Law, Science and Technology
MSCA ITN EJD n. 814177

RIoE
Rights of Internet of Everything

Law and Smart contracts

Mirko Zichichi

Outline

- “Law + Technology” approach
  - Smart Contract and its variety
  - Oracles
- Smart Legal Contract
  - From trust in the contracting party to trust in the code in the execution of the contract
  - Issues
- Intelligible Contract
  - Ricardian Contract

RIOE
Rights of Internet of Everything

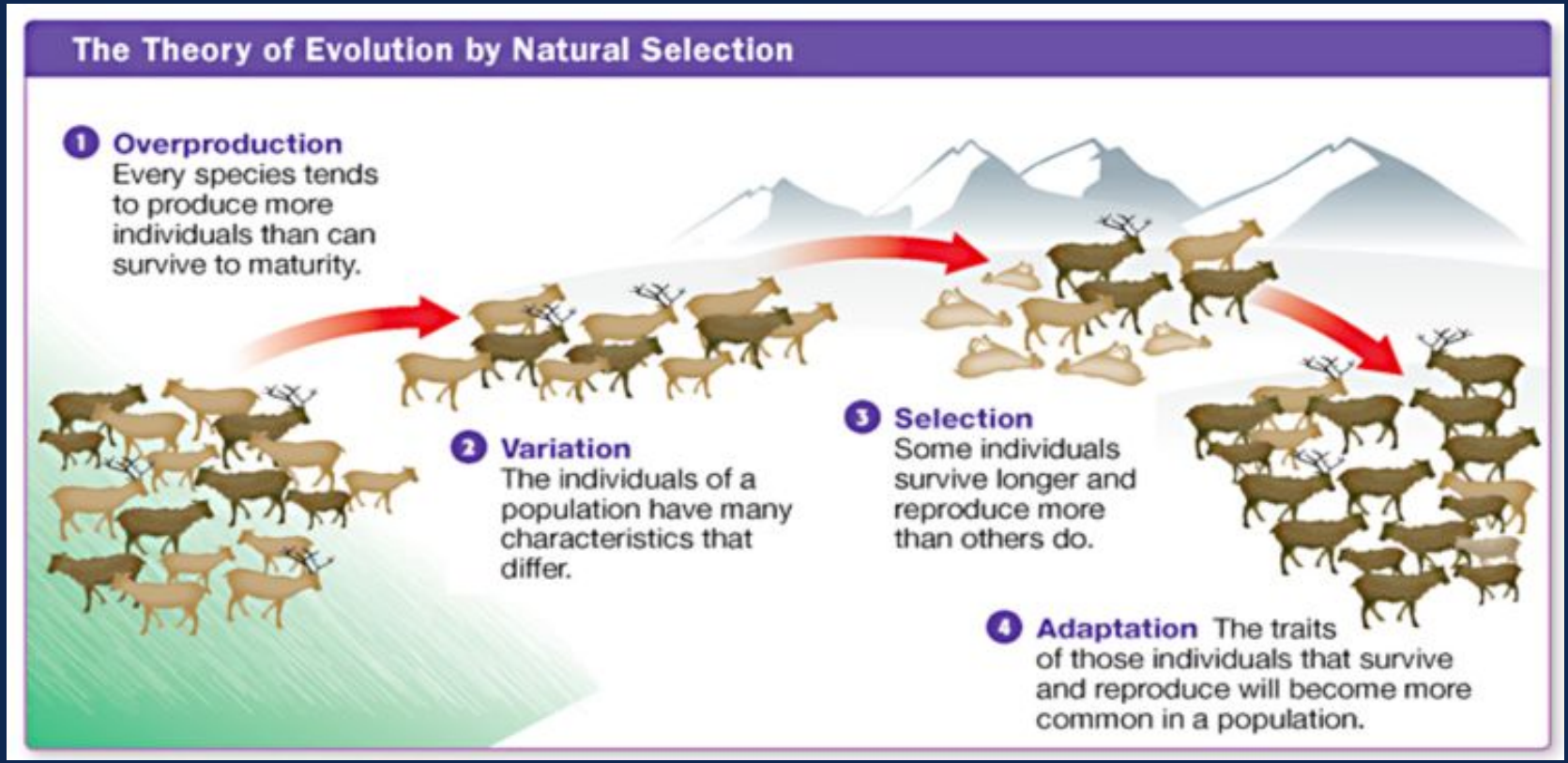
Law + technology

Approach

“Smart Contracts and the Digital Single Market Through the Lens of a ‘Law + Technology’ Approach”, DR. THIBAULT SCHREPEL, LL.M. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3947174

RIoE
Rights of Internet of Everything

Evolution point of view

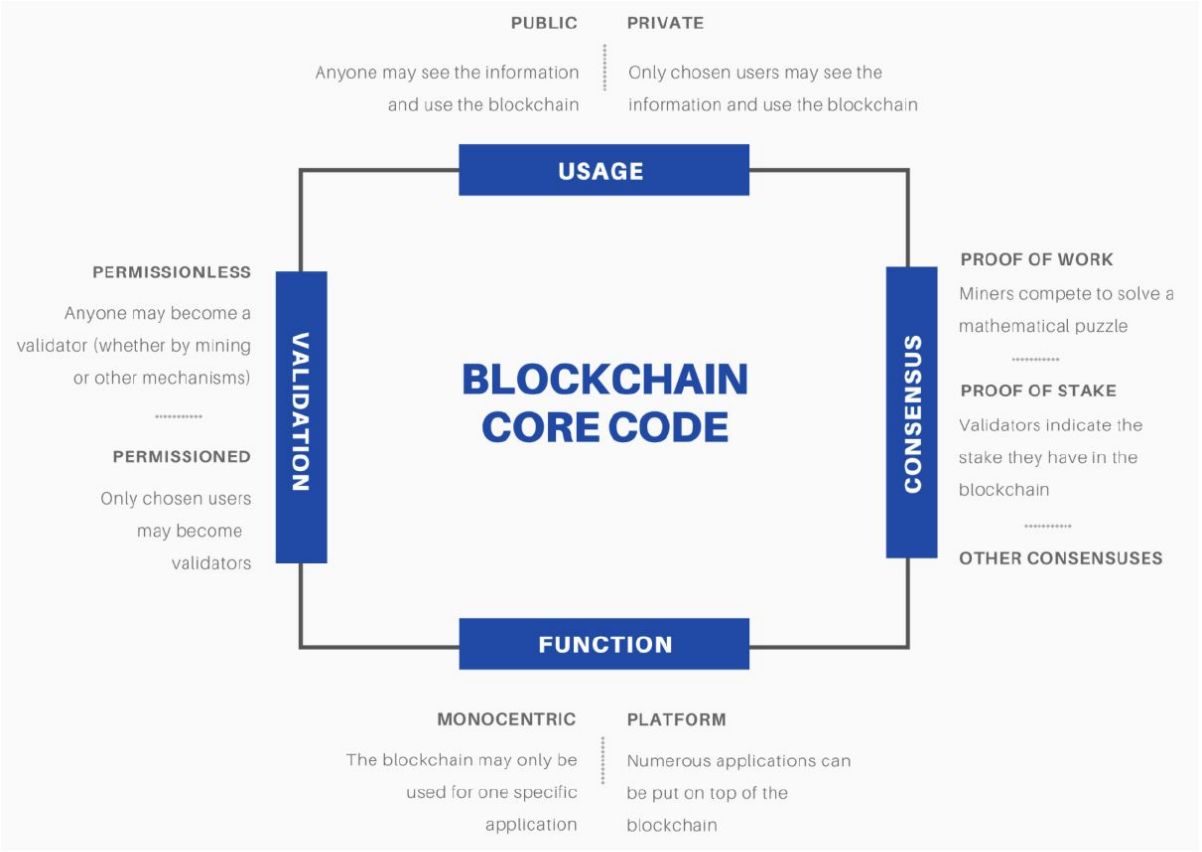


Evolution point of view

- Bitcoin refers to and makes use of past research, concepts and techniques, combining these pre-existing elements to give rise to the **blockchain**

- **Once a new class of technology has emerged, a Darwinian process of natural selection follows**
  - The technology → **species**
  - moves in different directions simultaneously, leading to the emergence of different → **varieties**
  - The varieties that survive multiply and seek to expand their territory, come into contact with other species, and begin to compete with them.

Blockchain Species and its Varieties



A look at blockchain varieties
© Thibault Schrepel[7]

The blockchain competition

- Blockchain is just beginning to compete with centralized transactional media
  - cryptocurrencies vs. fiat money

- The competition that is initially strong between varieties of blockchain is regimenting a competition between species
  - blockchain vs. centralized ecosystem

- Blockchain will **survive** only if it maintains strong **differentiators to gain a competitive advantage** over other species in a given environment

Smart Contracts species

- Smart contract technology leverages blockchains just as one species depends on another

- The smart contracts environment has
  - *legal dimensions*, i.e. soft law, regulations, case law, etc.
  - *technical dimensions*, the blockchain

- Must be combined → in the absence of **cooperation between law and technology**, these two aspects would struggle to take over

- A more cooperative and harmonized approach is therefore preferable so that smart contracts can grow in a cohesive and enduring environment

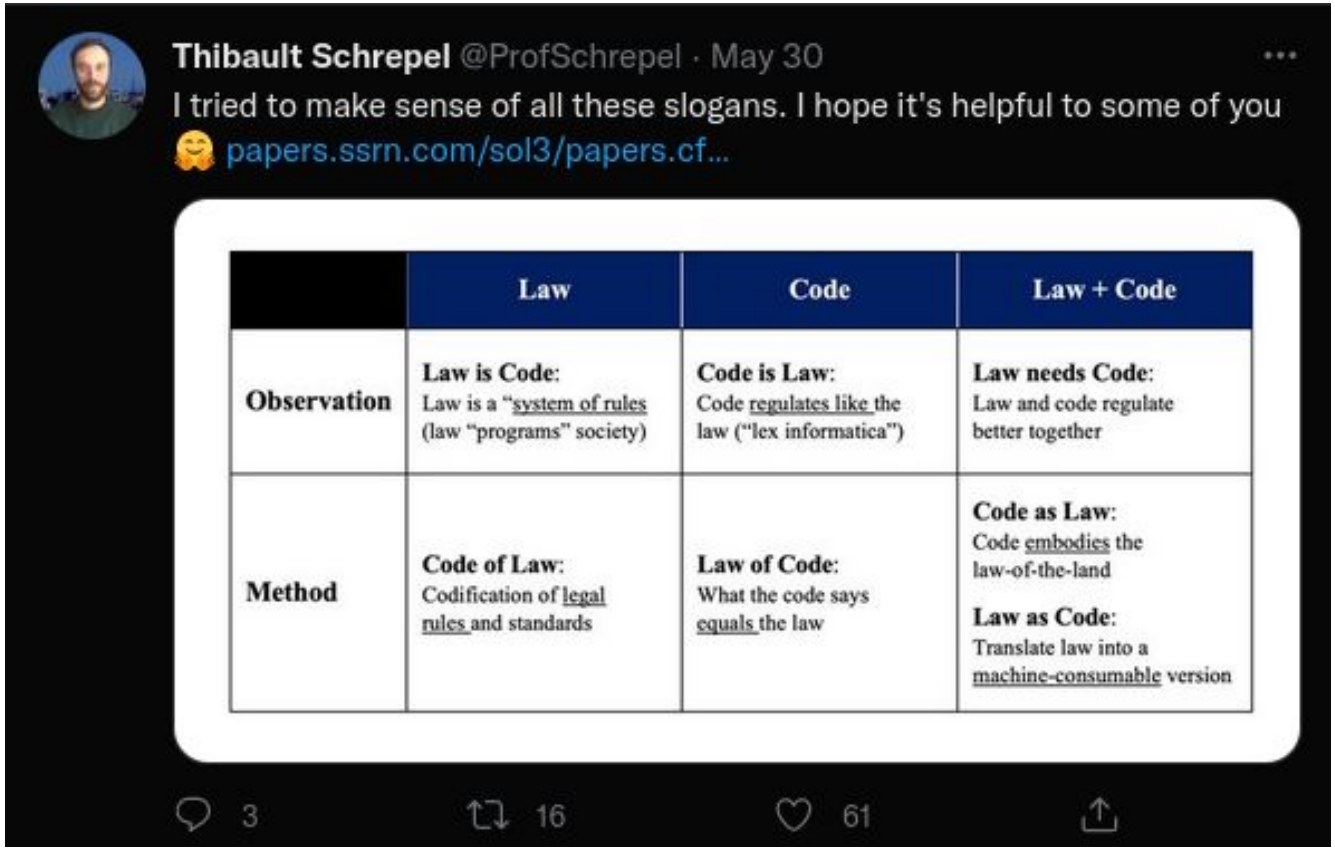
Possible Approaches

Absolutist
- Law perspective:
Creating laws without looking for ways to approach technology
- Technology perspective:
Technical fundamentalism is to design technology without relying on laws, leading to the creation of "temporarily autonomous zones" (TAZs).

Disadvantages:
→involves enforcing legal rules and standards without seeking to preserve the differentiating elements necessary for the survival of the technology
→as soon as the technology extends its territory and leaves the TAZ, law enforcement can lead to the extinction of the technology.

Cooperative
- **Law + technology**:
complement each other while trying to preserve their sphere of influence and building on each other's strengths
- Maintaining the distinctive features of blockchain while being allowed to enforce the law

Advantages:
→you can use smart contracts where contract law is difficult to enforce, for example, because jurisdictions are unfriendly
→used where the law cannot achieve a goal on its own, such *as preventing corruption*



Thibault Schrepel @ProfSchrepel · May 30

I tried to make sense of all these slogans. I hope it's helpful to some of you 🤗 papers.ssrn.com/sol3/papers.cf...

	Law	Code	Law + Code
Observation	**Law is Code**: Law is a "system of rules (law "programs" society)	**Code is Law**: Code regulates like the law ("lex informatica")	**Law needs Code**: Law and code regulate better together
Method	**Code of Law**: Codification of legal rules and standards	**Law of Code**: What the code says equals the law	**Code as Law**: Code embodies the law-of-the-land **Law as Code**: Translate law into a machine-consumable version

3 16 61

Code as Law + Law as Code

Key features of the Smart Contract species

1. Functioning
2. Immutability
3. Varieties of the species
4. Interactions between varieties and with the outside world

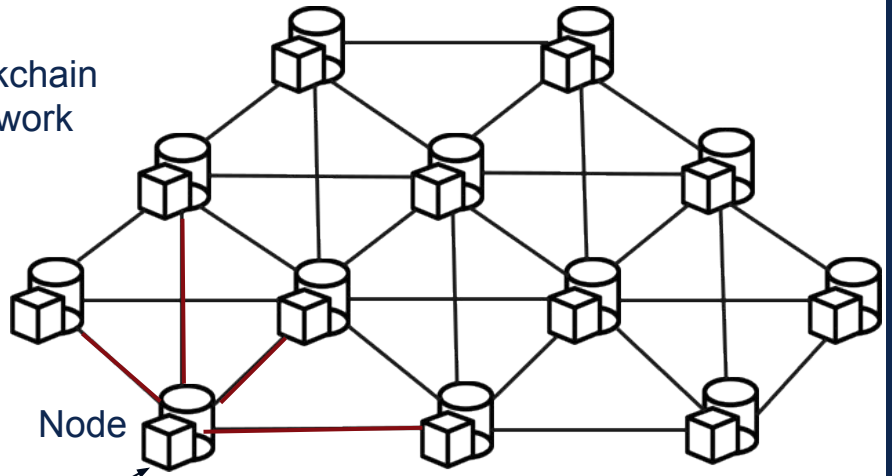
1. Functioning

Ethereum Smart Contracts

- Allows data structures to be easily maintained in the blockchain

- A new transaction refers to a previous one and updates the system state
  - In this case, the system state considers not only monetary transactions, but also data structures in smart contracts
  - The previous transaction refers to one that maintains the code and state of the smart contract
  - The new transaction refers to a set of instructions to be executed in the contract

Example: Smart Contract Vote



execute(

```
function _vote(uint256 challenge, bool inFavour) public {
    if (inFavour) {
        challenge.inFavour.add(msg.sender); // Alice
    } else {
        challenge.against.add(msg.sender); // Alice
    }
}
```

)

2. Immutability

2. Immutability

Smart contracts embedded in a blockchain are said to be immutable by default.

In fact, the source code (bytecode) of a smart contract is recorded in a transaction that is "mined" in a blockchain along with other transactions:

TX 1 : 5 btc |--> Charlie Pub
TX 2 : TX 1 |--> Bob Pub
TX 3 : TX 2 |--> Alice Pub

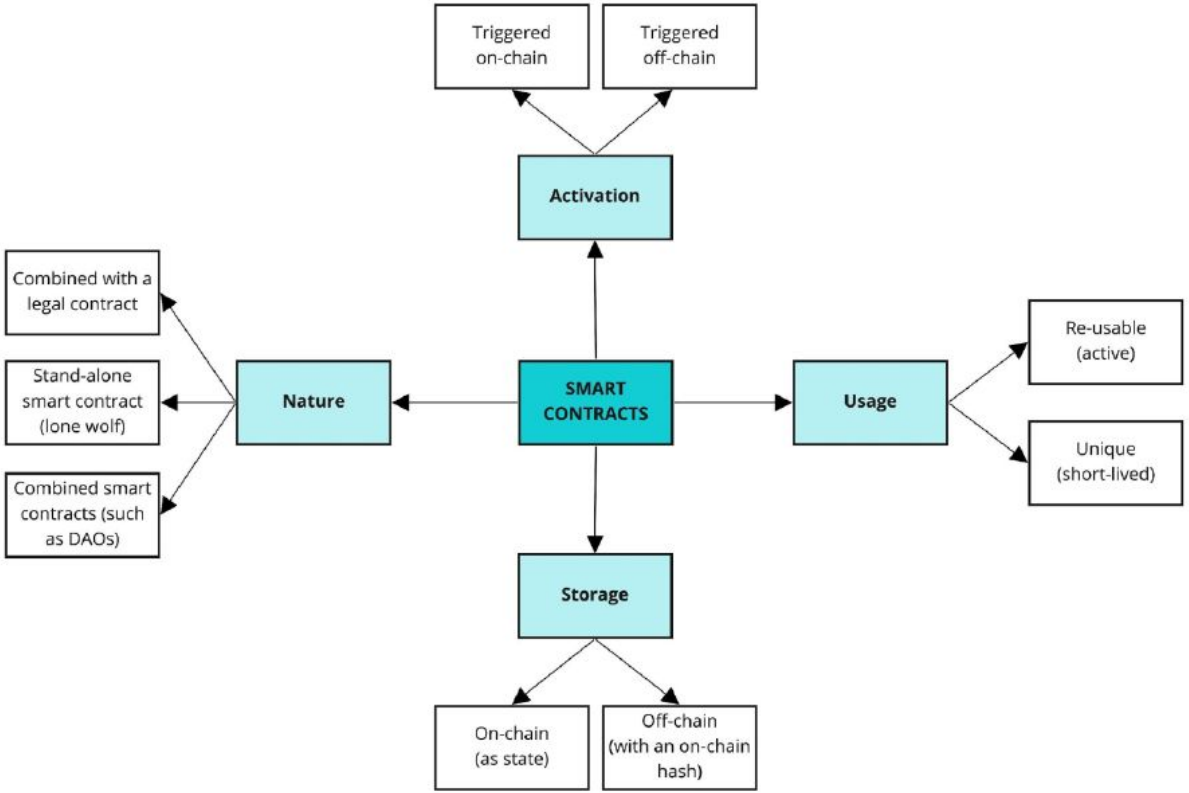
+ TX 4 : **Bytecode** |--> Contract Address

sign(TX 4, Alice Priv)

Alice's Wallet

3. Varieties of the species

Varieties



Title: An overview of blockchain smart contracts

Variety: **Nature**

1. **Combining a smart contract with a "legal contract"**
   a. e.g., a rental contract could be written in prose between an apartment owner and a tenant, while the smart contract could automate payment.

2. **Smart contract without the support of a legal contract**
   a. Most of the smart contracts in circulation today
   b. "lone wolves"→because they intend to be self-sufficient.

3. **Smart contracts combined with other smart contracts**
   a. create the conditions for decentralized governance of ecosystems
   b. Decentralized Autonomous Organizations (DAOs)

Variety: **Use**

1. Smart contract contingent on real-life events that only one of the two (or more) parties to the contract can invoke -> intuitu personae
   a. Conditions for invoking a smart contract are specific to a single party

2. Smart contracts invoked regularly, either by a single party or by a multitude of parties
   a. any user can invoke them
   b. called "active"

Variety: **Activation**

1. Activated **on-chain**: they are invoked following a blockchain event
   a. e.g., a smart contract can be designed to be invoked only when the value of an asset in the blockchain exceeds a certain level

2. **Off-chain** triggered: they are invoked as a result of an event outside the blockchain
   a. oracles

Variety: **Storage**

1. **On-chain**: the bytecode of a smart contract is stored on a transaction put on-chain
   a. doing so ensures immutability but also a lack of secrecy

2. **Off-chain**: the data (including the bytecode) can also be stored off-chain, with only the hash being recorded on the blockchain
   a. the immutability of the smart contract actually remains guaranteed because changing it automatically generates a new hash value that does not match the original one recorded on the blockchain.

4. Interactions between varieties and with the outside world

Interactions between varieties

Inter-blockchain

- smart contracts interact with each other, both to compete and to cooperate

- different blockchains compete for smart contracts and, depending on the technology on which they are built, have unique characteristics

Examples:
→ Polkadot, Cardano and EOS smart contracts are, on average, validated faster than Ethereum
→ Tezos allows for more secrecy
→ Polkadot uses bridges to enable the transfer of tokens or data from one blockchain to another

Intra-blockchain

- there is also competition and cooperation among smart contracts built on the same blockchain

- some become more attractive than others because they are better designed, introduce new features, or are more supported

Examples:
→Uniswap 1, 2, 3
→Smart contracts cooperate when they are technically linked together.
For example, many smart contracts automatically transfer the same type of ERC20 Token

Interactions with the outside world

Oracles allow smart contracts to interact with the outside world

- Originally, an oracle was a person charged with reporting prophecy whispered from divine sources
- As for blockchain, it generally designates the intermediary who reports information from the real world to the blockchain or vice versa
- Alternatively, the oracle may have a computational function when performing off-chain calculations

Oracles Varieties: **Direction, Data Collection, Sources**

1. Information can take two **directions**:
   1.1. *outbound*, information from the blockchain is brought to the outside world
   1.2. *inbound*, information is brought into the blockchain.

2. When inbound, several ways of **collecting information** are distinguished:
   2.1. *software*, interacts with (existing) information online and then transmits it
   2.2. *hardware*, transforms real-world measurements into digital information
   2.3. *human*, trusted third party providing real-world information.

3. The oracle can use a single **source** or several of them:
   3.1. single source, ``recentralizes'' the blockchain by introducing a **single point of failure** and requiring trust in a single point of entry
   3.2. combination of several sources, is preferable but nevertheless requires well-designed governance rules.

Oracles Varieties: **Validation, Integration, Use**

4. One must then **validate** the information once it has been transmitted:
   4.1. *automatic*, if the user decides to trust the oracle
   4.2. *voting*, subject to a vote submitted to the users of the blockchain (DAO).

5. Information must be **integrated**:
   5.1. *without intermediaries*, directly distributed to the blockchain network
   5.2. *custom* smart contract interface, e.g. dApp
   5.3. *software module* for data pre-processing
   5.4. *customized solution*, to prevent forgery e.g. fingerprinting

6. Once the information is integrated, its **uses** can be:
   6.1. *contract-specific*, use in a single smart contract
   6.2. *multiple smart contracts use*, such as a database e.g., financial data

Smart Legal Contract

From trust in the contracting party to trust in the code in the execution of the contract

"From Trust in the Contracting Party to Trust in the Code in Contract Performance", Chantal Bomprezzi https://kluwerlawonline.com/api/Product/CitationPDFURL?file=Journals%5CEuCML%5CEuCML2021032.pdf

"Smart legal contracts: advice to Government", UK's Law Commission
https://www.lawcom.gov.uk/project/smart-contracts/

RI E
Rights of Internet of Everything

Inviolable Contracts?

The use of smart contracts can be implemented either for business-to-business, peer-to-peer or even business-to-consumer (B2C) commercial contracts.

In the B2C case, there can be several situations where self-execution of a smart contract leads to the breach of that contract:

1. The content of the code does not correspond to the will of the parties, thus resulting in the execution of the contract not satisfying the consumer.

2. Technical issues impacting the execution of the contract.

3. Other issues due to the blockchain being closed to the outside world, i.e., when there is a need to connect the smart contract with the off-chain world.

Inviolable Contracts? -> (1) Content of the code

When the code does not work as intended by the consumer and agreed upon in the contract, the contract is breached.

Inviolable Contracts? -> (2) Technical issues

Blockchain-based applications consist of multiple components, and various technical issues can adversely affect these components:

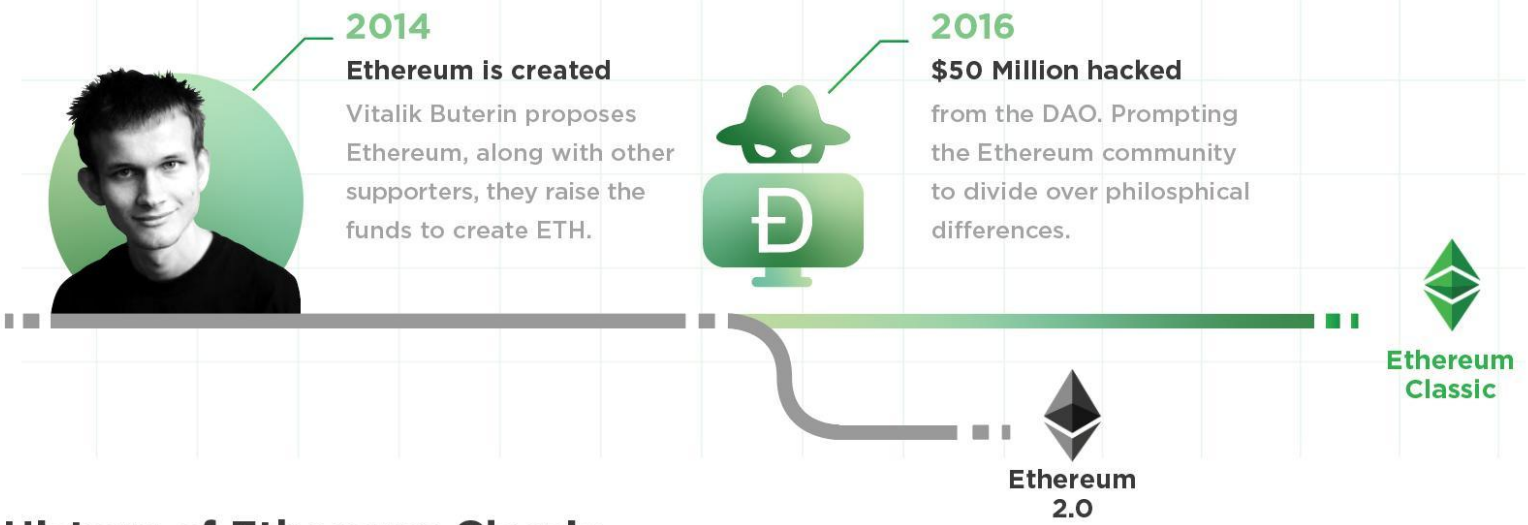
- The smart contract can be prone to bugs, like any other computer program.
- Problems can also arise from the underlying blockchain, such as from attacks that can give room for manipulation of the execution of a smart contract.
- In addition, oracles can be compromised, as the external data source may fail or become inactive.

Inviolable Contracts? -> (2) Technical issues
Bugs negli Smart Contracts

- The **Reentrancy attack** in Solidity.
  It occurs when a function makes an external call to another untrusted contract. Then the untrusted contract makes a recursive call to the original function in an attempt to drain some funds.

- Although the reentrancy attack is considered quite old, cases such as:
  - *Uniswap/Lendf.Me hacks (April 2020) – $25 million.*
  - *The BurgerSwap hack (May 2021) – $7.2 million.*
  - *The SURGEBNB hack (August 2021) – $4 million.*
  - *CREAM FINANCE hack (August 2021) – $18.8 million.*
  - *Siren protocol hack (September 2021) – $3.5 million.*
  - *Fei Protocol hack (April 2022) – $80 million.*

Inviolable Contracts? -> (2) Technical issues
Bugs negli Smart Contracts → The **DAO hack**



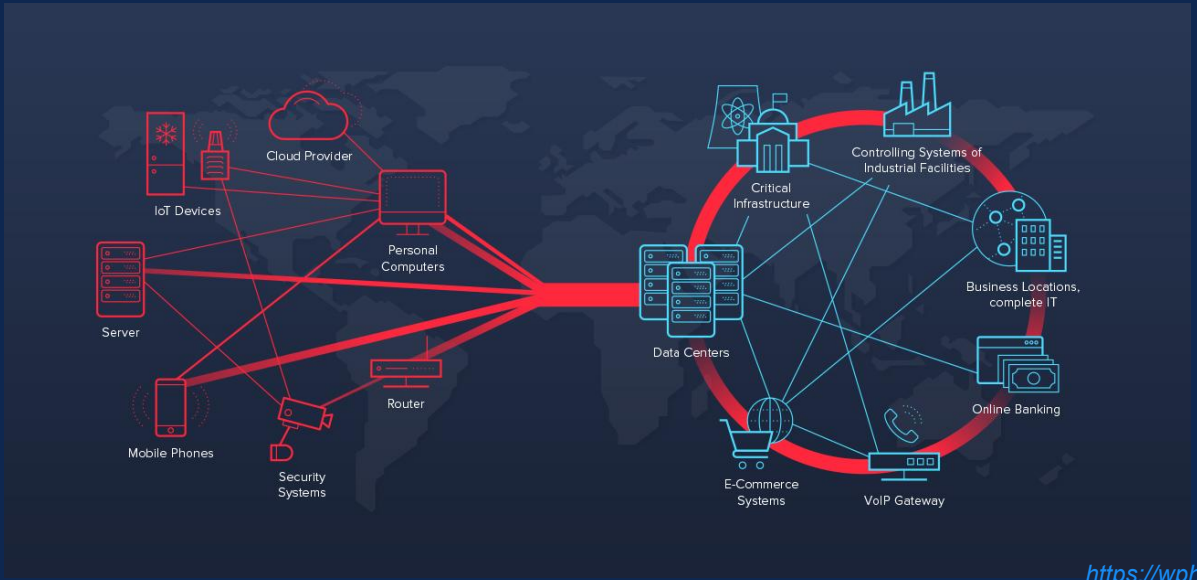
History of Ethereum Classic

The DAO Hack in 2016, one of the earliest projects on Ethereum called “The Dao”, was hacked for $50 million. The community was faced with a controversial dilemma: “Bailout investors like traditional bank systems have done for years or keep our values". The investors would receive their stolen funds, but at the cost of a foundational principle: immutability. This decision split the community, the majority supported “Bailout”, while the minority supported “Code is Law”. The chain forked, and the original Ethereum was labelled Classic.

https://twitter.com/eth_classic/status/1471329806671237121

Inviolable Contracts? -> (2) Technical issues
Blockchain Issues

- **Solana Went Offline for Four Hours ->** https://finance.yahoo.com/news/solana-latest-ddos-attack-leads-120022342.html
- Distributed Denial-of-Service (DDoS) attack



https://wphostinggeeks.com/cloud-vs-vps/

Inviolable Contracts? -> (3) Closure to the outside world

- If an oracle's information is not provided at all or is incorrect, the contract is not executed or is not executed correctly.
  - This may happen not only because of technical malfunctions, but also because of human errors or actions.

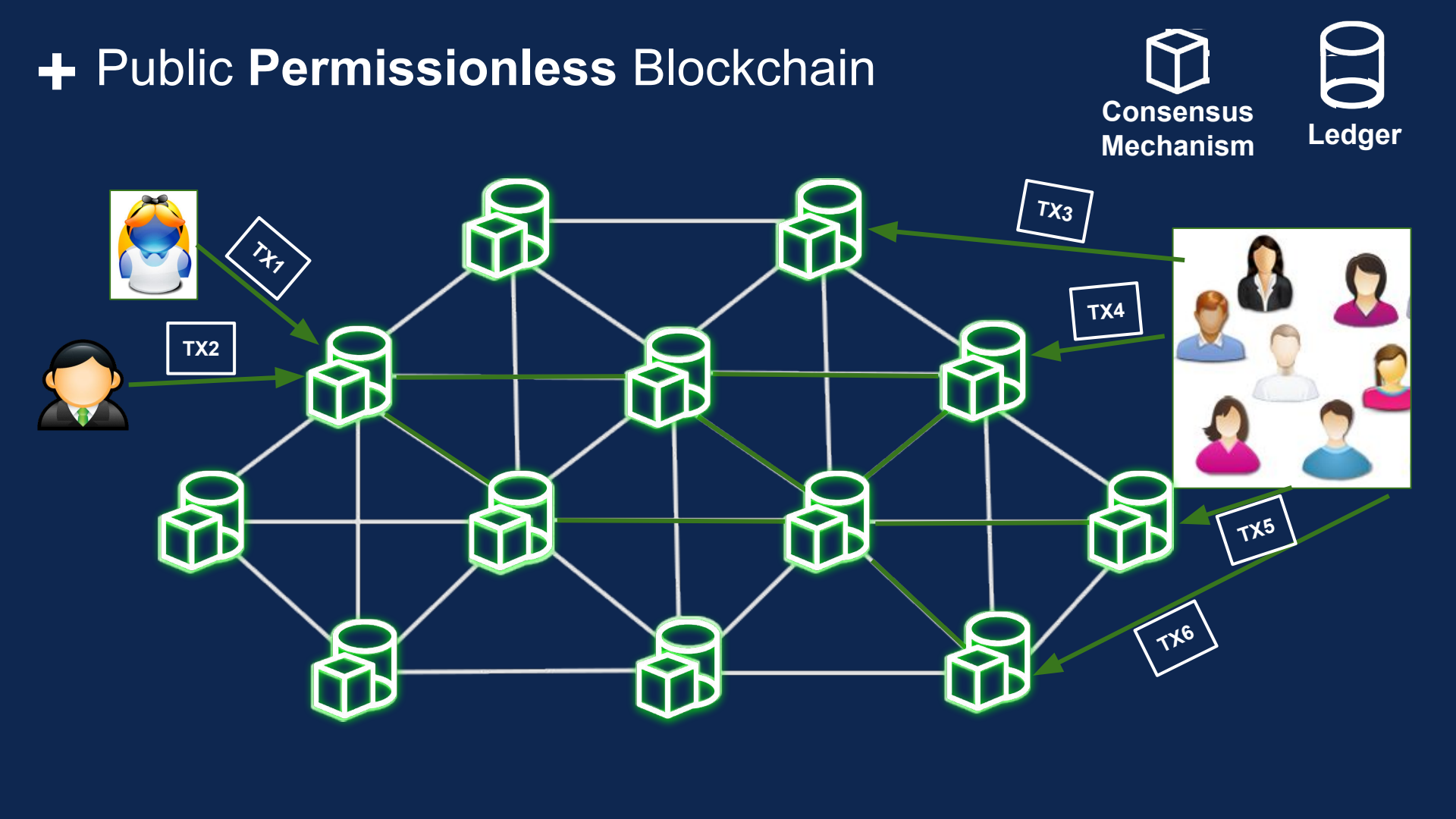
- For example, a courier reporting that the package was delivered to the specified address, while the package was not shipped, or the contents of the package differing from what the parties agreed to in the contract.

- Input into the blockchain is under someone's direct control and does not benefit from the decentralized nature of the blockchain.
  - "Garbage In → Garbage Out"

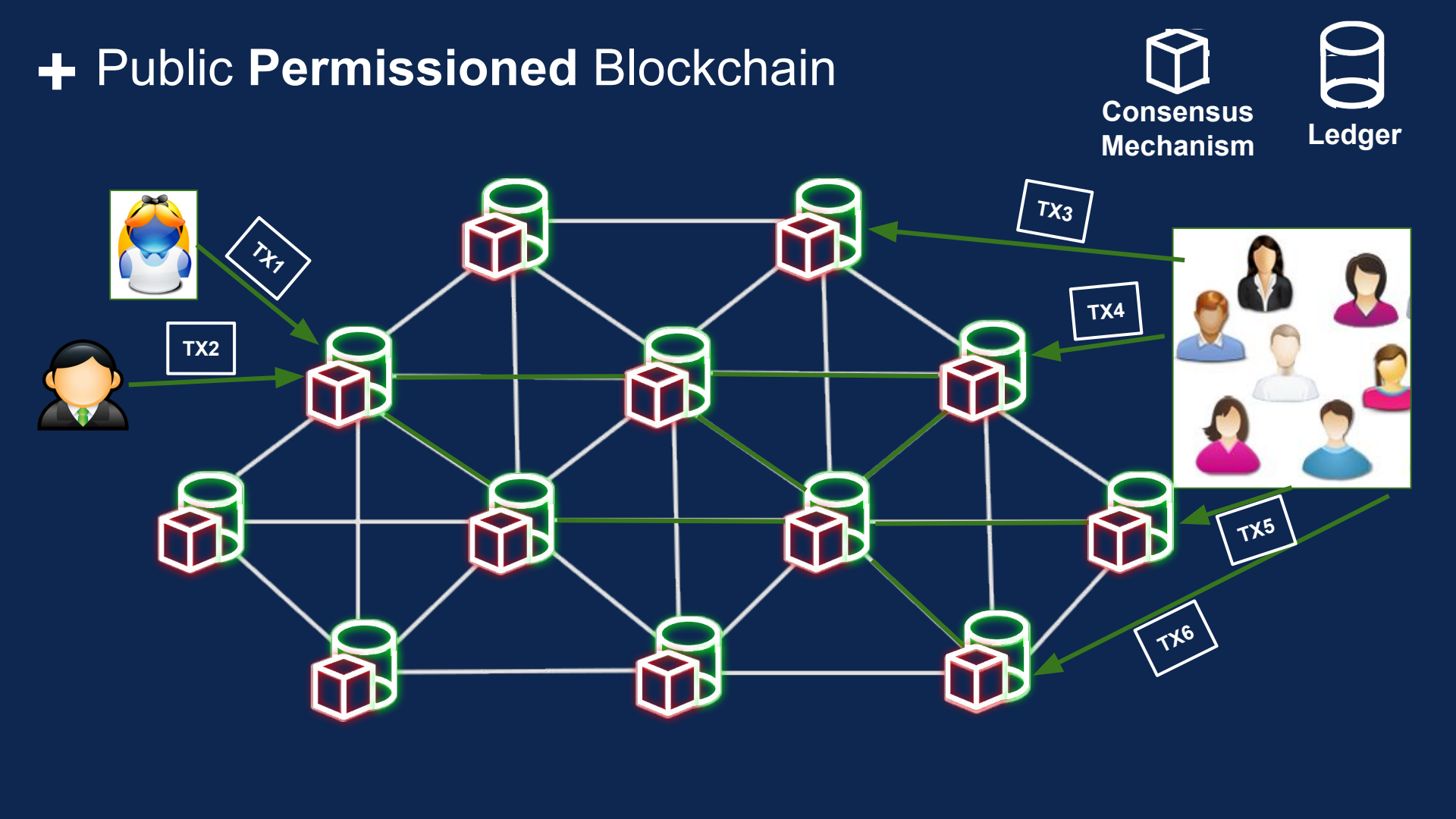
The discrepancy between decentralized technology and the absence of control over the execution of a contract

- Blockchain is a decentralized technology.
  - There is much confusion about the meaning of the term "decentralization."
- The latter could refer either to the technology itself or to the governance of the application running on a blockchain.
- De/centralized governance ->
  - Consensus mechanism + Software development
- De/centralized technology ->
  - distributed or centralized ledger

+ Public **Permissionless** Blockchain



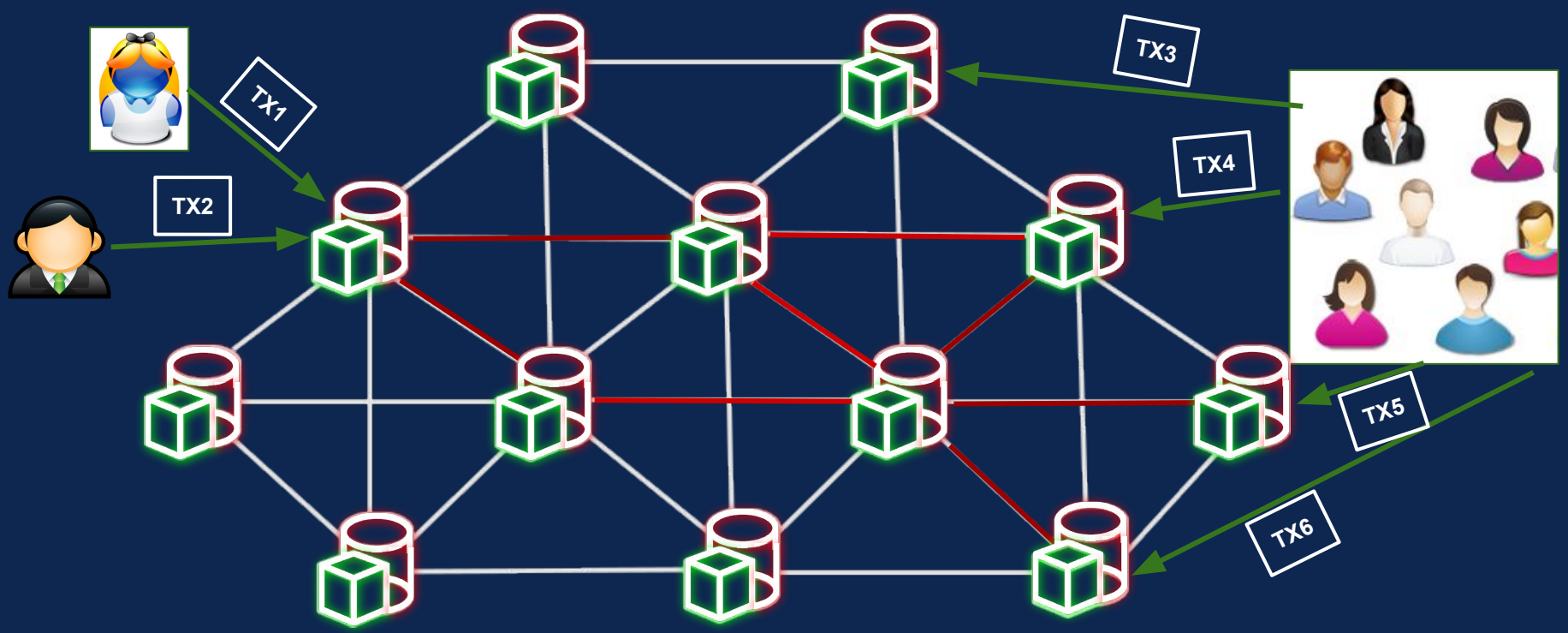
+ Public **Permissioned** Blockchain



+ Private **Permissionless** Blockchain

Consensus Mechanism

Ledger



+ Private **Permissioned** Blockchain

The discrepancy between decentralized technology and the absence of control over the execution of a contract

- *Private Permissioned*: does not involve consumers, is an example of centralized blockchain governance.
- As a result, it acts as a standard client-server architecture (e.g., the Cloud). Contract execution is under centralized control, with no additional benefits to consumers.

- *Public Permissioned*: the problem is similar, but if designed well it can provide a decentralized control mechanism similar to *(or better from)* permissionless.

- Permissionless: however, even in this case there may be centralized control over the company's performance of the contract. The latter does not depend on centralized governance of enforcement, but rather on the subject matter of the contractual obligation.

The discrepancy between decentralized technology and the absence of control over the execution of a contract (source code)

- The smart contract is executed according to contractual provisions.

- To protect consumers, it is vital to ensure that they are aware of the content of the contract
  - That the contract is not too skewed in favor of the company
  - That it does not contain unfair terms.

- If the conclusion of the contract is in the hands of the company:
  - it does not matter that smart contracts are capable of self-execution;
  - the company can indirectly influence the execution of the contract.

The discrepancy between decentralized technology and the absence of control over the execution of a contract (source code)

- The smart contract code is a human creation.
- Therefore, it could be argued that such creators, or those who hired them, should be held liable for failures in the code that caused the breach of contract.
- If the code is produced by the company, consumers should still trust the company and traditional legal remedies for breach are applicable.

+ The **code is law**?

- Non-programmers have to rely completely on experts (smart contracts programmers) to explain the contract, which brings additional challenges and places an even greater emphasis on liability.

- "Using the analogy with lawyers, smart contracts programmers could become a regulated profession and, similar to lawyers, could be required to take out liability insurance."

- However, lawyers can " help parties determine what would be the best contractual structure for a particular transaction and explain to them potential risks, such as those related to security or the objective nature of smart contracts, that leave less room for negotiation."

Dora Kadar, https://tech.eu/2022/05/09/can-smart-contracts-replace-lawyers-in-europe/

UK's Law Commission **Smart Legal Contract**

- Is a binding contract in which some or all contractual obligations are defined and/or performed automatically by a computer program

- Subset of smart contracts

- Can take a variety of forms with varying degrees of automation:
  - Natural language contract with automatic execution by code
  - Hybrid contract
  - Contract drafted solely by code

Natural language contract with automatic execution by code

- "Classic" natural language contract, in which some or all contractual obligations are automatically performed by the code.
- The code itself does not define any obligations, but is only a tool used to fulfill the contractual obligations.
- It is the currently most widely used form of legal smart contract. This form raises no (or few) new legal problems in the context of contract creation and interpretation.
- **The biggest issue remains understanding the proper translation from natural language to code.**

Hybrid contract

- A hybrid smart legal contract is one in which some contractual obligations are defined in natural language and others are defined in computer program code.

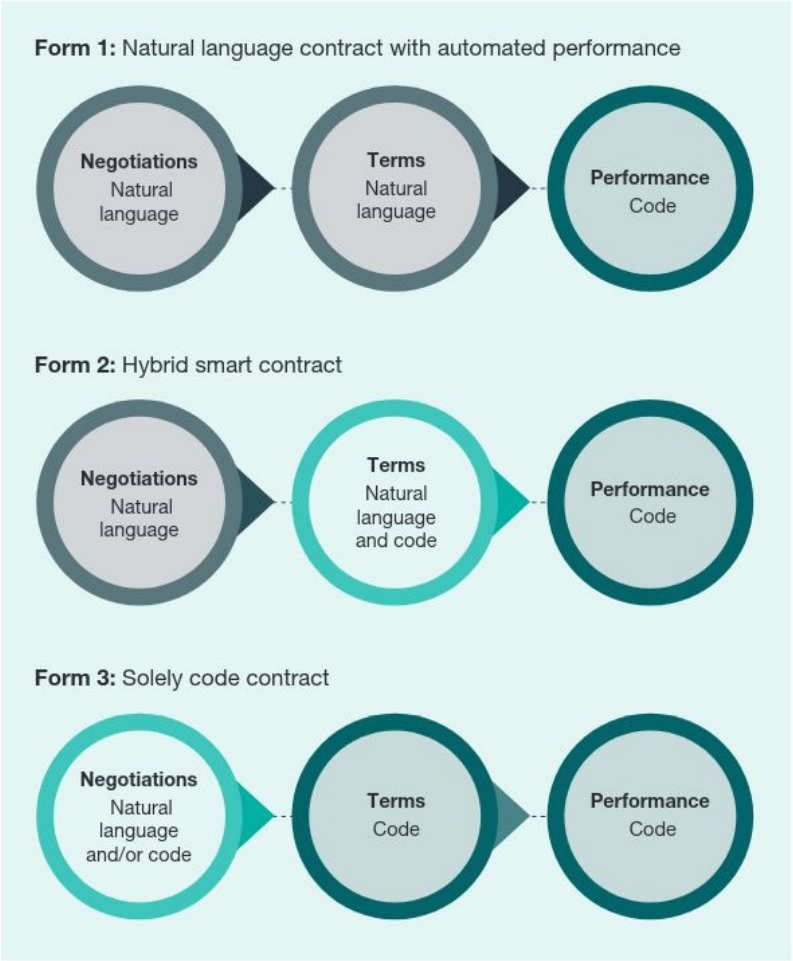
- Some or all of the contractual obligations are executed automatically:
  - mainly written in code with some natural language terms that establish, for example, the applicable law and jurisdiction.
  - mainly written in natural language and include only one or two terms written in code.

- Natural language terms may be in a separate document or transposed into natural language comments included in the code.

+ Contract drafted solely by code

- All contract terms are defined in the code and executed automatically. There is no natural language version.
- This type of smart legal contract presents the greatest challenges from a contract law perspective, in terms of determining if and when a legal contract is formed and how that contract can be interpreted.
- Commercial contracts are typically too nuanced to be reduced solely to code.

Smart Legal Contract



Smart legal contracts: Summary

Some Issues

- Since deeds have various formality requirements (e.g., they must be witnessed and attested), it is **difficult to use hybrid or code-only contracts** to create a deed in the *current state of affairs*.

- Difficulties may arise in relation to **determining jurisdiction and applicable law** for some smart legal contracts.
  - particularly when they are one-sided and solely in code, or formed by the autonomous interaction of computer programs, e.g., other smart contracts

- **Digital localization**, i.e., the need to attribute real places to digital assets and actions that "take place" on a distributed ledger, is also a significant challenge.

+ Other issues related to the processing of personal data

- **Protection and location of personal data**.
- Smart contracts may make use of personal data
  - GDPR may apply to them depending on the data they use and generate.

- **Key tensions**:
  - How to handle immutability and the right to be forgotten?
  - How to enforce accountability of data controllers in a permissionless blockchain are identified by pseudonymous addresses?
  - Obligations to store data within the European Union or an EU member state -> not applicable in a permissionless blockchain

+ Possible Solutions

- *UK's Law Commission* **“Reasonable coder”**:
  - interpretation of a contract term in code form should be determined by asking **what the term would mean to a reasonable person** with knowledge and *understanding of the code*.
  - The answer to this question will be what the code seemed to instruct the computer to do, in the reasoned opinion of that person.

- Develop established practices and model contracts that parties can use to negotiate and draft their smart contracts

- Technologies and methods for protecting personal data and trade secrets
  - e.g., cryptography (hashing, zero knowledge proof)
  - e.g., multi-layered DLTs

Intelligible Contract

Luca Cervone, Monica Palmirani, Fabio Vitali

http://hdl.handle.net/10125/63959

Original slides by Luca Cervone

http://lbl.cirsfid.unibo.it/wp-content/uploads/2020/01/The_Intelligible_Contract-v4.pdf

RIDE
Rights of Internet of Everything

Ricardian Contracts

- Ricardian contracts attempt to **bridge** the gap between **legal prose** (natural language) and **executable code**.
- The developer describes a triple <**P**, **C**, **M**> where:
  - **P** describes the denotative semantics of the contracts (the legal prose);
  - **C** describes the operational semantics of the contracts (the source code);
  - **M** is a mapping between the operations expressed in C and the legal prose in P.

Smart Contract Templates

- Smart Contract Templates are an implementation of Ricardian contracts whose operating code is standardized and whose behavior is controlled by parameters contained in a smart contract.
- **Legal drafting tools enable developers and legal experts to create smart contract templates together**
- Legal prose is serialized through standard and flexible vocabularies
- **A document mark-up links** the elements of a contract to standard ontologies
- Some **"features" link legal prose to operational code**

+ Intelligible Contracts

- Intelligible Contracts are smart legal contracts written in natural language that can be mapped, in whole or in part, to blockchain-based smart contract code.

- Intelligible Contracts extend the Ricardian Contracts and Smart Contract Templates by providing specifications for the **intelligibility of digital contract contracts**.

- They fill the gaps in the Ricardian Contracts and Smart Contracts Templates.

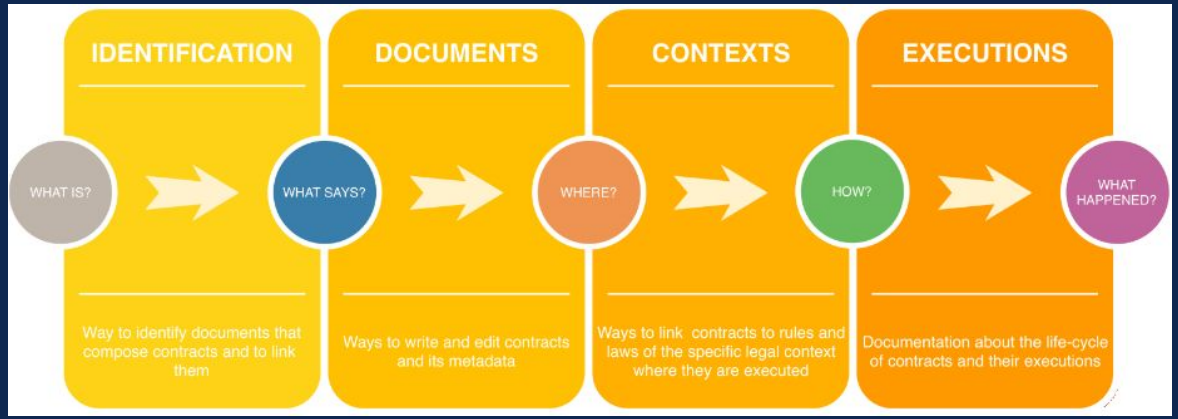
->

Intelligible Contracts - Features

- Provide links between contracts and other legal and nonlegal resources and documents
  - e.g., regulatory references in contracts
- Provide a description of the legal context of the contracts
  - e.g., jurisdiction of the facts.
- Provide information on the operational context of contracts
  - e.g., the type of blockchain.
- Report the automatic execution of contracts.

Intelligible Contracts - Features 2

- Link all the resources that make up contracts or define their legal contexts
- Link the agents involved in the life cycle of contracts
- Link the digital resources that
  - describe how the operating code is executed
  - report what happens during contract execution.

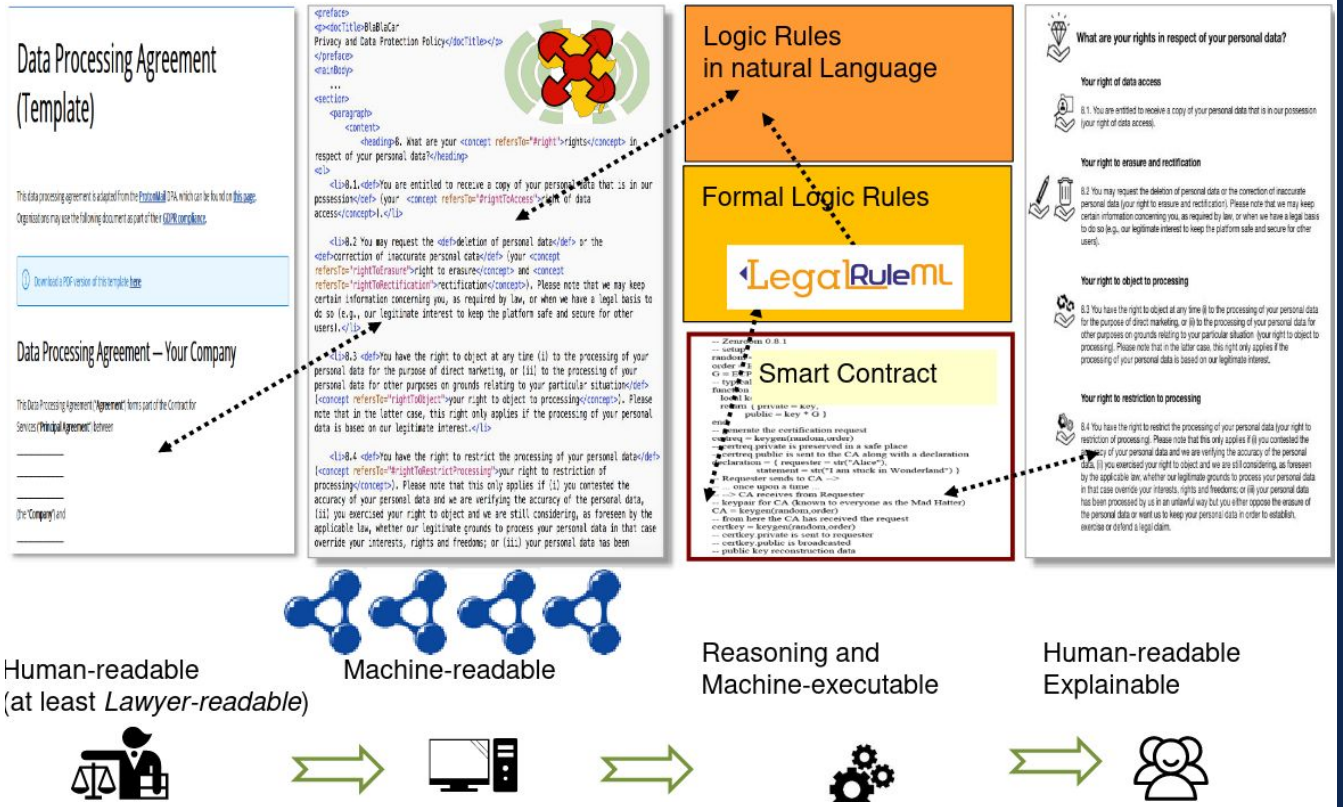


Denotative definition of Intelligible Contracts

```
Intelligible Contract::=
     UID and
     Context+ and,
     Document+ and,
     Execution Report+

UID::= URI => HASH

Context::= UID and Legal Context+ and Operational Context+
     Legal Context::= (Legal Document Ref  or Legal Document)+
     Operational Context::= Op Environment Ref+ and Op Code Ref+
          Operational Environment ::= URI
          Operational Code::= UID  and Bitcode+

Document::= UID and (Generic Document or Generic Document Ref)+
or (Legal Document or Legal Document Ref)+
     Generic Document::= Bitcode+
     Legal Document::= Legal Prose+ and Metadata+
          Legal Prose::= Human Natural Language Statement +
          Metadata::= <Legal Prose,Operational
Context,Descript.>

Execution Report::= UID and Document+
```

(::=) means "is defined as";
(*) means "zero or more occurrences";
(+) means "one or more occurrences";
if there are neither (*) nor (+), then there must be "exactly one occurrence";
(x and y) means "both x and y";
(x or y) means "x or y or both."
(A ⇒ B) to indicate that "A and B are both mandatory and B is a function of the content of A."

Scenario


Data Processing Agreement (Template)
Data Processing Agreement – Your Company
Logic Rules in natural Language
Formal Logic Rules
LegalRuleML
Smart Contract
What are your rights in respect of your personal data?
Your right of data access
Your right to erasure and rectification
Your right to object to processing
Your right to restriction to processing
Human-readable (at least Lawyer-readable)
Machine-readable
Reasoning and Machine-executable
Human-readable Explainable

MPEG-21
Smart Contract for Media

ISO/IEC 21000 - Part 23 standard

https://www.iso.org/standard/82527.html

RIoE
Rights of Internet of Everything

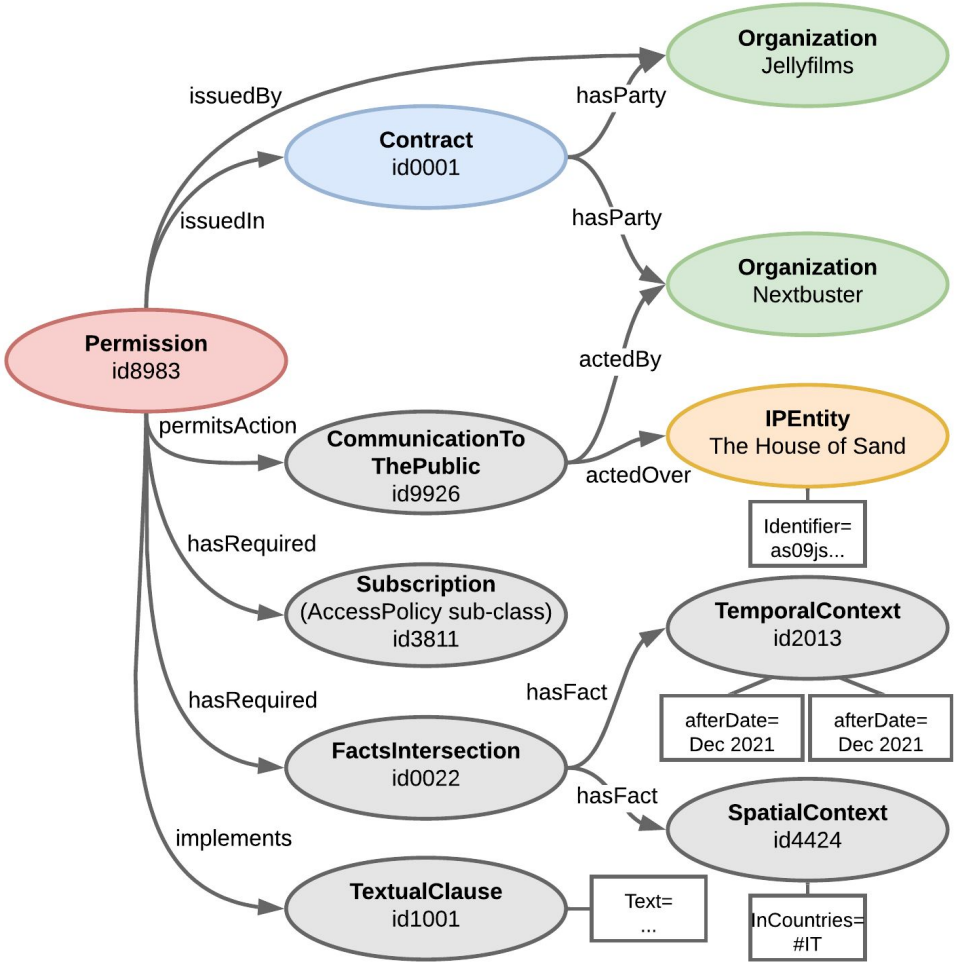
ISO/IEC 21000 MPEG-21 Framework

- MPEG stands for ``Moving Picture Experts Group'' a working group of the ISO/IEC, that develops media encoding standards.

- **ISO/IEC 21000 MPEG-21 standard**
  - framework for the delivery and consumption of multimedia
  - for use by all actors in the delivery and consumption chain
  - MPEG-21 parts include
    - digital copyright protection
    - payment systems
    - verification and quality assessment
    - a set of ontologies for encoding **Intellectual Property (IP) rights** information about media

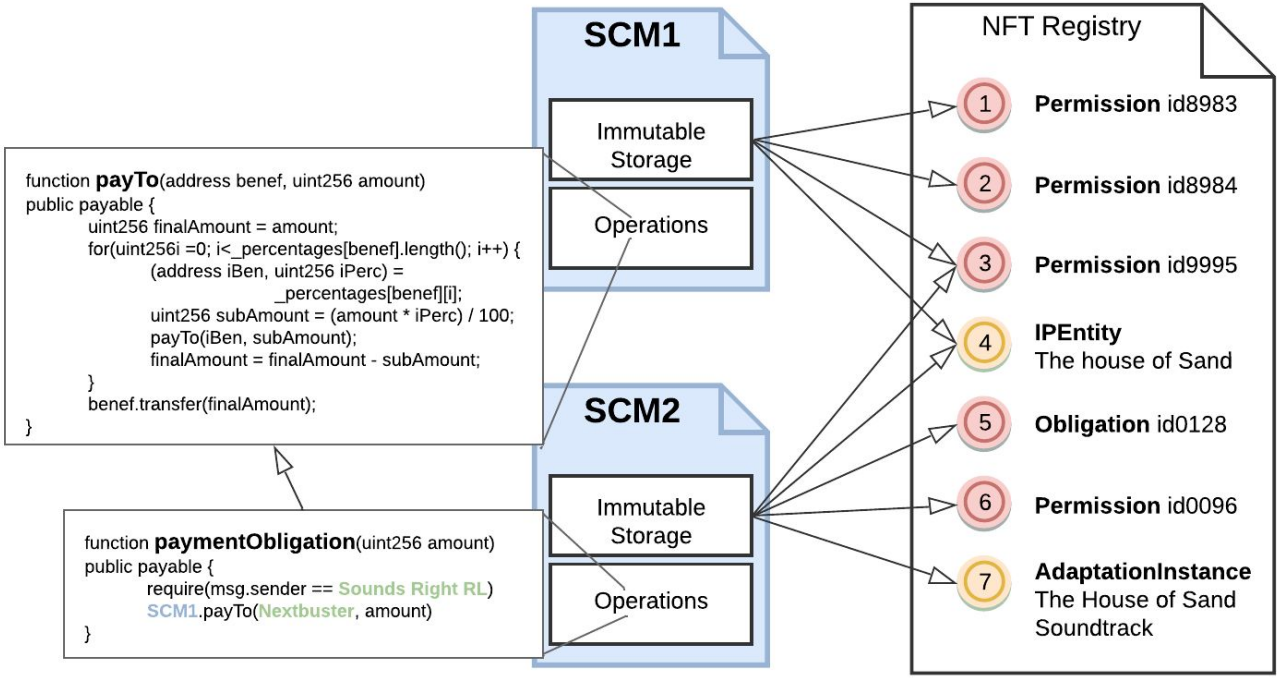
Smart Contract for Media

- Smart contracts can be used to encode the terms and conditions of a contract for media-related asset trading.
- Establish and enforce IP agreements such as licenses and enable the transmission of real-time payments to IP owners
- IP rights information in protected media content, then, can be encoded using the MPEG-21 framework and directly and uniquely linked to a smart contract
- e.g., smart contracts could allow music and media royalties to be administered almost instantaneously and manage usage allowances and restrictions.



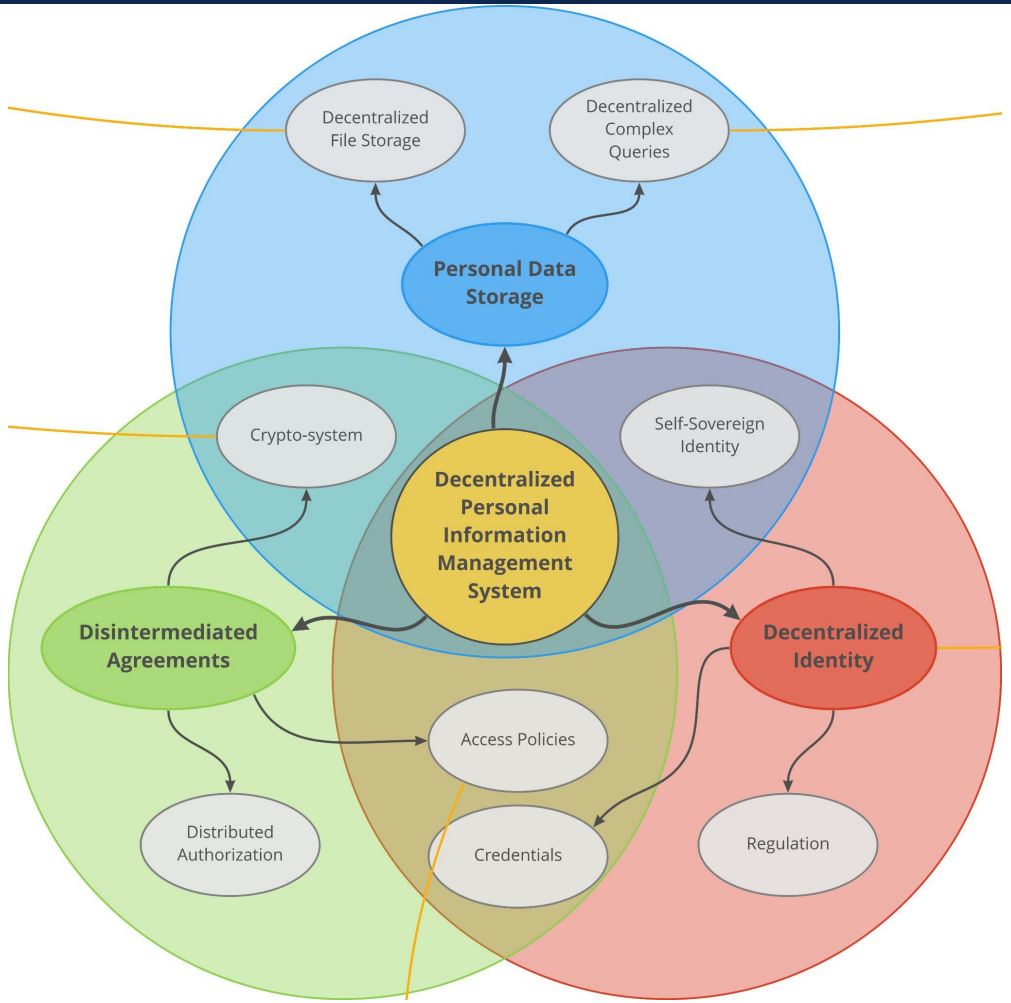
Video On-demand Services and Independent Producers

Smart Contract for Media



Decentralized Systems for the Protection and Portability of Personal Data

Systems Overview



“Complex” GDPR-compliant privacy policies (and consent)