time-consuming computations

1. **Hashing to Compare Data**

   The goal is to compare data without looking at their content. Also, the comparison should be fast and completely independent on the type of data and its size. So here is the idea: just compare hash values. How: if all the hash values are different, then all the data chunks are different as well. It works because hash functions are collision resistant.

2. **Hashing to Detect Changes in Data**
   If we can compare, we can also detect changes. We want to be able to say if data which should remain unchanged, has changed at a certain point in time. So we throw a comparison between two hash values belonging to the same chunk of data. If both hash values are identical it implies no change. Once again, detecting changes in data that are supposed to stay unchanged works due to collision resistance of cryptographic hash functions.

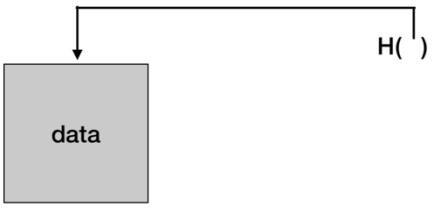
3. **Having to Refer to Data in a Change-Sensitive Way**
   Comparing and detecting changes are the basics. A more advanced approach is **hash references** and then ensure the data remain unchanged. To achieve this, we combine the hash value itself associated with stored data with information about where that data is located. The moment there is a change, both information will be no more consistent: the hash reference becomes invalid.
   Example: a cloakroom ticket is a hash reference to your jacket.
   - The physical data is the jacket.
   - Add-on information is the place where your jacket has been stored.

   Computers use reference addresses to remember where they store things that we told them to store. Hash references refer to data and at the same time they verify the data has not changed since the reference was created.
   Example: the cloakroom ticket points at an empty cloaca hook.
   We use hash references to protect users from retrieving wrong data (i.e. unintentionally or intentionally changed). The whole idea is properly based on the fact that hash values are encrypted: it is very unlikely that different chunks of data have identical hash values.

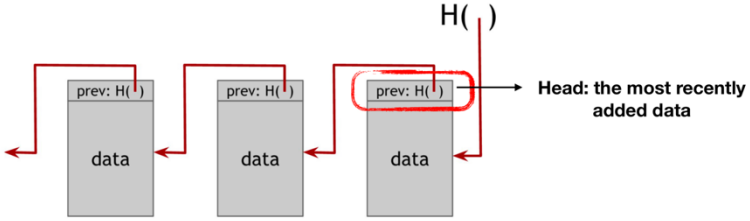
4. **Hashing to Store Data in a Change-Sensitive Way**
   The goal is to extend the reference method to be able to store data through hash values.
   Cloakroom tickets work in a straightforward way under optimal conditions … but.
   Let's play with this:
   - We give a jacket and we get a ticket.
   - We take the ticket and we put it in the pocket of another jacket.
   - We give this second jacket and we get another ticket.
   - Imagine to keep going on with this to form a very long list of tickets…

   We can do the same with data and form a chain of data. If at any point in time either the data or the hash references are changed all the hash references are broken. But this is the warning that advises us that a change occurred after the initial reference was created. There exist two patterns that can be used to store data in a change-sensitive way:
   - The chain
   - The tree

Hash Pointers and Data Structures
Definition: a hash pointer is simply a pointer to where some information is stored together with a cryptographic hash of the information.
Whereas Ed a regular pointer gives you a way to retrieve the information, a hash pointer also **gives you a way to verify that the information hasn't changed**.

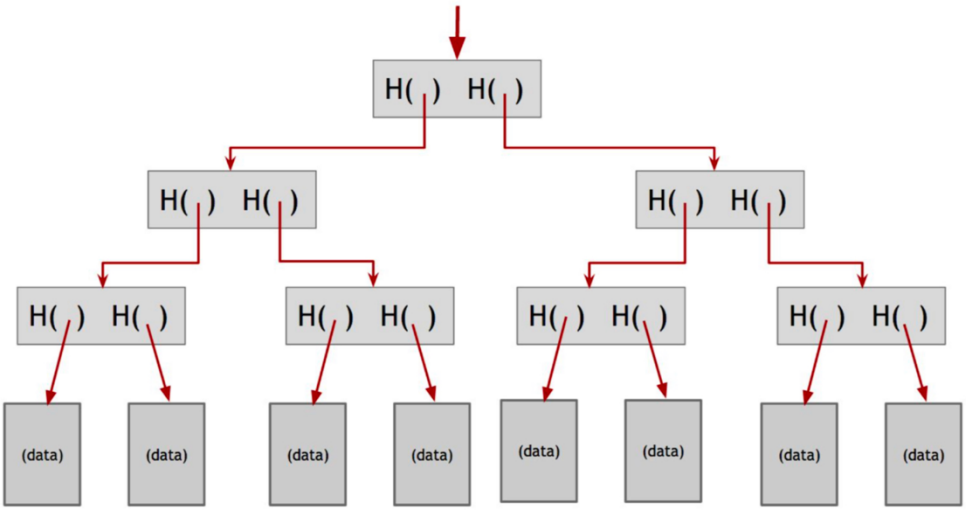


The Chain Pattern
Here is an example of a blockchain with hash pointers. Each block tells us where the value of the previous block was and a digest of that value



which allows us to verify that the value itself hasn't changed. This comes in handy when we know data will arrive step by step.

The Tree Pattern
This structure is called **Merkle Tree** and was proposed by the computer scientist Ralph Merkle in 1979. It is an important structure because it allows to group distinct pieces of data available at the same time and make them accessible via a single hash value.



5. **Causing Time-Consuming Computations**
   So far, we know we can combine, refer and store data securely and efficiently. Here we want to allow computers to challenge other computers with games. This is a fundamental step to understand blockchain which is strongly based on. It's a game, but with certain rules:
   - It should not be possible to solve the game based on knowledge or previously stored data.
   - Forget about being smart.
   - The only way is by brute computational power.

   How could you open a lock if you don't know the sequence?
   - Well, we try all the possible combinations.
   - Time consuming!

   The process of just trying all the combinations implies no knowledge nor reasoning of any sort. You just put hard work on this task. Hash puzzles (games) are computational games which can be considered as the equivalent of opening a lock.

Authorizing Transactions with Digital Signature
We know a bit about cryptography. We know a bit about hash functions, how they work and how we could use them. Now, we need to understand how to rightfully transfer a property from one account to another. In other words, we need an authorization. We obtain authorization through digital signatures. Think about handwritten signatures: through them we state agreement. We accept them because we think they are unique. Now, we transfer this concept tot the digital world of the blockchain. Digital signatures is a crucial concept for the security of individual transactions in a blockchain.

The Importance of Ordering
One of the fundamental problem with blockchain is that the order of blocks doesn't always reflect the one of transaction. Recognizing the order in which transactions occurred is key to achieve the

same identical results. Changing the order of transaction data implies a change in the aggregation of them. Receiving a payment from a friend seems to be the trivial case.

- I don't care about the order as far as I get the money.
- The logical thing is: money are transferred from my friend's account and then put on mine.

What if the two transactions occurred in the opposite order?

Integrity of the Transaction History

The history of transactions is really the core of the blockchain. Why? Because history is the way we reconstruct the state of the ownership. We want the history of data to be safe, complete, correct and consistent such that the integrity of the system is maintained. We need a system to validate transaction:

- **Formal correctness**, implies that the description of a transaction contains all the required data and those data are correctly formatted.
- **Semantic correctness**, deals with the meaning of a transaction and its scope.
  - Preventing double-spending
- **Authorization**, implies that every transaction carries all the necessary information to prove that the owner of the account agrees with the transfer.

The Double-Spending Problem

The idea: if we spend $1 to buy an apple, we can't use that same $1 to buy an orange. Why? Just because you have to give $1 to the vendor. With digital currencies, or assets, there is no such thing. We can spend the same currency for two or more transactions, in principle. Let's imagine we have a P2P system for managing ownership of real estate. Ledgers is maintained by individual computers so each peers has its own copy. The minute the ownership of a house is transferred from one account to another, all the ledgers must be updated to match the latest version of reality.
Issue: someone who already know the latest info could do something bad to someone else who is not up-to-date.

Example: *Frodo is sick and tired of carrying the ring around. So he decides to sell it but before doing it, he hides the ring where no one can spot it. He then goes to one of his fellow hobbits telling him about the ring and closes the deal for a considerable amount of pipeweed. However, Frodo is very smart. He wants more pipeweed so he quickly goes to another hobbit and sells the same ring for another honest amount of pipeweed.*

- The first transaction is a transfer of ownership and is documented in one of the ledgers in the system.
- We might have a problem with the second transaction… Hobbits are slow we all know that… (i.e., the system needs time to get updated).
- What if the second hobbit doesn't know about the first transaction? He will approve the deal!
- Frodo would have been able to sell the ring twice.

The term double-spending can be used to refer to different concepts:

- A problem of **copying digital goods**.
  - Easy: on a computer you can take as many copies as you want.
  - No noticeable limitations.
  - We have seen how digital money can be copied such that it can be used more than once.
  - Digital equivalent to replication bank notes with a copying machine.
  - This violates the core principle of money: an identical piece of money cannot be given to different people at the same time.

    - The ability to copy and spend digital money multiple times decreases value of the money, hence, the double-spending problem.
- A problem that can occur in **distributed P2P systems**.
    - Transferring (distributing) information to all the nodes requires time.
    - This implies that not all nodes know the latest state of the system.
    - This is a threat which can be exploited who already knows the latest information.
    - As a result, one may be able to transfer ownership more than once, resulting in double-spending.
- A threat that can **violate the integrity** in purely distributed P2P systems.
    - Distributed systems are not restricted to just the management of ownership.
    - One other problem is to maintain data consistency in such systems.
    - Data consistency is one of the aspect of integrity.
    - We can argue that the double-spending problem is an example of violated system integrity.

Hash Structures and Consensus Protocols – Class 06

Blockchain is a Distributed System

Blockchain consists of different actors. Each actor acts depending on personal incentives and on available information. When a new transaction is broadcasted to the network, nodes can decide if they want to include it as a copy in their ledger or to ignore it. When the majority of the actors decides on a single defined state, the consensus is achieved.

What is the Consensus?

The consensus is simply the common agreement on something. It is a fundamental problem in distributed computing. The problem is: how can we reach a consensus through computers? Well, we need an algorithm. **Definition**: a consensus algorithm is a process used to achieve agreement on a single data value among distributed processes or systems. Let's try to improve our definition … more formally.

Distributed Consensus Protocol: There are n nodes that each have an input value. Some of these nodes are faulty or malicious. A distributed consensus protocol has the following two properties:

- It must terminate with all honest nodes in agreement on the value.
- The value must have been generated by an honest node.

The context of Bitcoin

Let's focus for a second on Bitcoin: a peer-to-peer system. What happens when Princess Leia wants to send the money that Han Solo deserves? Princess Leia has to broadcast the transaction to all the nodes within the network.

Ian's node might be in the network (not a requirement). Actually, he is running one of the nodes. He wants to be notified when the transaction did happen.

Distributed Consensus

Several users are broadcasting this transaction. The nodes must agree on two things:

- All the nodes in the P2P network have a ledger consisting of a sequence of blocks, each containing a list of transactions, that they've reached consensus on.
- This leads to a single and global ledger.

Also, some of the nodes might have not being informed —> the network is not perfect.
How exactly do nodes come to consensus on a block? At regular intervals, say every 10 minutes, every node in the system proposes its own outstanding transaction pool to be the next block. Then the nodes execute some consensus protocol, where each node's input is its own proposed block. If the consensus protocol succeeds, a valid block will be selected as the output. There are a number of technical problems with this approach.

Issues with Distributed Consensus
First: consensus in general is a hard problem

- Nodes can crash
- Nodes can be malicious

Second: the network is imperfect

- Not all pairs of nodes are connected to each other
- Poor internet connectivity

Third: for real cases, **latency is a real problem**

- No global time so there is no common ordering of events based on timestamps.

Request: Fault Tolerance
The goal is to achieved overall system reliability in the presence of a number of faulty processes. So, this requires to agree on some data value needed during computation. What happens when an actor decides to not follow the rules and to tamper with the state of the ledger? What happens when these actors are a large part of the network, but not the majority? In order to create a secure consensus protocol it must be **fault tolerant**.

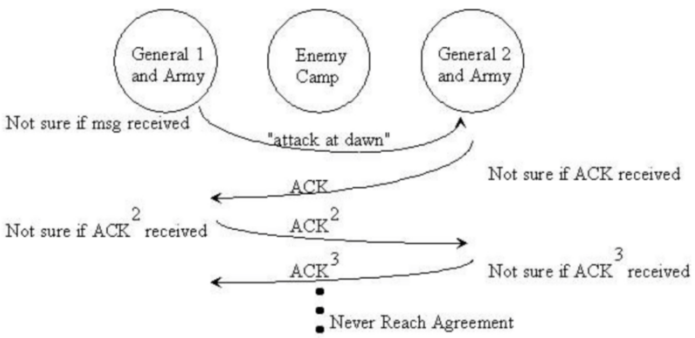
Unsolvable Problems
Focus on two well-known problems:

- The Two Generals Problem
- The Byzantine Generals' Problem

Plus:

- Byzantine Fault Tolerance

The Two Generals Problem (Akkoyunlu te al. 1975)
The scenario: two generals are attacking a common enemy. General 1 is the leader, while General 2 should obey General 1's orders. Each general's army has not enough power to defeat the enemy. They need to cooperate and coordinate the attack. There is one big caveat though. Since they have to agree on the attack time, General 1 sends out a messenger to General 2's camp. There is a chance that the messenger can get caught by enemy. If the messenger does not get caught, then General 2 has to **acknowledge** the message by sending the messenger back to General 1's camp. Again, there is a chance that the messenger can get caught by enemy.



There is no way to guarantee that each general be sure the other has agreed to the attack plan! Both generals will always be left wondering whether their last messenger got through.

The Byzantine Generals' Problem (Lamport et al. 1982)
In this scenario, the authors generalize the Two Generals Problem. So, we have manifold of Generals who need to agree on the time to attack the enemy camp. There is a twist! One or more generals can be a traitor. A traitor is a liar, and he can lie about his choice. Each general has a certain number of lieutenants. To achieve consensus, both the general and all his lieutenants must agree on the same choice. For simplicity, the choice is binary: *attack* or *retreat*.

Byzantine Generals Problem. A commanding general must send an order to his $n - 1$ lieutenant generals such that

IC1. All loyal lieutenants obey the same order.
IC2. If the commanding general is loyal, then every loyal lieutenant obeys the order he sends.

IC = Interactive Consistency conditions

Even if the General is a traitor, consensus must be achieved anyway. Thus, all lieutenants take a majority vote.
For any m, OM(m) (Oral Message) reaches the consensus if there are more than 3m generals and at most m traitors.

Algorithm OM(0).

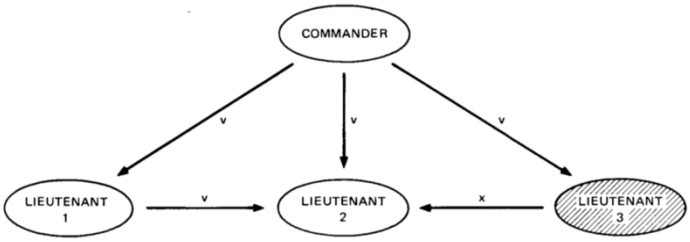
(1) The commander sends his value to every lieutenant.
(2) Each lieutenant uses the value he receives from the commander, or uses the value RETREAT if he receives no value.

Algorithm OM(m), $m > 0$.

(1) The commander sends his value to every lieutenant.
(2) For each i, let v_i be the value Lieutenant i receives from the commander, or else be RETREAT if he receives no value. Lieutenant i acts as the commander in Algorithm OM($m - 1$) to send the value v_i to each of the $n - 2$ other lieutenants.
(3) For each i, and each $j \neq i$, let v_j be the value Lieutenant i received from Lieutenant j in step (2) (using Algorithm OM($m - 1$)), or else RETREAT if he received no such value. Lieutenant i uses the value *majority*($v_1, \ldots, v_{n-1}$).

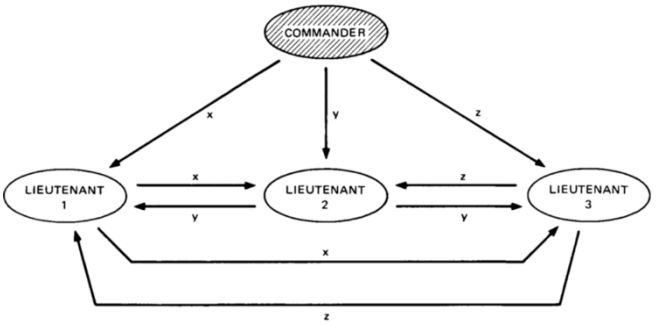
The algorithm can reach consensus as long as 2/3 of the actors are honest. If the traitors are more than 1/3, consensus is not reached, the armies do not coordinate their attack and the enemy wins.

Case 1



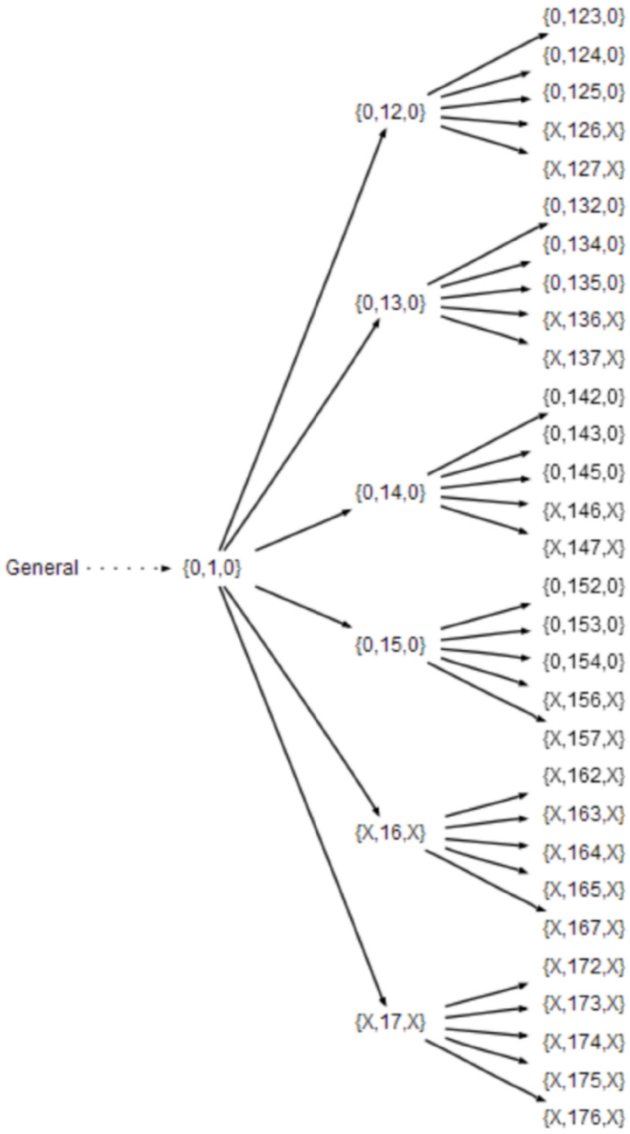
- The commander sends v to all lieutenants.
- L1 sends v to L2 && L3 sends x to L2.
- L2 <— majority(v, v, x) := v.
- L3 is the traitor!
- The final decision is the majority vote from L1, L2, L3.
- The goal is for the majority of all the L* to pick the same decision not a specific one.

Case 2



- The commander sends x, y, z to L1, L2, L3, respectively.
- L1 sends x to L2, L3 && L2 sends y to L1, L3 && L3 sends z to L1, L2.
- L1 <— majority(x, y, z) | L2 <— majority(x, y, z) | L3 <— majority(x, y, z).
- Commander is the traitor!
- They all have the same value.
- If x, y, z are different commands, a default option would be to retreat.

A tree with n = 7 and m = 2:



Byzantine Fault Tolerance (BFT)

Byzantine Fault Tolerance is the characteristic which defines a system that tolerates the class of failures that belong to the Byzantine Generals' Problem. Byzantine Failures are the most difficult class of failures. There are no restrictions and no assumptions on the kind of behavior and data a node can inject in the network. BFT is required in airplanes, nuclear power plants and even SpaceX. SpaceX requires it to handle situations where the computers do not agree (e.g., changing values in memory/registry due to radiation). In the context of Byzantine Generals Problem, this is indeed BFT as long as the number of traitors do not exceed 1/3 of the generals. Blockchains are distributed systems with no central authority. What is stored in the ledgers could be of high value so there are relevant economic incentives by malicious nodes to cause faults. With not BTF, a malicious node can transmit false transactions. This would then impede the achievement of integrity.

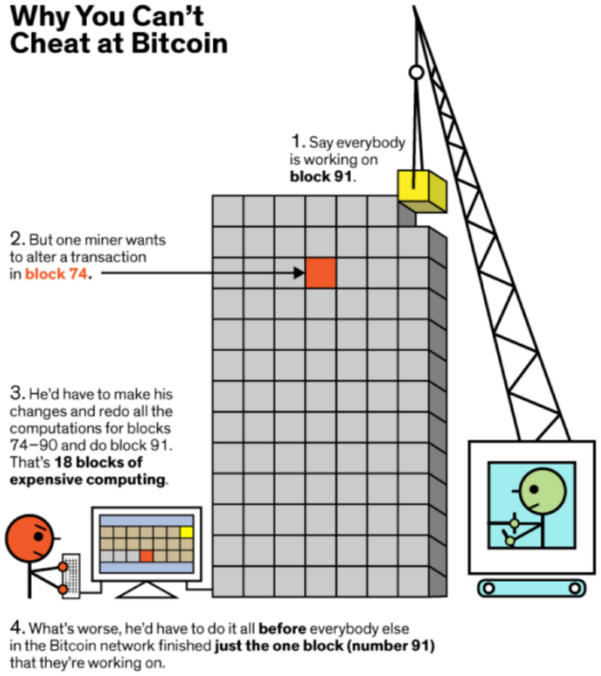
Proof-of-Work (PoW)

Bitcoin solves the Byzantine Generals Problem… Under a probabilistic flavor. Here, the leader is the responsible for transmitting the block to the network so that the other peers can verify it. To be elected as a leader and choose the next block, the network has to solve a mathematical puzzle.

Given data X, find a number n such that the hash of n appended to X is a number less than Y.

```
Y = 10; X = "test"

hash( X )     = hash( "test" )  = 0x0f = 15  # Y > 10
hash( X + 1 ) = hash( "test1" ) = 0xff = 255 # Y > 10
hash( X + 2 ) = hash( "test2" ) = 0x09 = 9   # Y < 10 OK, Solved.
```

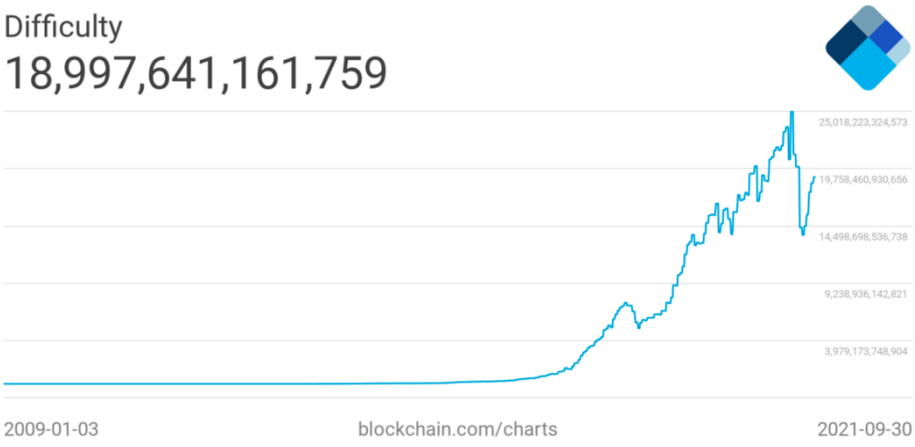


Since hash() is a cryptographic hash function, we must use brute force to find a solution. In probability, the node that will solve the problem first is probably the one with more computing power. These nodes are called **miners**.
It is a very popular algorithm, for which is very hard to find a solution. When the solution is found, it is very easy to verify it. When a miner finds a solution, it gets rewarded —> **incentive!**

An attack to the whole network would cost a lot. Energy, computational power, hardware, potential missed rewards.

Proof-of-Work in Bitcoin
Each block is mined every 10mins. For a safe transaction, it can take up to 1 hour. **Miners are rewarded 6.25 Bitcoins for mining each block**. Remember how computational power increases the chances of breaching algorithms? Mining difficulty increases with time. The mining difficulty is adjusted every 2016 blocks. The difficulty can even go down if the supply of computational power decreases.

Bitcoin Difficulty:	6,379,265,451,411
Estimated Next Difficulty:	6,541,721,510,871 (+2.55%)
Adjust time:	After 1499 Blocks, About 10.3 days
Hashrate(?):	45,867,201,622 GH/s
Block Generation Time(?):	1 block: 9.9 minutes 3 blocks: 29.6 minutes 6 blocks: 59.2 minutes
Updated:	15:0 (200.2 days ago)



Proof-of-Stake (PoS)
Let's start with an analogy, a probabilistic analogy

- Think about a lottery
- If Alice has more tickets than Bob
- Then Alice is more likely to win the lottery.

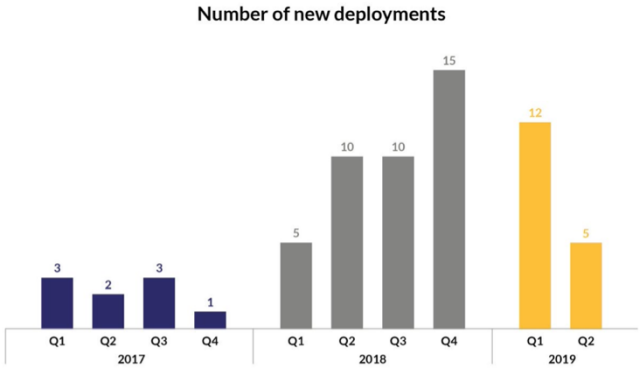
Similarly, under PoW if Alice had more computational power than Alice, she is more likely to be able to mine the next block.
Similarly, under PoS if Alice had more stake than Bob, she is more likely to be able to mine the next block.
The big difference between PoW and PoS is that the latter replaces the computational power with the stake. We refer to the stake as a given amount of currency that a given wallet is willing to lock up and freeze for a certain amount of time. In return, you get a chance of mining the next block proportional to the stake you froze. Issue: **nothing-at-stake** —> nodes are not disincentivized in mining forked chains. Some hybrid consensus algorithms PoS-PoW have been developed.

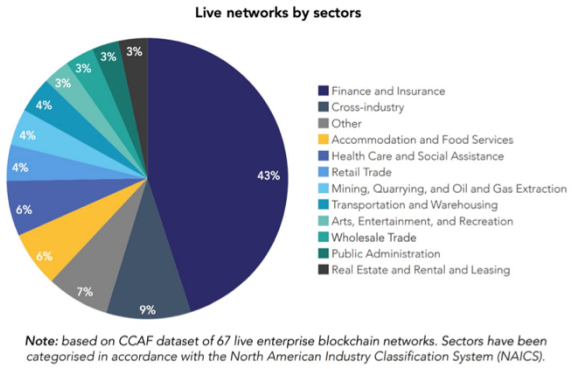
Consensus Algorithms
We need robust consensus algorithms! These algorithms verify the validity of transactions:

- Through them we reach the consensus.
- Through them we avoid the double-spending problem.

Blockchain & Accounting – Class 13

"What is a Blockchain? Is it a Hype?" – Mew York Times

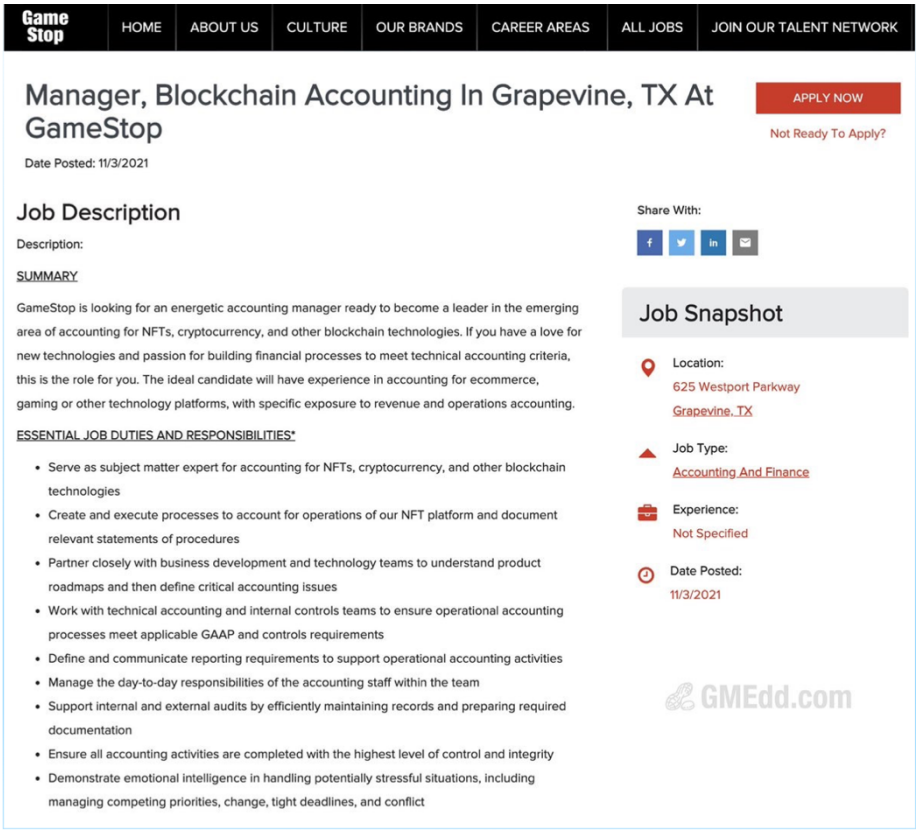




Blockchain Ecosystem evolved, but still early

- In 2017, "the industry landscape was mostly dominated by half-hearted experiments and short-lived proofs-of-concepts – often announced with great fanfare and publicity".
- "[The] hype was gradually given way to genuine development of sustainable blockchain networks that are increasingly being deployed in production environments".
- "77% of live enterprise blockchain networks have little in common with multi-party consensus systems apart from incorporating some of the same technology components (e.g., cryptography, peer-to-peer networking) and using similar nomenclature".
- Attitudes about blockchain may be improving, but 43% [of executives surveyed] still see blockchain as overhyped, up from 39% [in 2018]".

A demand for Blockchain Accounting Managers…



Game Stop | HOME | ABOUT US | CULTURE | OUR BRANDS | CAREER AREAS | ALL JOBS | JOIN OUR TALENT NETWORK

Manager, Blockchain Accounting In Grapevine, TX At GameStop

APPLY NOW

Not Ready To Apply?

Date Posted: 11/3/2021

Job Description

Description:

SUMMARY

GameStop is looking for an energetic accounting manager ready to become a leader in the emerging area of accounting for NFTs, cryptocurrency, and other blockchain technologies. If you have a love for new technologies and passion for building financial processes to meet technical accounting criteria, this is the role for you. The ideal candidate will have experience in accounting for ecommerce, gaming or other technology platforms, with specific exposure to revenue and operations accounting.

ESSENTIAL JOB DUTIES AND RESPONSIBILITIES*

- Serve as subject matter expert for accounting for NFTs, cryptocurrency, and other blockchain technologies
- Create and execute processes to account for operations of our NFT platform and document relevant statements of procedures
- Partner closely with business development and technology teams to understand product roadmaps and then define critical accounting issues
- Work with technical accounting and internal controls teams to ensure operational accounting processes meet applicable GAAP and controls requirements
- Define and communicate reporting requirements to support operational accounting activities
- Manage the day-to-day responsibilities of the accounting staff within the team
- Support internal and external audits by efficiently maintaining records and preparing required documentation
- Ensure all accounting activities are completed with the highest level of control and integrity
- Demonstrate emotional intelligence in handling potentially stressful situations, including managing competing priorities, change, tight deadlines, and conflict

Share With:

Job Snapshot

Location: 625 Westport Parkway Grapevine, TX

Job Type: Accounting And Finance

Experience: Not Specified

Date Posted: 11/3/2021

GMEdd.com

Blockchain and its Impact on Accounting

What we will discuss in the remainder of the course:

- Blockchain and accounting: triple entry accounting.
- Blockchain as an auditing tool.
- Blockchain and the future accounting/auditing profession.
- Blockchain and corporate governance, taxation, etc.

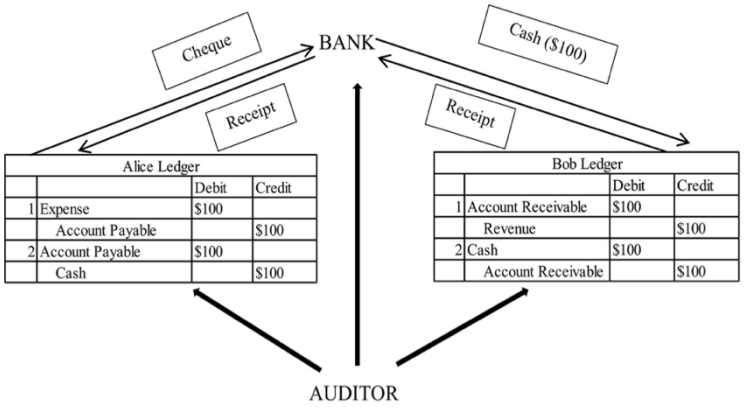
Single Entry Accounting

A one-sided accounting entry for each transaction. Assets are entered and crossed off as they move in and out. Accounting ledger and the dollar amount is recorded once per transaction. Subject to serious limitations as errors cannot be detected and traced, providing ample opportunities for fraud.

Double Entry Accounting

Each financial transaction requires at least two accounting entries (debit and credit). Preserves a verifiable audit trail: as dollar amounts are recorded twice for each transaction on both sides, the total of debits must equal the total of credits. Each debit and credit can be traced back to the original entry and transaction source document. Even if the debits equal the credits, it is possible to do so in a false or misleading manner. As a firm records transactions completed independently and privately, there is the potential for the creation of fabricated transactions. To confirm the integrity of a firm's accounting, shareholders and governments require auditing on a regular basis. Auditing: sampling, timing, costs.

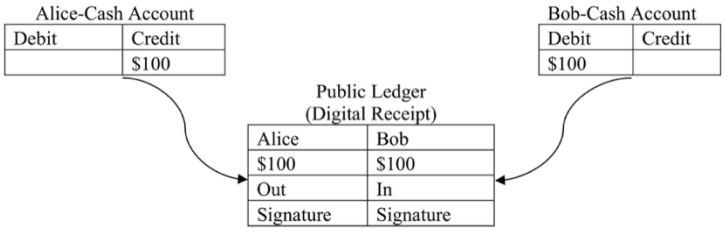
A Payment Transaction in a Double-Entry System



Triple Entry Accounting by Ian Grigg

Triple-entry accounting was a process introduced by financial cryptography expert Ian Grigg in December 2005. Companies should not be the sole recorders of business transactions. A third-party, cryptographically secured entry can be recorded at the same time for transactions between entities. In this third entry, the debit recorded by one entity is the credit recorded by the counterparty.

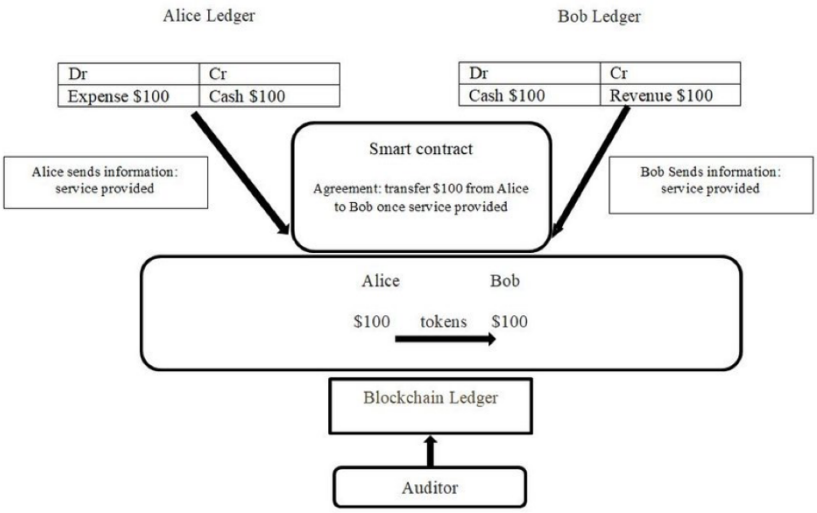
A Payment Transaction in a Triple-Entry System



"It's though to lie when everybody is watching"- Ian Grigg

Unclear who would act as the trusted and neutral third party to control the third shared ledger. The emergence of Bitcoin and its underlying Blockchain protocol demonstrated that a trusted and neutral third-party is NOT required. The third public ledger in Grigg (2005) can be decentralized, immutable, secure and automated using blockchain.

Triple-Entry with a Smart Contract

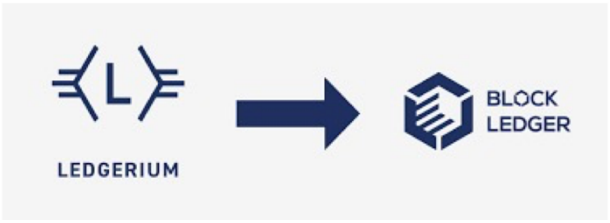


Key Features of a Payment Record on Blockchain Ledger

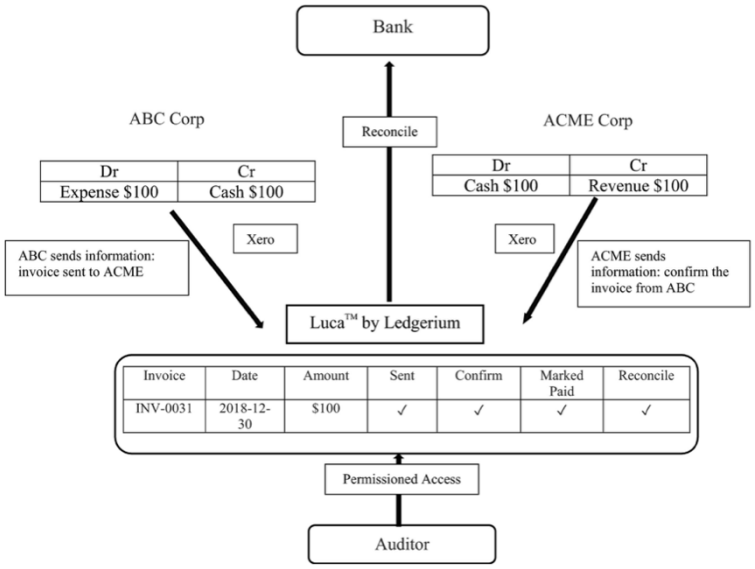
The payment is made in the form of tokens (cryptocurrency) which disintermediates the traditional bank. This payment transaction is recorded in chronological order and this record is permanent without change. If there is an amendment, a new record will be required. This record is not maintained by a centralized server, so security threats are reduced. This record creates a linkage

between the internal records of Alice and Bob so it is less prone to errors and fraud. This record is verifiable, creating an easy audit trail.

LUCA by Ledgerium / BlockLedger



A Payment Transaction Using LUCA



Advantages of LUCA

- A clear, specific, and manageable task: design a public ledger with only two main accounts: accounts receivable and accounts payable.
- Increases the transparency and efficiency of the accounts receivable-payable business cycle.
- Integrates with companies' existing accounting software and banking.
- No extensive changes to companies' internal systems.

Challenges of LUCA

Privacy concerns

- Only committing to hashes of transactions on the ledger – does not support public verifiability.
- Using trusted third parties to independently verify transactions – content revealed.
- Using cryptographic schemes to hide the content of transactions.

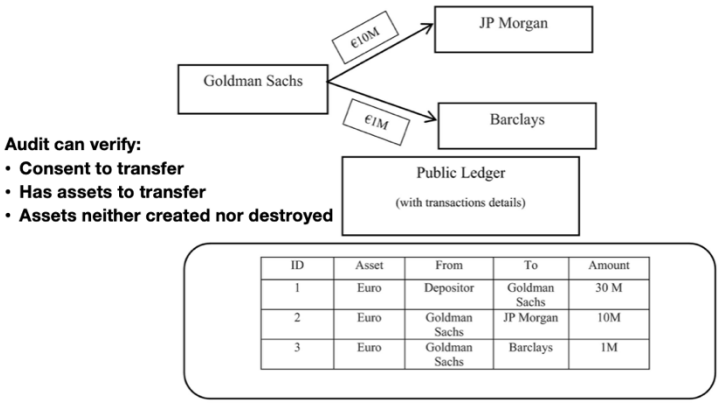
Scalability

- Prevents the mass adoption of blockchain.
- Triple-entry accounting requires both parties to use a common ledger.
- Everybody needs to participate and cooperate.
- This decentralization and the many distributed copies of a blockchain limit the number of transactions per second.

zkLedger – Privacy preserving auditing for distributed ledgers

digital currency initiative | mit media lab

zkLedger – A Record of the Transaction



Bank Care about Privacy

ID	Asset	From	To	Amount
1	Euro	Depositor	Goldman Sachs	30 M
2	Euro	Goldman Sachs	JP Morgan	10M
3	Euro	Goldman Sachs	Barclays	1M

zkLedger

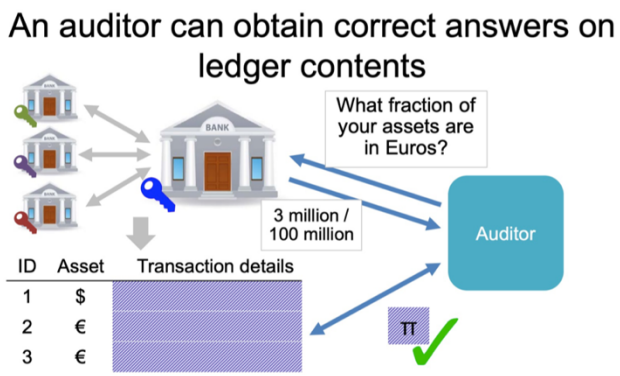
(Transactions are hidden by Pedersen commitments)

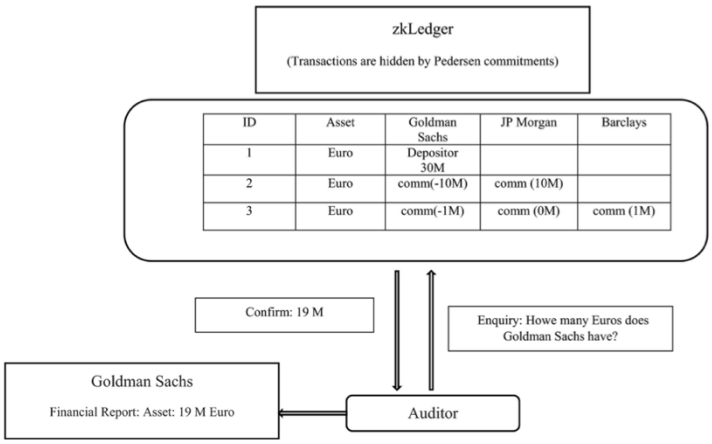
ID	Asset	Goldman Sachs	JP Morgan	Barclays
1	Euro	Depositor 30M		
2	Euro	comm(-10M)	comm (10M)	
3	Euro	comm(-1M)	comm (0M)	comm (1M)

zKLedger – A Private, Auditable Transaction Ledger

- **Privacy**, it hides transacting banks and amounts.
- **Integrity** with public verification, everyone can verify transactions are well-formed.
- **Auditing**, compute provably-correct linear functions over transactions.

How to Audit the Ledger Content





Which Measurements does zkLedger Support

- Ratios and percentages of holdings
- Sums, Averages, Variance, Skew
- Outliers
- Approximations and Orders of Magnitude
- Changes over time
- Well-known Financial Risk Measurements

Advantages of zkLedger

- **Privacy**, the auditors and non-involved parties cannot see transaction participants or amounts,
- **Completeness**, banks cannot lie to the auditor or omit transactions.
- **Integrity**, banks cannot violate financial invariants (Honest banks can always convince the auditor of a correct answer).
- **Progress**, a malicious bank cannot block other banks from transacting.

Challenges of zkLedger

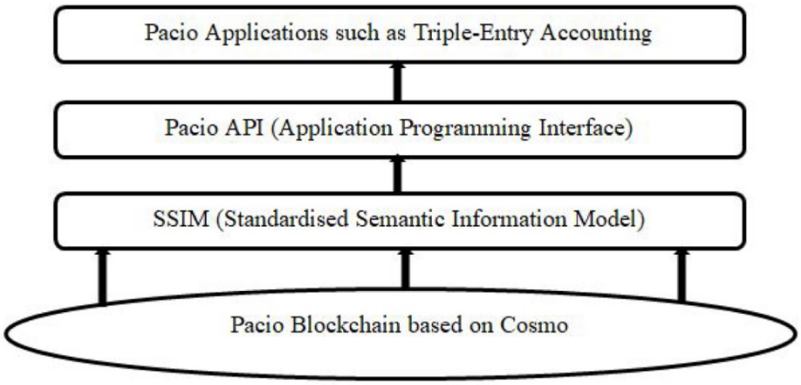
- Banks might attempt to steal or hide assets, manipulate balances, or lie to the auditor.
- Banks can arbitrarily collude.
- Banks or the auditor might try to learn transaction contents.
- Scalability remains an issue.

COSMOS Network

It is a decentralized network of independent, scalable, and interoperable blockchains, creating the foundation.

Pacio Solution

"Pacio will lead the world into the next advance in business record keeping – Triple Entry Accounting (TEA) – and associated systems and services to reduce the mid-decade $27 trillion per annum opportunity losses and costs resulting from current accounting and management deficiencies" – David Hartley, CEO of Pacio.



A blockchain ecosystem with triple-entry accounting.

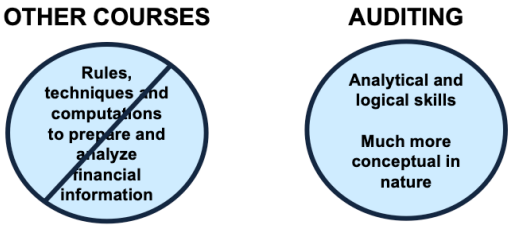
Potential Benefits of Triple-Entry Accounting

- Introduces a public ledger where all participating entities mandatorily host all accounting entries.
- Removes dependency on auditors to verify accuracy and completeness of financial statements.
- Establishes a self-regulated and shared environment amongst all stakeholders.
- Less errors and fraud, tamper-proof and audit trail.
- Streamlines reconciliations and financial recording.
- "Last but not least, with triple-entry accounting, for the very first time, we can seamlessly follow the world's money."- David Hartley.

Blockchain & Accounting – Class 15

Blockchain as an auditing Tool

One of the main areas where blockchain can be applied in accounting is Auditing. The study of auditing is different from other accounting courses that you have taken in college because…

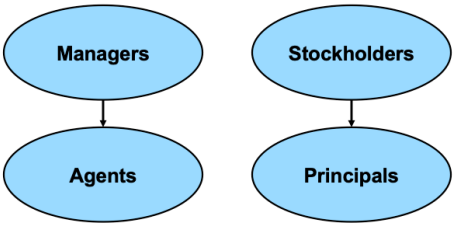


The Demand for Auditing and Assurance

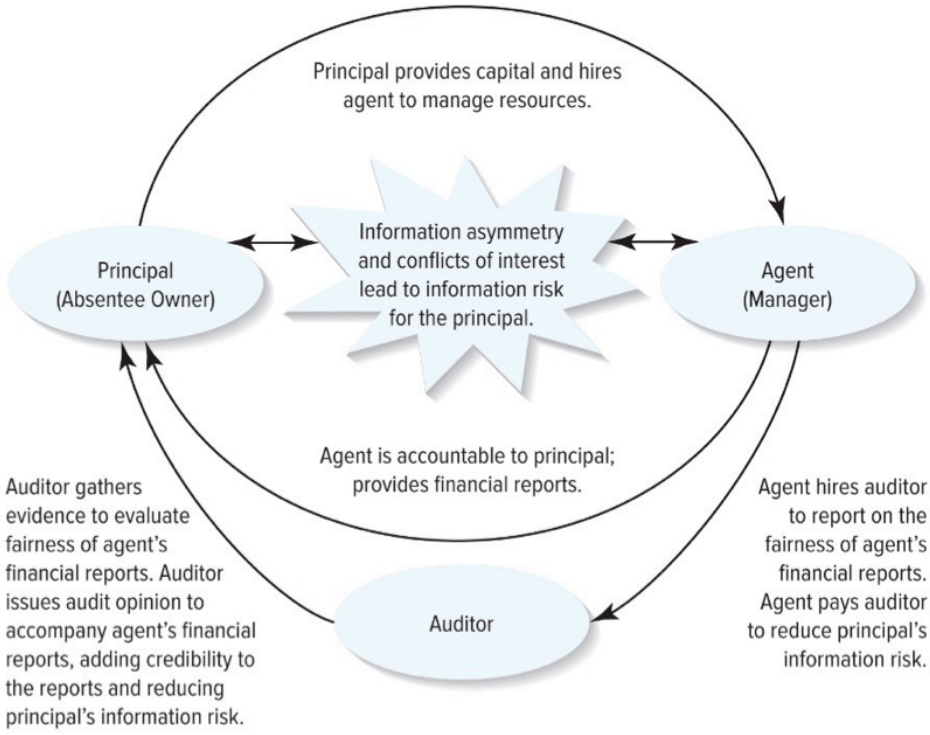
The development of the corporate form of business and the expanding world economy over the last 200 years have given rise to an explosion in the demand for assurance provided by auditors.

Principals and Agents

A public company is a company that sells its stocks or bonds to the public, giving the public a valid interest in the proper use of the company's resources.



Principal-Agent Relationship and Demand for Auditing



Auditing Demands Logic, Reasoning and Resourcefulness

An auditor needs to understand more than just the accounting concepts and techniques. Auditing is a fundamentally logical process of thinking and reasoning – so use your common sense and reasoning skills. Being a good auditor sometimes requires imagination and innovation. Understanding audit concepts is useful for all business professionals, consultants, etc.

Types of Auditors

- External Auditors
- Internal Auditors
- Government Auditors
- Forensic Auditors

Types of Audit Services

Audit Services	Internal Control Audits, Compliance Audits, Operational Audis, Forensic Audits
Attest Services	Reporting on nature and quantity of inventory stored in a company's warehouse so that the company can obtain a bank loan
Assurance Services	Auditing is a specialized form of assurance service
Non-audit Services	Tax Preparation and Planning Services, Management Advisory Services, Compilation and Review Services

Many Regulators and Standard-Setters Affect Auditing

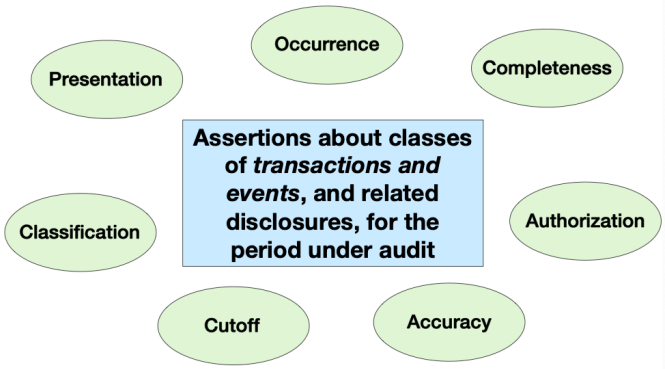
In the US:

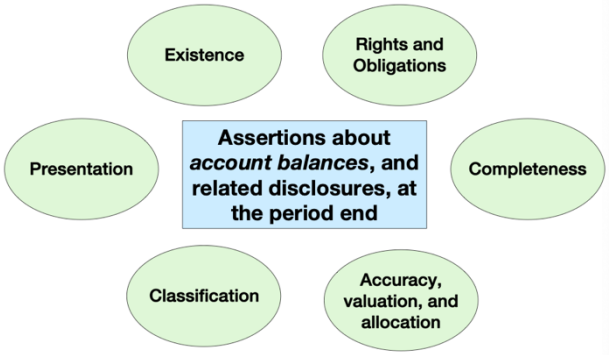
- SEC Securities and Exchange Commission
- AICPA American Institute of Certified Public Accountants
- PCAOB Public Company Accounting Oversight Board
- FASB Financial Accounting Standards Board
- ASB Auditing Standards Board

Society's Expectations and the Auditor's Responsibilities
The auditor's responsibility is to provide **reasonable assurance** that the financial statements are free of material misstatement, whether caused by error, fraud, or illegal acts. Because of the nature of audit evidence and the characteristics of fraud, the auditor is able to obtain **reasonable**, but not **absolute**, assurance that material misstatements are detected.

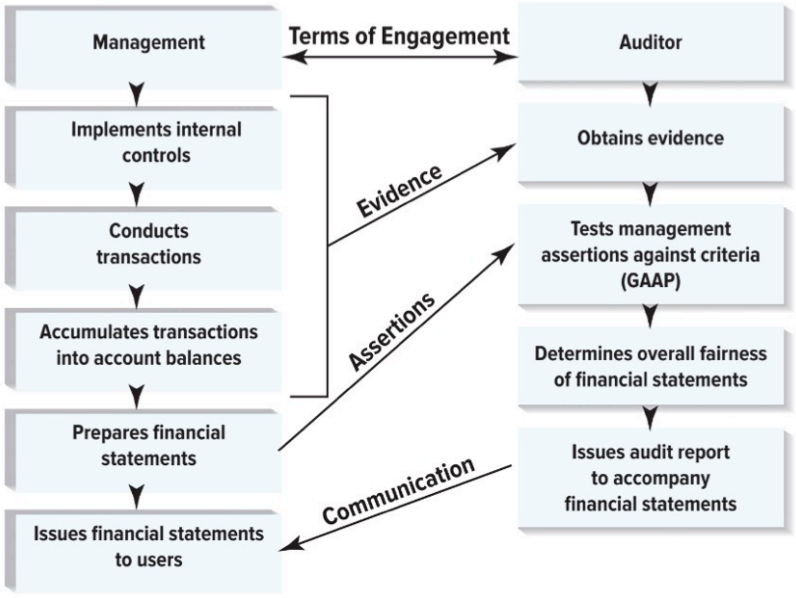
Responsibility for the Financial Statements
While auditors have important responsibilities, **management** is primarily responsible for maintaining effective internal control and for ensuring the fairness of the company's financial statements.

Management Assertions





Overview of Financial Statement Audit



Major Phases of an Audit

Fundamental Auditing Concepts

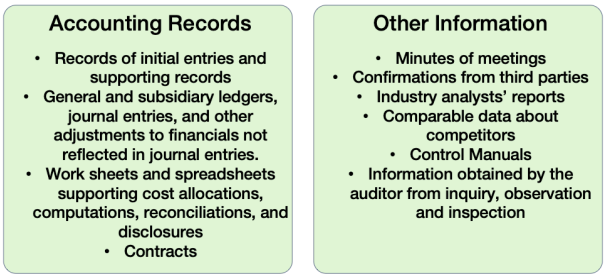
- **Materiality**
  The magnitude of an **omission** or **misstatement** of accounting information that, in light of surrounding circumstances, makes it probable that the judgement of a reasonable person relying on the information would have been changed or influenced by the omission or misstatement.

- **Audit Risk**
  Audit risk is the risk that the auditor mistakenly expresses a clean audit opinion when the financial statements are materially misstated.
  Auditing standards make it clear that the audit provides only **reasonable assurance** that the financial statements do not contain material misstatements.
  Reasonable assurance implies some risk that a material misstatement could be present in the financial statements and the competent auditor will fail to detect it.

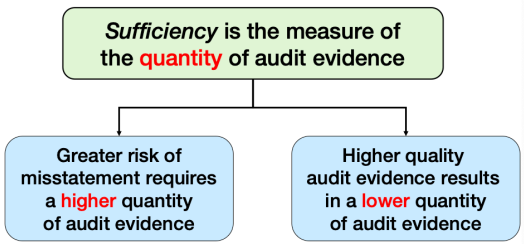
Knowledge Assessment
Which of the following best describes the concept of audit risk?

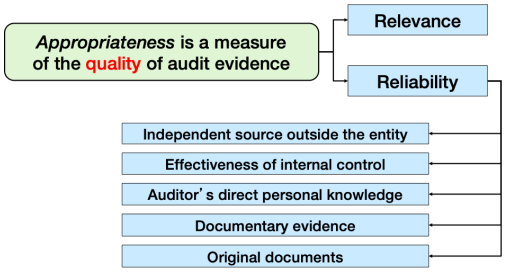
A. The risk of the auditor being sued because of association with an auditee.
B. The risk that the auditor will provide an unqualified opinion on financial statements that are, in fact, materially misstated.
C. The overall risk that a material misstatement exists in the financial statements.
D. The risk that auditors use audit procedures that are inappropriate.

- **Audit Evidence**
  The information used by the auditor in arriving at the conclusions on which the audit opinion is based.
  - **The Concepts of Audit Evidence**
    - **Nature of Audit Evidence – Refers to the form or type of information**



    - **Sufficiency and Appropriateness of Audit Evidence**





    - **Evaluation of Audit Evidence**
  - **Audit Evidence Regarding Management Assertions**
    Evidence that assists the auditor in evaluating management's financial statement assertions and consists of the underlying accounting data and any additional information available to the auditor, whether originating from the client or externally.
    - **Relevance** – Is the evidence related to the specific assertion being tested?
    - **Reliability** – Can the evidence be relied upon to signal the true state of the specific assertion being tested?
  - **Sampling Inferences Based on Limited Observations**

Auditors use a sampling approach to examine a subset of the transactions based on previous audits, an understanding of the company's internal control system, or knowledge of the company's industry.

- It would be too costly for the auditor to examine every transaction.
- Data analytics will sometimes allow for testing entire populations.

Knowledge Assessment

Why do auditors generally use a sampling approach to evidence gathering?

A. Auditors are experts and do not need to look at much to know whether the financial statements are correct or not.
B. Auditors must balance the cost of the audit with the need for precision.
C. Auditors must limit their exposure to their auditee to maintain independence.
D. The auditor's relationship with the auditee is generally adversarial, so the auditor will not have access to all of the financial information of the company.

Which of the following sources of evidence are more reliable?

1. Inquiry of an accounts receivable clerk regarding the accounts receivable balance – or – **Accounts receivable confirmations sent to a sample of customers.**
2. **Physical examination of lumber inventory performed by the external auditor** – or – physical examination of inventory performed by internal auditors.

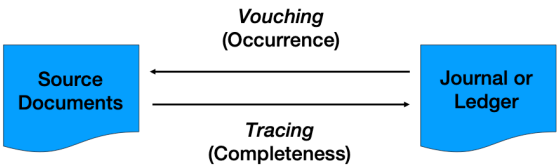
Examples of Audit Procedures to Collect Evidence

Audit Program for Accounts Receivable	
Management Assertions	**Example Audit Procedures**
Existence	Confirm accounts receivable
Rights and obligations	Inquire of management whether receivables have been sold
Completeness	Agree total of accounts receivable subsidiary ledger to accounts receivable control account
Valuation or allocation	Test the adequacy of the allowance for doubtful accounts

Collection Evidence: Inspection of Records

Inspection of records and documents

- Evidence obtained from external documents is more reliable than evidence obtained from internal documents



What May Change: Inspection

Current Method	BC-based Method
Vouch, trace, and scan different forms of records or documents (paper form, electronic form, other media)	Inspect documents that support the configuration and governance of the peer work which is operating the blockchain process
Sampling method	Inspect entire transactions recorded on blockchain continuously

Collection Evidence: Inspection of Assets/Observation

Inspection of tangible assets

- Physical examination of a tangible assets

Observation

- The process of watching a process or procedure being performed by others

What May Change: Observation

Current	BC-based
Look at a process or procedure being performed by the entity's personnel on site Observe the performance of control activities	Blockchain itself provides proofs of observation via consensus New blockchain risk control observation: If timestamps are added, if transaction information being hashed or if the hashing algorithm exists etc

Collection Evidence: Inquiry

Inquiry

In conducting inquiry, the auditor should:

- Consider the knowledge, objectivity, experience, responsibility, and qualifications of the individual to be questioned
- Ask clear, concise, relevant questions
- Use open or closed questions appropriately
- Listen actively and effectively
- Consider the reactions and responses and ask follow-up questions
- Evaluate responses

What May Change: Inquiry

Current	BC-based
Seek information of knowledgeable persons (written or oral inquiries)	Obtain evidence from the peer network/ transaction participants

Collection Evidence: Confirmation

Confirmation

- Audit evidence obtained by the auditor as a direct written response to the auditor from a third party

The reliability of evidence obtained through external confirmations may be affected by factors such as:

- The form of the confirmation.
- Prior experience with the entity.
- The nature of the information being confirmed.
- The intended respondent.

What May Change: Confirmation

Current	BC-based
Verify accounts balances and elements from third party in paper, electronic or other medium Waiting periods are expected	Automatic communication/reconciliation with members of the peer network Online and in real-time Reconciliation requests and results are recorded and stored on blockchain in time

Collection Evidence: Recalculation/Reperformance

Recalculation

- Checking the mathematical accuracy of documents or records

Reperformance

- The auditor's independent execution of procedures or controls that were originally performed by company personnel

What May Change: Recalc./Reperf.

Current	BC-based
Verify the mathematical accuracy of documents or records figures manually or electronically Independently re-execute the entity's procedures or internal controls if needed	Verify transaction blocks with the peers of the network automatically based on the verifiability of distributed ledger technology Automatically re-execute transactions and identify exceptions

Collection Evidence: Analysis and Scanning

Analytical Procedures

- Evaluations of financial information made by a study of plausible relationships among both financial and nonfinancial data

Scanning

- Judgmentally review accounting data to identify significant or unusual items to test

What May Change: Analytical Procedures

Current	BC-based
Analyze and compare financial and nonfinancial data to understand changes in business Data analysis	Real-time monitor and analysis on a continuous basis or desired intervals

What Can Blockchain Change in Auditing

- **Governance, Transparency and Trust**
  A "truth machine"- unprecedented levels of trust and transparency - (Casey and Vigna, 2018)

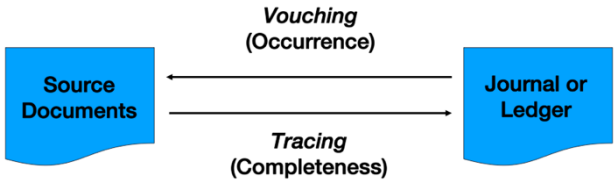
- Peer-to-peer auditing with no institutional intermediation
- Transparent and immutable accounting records
  - Manipulating and destroying is virtually impossible
  - Disclose off-book and hidden transactions
- Immediate access to accounting data
- Differentiated access to information on private, permissioned blockchains

- **Continuous Auditing**
  - Contemporary audit is labor intensive and mostly retrospective. It requires the approval of transactions and balances at the end of reporting periods. If transactions are recorded and validated in real-time, auditors could move to an "always-on", "real-time auditing", continuously.
  - **Blockchain-Enabled Continuous Auditing**
    - Monitoring what happens in real time not in retrospect.
    - Combining the processing of transactions with the recording and reconciling > efficiencies.
    - Sampling > up-to-date, immutable historical audit trail of all transactions.
    - Real-time systems highlight anomalies at the time of occurrence > timely fraud investigations.
    - Better auditors' understanding of clients' businesses.

- **Smart Contracts**
  - Smart contracts extend blockchains' utility from simple record-keeping of transaction entries to automatically implementing terms of multiparty agreements.
    - Allow autonomous recording of transactions in compliance with agreed terms: automatic audit review and verification.
    - Automate transaction reconciliation procedure while providing more transparency to stakeholders.
    - Saves time and human error
  - Accounting rules can be encoded into smart contracts
    - Auditors monitor if transactions are compliant with these accounting rules and highlight cases of mismatch.
    - Smart contracts can revoke transactions if the system detects that rules and standards encoded into the contract are disobeyed.
    - May offer a predicting function, for instance by encoding default or credit rating prediction model, monitoring debtor's default risk based on financial data, and adjust bad debt estimations accordingly.
  - Artificial Intelligence technologies may extend the potential of smart contract applications to the assessment and recording of the physical conditions of goods.
  - In combination with AI, smart contracts could detect and measure damage on inventory and other assets and potentially automate the accounting measurement of those assets.

Transaction Verification

"Just because a transacted record is computerized and "blockchained" does not necessarily imply that its physical world counterpart material of commerce has not been tampered with" - (Apte and Petrovsky, 2016).



Are Blockchains completely fraud-free?

- Committing fraud is still possible on blockchains, as "lies encoded on the blockchain are still lies. They're just immutable lies" – (Bradbury, 2015).
- Cannot eliminate fraud completely but may help identify fraud in real-time. - (Wang and Kogan, 2018).

What are the Auditing Challenges in Blockchain?

- Only financial statement audit, not audit of internal controls, compliance.
- Not all assertions.
- Transaction can still be fraudulent, illegal, misclassified, related-parties, side-agreement.
- Record in BC != authenticity.
- No help with estimates, complex confirmations, valuation of complex instruments.

Blockchain & Accounting – Class 20

What Blockchain Offers to Auditors

- Distributed ledger:
  - Each user has a live copy of the ledger → real time updates and reconciliations.
  - No single point of failure
  - No intermediaries of central authorities
- Immutability and consensus protocols
  - Validated by majority, reply on cryptographic proofs
  - Chronological, tamperproof → audit trail
  - Difficult (but not impossible) for fictious and erroneous transactions
- Resilience → more security and robustness → increased reliability of accounting records.

What Can Blockchain Change in Auditing

- **Continuous Auditing**
  - Contemporary audit is labor intensive and mostly retrospective.
  - Requires the approval of transactions and balances at the end of reporting periods.
  - If transactions are recorded and validated in real-time, auditors could move to an "always-on", "real-time auditing", continuously.
  - **Definition (1999)**: a continuous audit is a methodology that enables independent auditors to provide written assurance on a subject matter, for which an entity's management is responsible, using a series of auditors' reports issued virtually simultaneously with, or short period of time after, the occurrence of events underlying the subject matter.

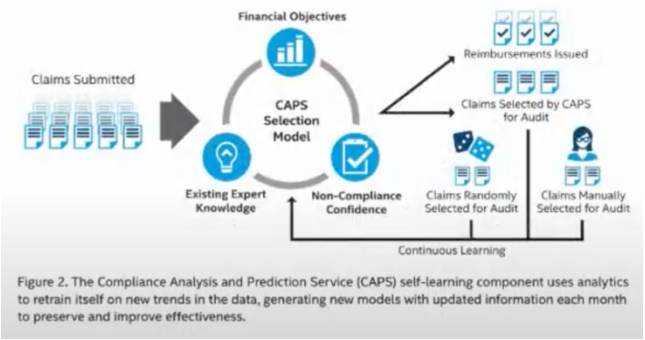
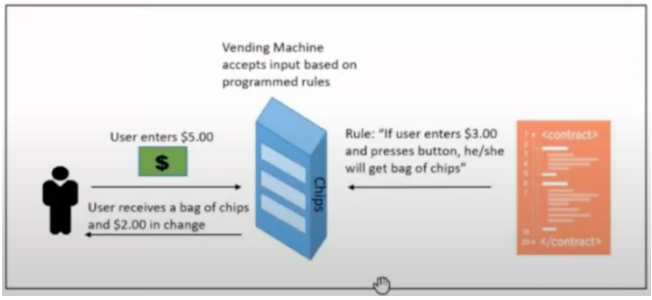


Figure 2. The Compliance Analysis and Prediction Service (CAPS) self-learning component uses analytics to retrain itself on new trends in the data, generating new models with updated information each month to preserve and improve effectiveness.

  - Monitoring what happens in real time not in retrospect.
  - Combining the processing of transactions with the recording and reconciling > efficiencies.
  - Sampling > up-to-date, immutable historical audit trail of all transacitons.
  - Real-time systems highlight anomalies at the time of occurrence > timely fraud investigations.
  - Better auditors' understanding of clients' businesses.

- **Smart Contracts**

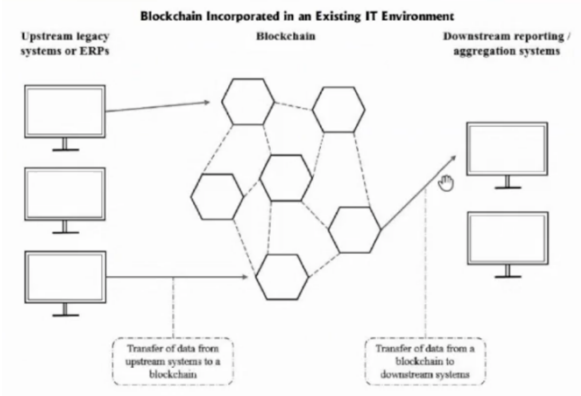


  Smart contracts extend blockchains' utility from simple record-keeping of transaction entries to automatically implementing terms of multiparty agreements.
  - Allow autonomous recording of transactions in compliance with agreed terms: automatic audit review and verification.
  - Automate transaction reconciliation procedure while providing more transparency to stakeholders.
  - Saves time and human error.
  - Smart contracts are "computerized transaction protocol that executes the terms of a contract".

  Accounting rules can be encoded into smart contracts:
  - Auditors monitor if transactions are compliant with these accounting rules and highlight cases of mismatch.
  - Smart contracts can revoke transactions if the system detects that rules and standards encoded into the contract are disobeyed.
  - May offer a predicting function, for instance by encoding default or credit rating prediction model, monitoring debtor's default risk based on financial data, and adjust bad debt estimators accordingly.
  - Artificial Intelligence technologies may extend the potential of smart contract applications to the assessment and recording of the physical conditions of goods.
  - In combination with AI, smart contracts could detect and measure damage on inventory and other assets and potentially automate the accounting measurement of those assets.
  - Smart audit procedure can help reduce the expectation gap between the procedures auditors perform versus those procedures audit inspectors, and investors, expect them to perform.

- **Blockchain as an IT System**

  Verification of BC reliability and Environment



  - Central locations to obtain audit data & evidence directly from blockchains.
  - Still risk that the information is inaccurate due to error or fraud.
  - This will present new challenges because a blockchain likely would not be controlled by the entity being audited.
  - Need to extract the data from the blockchain and also consider whether it is reliable.

  IT Control Environment
  - With a blockchain consortium, this governance becomes complex.
  - Does the consortium has a steering committee in place, the powers granted to this committee, and the voting power of each member?
  - Resources committed by each member (financial, personnel, or computational), and what evidence that proper controls are implemented and maintained across the blockchain (independent attestation).

  Evaluation of Consensus Protocols

- Which nodes are authorized to participate in consensus.
- Distribution of computational power among members.
- Approach to consensus (e.g., POW, POS, all nodes validate all blocks and consensus is reached when a majority agree on the current state).
- Balance of computational power among validating nodes.

Change Management (CM)

- Changes to systems require a controlled process to request, develop, test, authorize, and implement the update.
- With blockchain, changes occur with:
  - Consensus protocol
  - Communication protocol
  - Smart contracts
    - May involve a change in a referenced oracle
    - What if change is not accepted by all
  - Decentralized applications, submitted as transactions, virtually unstoppable once loaded.
  - Source code of the chain, chain might pause while updates load
- CM committee with elected representatives or one official from each member organization.
  - Mechanism to agree on changes is crucial.

Interface with Other Applications

- Important to maintain and enhance controls over legacy systems
  - Data is tamper-proof in BC, but vulnerable outside of BC
  - Upstream interface might be particularly critical
- Connection with third-party oracles
  - Data not compromised, free of bugs
  - Properly linked and coded
  - Physical control of IoT devices etc.

Application and Infrastructure Access

- User/node access provisioning (grant/modify/remove)
  - Permission to submit, relay, verify transactions or validate new blocks.
  - Formal request and approval and subject to periodic reviews.
- Access to manipulate smart contracts, decentralized applications, protocols and source code, who in the consortium provisions/revokes access.
- If private keys are used. Storage, how to regain control. E.g., access to terminated employees.
- Robust network security.

Some IT Risks are Potentially Eliminated

- Data Backup
- Disaster Recovery
- Batch Processing
- Unauthorized changes to historical data

- **New Auditing Roles**

Auditors of Smart Contracts

- Smart contracts embedded to automate business processes.
- Verification that smart contracts are implemented with the correct business logic. Verify the interface between smart contracts and external data sources.
- Without an independent evaluation, users of blockchain technologies face the risk of unidentified errors or vulnerabilities.

- In the context of a financial statement audit, management will be responsible for establishing controls to verify the smart-contract source code is consistent with the intended business logic.

Service Auditor of Consortium Blockchains

- Prior to launching a new application on an existing blockchain platform or leveraging or subscribing to an existing blockchain product, users of the system may desire independent assurance as to the stability and robustness of its architecture.
- Critical blockchain elements (e.g., cryptographic key management) should be designed to include sophisticated ITGCs that provide ongoing protection for sensitive information, as well as processing controls over security, availability, processing integrity, privacy and confidentiality.

Administrator Function

- Permissioned blockchain solutions may benefit from a trusted, independent and unbiased third-party to perform the functions of a central access-granting administrator.
- Verification of identity or a further vetting process to be completed by a participant before they are granted access to a blockchain.
- This central administrator could validate the enforcement and monitoring of the blockchain's protocols. If this function is performed by a user/node of the blockchain, then an undue advantage could exist and trust among consortium members could be weakened.
- Create trust for the blockchain as a whole:
    - Legal considerations?
    - How to ensure independence?
    - Can be combined with financial statement audits?

Arbitration Function

- Business arrangements can be complex and result in disputes between even the most well-intentioned parties.
- For a permissioned blockchain, an arbitration function might be needed in the future to settle disputes among the consortium-blockchain participants.
- Analogous to the executor of an estate trust. Participants on the blockchain may require this type of function to enforce contract terms where the spirit of the smart contract departs from a legal document, contractual agreement or letter.
    - What legal framework would be used to settle disputes?
    - Could this role create unintended threats to independence?

Auditors Already Have Important Skills

- Unstructured data
- Big data, data analytics and data visualizations
- Efficiency of audits and speed of validation.
- Computational capabilities: move from sampling to entire population.

Yet the Profession is Slow to Adapt…

- Auditor social intelligence: client relationship management, collection of inquiry evidence, and detecting management's intentions to commit financial statement fraud.
- Business domain still critical.
- Many audit conclusions are still highly subjective, e.g., "going concern".
- Other fields (computer science) may start providing audit functions.

- **Points of Caution**

Auditing Challenges in Blockchain

- Only financial statement audit
- Transaction can still be fraudulent, illegal, misclassified, related-parties, side-agreement.
- Record in BC != authenticity
- No help with estimates, complex confirmations, valuation of complex instruments.
- Lack of guidance on risk and audit evidence for cryptoassets.
- Blockchain environment and reliability.

Blockchain & Accounting – Class 21

Corporate Governance is Not New

"The directors of companies, being the managers of other people's money rather than their own, cannot well be expected to watch over it with the same anxious vigilance with which (they) watch over their own" – Adam Smith, The Wealth of Nations, 1776.

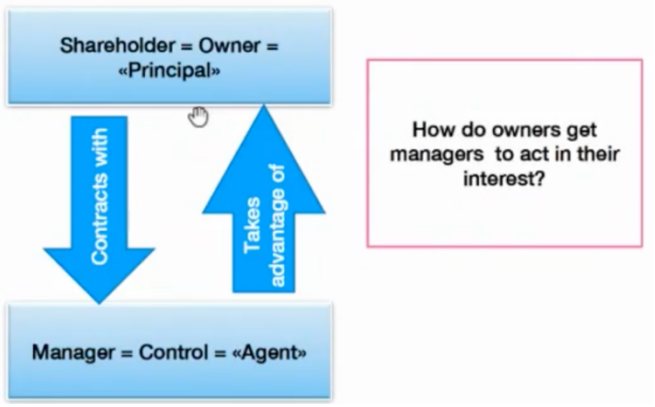
Financial Scandals and Collapses

Enron, Parmalat, Satyam, Olympus Corporation, …

The Root Cause of Failures

- Consummate greed for money, power or both (**self-interest**).
- Lack of **detection** and punishment for unethical actions.
- Lack of individual **integrity** (knowledge that certain actions are inherently wrong even if they are undetected and left unpunished).
- Lack of consideration for others affected.

Agency Theory



Corporate Governance Definition

- No uniform definition among scholars.
- "A **collection of control mechanisms** that an organization adopts to prevent potentially self-interested managers from engaging in activities detrimental to the shareholders and stakeholders".

In Other Words…

Corporate governance is a set of principles and policies

- By which a company is **directed.**
- Which influences the **rights and relationships** among stakeholders.
- And ultimately how a firm is **managed**.

Some Mechanisms of Corporate Governance

- Board of Directors
- CEO role and compensation

- Regulatory framework of the country
- Audit
- Market for control
- Shareholder activism
- Ethics and tone from the top

Blockchain: Transparency and Trust

A "truth machine" – unprecedented levels of trust and transparency.

- Peer-to-peer auditing with no institutional intermediation.
- Transparent and immutable accounting records
  - Manipulating and destroying is virtually impossible.
  - Disclose off-book and hidden transactions.
- Immediate access to accounting data.
- Differentiated access to information on private, permissioned blockchains.

BD as a Ledger for Securities Transactions

- Slow info dissemination on transactions with company's security leads to "empty voting".
- Clearing and settlement as a solution
- Examples of security exchange pilots with blockchain/DLT
  - **Depositary Trust and Clearing Corporation** (DTCC) finished a PoC of project Ion, an alternative settlement platform that leverages distributed ledger technology.
  - **Australian Stock Exchange** CHESS – delayed until 2023.
  - **Japan Stock Exchange** DLT trial – ongoing latest update on pilot in Dec. 2020.
  - **Luxemburg Stock Exchange** trial for BC-based platform for investment fund industry.

Smart Contracts to Mitigate Conflicts in the Board of Directors

- **Shareholders' e-voting**
  - Will allow board to focus on strategic matters and less administrative.
  - Selection of board of director replacement members instantaneous/
- Auditing and accounting exceptions as smart contracts.
- Examples:
  - NASDAQ/Estonai e-voting in remote annual general meetings.
  - Central Securities Depositaries (CSD) Consortium for general meeting proxy voting on DLT.

BC in Compensation Schemes

- "Smart contracts could be used to enable employees to be paid on an hourly or daily basis with taxes remitted to a governmental body in real-time".
- "Smart contracts may be used for compensation, and for authentic payments when performance goals are achieved".
- Compensation using BC-based tokens.
- Lack of regulation and still prone to fraud.

BC and Shareholder Activism

- Hostile takeovers may lead to accumulation of shares in a target company which will allow to block certain managerial decisions.
- Visibility, transparency or ownership, immediate market reaction to takeover.
- Abnormal returns due to name change or disclosures related to blockchain.

Governance as a Broader Concept

- Government: the office, authority or function of governing.
- Governing: having control or rule over oneself.
- Governance: the activity of governing. A set of decision and processes made to reflect social expectations through the management or leadership of the government.

Blockchain-Based Governance

- Efficient
- Decentralized
- Consensus Driven

Some Principles of the BC-Based Governance

- State as a Single Point of Failure
- Distributed Architecture and Trust-by-Computation – "Code is Law" – Lessig 1999
- Power of individuals by instant, atomic interactions
- A do-it-yourself public administration
- Borderless, globalized government services
- Authority floating feely and societal maturity

Decentralized Autonomous Organizations (DAO)

"… a concept derived from AI. Here, a decentralized network of autonomous agents perform tasks, which can be conceived in the model of a corporation running without any human involvement under the control of a set of business rules. In a DAO, there are smart contracts as agents running on blockchains that execute ranges of prespecified or preapproved tasks based on events and changing conditions" – M. Swan.

The DAO

- Slock.it created The DAO – an investment fund – using the theoretical framework of decentralized autonomous organization.
- The DSO was to be managed by software only.
- The DAO was launched April, 2016 going live with roughly $150 million worth of ETH contained within its contract.
- More democratic investment logic, effective economic decisions, independent of private and public bodies.
- There was an initial two-weeks "debate period" during which the community was supposed to decide how to allocate funds, and which projects were most attractive to the investors.
- Cryptographers and academics warned about flaws in the code.

"The DAO's smart contract code governs the Creation of DAO tokens and supersedes any public statements about The DAO's Creation made by third parties or individuals associated with The DAO, past, present and future".

The DAO Hack

- The DAO was "hacked" on June 17, 2016 by an anonymous "hacker" that used the terms and conditions of the smart contracts in such a way that about 40 to 50 million USD could be diverted from the fund.
- Virtual version of an out-of-control ATM.
- "Using a legal loophole to effect a result that was clearly within the letter of the law, but now within its spirit".

- In response, the majority of the shareholders in the blockchain decided to recapture the funds, thereby actually altering the allegedly immutable code and undermining trust between parties as one of the pillars of the blockchain technology.
- The SEC investigation.

Potential Issues with BC Governance

- Trade-oof between network dimension and decentralization.
    - Scalability leads to technical centralization.
    - Secretive operations, collusions, cartels.
    - Private interests.
- Inherent volatility of BC
    - Can be forked or dismissed, becomes obsolete.
    - Invalidates contract and governance services.
    - Reliant on network connectivity.