Utrecht University

“I’m still / I’m still / Chaining from the Block”

An Outlook of the Ongoing and Future Relationship between Blockchain Technologies and Process-aware Information Systems

Claudio Di Ciccio | https://diciccio.net/ | c.diciccio@uu.nl
Utrecht University, Netherlands

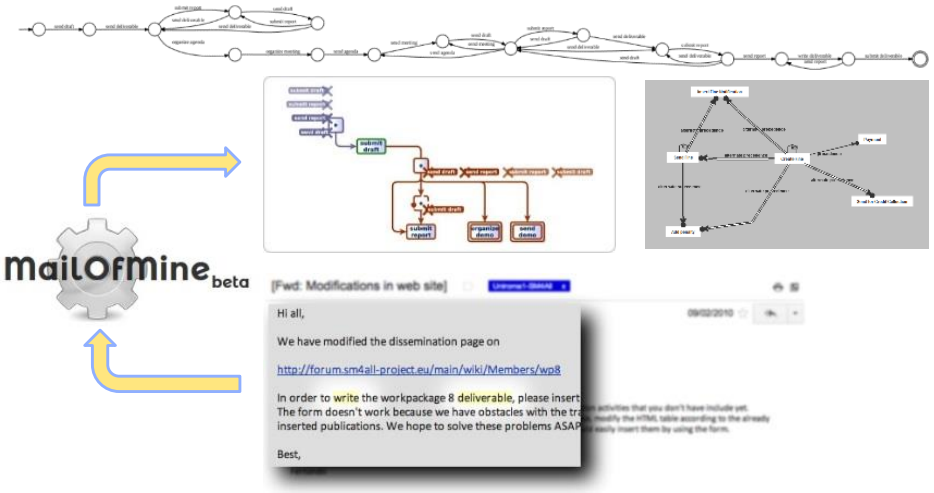
Claudio Di Ciccio

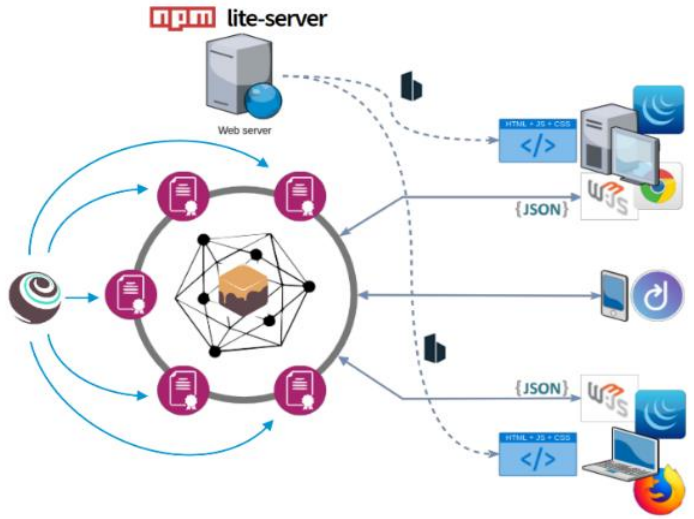
Associate professor
Ph.D. in Computer Science and Engineering

Faculty of Science /
Software Division /
Process Science Group

Main research interests:
formal methods &
logic and computation,

applied in
process analytics &
blockchain architectures



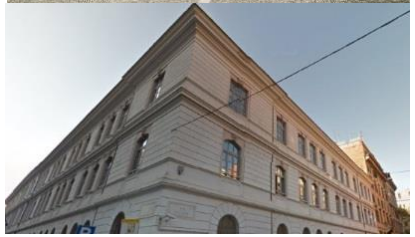


My experience so far

Utrecht University

Latina, Italy
(B.Sc)

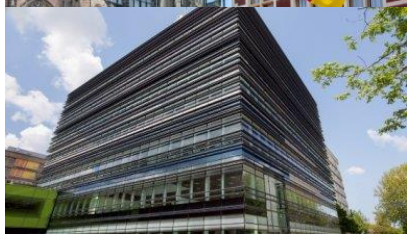
Rome, Italy
(M.Sc, Ph.D)



Vienna, Austria
(Post-doc,
Assistant Prof.)

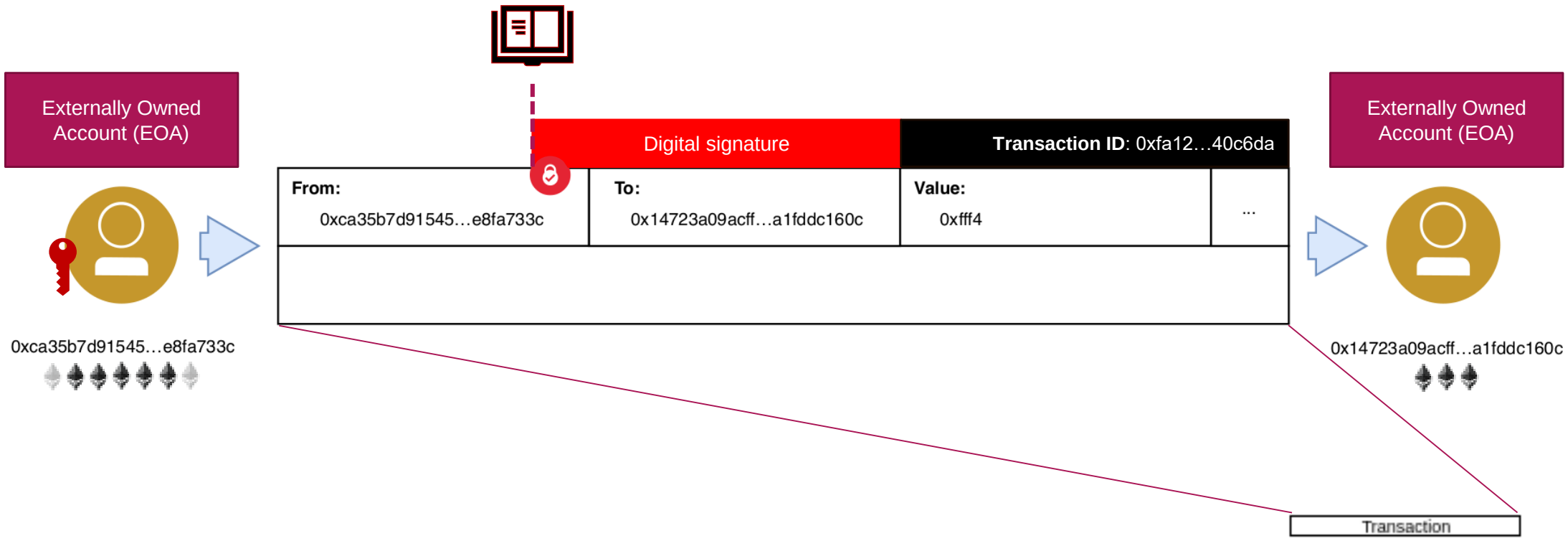
Rome, Italy
(Assistant Prof.,
Associate Prof.)

Utrecht, Netherlands
(Associate Prof.)



Transaction

- Transfer of **(crypto)assets** (Ether, Bitcoin, Algo, …) from **account** A to **account** B



Ledger

- Ordered collection of transactions
- The **order** matters!

Amsterdam, 1856

- About **2000 ships** departed on an annual basis
- **Seafarers**
  - **numerous**
  - a vital contribution to **trade**
  - wages paid after a journey (always in **need of credit**)
- **Non-bank credit markets**
  - Shopkeepers and boarding-house keepers as lenders
- The **Discipline Act** (1856)
  - Forbids the use of seafarers' wages as redemption payments

Image source: https://www.istockphoto.com/nl/vector/amsterdam-in-the-19th-century-gm538063610-95625387

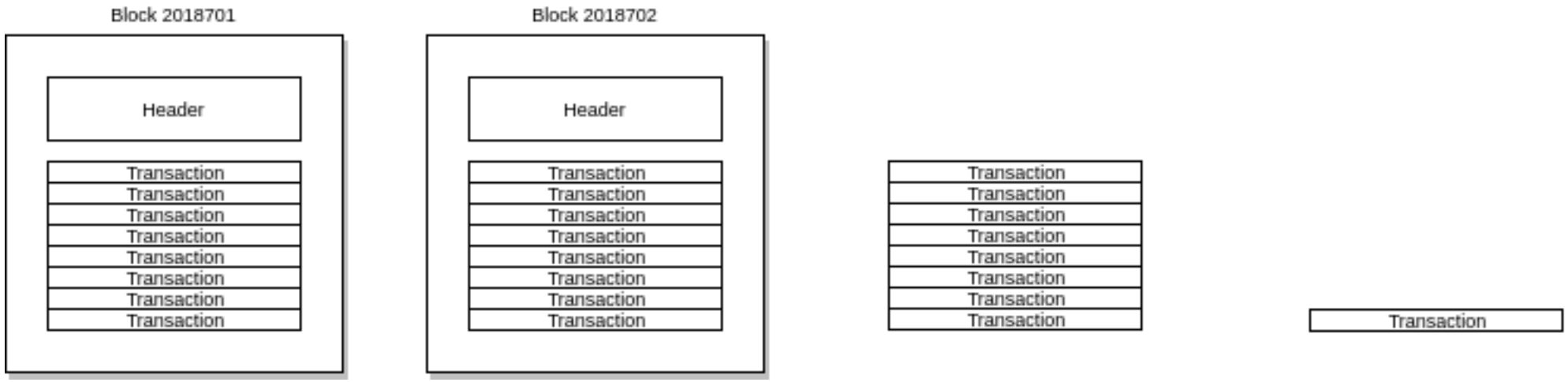
The ledger of the water bailiff's

- The **Discipline Act** prescribed that lenders had to present their unredeemed IOUs to the **water bailiff**'s during the month of **July 1856**
- Every IOU recorded basic information, including:
  - the **date** on which it was entered
  - the names of **lender** and **borrower**
  - the unredeemed **amount**
- 13,708 loans were registered in a 443-page **ledger**

A hand-drawn ink lithography of men writing a ledger in the water bailiff's office in 1856 (DALL-E, https://openai.com/dall-e-2/)

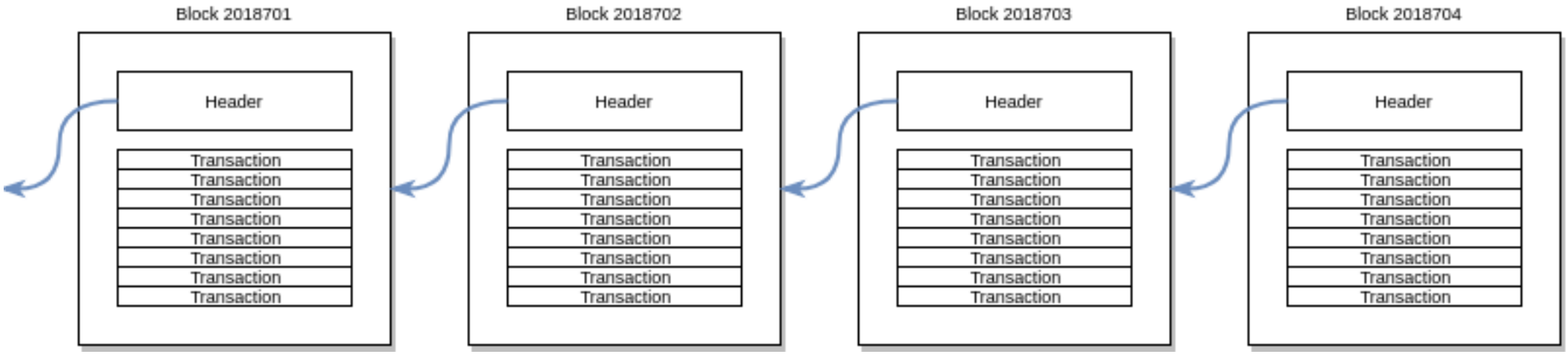
Block

- Blocks group and collate transactions
- The order matters!

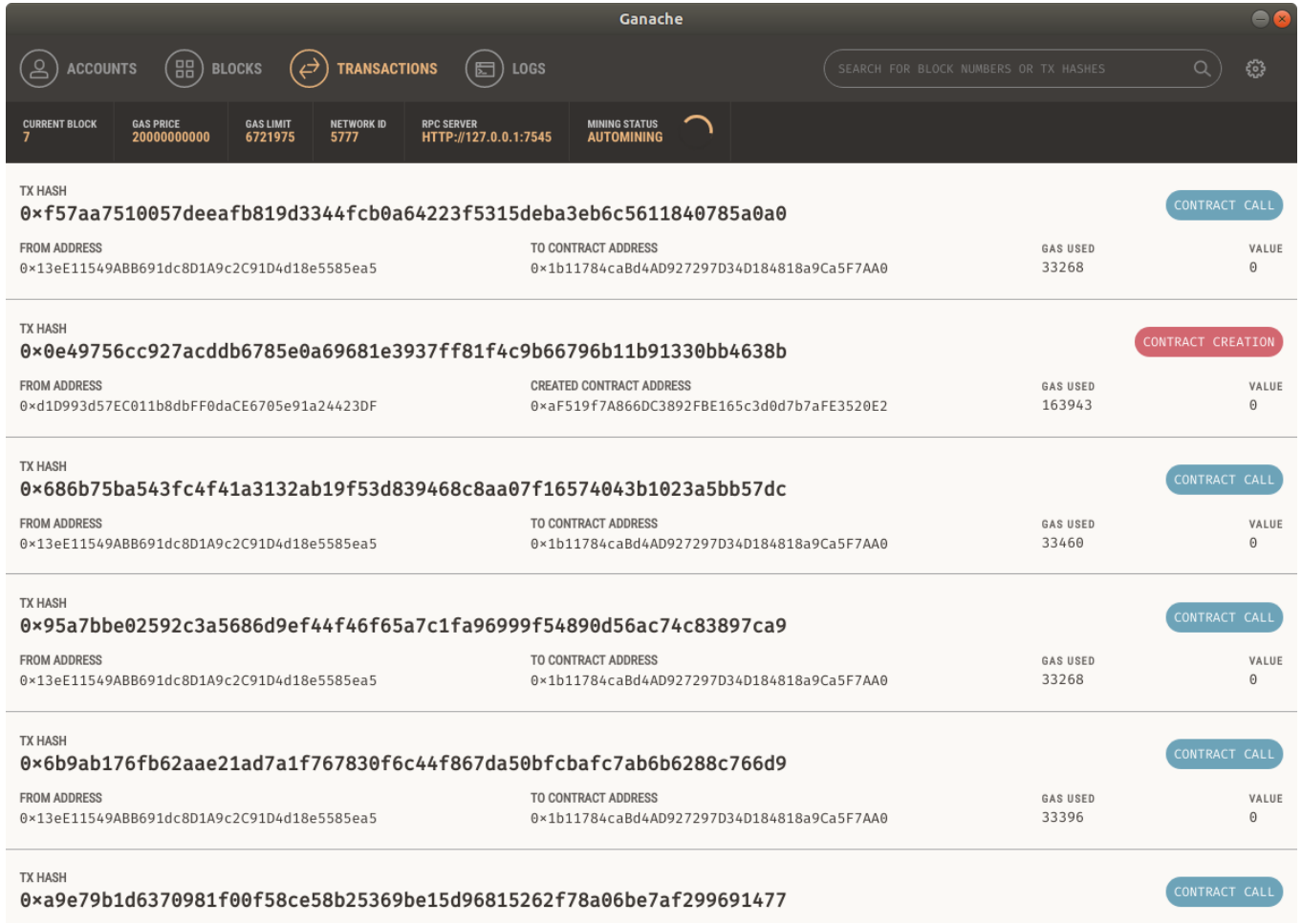


Hashing the previous block for immutability

- Blocks refer back to direct predecessors via **hashing**
- The order matters!



The blockchain remembers



Centralised ledger

"In Amsterdam, the water bailiff's office was located in the […] middle of one of the seafarers' quarters […] open for registering IOUs six days per week.

On one occasion, clerks […] worked **overtime on a Sunday**: presumably because the company of H. Lond, one of the largest lenders in town, had delivered its 1314 unredeemed IOUs the day before and they did not want to start the new week with such a backlog"

Decentralisation for persistence

Centralisation

Decentralisation

Warning: possible information inconsistency → proof-of-* and consensus

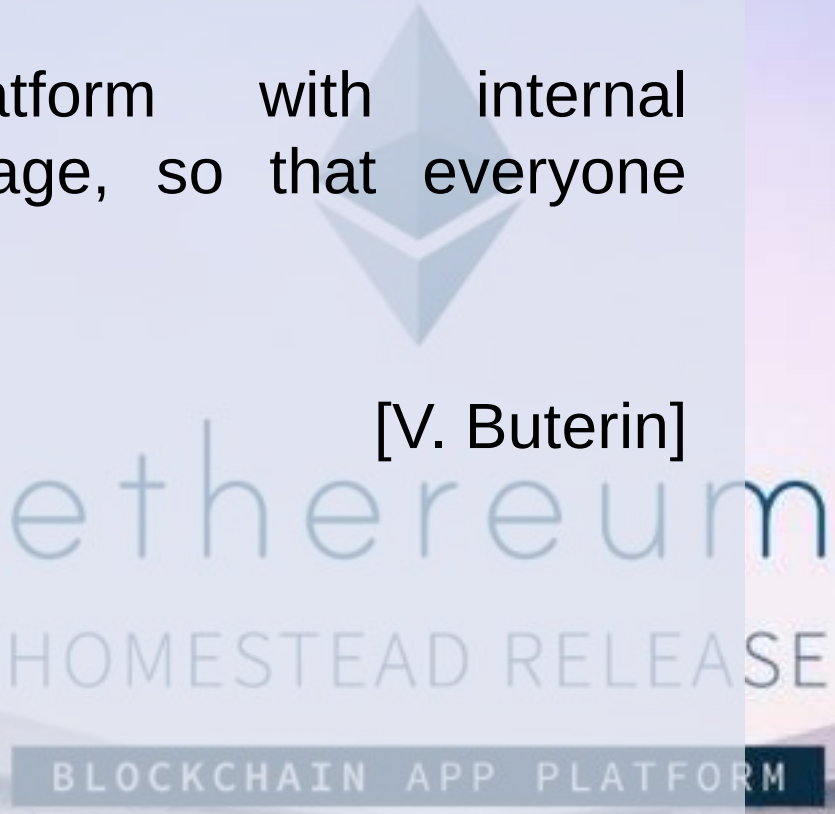
Private|public / Permissioned|permissionless

		Transactability / visibility	
		Private	**Public**
Consensus	**Permissionless**	**Selected** nodes can transact and view, **every** node can participate in consensus	**Every** node can transact and view and participate in consensus
	Permissioned	**Selected** nodes can transact and view, a subset of which can participate in consensus	**Every** node can transact and view, **selected** nodes participate in consensus

Anderberg et al. “Blockchain Now And Tomorrow” (2019) http://dx.doi.org/10.2760/901029

“A universal platform with internal programming language, so that everyone could write any app”

[V. Buterin]



From a peer-to-peer electronic cash system to a programmable distributed environment

Claudio Di Ciccio

Smart Contracts

```solidity
// SPDX-License-Identifier: CC-BY-SA-4.0
pragma solidity >=0.8.0 <0.9.0;

contract HelloToken {
    address public minter;  // The creator of the contract instance
    mapping (address => uint) public balances;  // The balances in Hello-Tokens
    uint public constant PRICE = 2000000000;  // The price of a Hello Token (2 Gwei)

    constructor() {  // Deploys new instances of the smart contract
        minter = msg.sender;  // The sender is the creator
    }

    function mint() public payable {
        // Request the minimum amount for a Hello Token, or terminate
        require(msg.value >= PRICE, "Not enough value for a token!");
        // Add new Hello Tokens to the balance of the sender
        balances[msg.sender] += msg.value / PRICE;
        // The value of the transaction is acquired by the Smart Contract account
    }

    function transfer(uint amount, address to) public {
        require(balances[msg.sender] >= amount, "Not enough tokens!");
        // Decrese the amount from the sender
        balances[msg.sender] -= amount;
        // Increase the amount of Hello Tokens to a specified address
        balances[to] += amount;
    }

    function terminate() public {
        // Only the contract creator can terminate this instance
        require(msg.sender == minter, "You cannot terminate the contract!");
        // Terminate the contract instance and transfer the balance amount to the creator
        selfdestruct(payable(minter));
    }
}
```

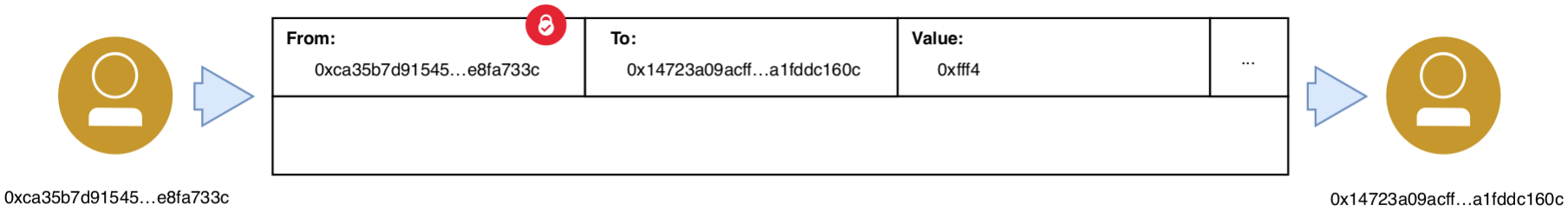
- Smart Contracts in Ethereum
  - live in the Ethereum environment
  - execute a function when called
  - have direct control over their own balance and key/value storage
  - exhibit a behaviour that is fully specified by their **code**

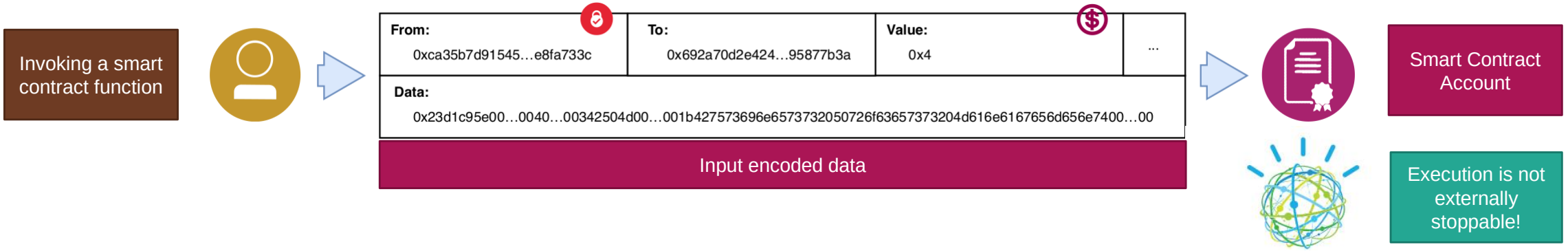
Expressive power of smart contracts

- Variants exist
- **Solidity** is a **Turing-complete** language for the Ethereum blockchain
- Smart contracts can potentially run any computable algorithm

Picture: https://doi.org/10.1093/ref:odnb/36578

A programmable distributed environment

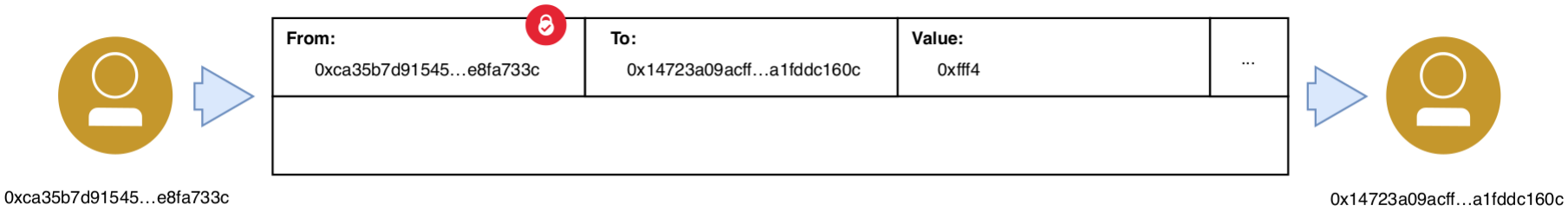




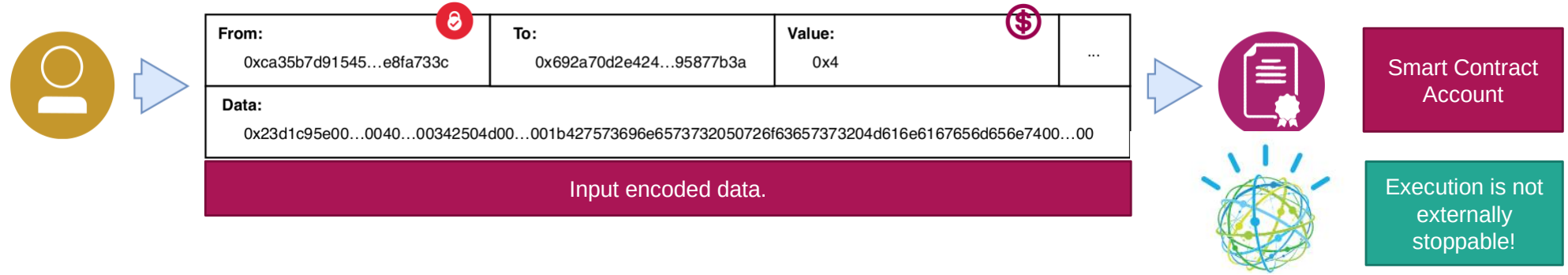
From high-level code to bytecode to bits and bytes

```solidity
// SPDX-License-Identifier: CC-BY-SA-4.0
pragma solidity >=0.8.0 <0.9.0;

contract HelloToken {
    address public minter;  // The creator of the contract instance
    mapping (address => uint) public balances;  // The balances in Hello-Tokens
    uint public constant PRICE = 2000000000;  // The price of a Hello Token (2 Gwei)

    constructor() {  // Deploys new instances of the smart contract
        minter = msg.sender;  // The sender is the creator
    }

    function mint() public payable {
        // Request the minimum amount for a Hello Token, or terminate
        require(msg.value >= PRICE, "Not enough value for a token!");
        // Add new Hello Tokens to the balance of the sender
        balances[msg.sender] += msg.value / PRICE;
        // The value of the transaction is acquired by the Smart Contract account
    }

    function transfer(uint amount, address to) public {
        require(balances[msg.sender] >= amount, "Not enough tokens!");
        // Decrese the amount from the sender
        balances[msg.sender] -= amount;
        // Increase the amount of Hello Tokens to a specified address
        balances[to] += amount;
    }

    function terminate() public {
        // Only the contract creator can terminate this instance
        require(msg.sender == minter, "You cannot terminate the contract!");
        // Terminate the contract instance and transfer the balance amount to the creator
        selfdestruct(payable(minter));
    }
}
```

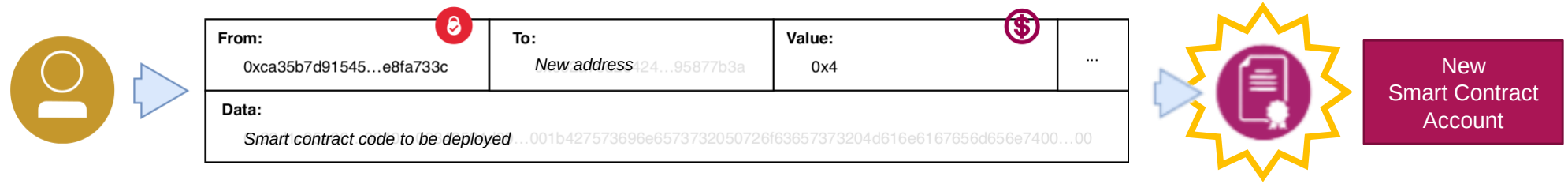
A programmable distributed environment



Invoking a smart contract function



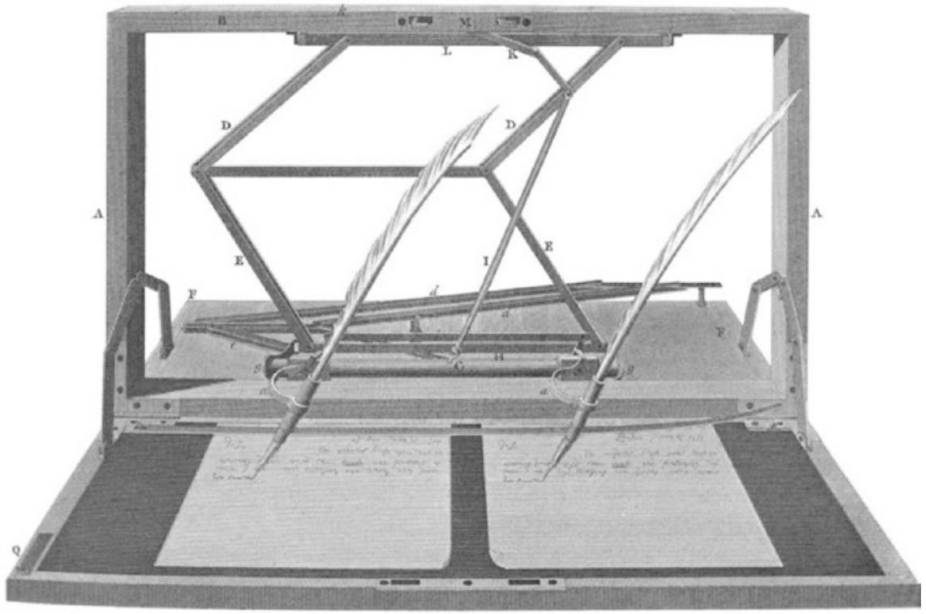
Deploying a new smart contract



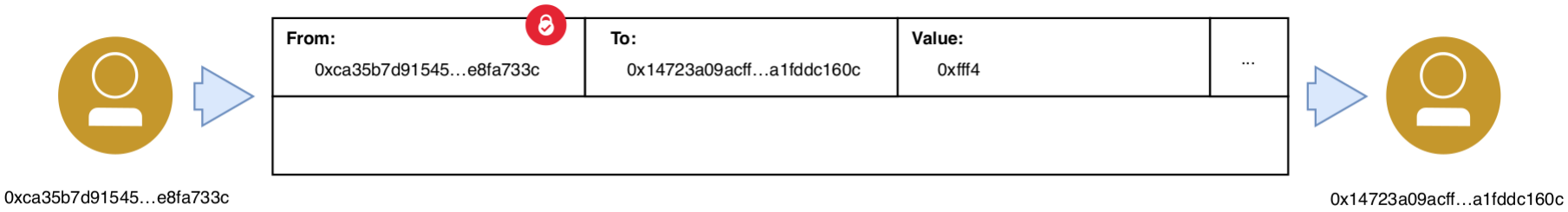
The polygraph machine

Where are Smart Contracts executed?

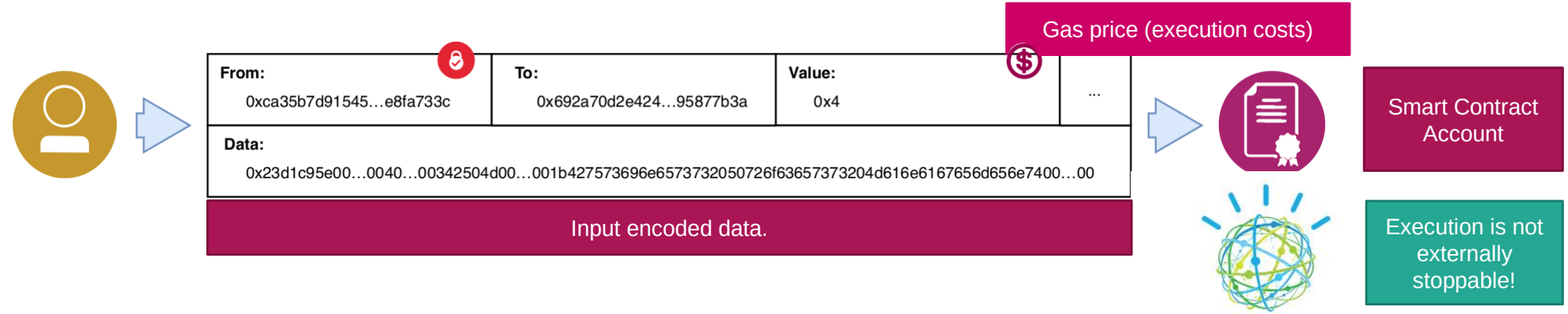
First on the mining nodes.
Then, potentially, on every node!



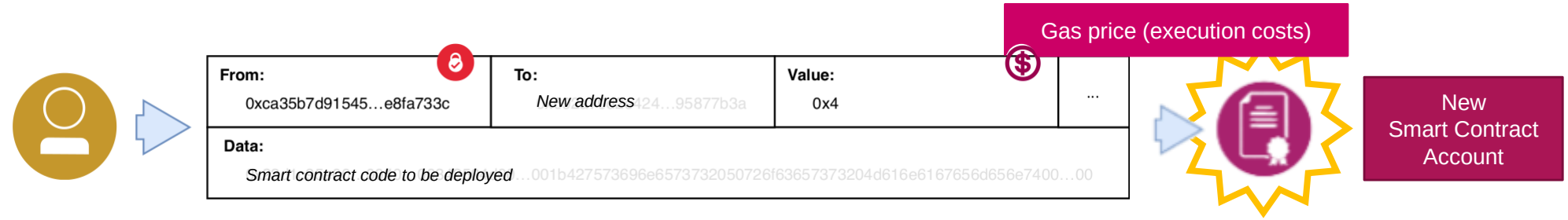
A programmable distributed environment



Invoking a smart contract function



Deploying a new smart contract



Execution is not for free (most of all, in public blockchains)

Name	Value	Description*
G_{zero}	0	Nothing paid for operations of the set W_{zero}.
G_{base}	2	Amount of gas to pay for operations of the set W_{base}.
$G_{verylow}$	3	Amount of gas to pay for operations of the set $W_{verylow}$.
G_{low}	5	Amount of gas to pay for operations of the set W_{low}.
G_{mid}	8	Amount of gas to pay for operations of the set W_{mid}.
G_{high}	10	Amount of gas to pay for operations of the set W_{high}.
$G_{extcode}$	700	Amount of gas to pay for operations of the set $W_{extcode}$.
$G_{balance}$	400	Amount of gas to pay for a BALANCE operation.
G_{sload}	200	Paid for a SLOAD operation.
$G_{jumpdest}$	1	Paid for a JUMPDEST operation.
G_{sset}	20000	Paid for an SSTORE operation when the storage value is set to non-zero from zero.
G_{sreset}	5000	Paid for an SSTORE operation when the storage value's zeroness remains unchanged or is set to zero.
R_{sclear}	15000	Refund given (added into refund counter) when the storage value is set to zero from non-zero.
$R_{selfdestruct}$	24000	Refund given (added into refund counter) for self-destructing an account.
$G_{selfdestruct}$	5000	Amount of gas to pay for a SELFDESTRUCT operation.
G_{create}	32000	Paid for a CREATE operation.
$G_{codedeposit}$	200	Paid per byte for a CREATE operation to succeed in placing code into state.
G_{call}	700	Paid for a CALL operation.
$G_{callvalue}$	9000	Paid for a non-zero value transfer as part of the CALL operation.
$G_{callstipend}$	2300	A stipend for the called contract subtracted from $G_{callvalue}$ for a non-zero value transfer.
$G_{newaccount}$	25000	Paid for a CALL or SELFDESTRUCT operation which creates an account.
G_{exp}	10	Partial payment for an EXP operation.
$G_{expbyte}$	50	Partial payment when multiplied by $\lceil \log_{256}(exponent) \rceil$ for the EXP operation.
G_{memory}	3	Paid for every additional word when expanding memory.
G_{txcreate}	32000	Paid by all contract-creating transactions after the *Homestead* transition.
$G_{txdatazero}$	4	Paid for every zero byte of data or code for a transaction.
$G_{txdatanonzero}$	68	Paid for every non-zero byte of data or code for a transaction.
$G_{transaction}$	21000	Paid for every transaction.
G_{log}	375	Partial payment for a LOG operation.
G_{logdata}	8	Paid for each byte in a LOG operation's data.
G_{logtopic}	375	Paid for each topic of a LOG operation.
G_{sha3}	30	Paid for each SHA3 operation.
$G_{sha3word}$	6	Paid for each word (rounded up) for input data to a SHA3 operation.
G_{copy}	3	Partial payment for *COPY operations, multiplied by words copied, rounded up.
$G_{blockhash}$	20	Payment for BLOCKHASH operation.
$G_{quaddivisor}$	100	The quadratic coefficient of the input sizes of the exponentiation-over-modulo precompiled contract.

Wood , "Ethereum: A secure decentralised generalised transaction ledger" (2014), https://ethereum.github.com/yellowpaper/paper.pdf

Challenges about costs

Home › Business Process Management: Blockchain, Robotic Process Automation, and Central and Eastern Europe Forum

Blockchain for Business Process Enactment: A Taxonomy and Systematic Literature Review

Fabian Stiehle & Ingo Weber

Conference paper | First Online: 07 September 2022

1018 Accesses | 2 Citations

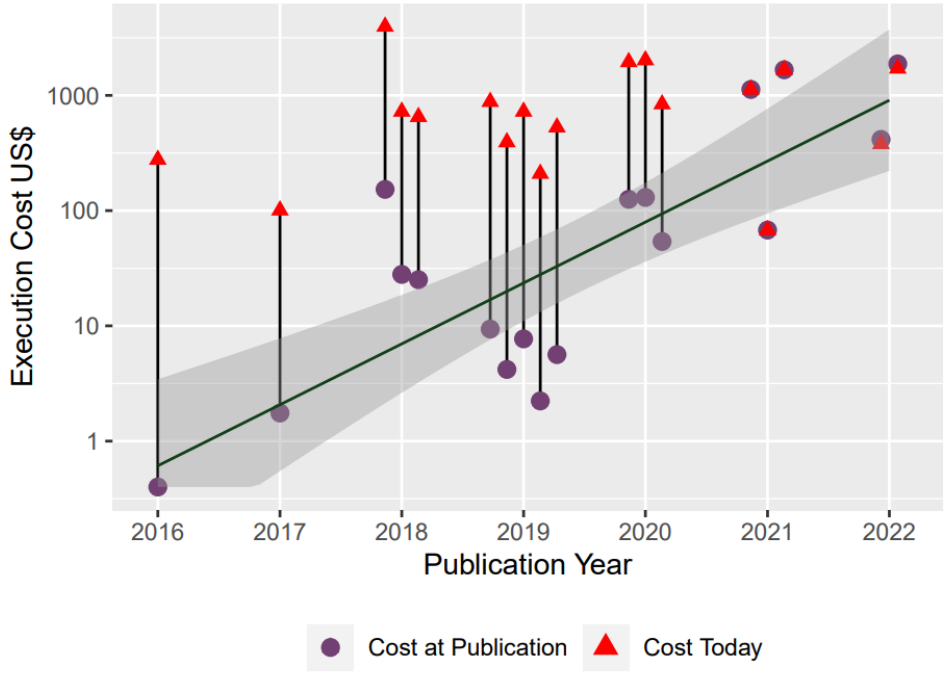
Part of the Lecture Notes in Business Information Processing book series (LNBIP,volume 459)

Abstract

Blockchain has been proposed to facilitate the enactment of interorganisational business processes. For such processes, blockchain can guarantee the enforcement of rules and the integrity of execution traces—without the need for a centralised trusted party. However, the enactment of interorganisational processes pose manifold challenges. In this work, we ask what answers the research field offers in response to those challenges. To do so, we conduct a systematic literature review (SLR). As our guiding question, we investigate the guarantees and capabilities of blockchain-based enactment approaches. Based on this SLR, we develop a taxonomy for blockchain-based enactment. We find that a wide range of approaches support traceability and correctness; however, research focusing on flexibility and scalability remains nascent. For all challenges, we point towards future research opportunities.

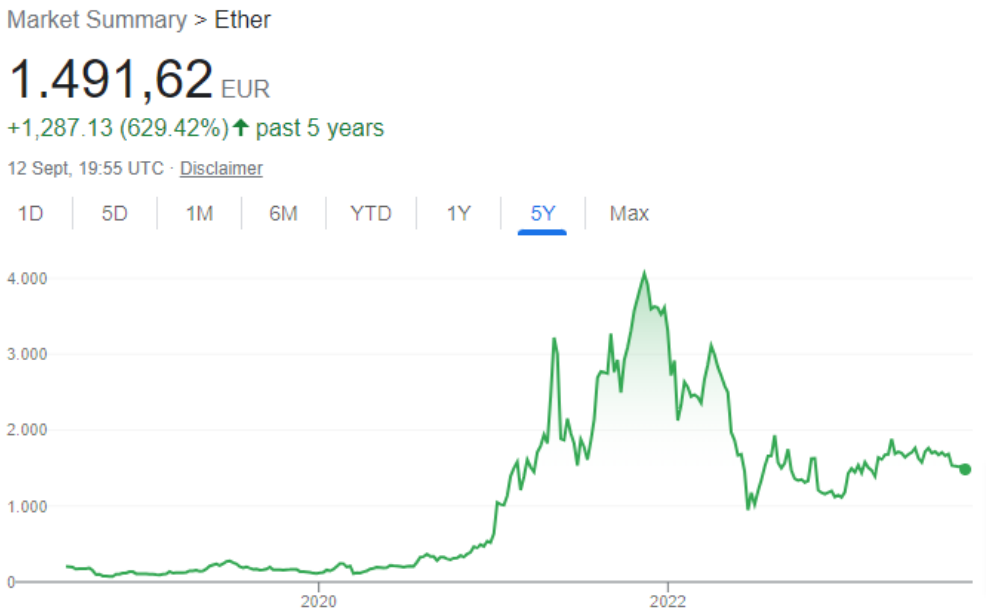
Keywords

Blockchain | Business process enactment | Business process execution | Interorganisational processes | Taxonomy | SLR

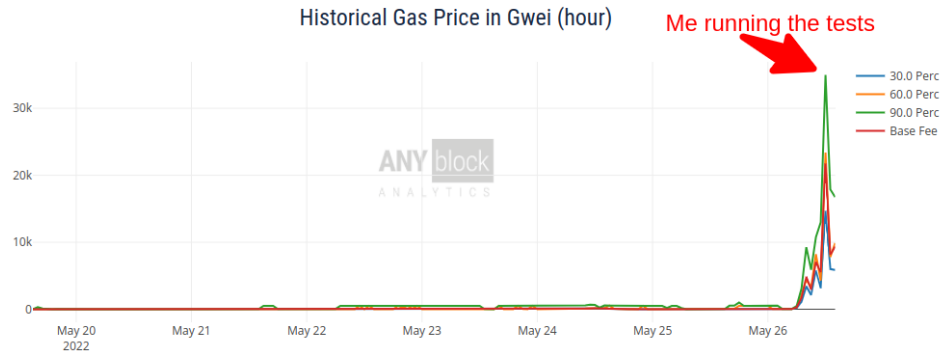


Price instability of cryptocurrency and gas prices

ETH/EUR exchange



Gas price on the Ropsten testnet



Cryptos and fiat money

Keep smart contracts lean!
Only absolutely needed instructions should be in the code.

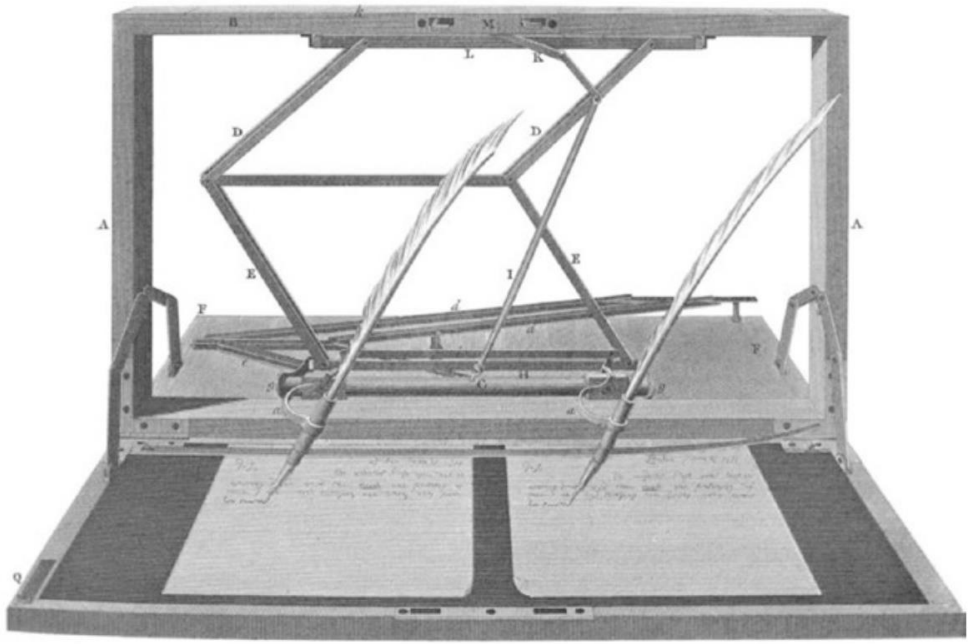


Image source: https://www.kbrillphotograp.top/products.aspx?cname=jlo+jenny+from+the+block+outfit&cid=140

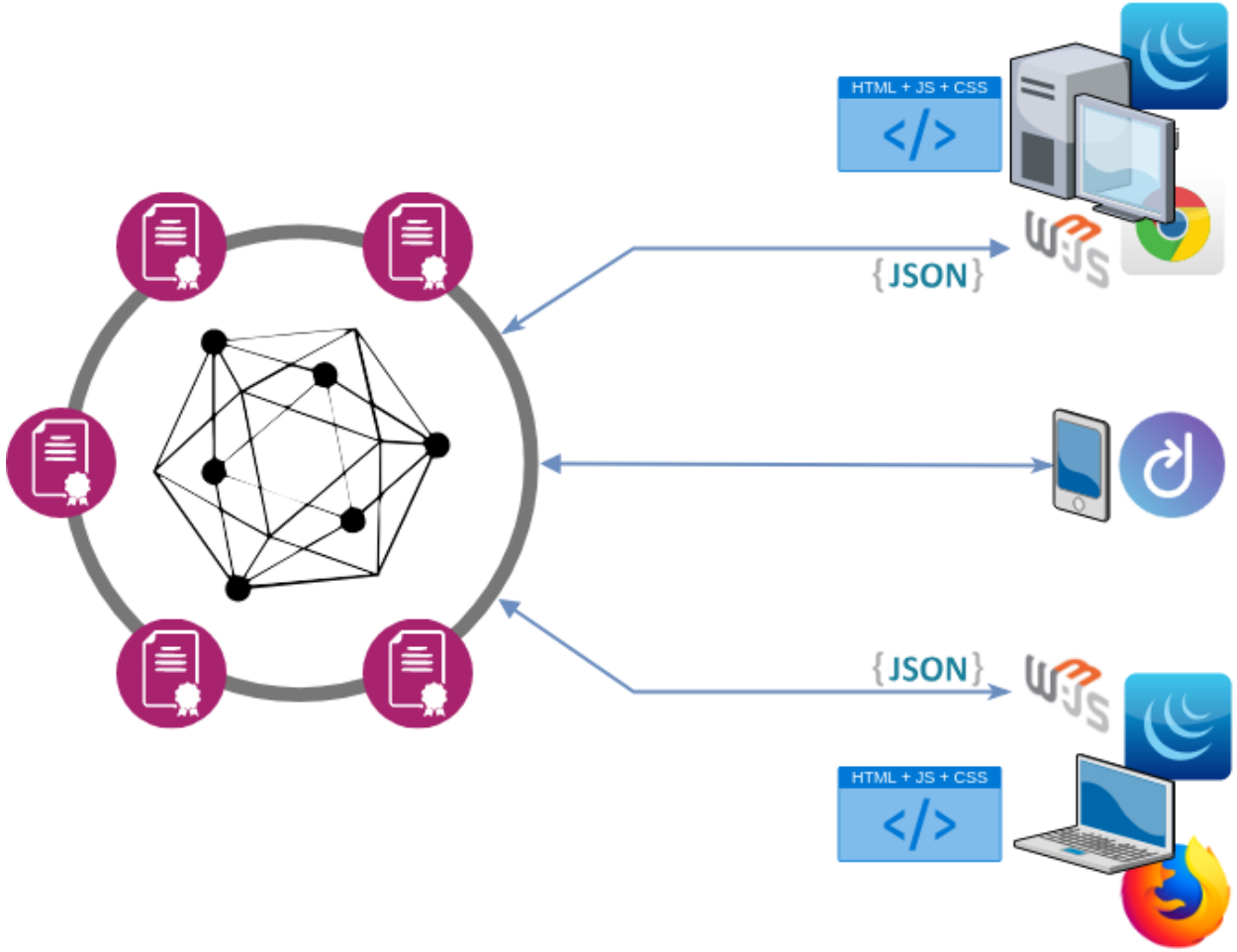
The paradigm

Mainframe

Terminal

L: https://commons.wikimedia.org/wiki/File:SiemensHicom300_1200.jpg; R: https://commons.wikimedia.org/wiki/File:HP_2647A_terminal.jpg

Web 3.0 and Decentralised applications (DApps)



Advantages and connection to processes

- Smart contracts → Programmability → Process rule enforcing
- Transactions → Asset transfer & function invocation → Process execution
- Distributed store → Data persistency → Process monitoring
- Ledger → Transaction ordering → Logging
- Hashing → Robustness → Secure storage
- Signatures → Authentication → Non-repudiability
- Consensus → Eventual consistency → Traceability

Layer of **trust** even in a regime of partial trust among actors

Blockchains for Business Process Management - Challenges and Opportunities

Authors: Jan Mendling, Ingo Weber, Wil Van Der Aalst, Jan Vom Brocke, Cristina Cabanillas, Florian Daniel, Søren Debois, Claudio Di Ciccio, Marlon Dumas, Schahram Dustdar, + 22

Authors Info & Claims

ACM Transactions on Management Information Systems, Volume 9, Issue 1 • Article No.: 4, pp 1–16 • https://doi.org/10.1145/3183367

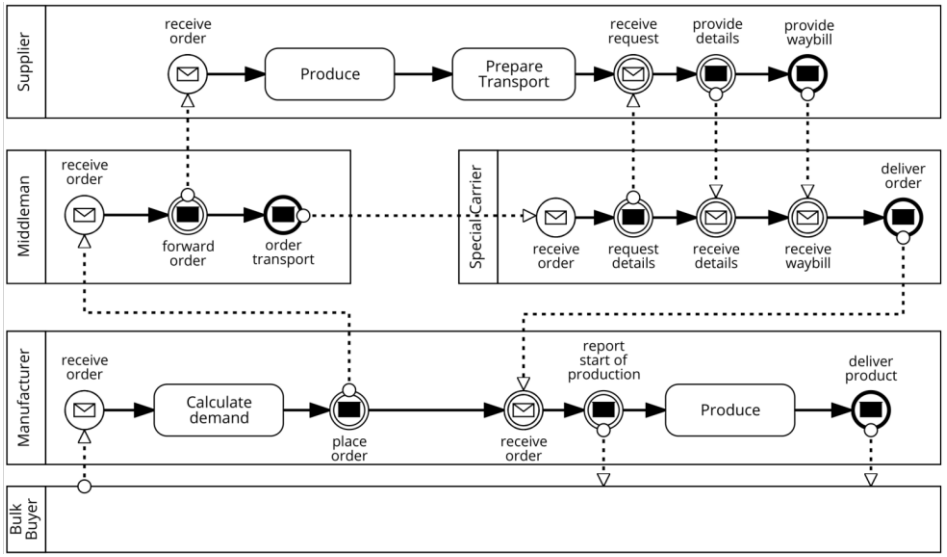
Dagstuhl Seminar 18332

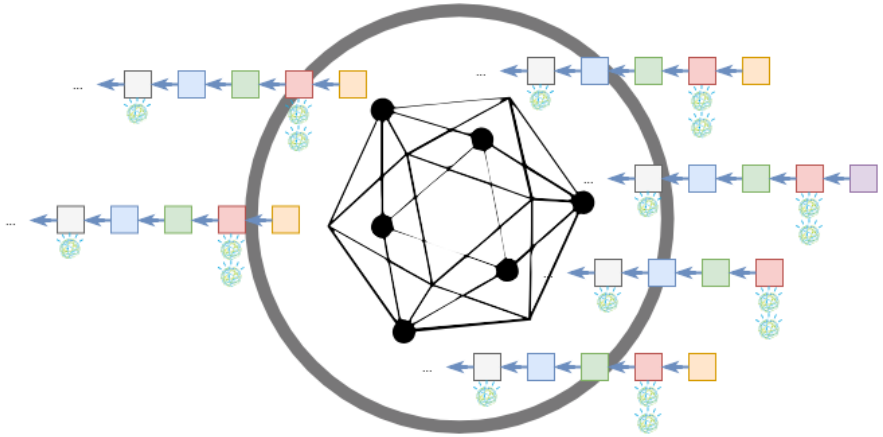
Blockchain Technology for Collaborative Information Systems

(Aug 12 – Aug 17, 2018)

Images: Left: https://doi.org/10.1145/3183367; right: https://www.dagstuhl.de/en/seminars/seminar-calendar/seminar-details/18332

Executing inter-organisational processes on the Blockchain: A model-driven approach

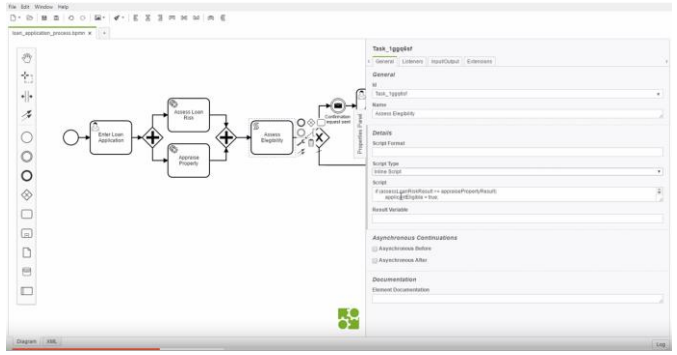




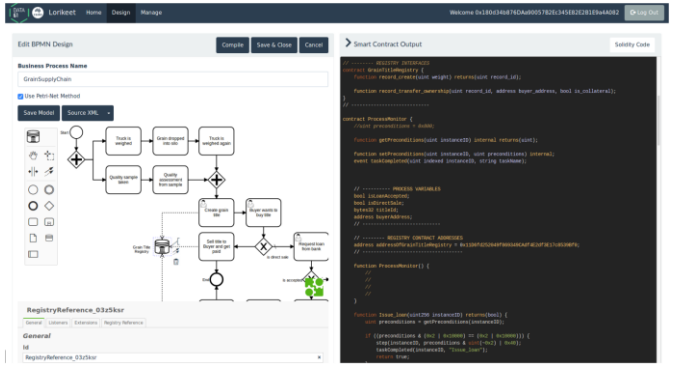
Executing inter-organisational processes on the Blockchain: A model-driven approach

López-Pintado, García-Bañuelos, Dumas, Weber. **Caterpillar**: A blockchain-based business process management system. In: BPM Demos. CEUR.ws, 2017.
Tran, Lu, Weber. **Lorikeet**: A Model-Driven Engineering Tool for Blockchain-Based Business Process Execution and Asset. In: BPM Demos. CEUR.ws, 2018.
Corradini, Marcelletti, Morichetta, Polini, Re, Tiezzi: Engineering Trustable and Auditable Choreography-based Systems Using Blockchain. ACM TMIS 13(3), 2022.

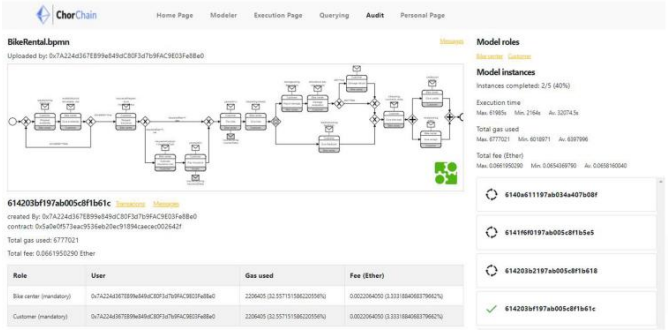
Caterpillar



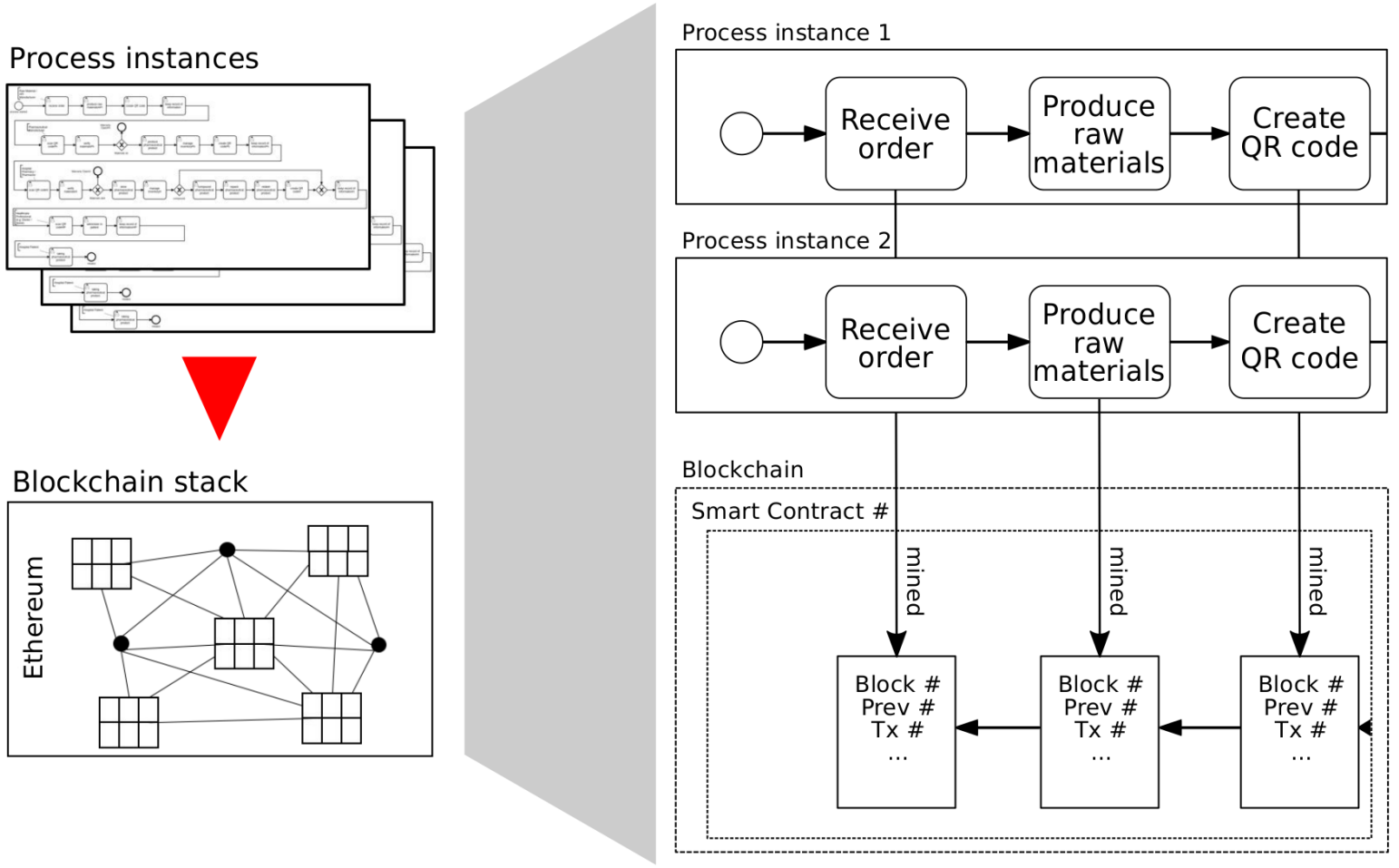
Lorikeet



ChorChain

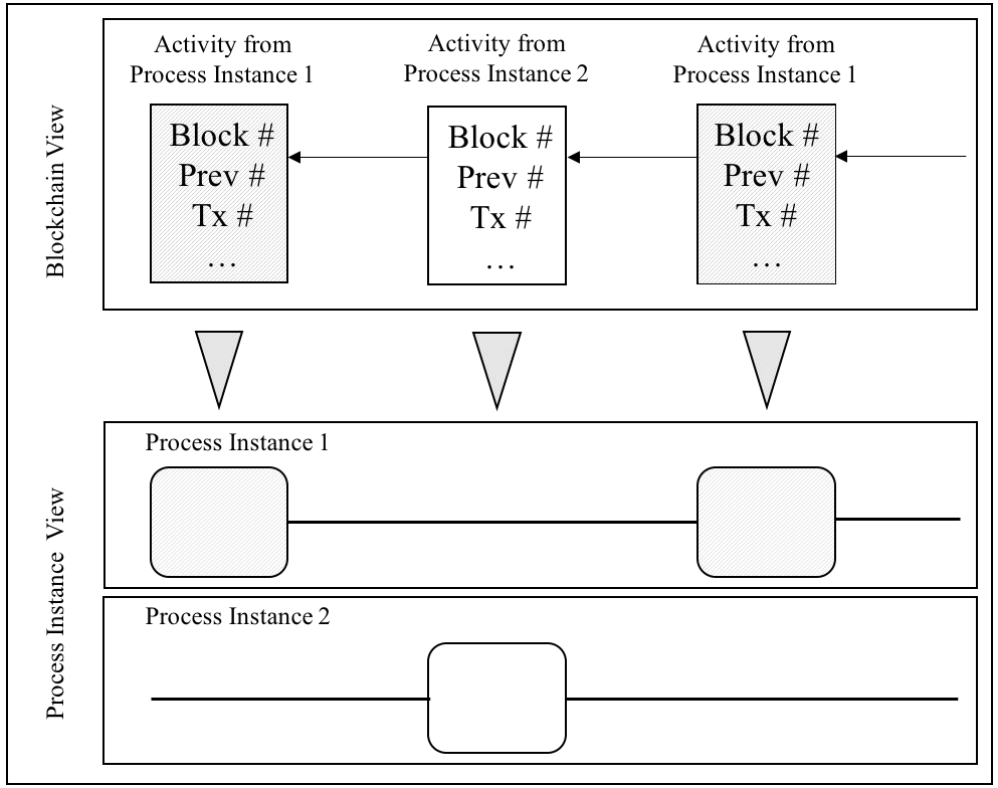


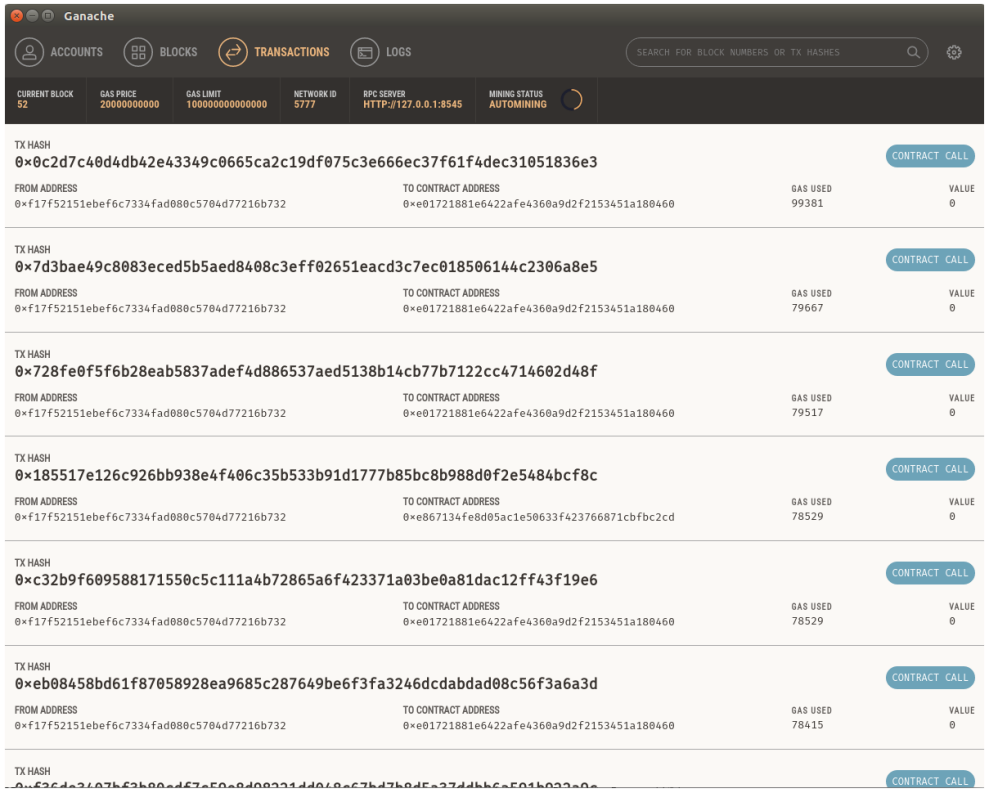
Tracking execution



D.C. et al. Blockchain-Based Traceability of Inter-organisational Business Processes. In: BMSD. Springer, 2018.

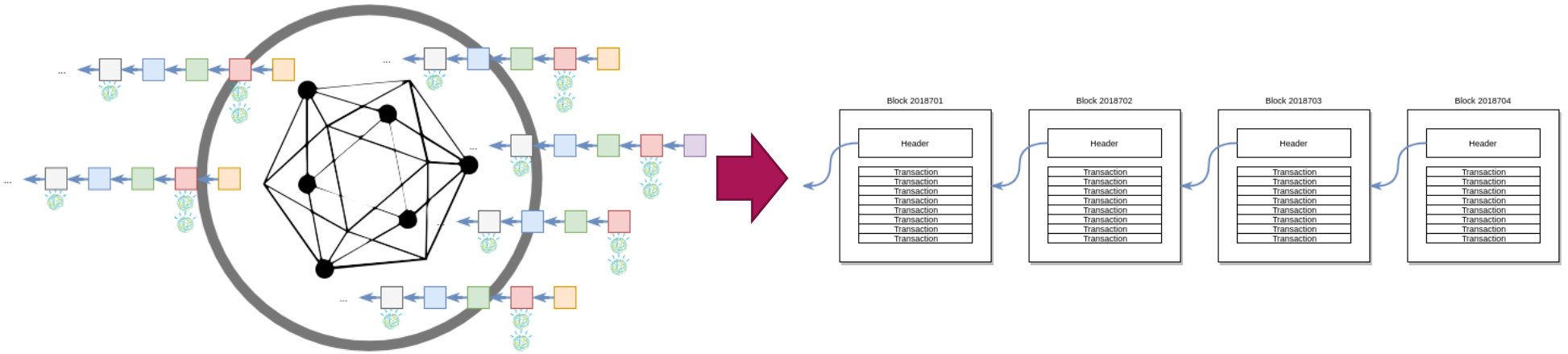
Traceability



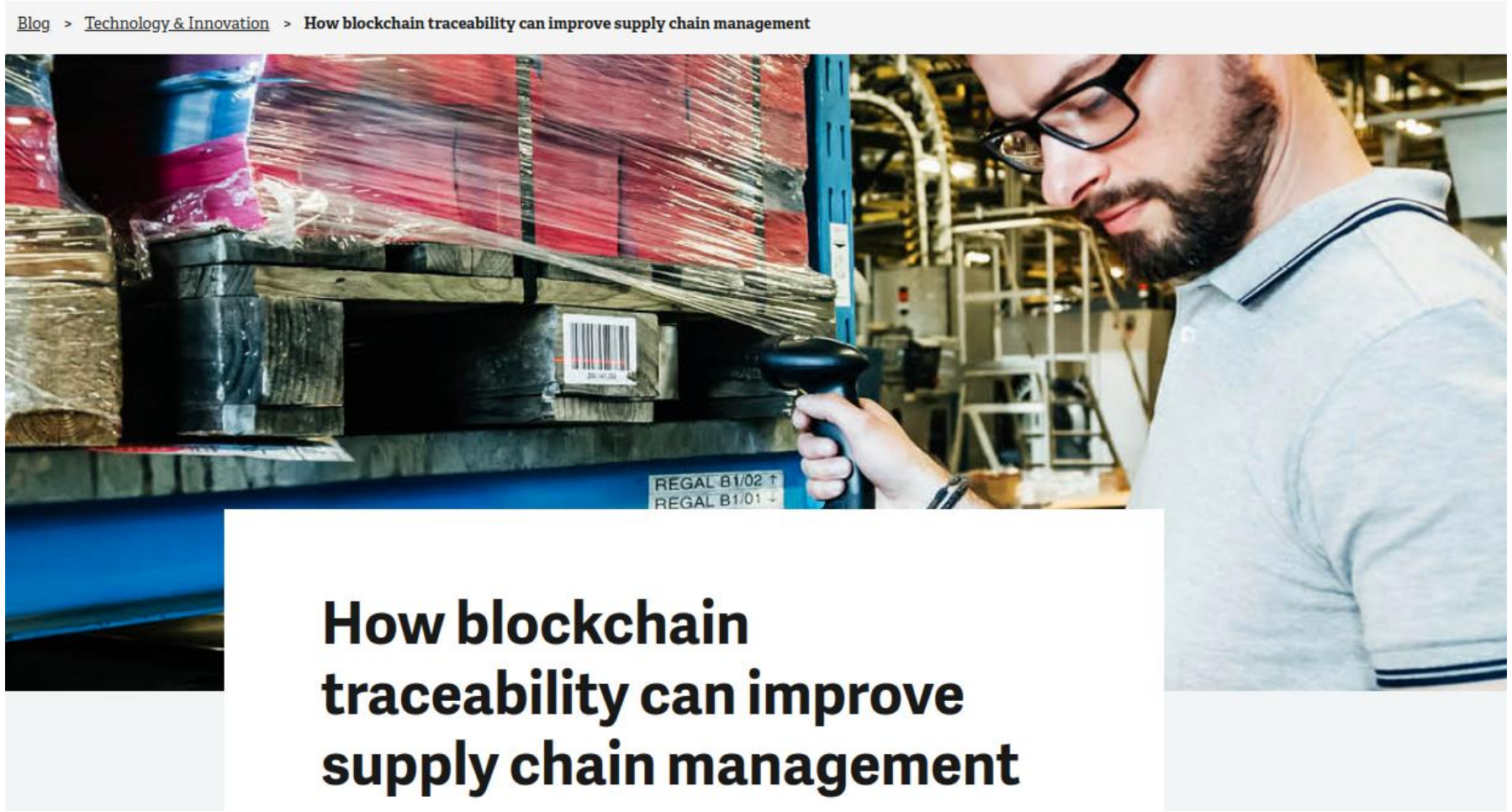


D.C. et al. Blockchain-Based Traceability of Inter-organisational Business Processes. In: BMSD. Springer, 2018.

From execution to ledgers

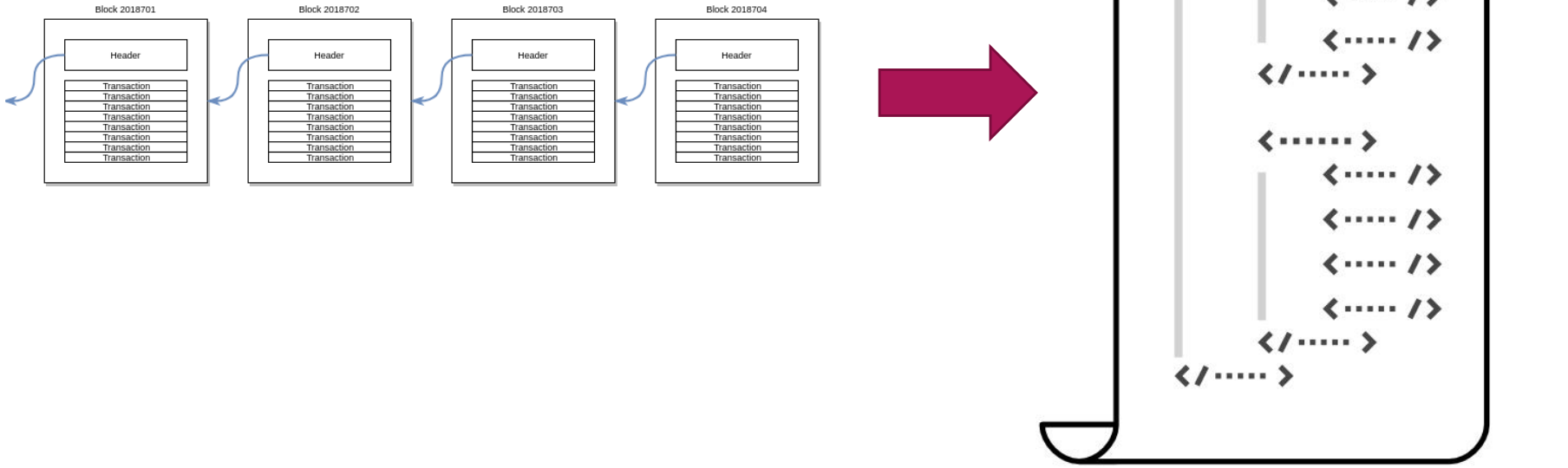


Traceability

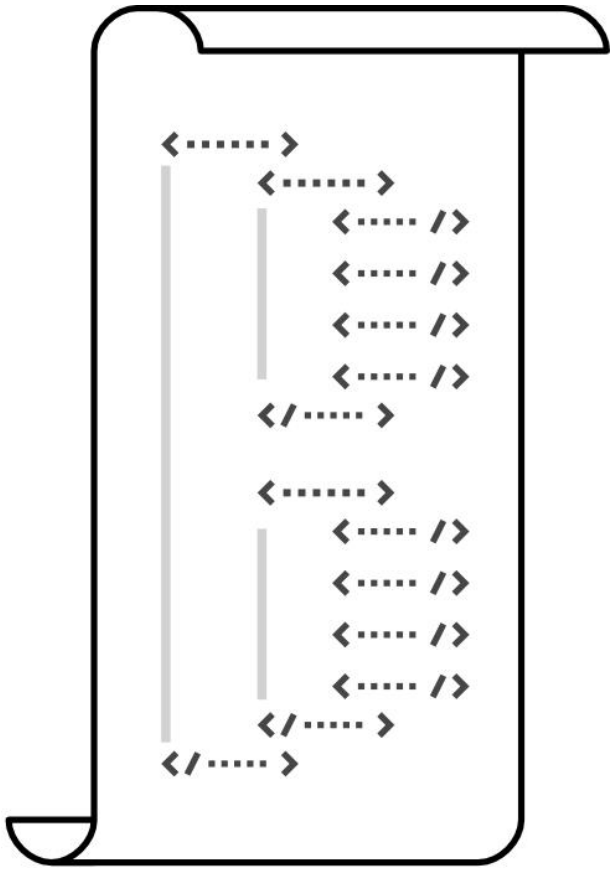


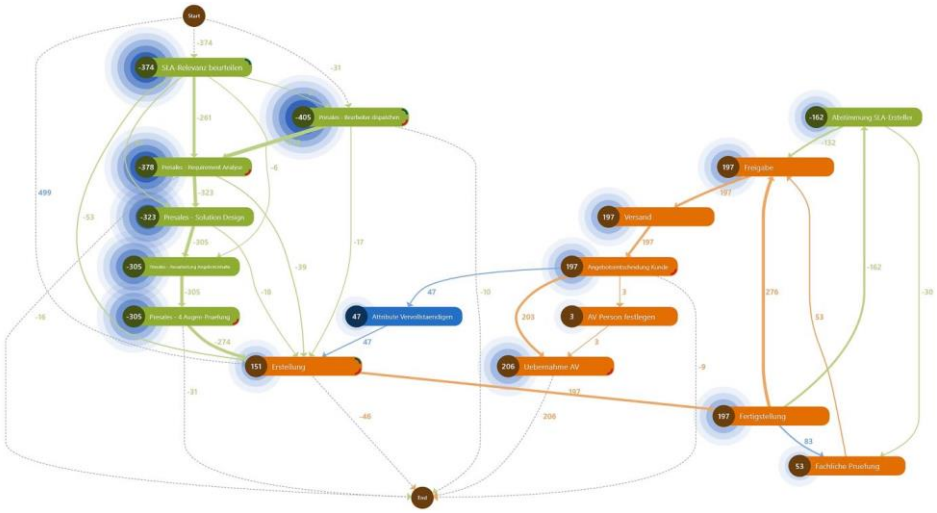
https://www.sage.com/en-gb/blog/blockchain-traceability-supply-chain/ Accessed: 19/02/2020

From ledgers to time-ordered datasets



From data sets to process mining and analytics





Mining blockchain processes

Home > Business Process Management: Blockchain and Central and Eastern Europe Forum > Conference paper

Mining Blockchain Processes: Extracting Process Mining Data from Blockchain Applications

Christopher Klinkmüller, Alexander Ponomarev, An Binh Tran, Ingo Weber & Wil van der Aalst

Conference paper | First Online: 26 August 2019

3957 Accesses | 18 Citations | 1 Altmetric

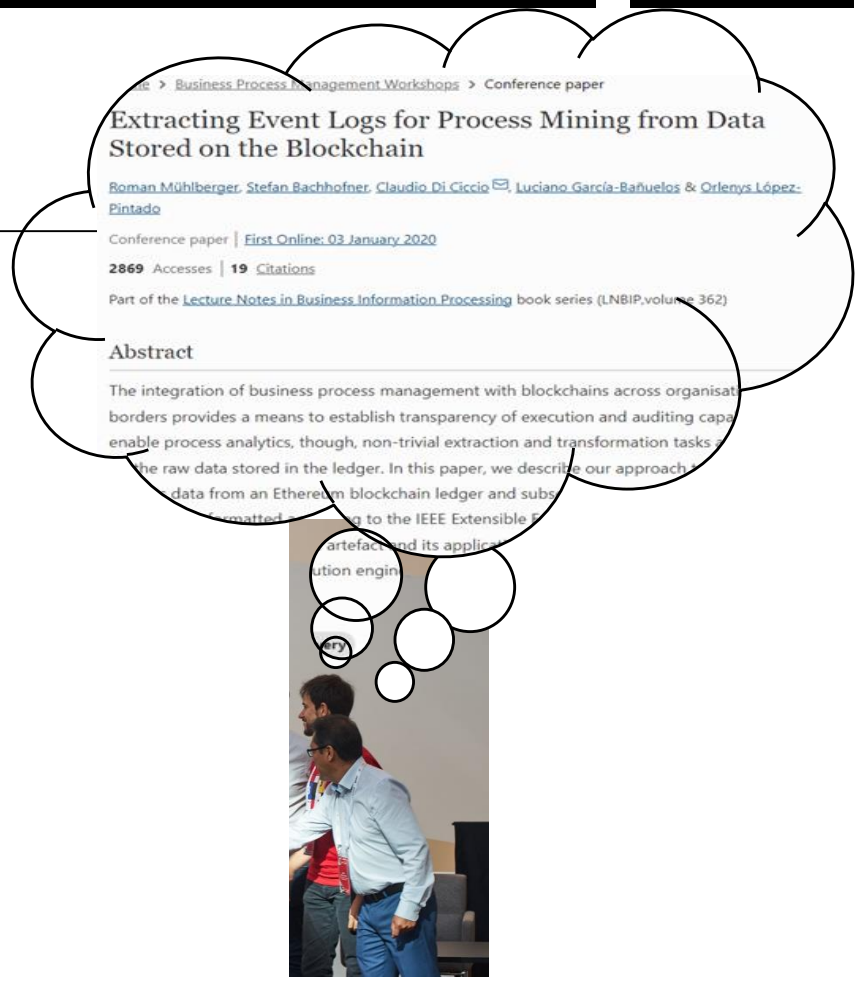
Part of the Lecture Notes in Business Information Processing book series (LNBIP,volume 361)

Abstract

Blockchain technology has been gaining popularity as a platform for developing decentralized applications and executing cross-organisational processes. However, extracting data that allows analysing the process view from blockchains is surprisingly hard. Therefore, blockchain data are rarely used for process mining. In this paper, we propose a framework for alleviating that pain. The framework comprises three main parts: a manifest specifying how data is logged, an extractor for retrieving data (structured according to the XES standard), and a generator that produces logging code to support smart contract developers. Among others, we propose a convenient way to encode logging data in a compact form, to achieve relatively low cost and high throughput for on-chain logging. The proposal is evaluated with logs created from generated logging code, as well as with existing blockchain applications that do not make use of the proposed code generator.

Keywords

Process mining | Blockchain | Smart contracts | Logging | XES



Mining blockchain processes

Home > Business Process Management: Blockchain and Central and Eastern Europe Forum > Conference paper

Mining Blockchain Processes: Extracting Process Mining Data from Blockchain Applications

Christopher Klinkmüller, Alexander Ponomarev, An Binh Tran, Ingo Weber & Wil van der Aalst

Conference paper | First Online: 26 August 2019

3957 Accesses | 18 Citations | 1 Altmetric

Part of the Lecture Notes in Business Information Processing book series (LNBIP,volume 361)

Abstract

Blockchain technology has been gaining popularity as a platform for developing decentralized applications and executing cross-organisational processes. However, extracting data that allows analysing the process view from blockchains is surprisingly hard. Therefore, blockchain data are rarely used for process mining. In this paper, we propose a framework for alleviating that pain. The framework comprises three main parts: a manifest specifying how data is logged, an extractor for retrieving data (structured according to the XES standard), and a generator that produces logging code to support smart contract developers. Among others, we propose a convenient way to encode logging data in a compact form, to achieve relatively low cost and high throughput for on-chain logging. The proposal is evaluated with logs created from generated logging code, as well as with existing blockchain applications that do not make use of the proposed code generator.

Keywords

Process mining | Blockchain | Smart contracts | Logging | XES

Business Process Management Workshops > Conference paper

Extracting Event Logs for Process Mining from Data Stored on the Blockchain

Roman Mühlberger, Stefan Bachhofer, Claudio Di Ciccio, Luciano García-Bañuelos & Orlenys López-Pintado

Conference paper | First Online: 03 January 2020

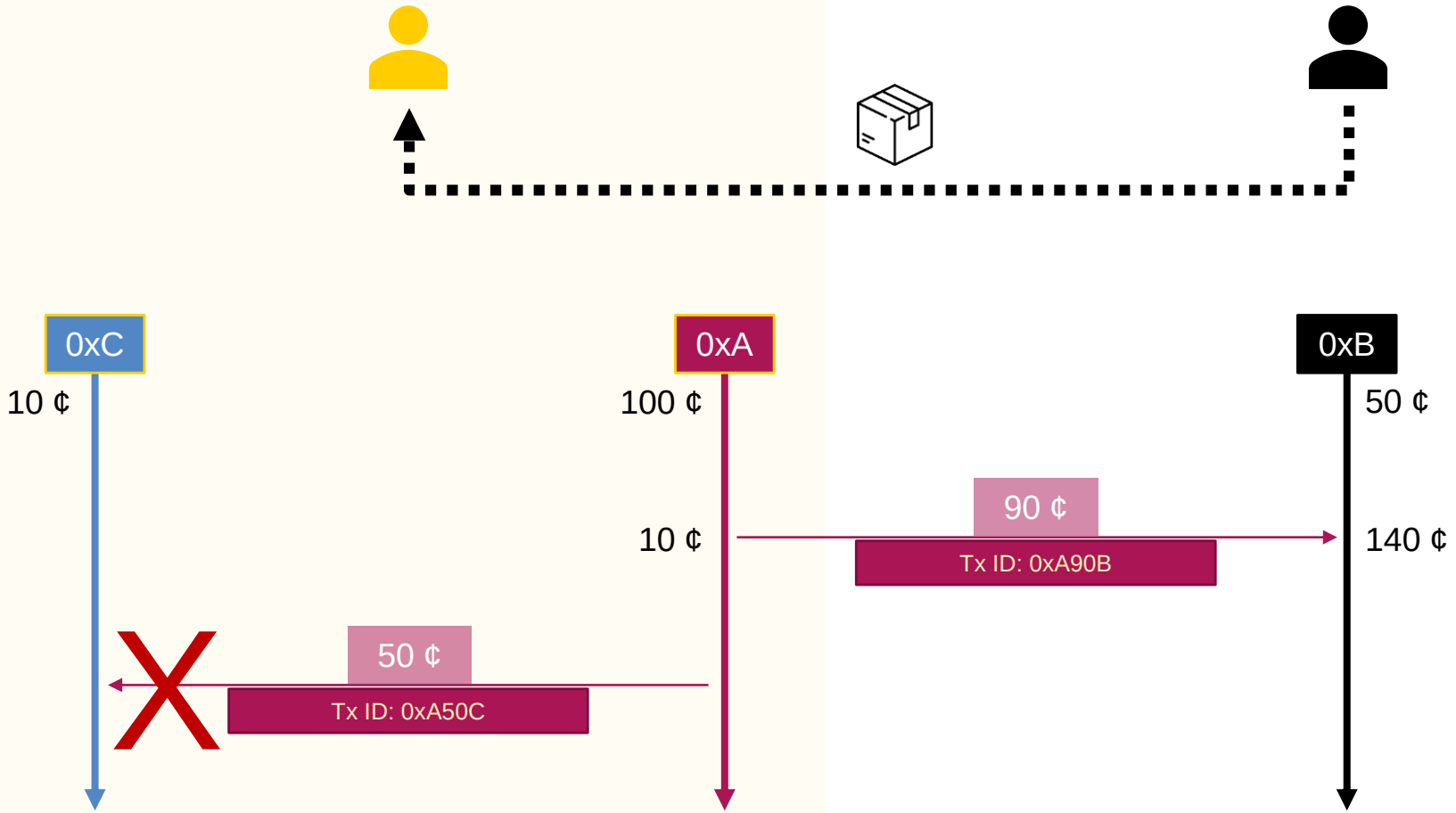
2869 Accesses | 19 Citations

Part of the Lecture Notes in Business Information Processing book series (LNBIP,volume 362)

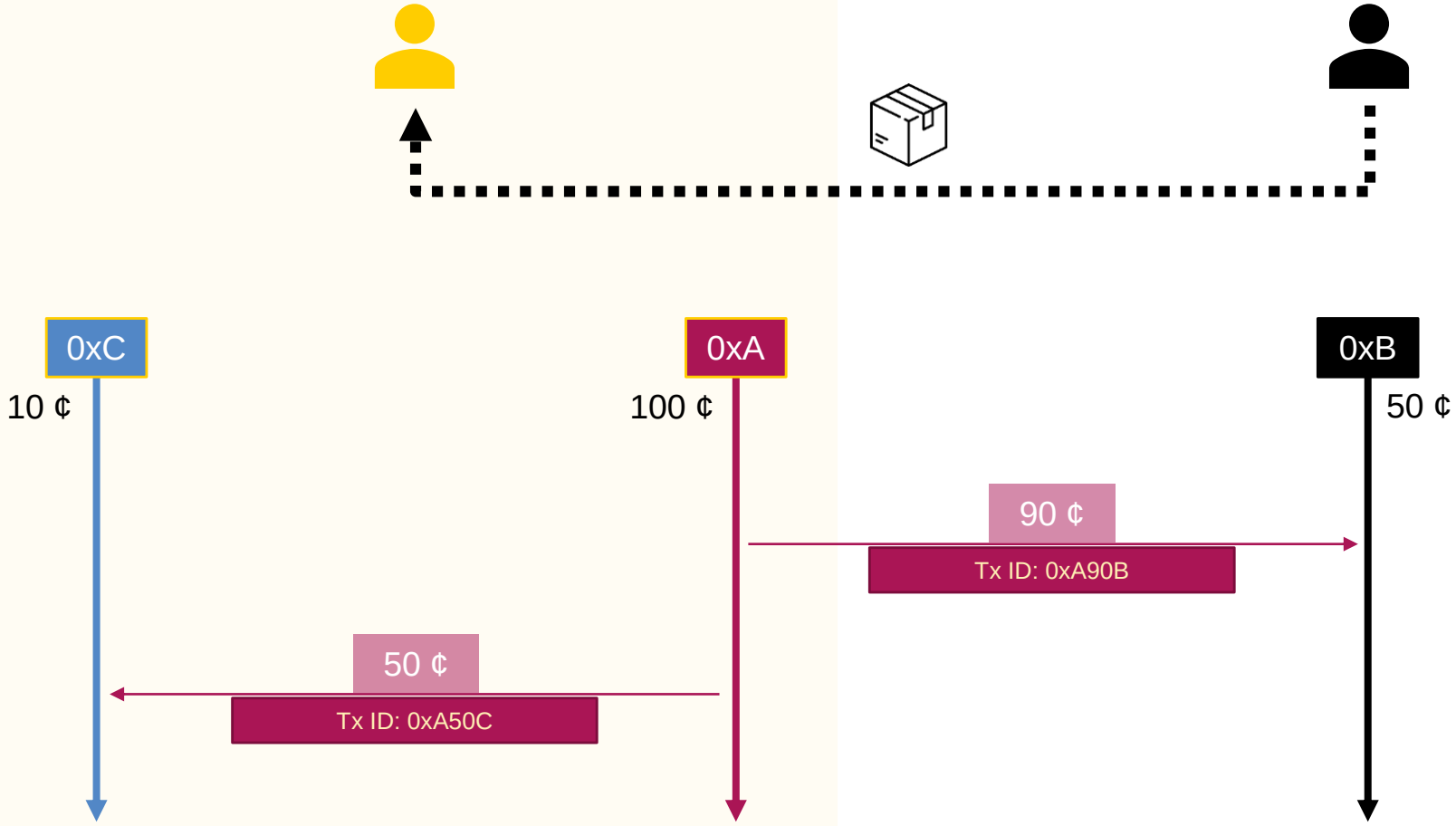
Abstract

The integration of business process management with blockchains across organisat borders provides a means to establish transparency of execution and auditing capa enable process analytics, though, non-trivial extraction and transformation tasks the raw data stored in the ledger. In this paper, we describe our approach data from an Ethereum blockchain ledger and subs to the IEEE Extensible artefact and its ution engin

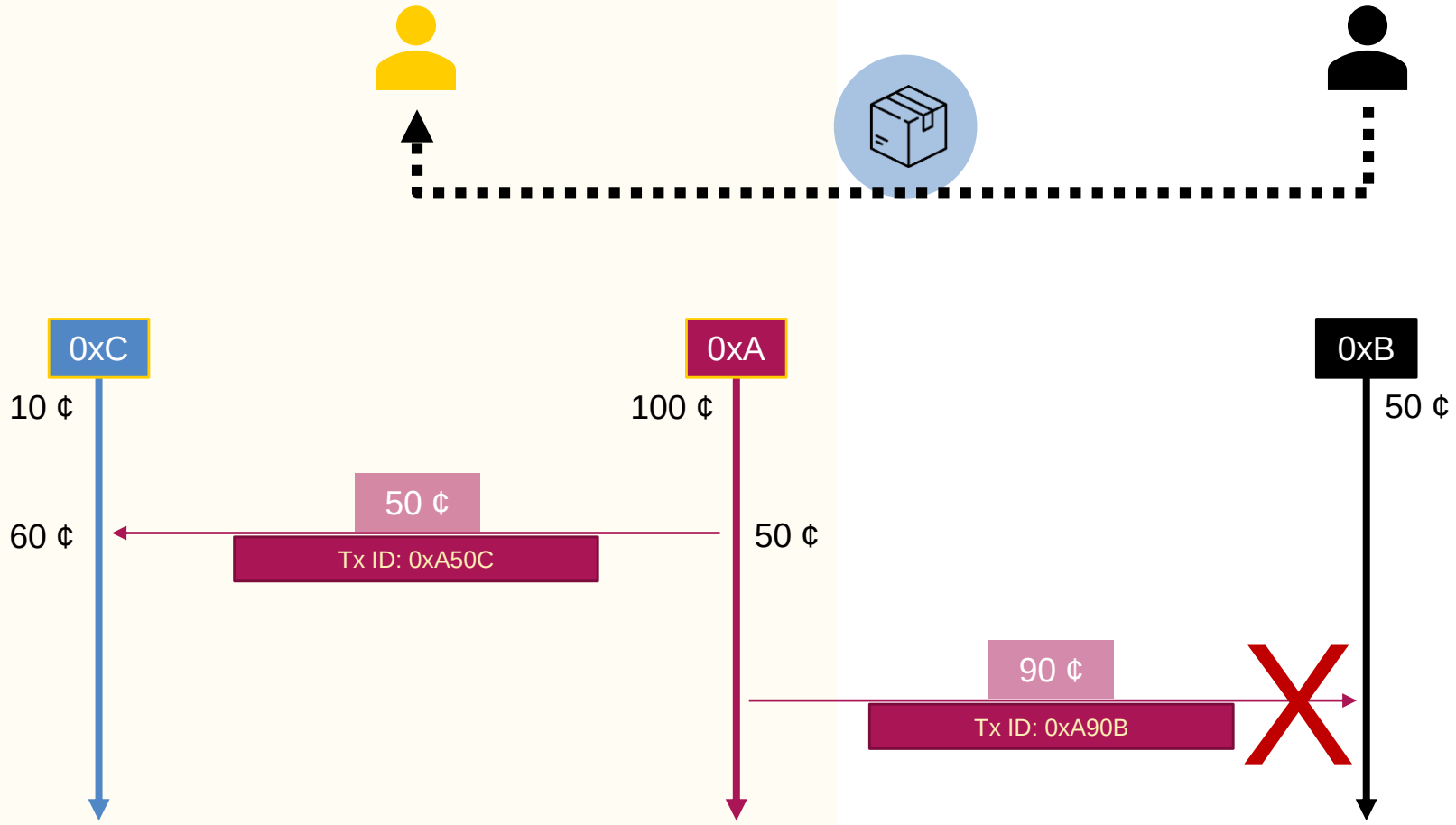
Double spending



Double spending



Double spending



On-chain vs off-chain

The broken link is that blockchain natively has no control on or view of off-chain objects

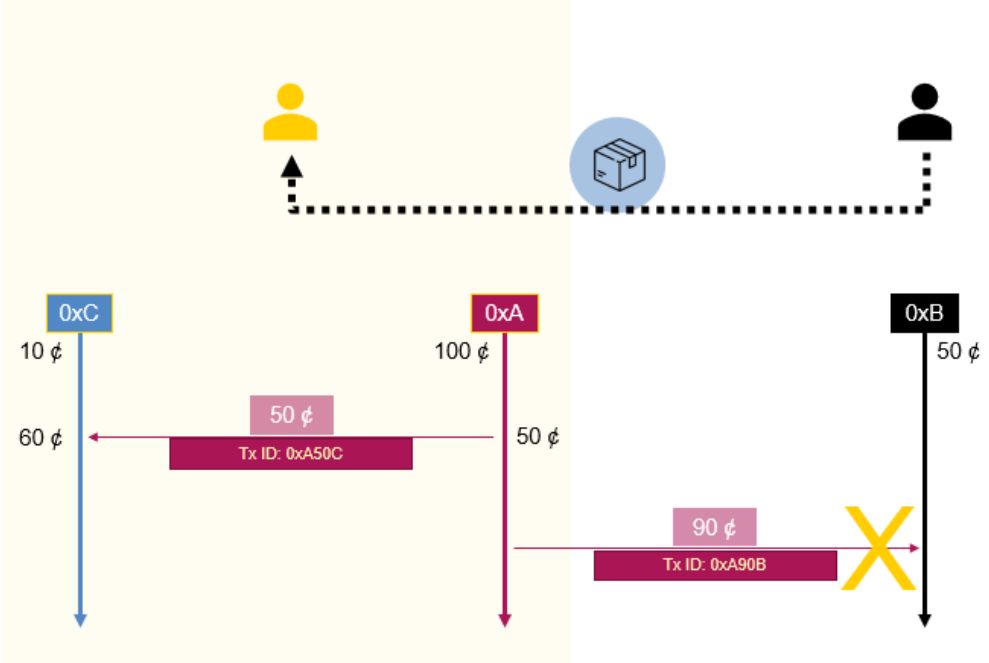
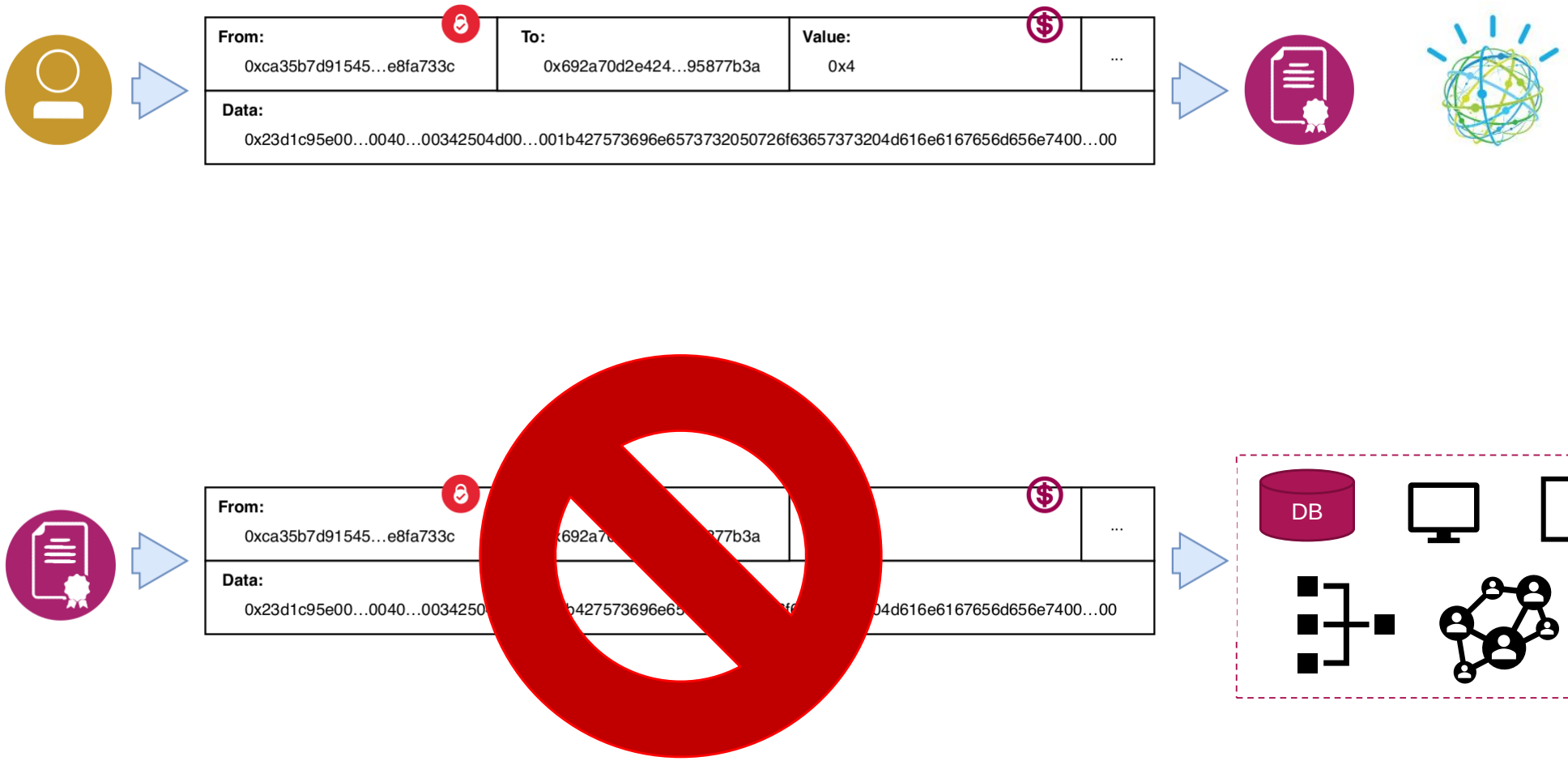


Image source: https://www.kbrillphotograp.top/products.aspx?cname=jlo+jenny+from+the+block+outfit&cid=140

The problem

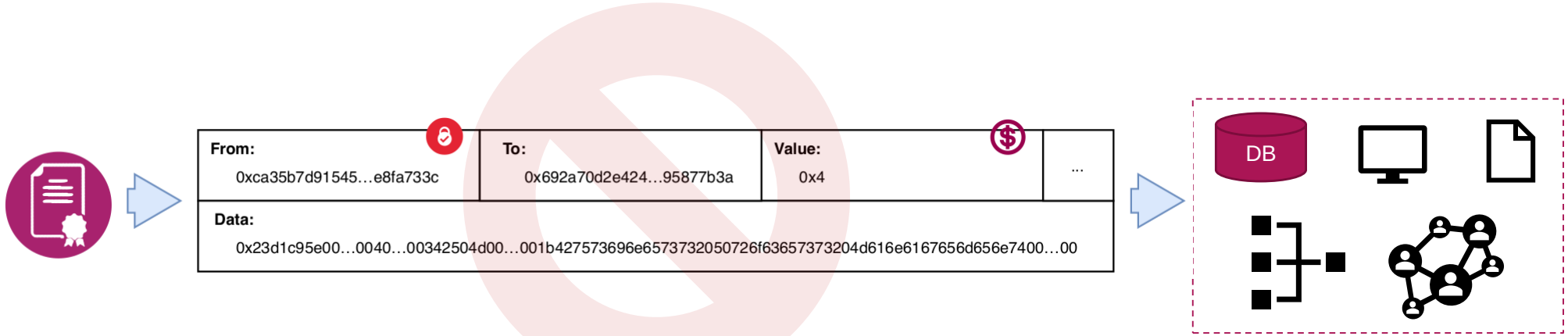

From:
0xca35b7d91545…e8fa733c
To:
0x692a70d2e424…95877b3a
Value:
0x4
...
Data:
0x23d1c95e00…0040…00342504d00…001b427573696e6573732050726f63657373204d616e6167656d656e7400…00
DB

How about the real world?

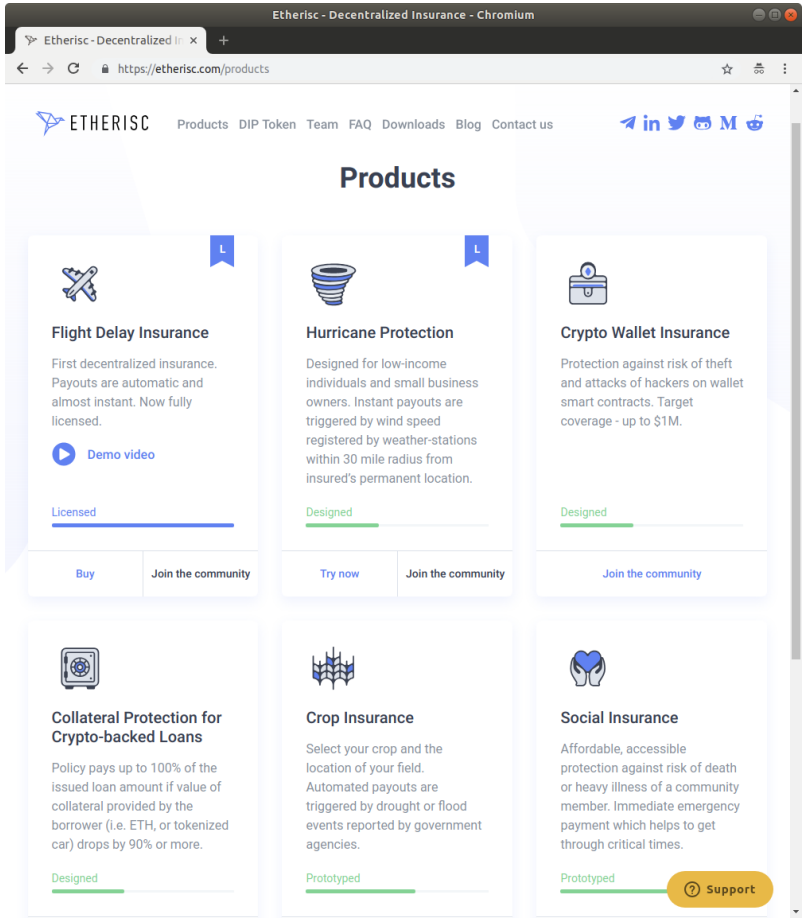
Oracles: From on-chain to off-chain and vice versa

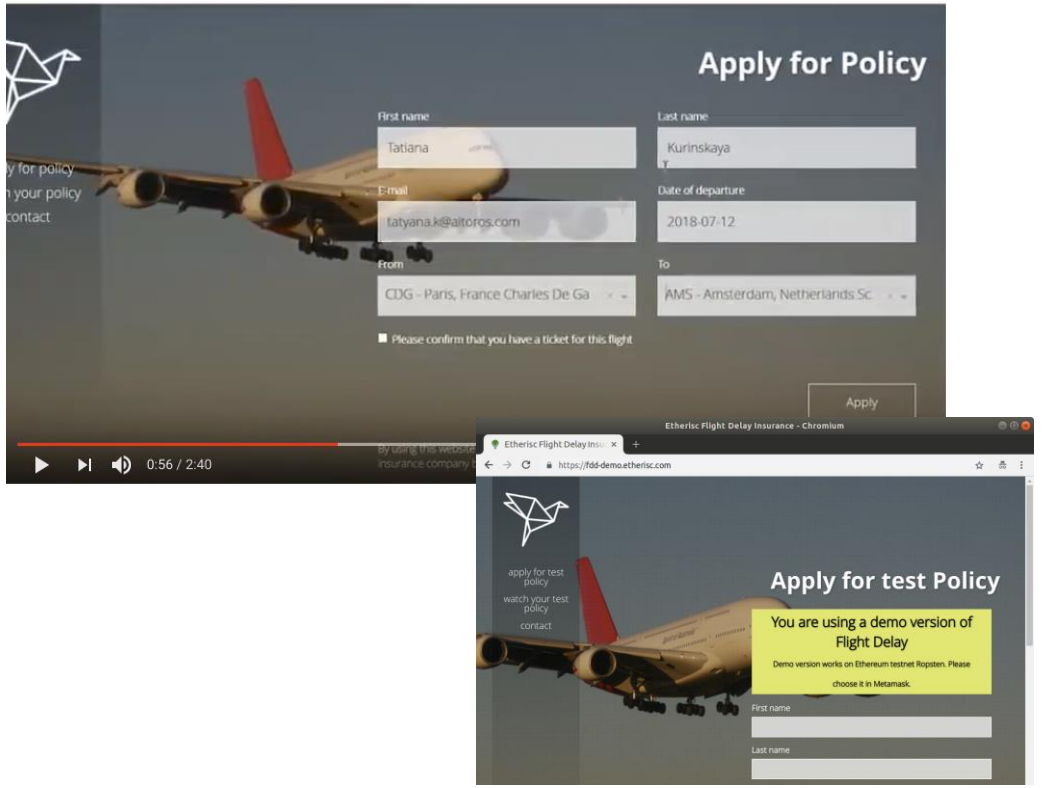
The Oracle

Source: http://matrix.wikia.com/wiki/File:The_Oracle_Making_Cookies.jpg



Etherisc


Etherisc - Decentralized Insurance - Chromium
https://etherisc.com/products
ETHERISC
Products DIP Token Team FAQ Downloads Blog Contact us
Products
Flight Delay Insurance
First decentralized insurance. Payouts are automatic and almost instant. Now fully licensed.
Demo video
Licensed
Buy
Join the community
Hurricane Protection
Designed for low-income individuals and small business owners. Instant payouts are triggered by wind speed registered by weather-stations within 30 mile radius from insured's permanent location.
Designed
Try now
Join the community
Crypto Wallet Insurance
Protection against risk of theft and attacks of hackers on wallet smart contracts. Target coverage - up to $1M.
Designed
Join the community
Collateral Protection for Crypto-backed Loans
Policy pays up to 100% of the issued loan amount if value of collateral provided by the borrower (i.e. ETH, or tokenized car) drops by 90% or more.
Designed
Crop Insurance
Select your crop and the location of your field. Automated payouts are triggered by drought or flood events reported by government agencies.
Prototyped
Social Insurance
Affordable, accessible protection against risk of death or heavy illness of a community member. Immediate emergency payment which helps to get through critical times.
Prototyped
Support


Apply for Policy
First name
Tatiana
Last name
Kurinskaya
E-mail
tatyana.k@altoros.com
Date of departure
2018-07-12
From
CDG - Paris, France Charles De Ga
To
AMS - Amsterdam, Netherlands Sc
Please confirm that you have a ticket for this flight
Apply
0:56 / 2:40
Etherisc Flight Delay Insurance - Chromium
https://fdd-demo.etherisc.com
Apply for test Policy
You are using a demo version of Flight Delay
Demo version works on Ethereum testnet Ropsten. Please choose it in Metamask.
First name
Last name

Flight delay insurance: the FlightDelayPayout contract

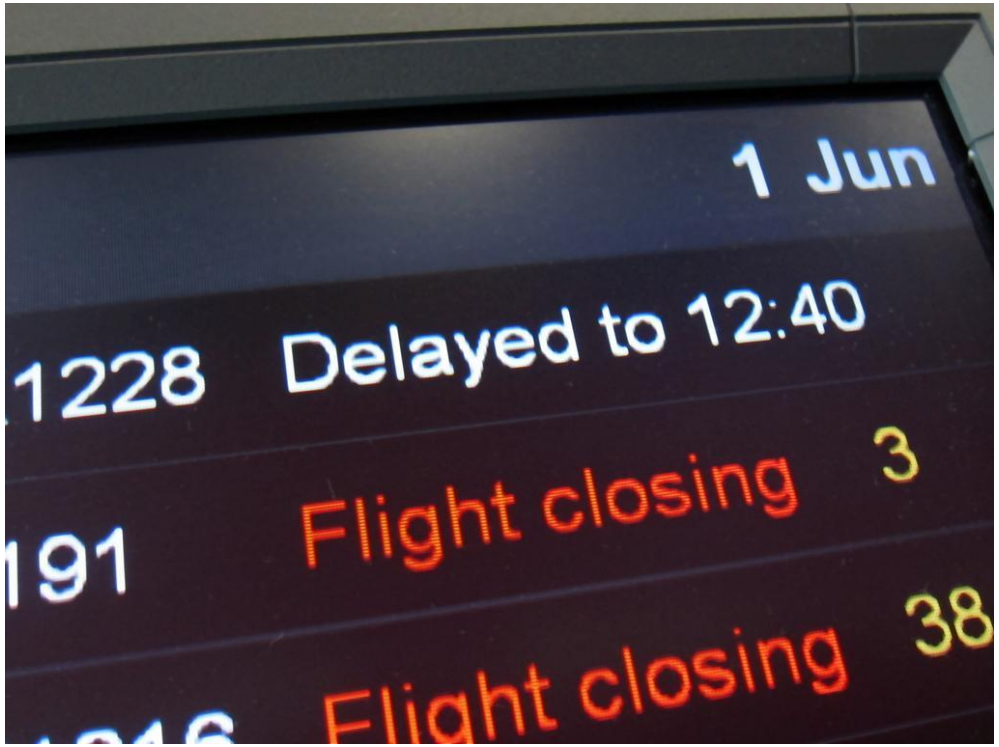
flightDelay/FlightDelayPayout.sol at master · etherisc/flightDelay · GitHub - Chromium

GitHub, Inc. [US] | https://github.com/etherisc/flightDelay/blob/master/contracts/FlightDelayPayout.sol

```
100     /*
101      * @dev Oraclize callback. In an emergency case, we can call this directly from FD.Emergency Account.
102      * @param _queryId
103      * @param _result
104      * @param _proof
105      */
106     function __callback(bytes32 _queryId, string _result, bytes _proof) public onlyOraclizeOr(getContract('FD.Emergency')) {
107
108         var (policyId, oraclizeTime) = FD_DB.getOraclizeCallback(_queryId);
109         LogOraclizeCallback(policyId, _queryId, _result, _proof);
110
111         // check if policy was declined after this callback was scheduled
112         var state = FD_DB.getPolicyState(policyId);
113         require(uint8(state) != 5);
114
115         bytes32 riskId = FD_DB.getRiskId(policyId);
116
117 // --> debug-mode
118 //          LogBytes32("riskId", riskId);
119 // <-- debug-mode
120
121         var slResult = _result.toSlice();
122
123         if (bytes(_result).length == 0) { // empty Result
124             if (FD_DB.checkTime(_queryId, riskId, 180 minutes)) {
125                 LogPolicyManualPayout(policyId, "No Callback at +120 min");
126                 return;
127             } else {
128                 schedulePayoutOraclizeCall(policyId, riskId, oraclizeTime + 45 minutes);
129             }
130         } else {
131             // first check status
132             // extract the status field:
133             slResult.find("\"".toSlice()).beyond("\"".toSlice());
134             slResult.until(slResult.copy().find("\"".toSlice()));
135             bytes1 status = bytes(slResult.toString())[0];  // s = L
136             if (status == "C") {
137                 // flight cancelled --> payout
138                 payOut(policyId, 4, 0);
139                 return;
140             } else if (status == "D") {
141                 // flight diverted --> payout
142                 payOut(policyId, 5, 0);
143                 return;
144             } else if (status != "L" && status != "A" && status != "C" && status != "D") {
145                 LogPolicyManualPayout(policyId, "Unprocessable status");
146                 return;
147             }
148
149             // process the rest of the response:
```

Contact with the off-chain world

Payout in case of signalled problems with the flight



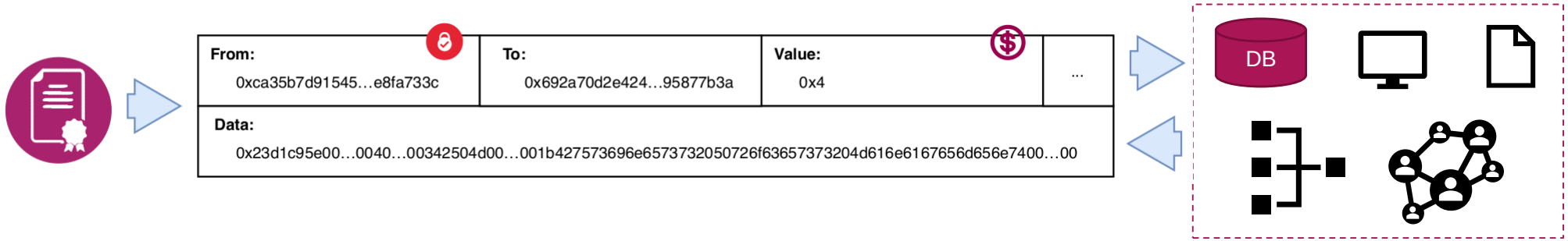
Source: https://www.flickr.com/photos/michaelduxbury/5824469025

The Oracle

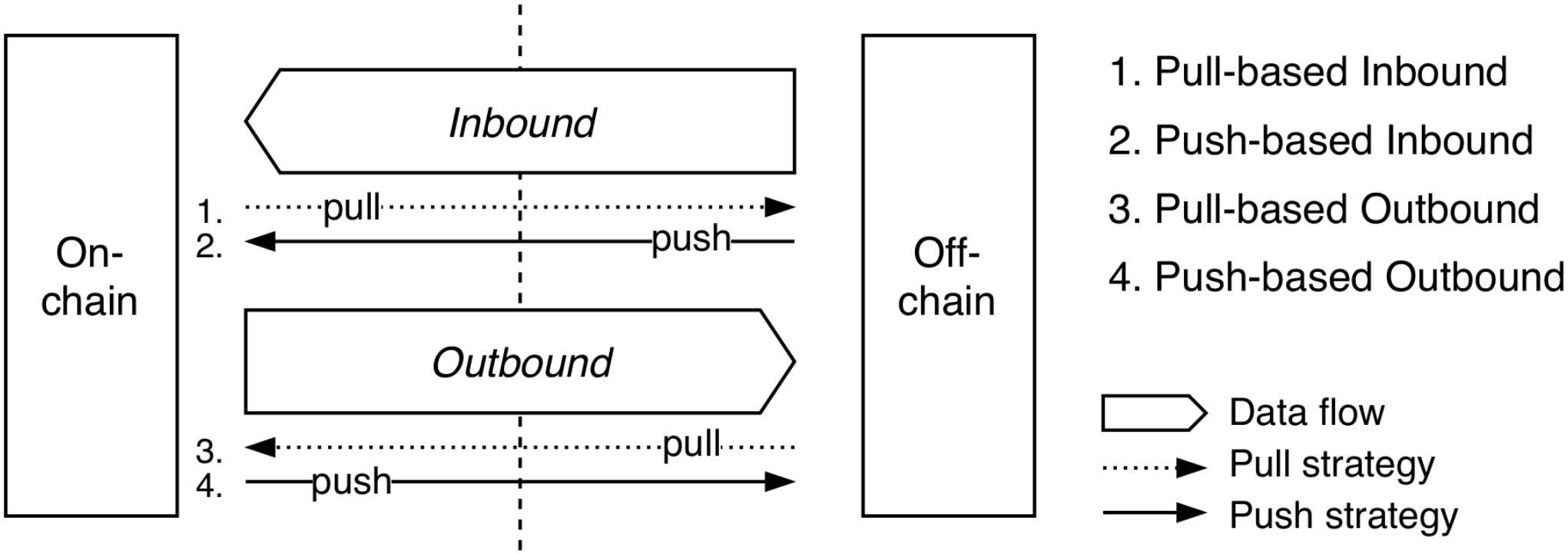
ISO/TC 307, ISO/TR 2345: "[A] **DLT Oracle** [is a] **service** that updates a distributed ledger using **data from outside** the distributed ledger system". (2019)

Previous literature: oracles as off-chain information providers.

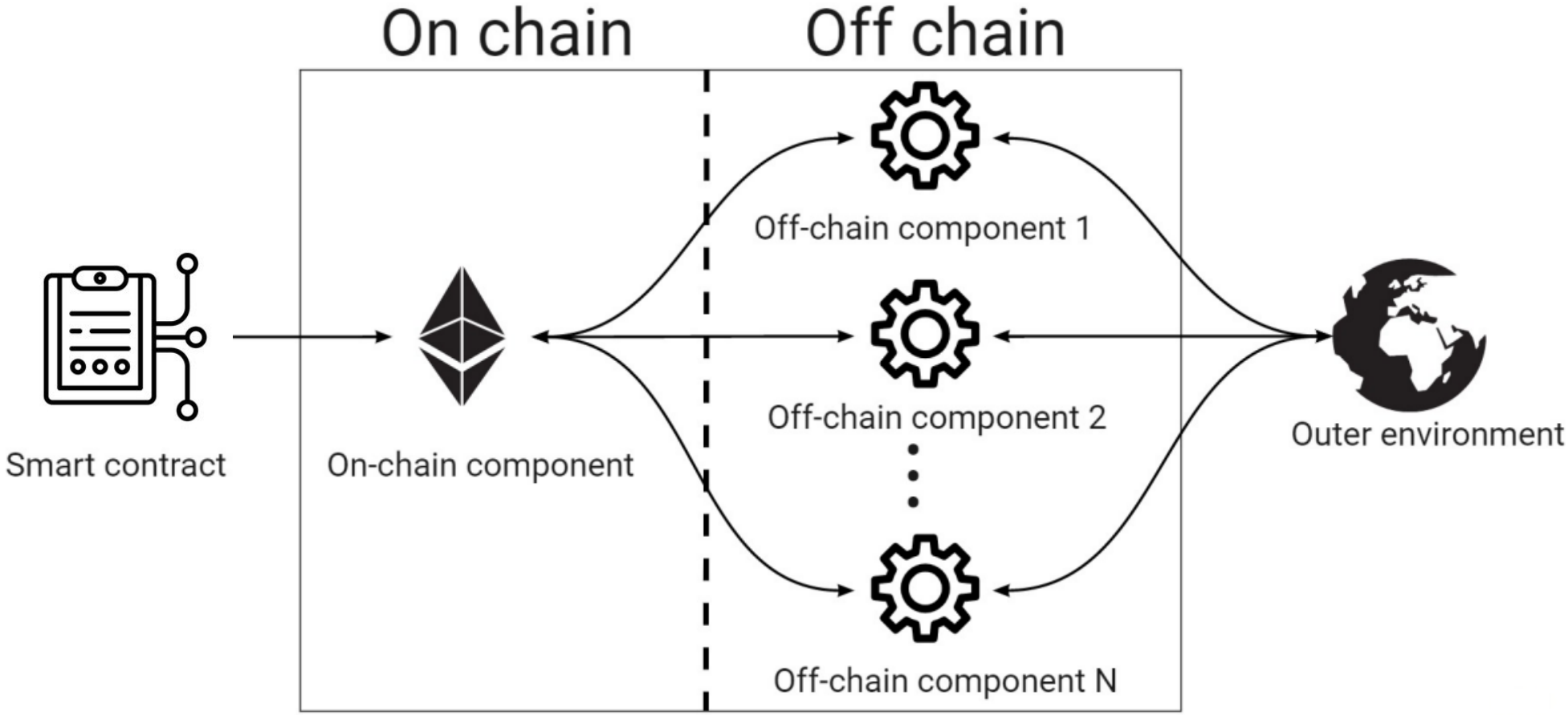
We see **oracles** as a **bridge**
between the on-chain and off-chain worlds.



Oracle patterns: Overview

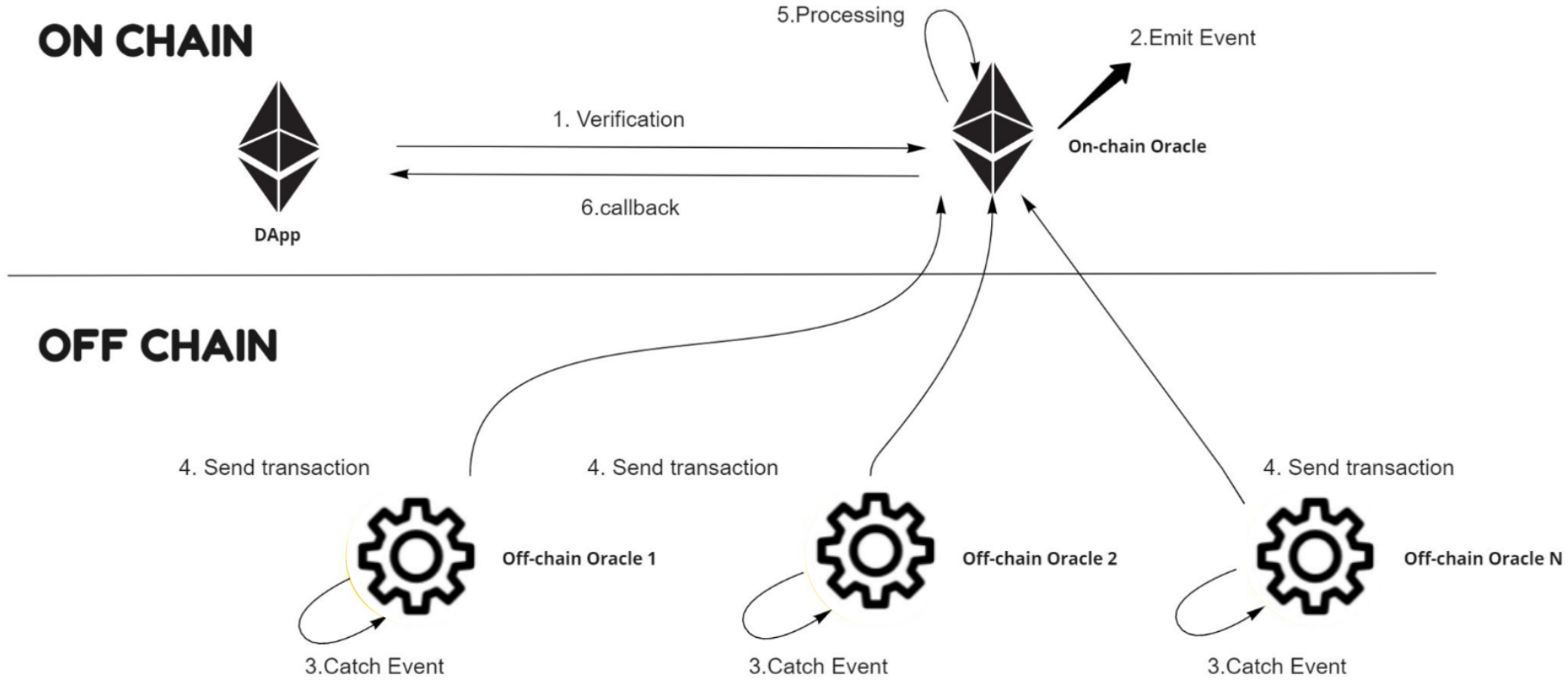


Decentralised oracles

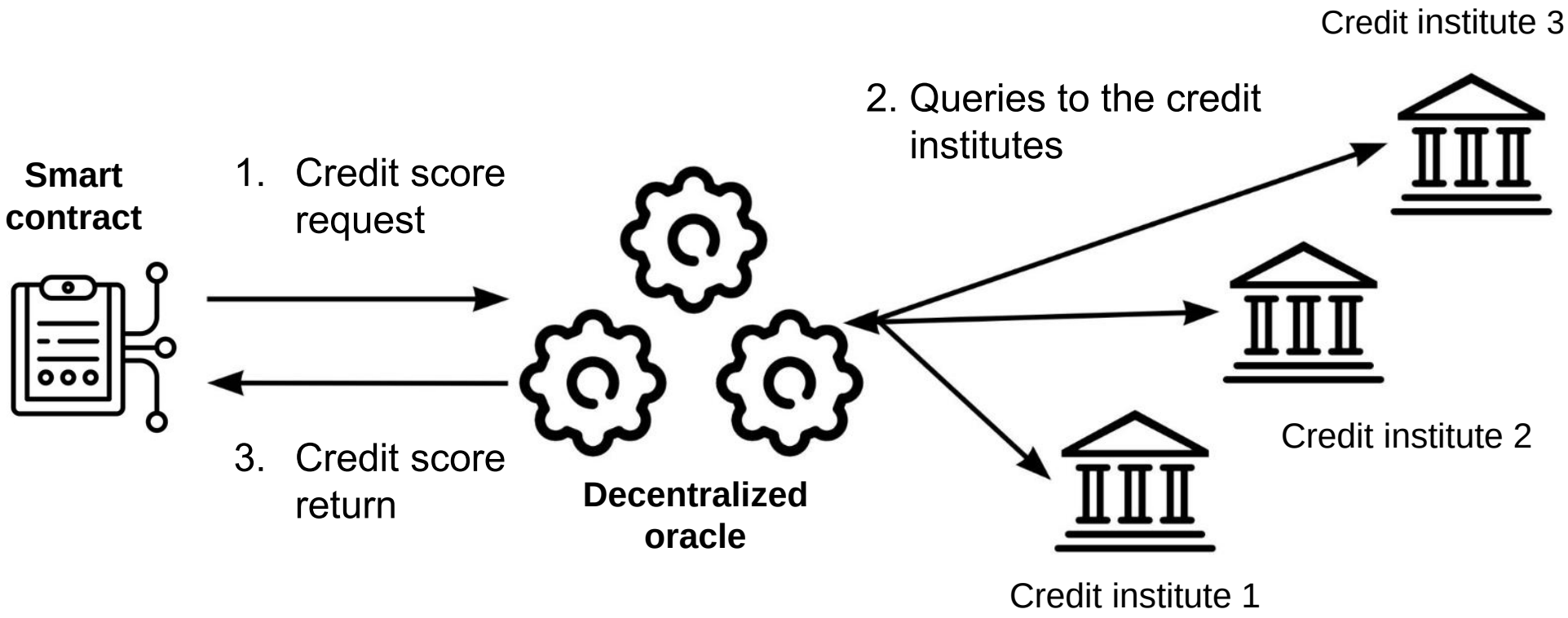


Enhancing Blockchain-Based Processes with Decentralized Oracles. In: BPM Blockchain and RPA Forum 2021, 102-118, Springer.

Decentralised oracles (example: pull-in)



Decentralised oracles (example: pull-in)



About privity

- Albeit **very costly**, we could inject all the information we need on-chain.
- Even if we were able to inject all the data in the world on chain, would we like the idea?
- "**Privity** strives for limiting the sharing of information within a contract to those parties of a contract who have a contractual need to know"

Home › Business Process Management: Blockchain and Central and Eastern Europe Forum › Conference paper

Balancing Privity and Enforceability of BPM-Based Smart Contracts on Blockchains

Julius Köpke, Marco Franceschetti & Johann Eder

Conference paper | First Online: 26 August 2019

3382 Accesses | 4 Citations

Part of the Lecture Notes in Business Information Processing book series (LNBIP,volume 361)

Abstract

Blockchains are a promising enabling technology for inter-organizational processes in untrusted environments and for the implementation of smart contracts in general. Smart contracts aim at three major objectives: observability, online enforceability and privity. Privity strives for limiting the sharing of information within a contract to those parties of a contract who have a contractual need to know. However, current BPM-based systems operating on blockchains do not address privity. The approaches deal with enforceability and privity as mutual exclusive properties. We show that the trade-offs between privity and enforceability can be considered in fine details and propose means to balance privity and enforceability in the design of smart contracts according to the application requirements. Besides this conceptual basis, we introduce patterns for encryption and key exchange allowing different levels of privity and for supporting proactive online enforceability in the presence of encrypted on-chain data.

Keywords

Inter-organizational business processes | Blockchain | Smart contracts | Privity | Confidentiality

Köpke, Franceschetti, Eder: Balancing Privity and Enforceability of BPM-Based Smart Contracts on Blockchains. BPM (Blockchain and CEE Forum) 2019: 87-102

While collaborators cooperate…

Signing of the Peace Protocol Between Spain and the United States, August 12, 1898
(source: https://library.whitehousehistory.org/fotoweb/archives/5017-Digital-Library/Main%20Index/Artwork/520.tif.info)

… the whole network observes

What about confidentiality?

Scene from The Truman Show (1998). Image source: https://redeemingculture.com/film/drama/4106-the-truman-show-empathy-and-reality

Ledger and secrecy

Every participant in the blockchain network can read the data on the ledger unless the platform is private and permissioned

Public permissionless platforms are more robust and guarantee non-repudiability

Image source: https://www.kbrillphotograp.top/products.aspx?cname=jlo+jenny+from+the+block+outfit&cid=140

Transaction information hiding via homomorphic encryption

Regulation-Friendly Privacy-Preserving Blockchain Based on zk-SNARK

Lei Xu(✉), Yuewei Zhang(✉), and Liehuang Zhu

School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing, China
{xu.lei,yueweizhang,liehuangz}@bit.edu.cn

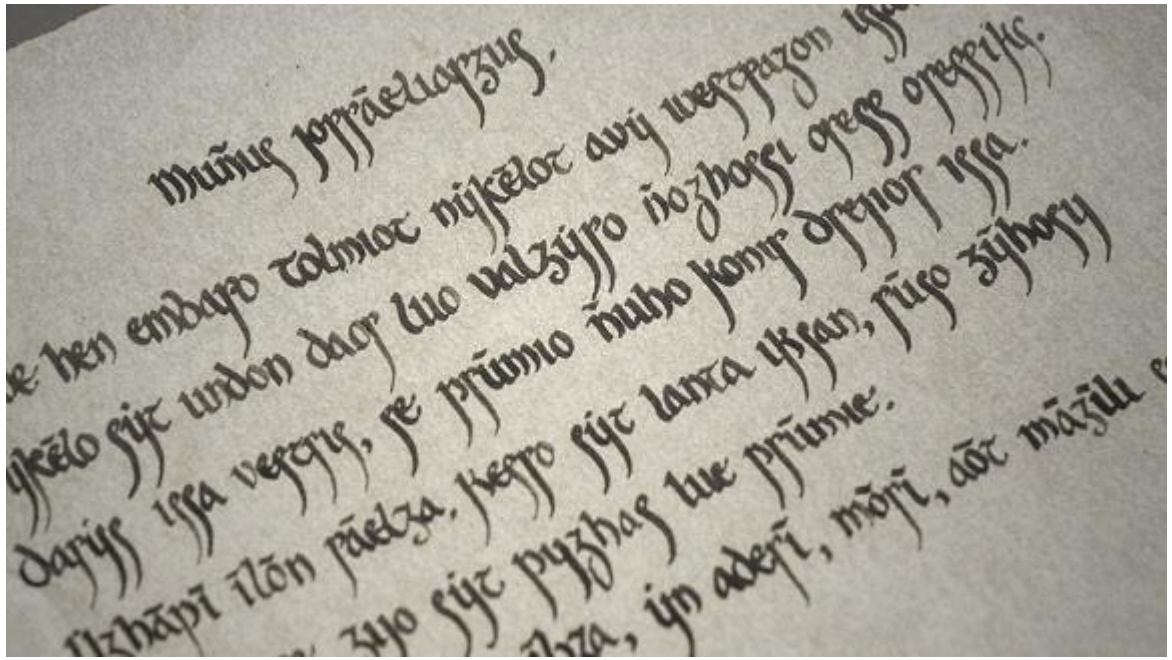
Abstract. Recently, blockchain has attracted much attention from industries, due to its good characteristics such as decentralization and tamper proofing. To ensure that sensitive transaction data are not disclosed to the public, many privacy protection methods have been proposed for blockchain, which generally conflicts with regulatory requirements. To resolve such a conflict, in this paper we propose a privacy-preserving account-based blockchain system which supports auditing on transactions. The proposed system protects the privacy of a transaction via homomorphic encryption. The validity of the transaction is guaranteed via zero-knowledge proof. Especially, details of a transaction are presented in form of ciphertexts on the public ledger, which can be decrypted by regulatory authorities. We have implemented a demo of the proposed system using the Substrate framework. Simulation results show that the system has acceptable performance.

Keywords: Blockchain · Privacy Preserving · Regulation Compliance · Homomorphic Encryption · Zk-SNARK

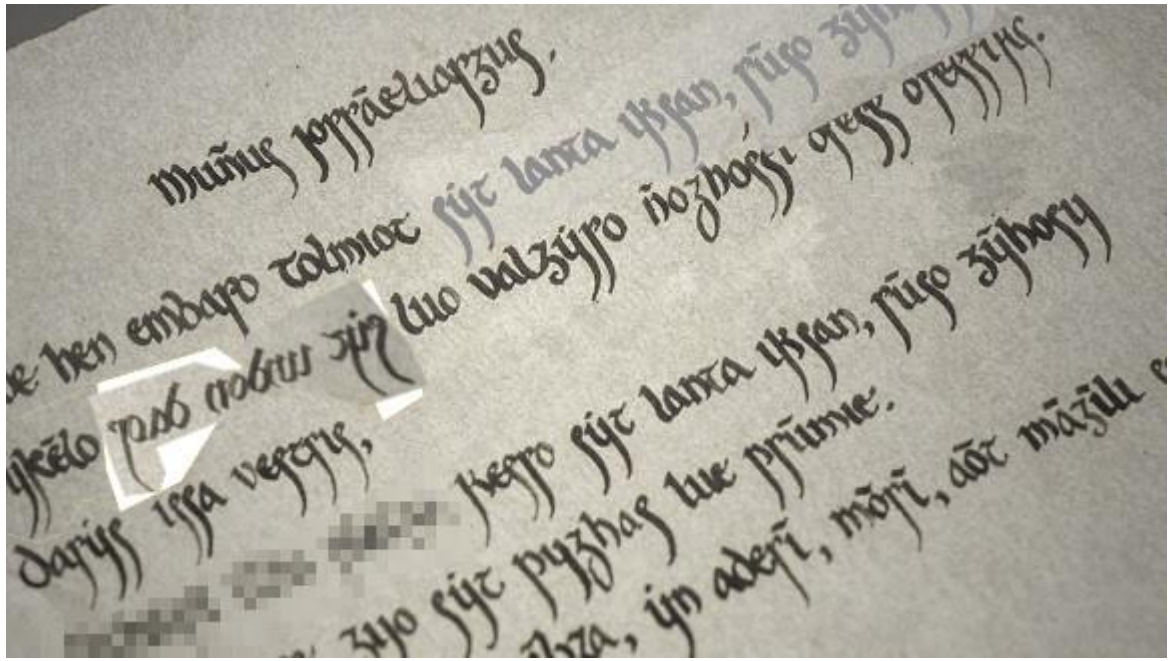
Coming next: *High-Performance Confidentially-Preserving Blockchain via GPU-Accelerated Fully Homomorphic Encryption* (Guan, Qi, Shen, Wang, Zhang and Cui)

Xu, Zhang, Zhu (2023). Regulation-Friendly Privacy-Preserving Blockchain Based on zk-SNARK. In: Advanced Information Systems Engineering Workshops. CAiSE 2023

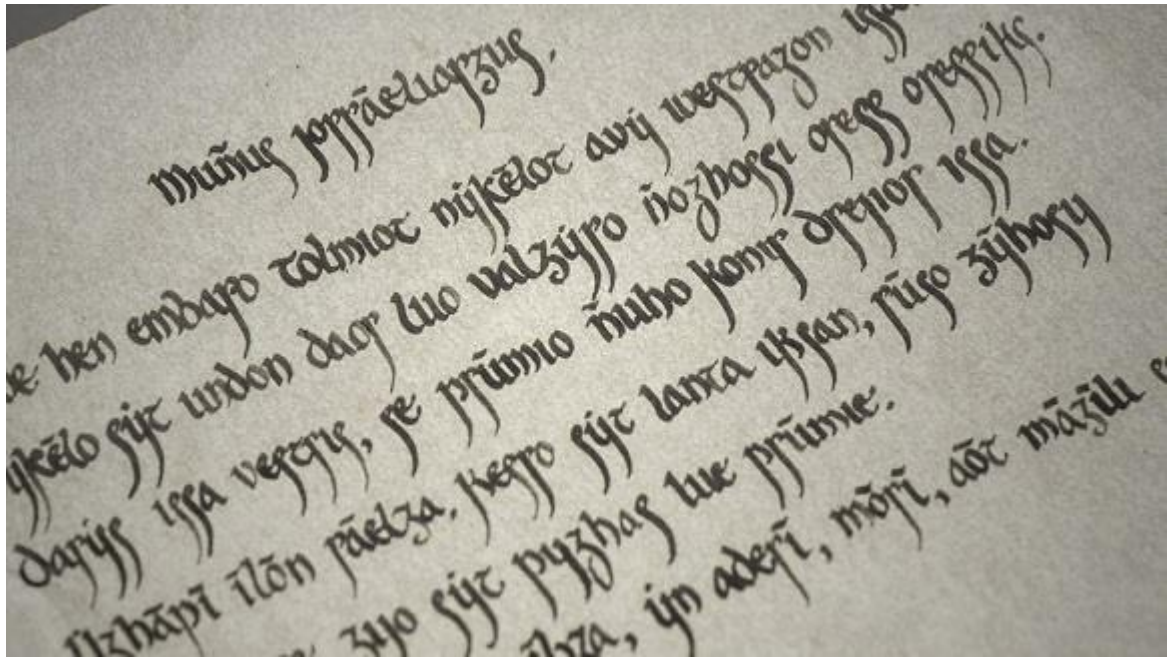
Fix this image in your memory



Is this the same image?



Validation v. understanding



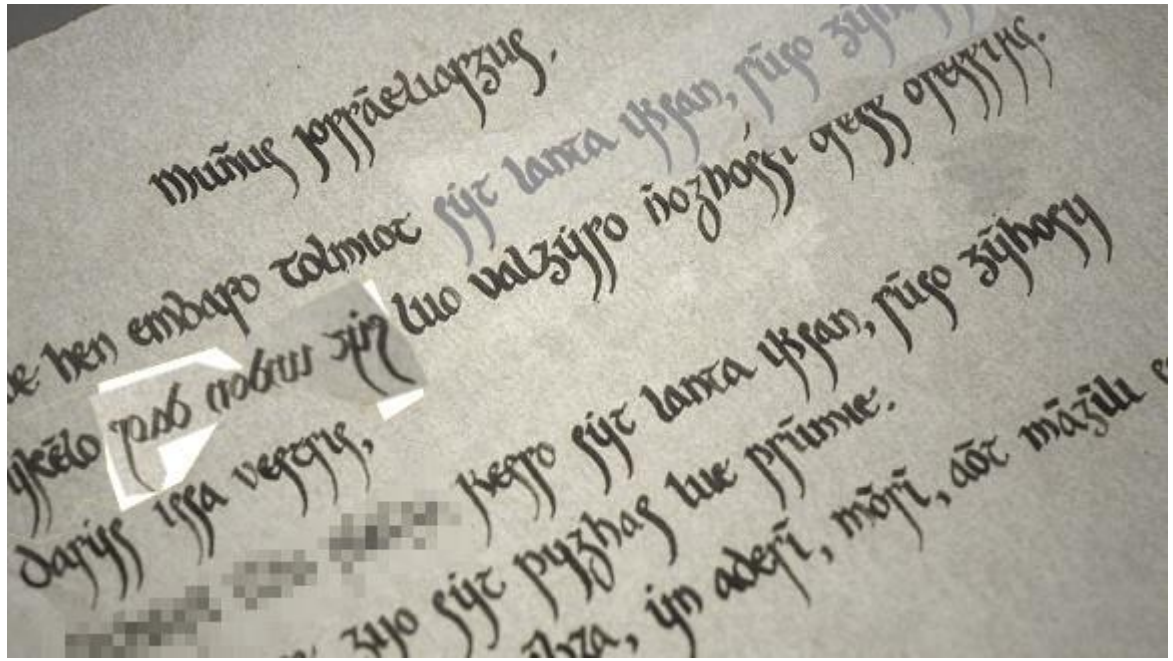
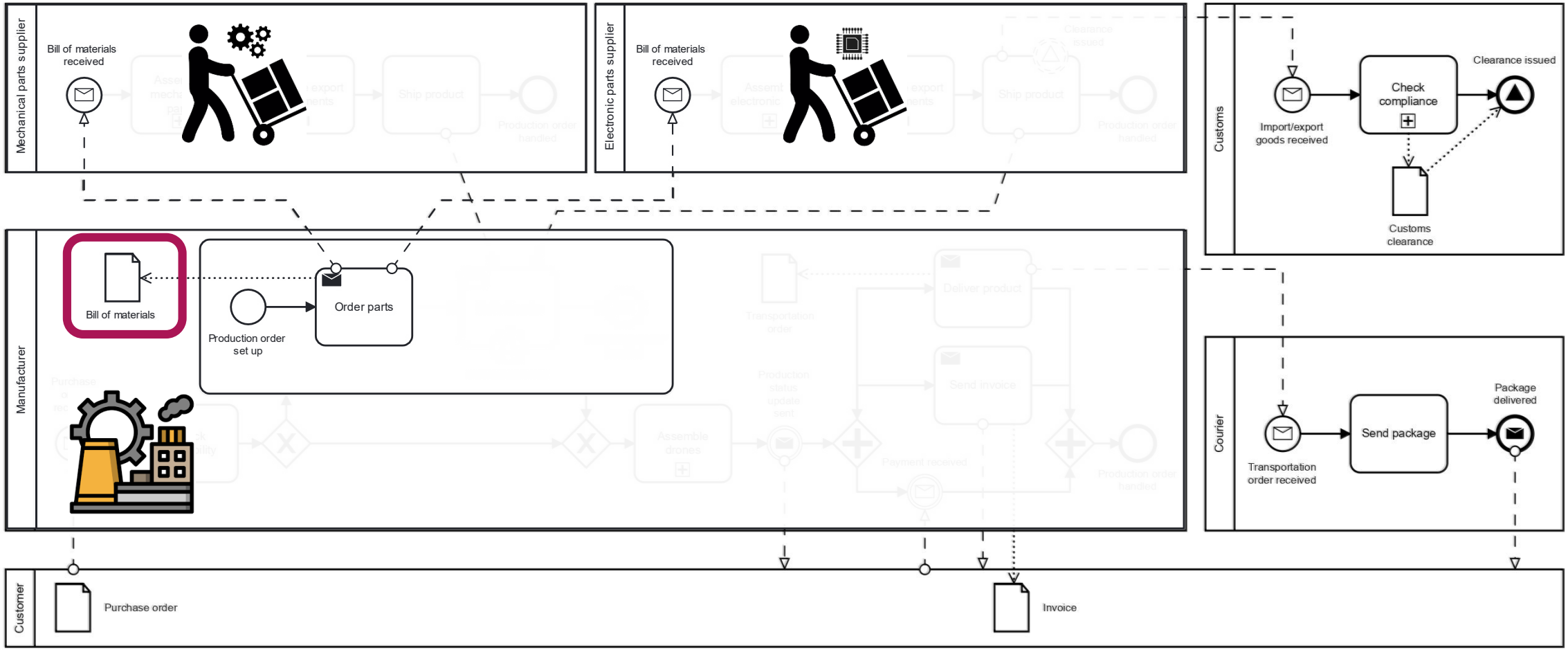


Image source: https://www.makinggameofthrones.com/production-diary/2013/5/29/leeches-dragons-and-a-bear-behind-the-scenes-of-episodes-307.html

Business Process Model and Notation (BPMN) collaboration diagram



The message, in clear (as seen by the manufacturer)

Original data	
Manufacturer_company:	Beta
Address:	82, Beta street
E-mail:	mnfctr.beta@mail.com
Frames_quantity:	8
Propeller_quantity:	80
PropellerGuard_quantity:	63
Camera_quantity:	30
Controller_quantity:	4
Amount_paid:	$12000
IMU_quantity:	6
ESC_quantity:	40
Engines_quantity:	9
Batteries_quantity:	25
Amount_paid:	$9850

Hash: 0xfd9ffe3578a42d81c4684bd47e6575d9a8f10cc6dab984bd7e19ee0dbacb6287

The message (as seen by external parties)

Hash: 0xfd9ffe3578a42d81c4684bd47e6575d9a8f10cc6dab984bd7e19ee0dbacb6287

The message (as seen by the electronic parts supplier)

Original data

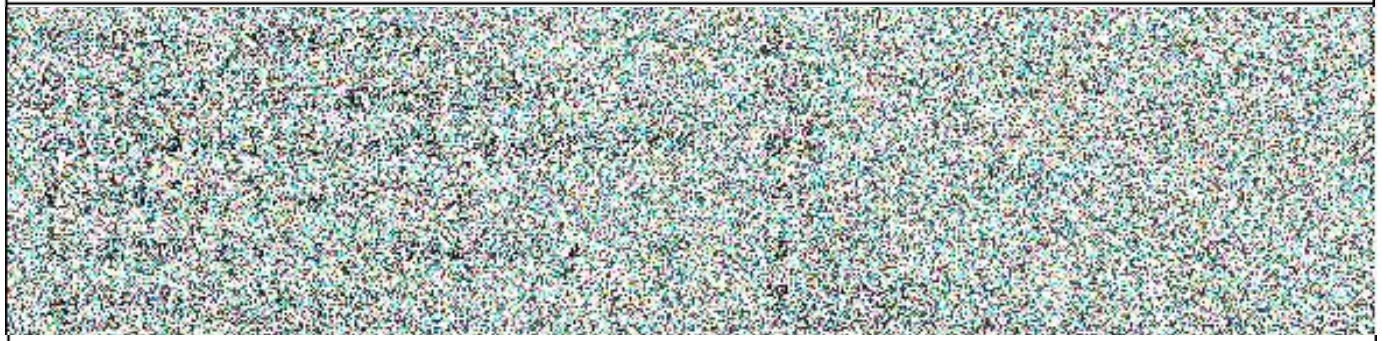
Manufacturer_company:	Beta
Address:	82, Beta street
E-mail:	mnfctr.beta@mail.com
Frames_quantity:	8
Propeller_quantity:	80
PropellerGuard_quantity:	63
Camera_quantity:	30
Controller_quantity:	4
Amount_paid:	$12000

Hash: 0xfd9ffe3578a42d81c4684bd47e6575d9a8f10cc6dab984bd7e19ee0dbacb6287

The message (as seen by the mechanical parts supplier)

Original data

Manufacturer_company:	Beta
Address:	82, Beta street
E-mail:	mnfctr.beta@mail.com



IMU_quantity:	6
ESC_quantity:	40
Engines_quantity:	9
Batteries_quantity:	25
Amount_paid:	$9850

Hash: 0xfd9ffe3578a42d81c4684bd47e6575d9a8f10cc6dab984bd7e19ee0dbacb6287

Hence the name: CAKE

Thursday, June 6, 14:00: *CAKE: Sharing Slices of Confidential Data on Blockchain* (Maragone, Spina, D.C., Weber)

Image source: https://heriots.co.uk/products/pistachio

Ingredients

- Blockchain platform
- Smart contracts
- InterPlanetary File System (IPFS)
- Ciphertext-Policy (CP) Attribute-Based Encryption (ABE)

CP-ABE

- Attribute-Based Encryption (ABE): type of public-key encryption
- Ciphertext-Policy ABE (CP):
  - We associate roles and process instance ID with attributes
    - (propositional literals)
  - Messages are associated with policies
    - (propositional formulae on attributes)

- Attributes:
  `14548487`, `Supplier`, `Electronics`, `Electronics`, `Manufacturer`

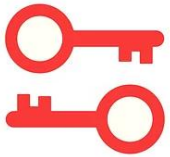
  Process instance ID — Roles

- Policy:
  `14548487 AND (Manufacturer OR (Supplier AND Electronics))`

CAKE

Given

Key pair



Public key (pk)

Master public key (mpk)

0: pre-phase

Attribute certification

1: ciphering

Plaintext

Policy

pk

Ciphertext

2: key generation

Attributes

Key pair

pk

mpk

Secret key

3: deciphering

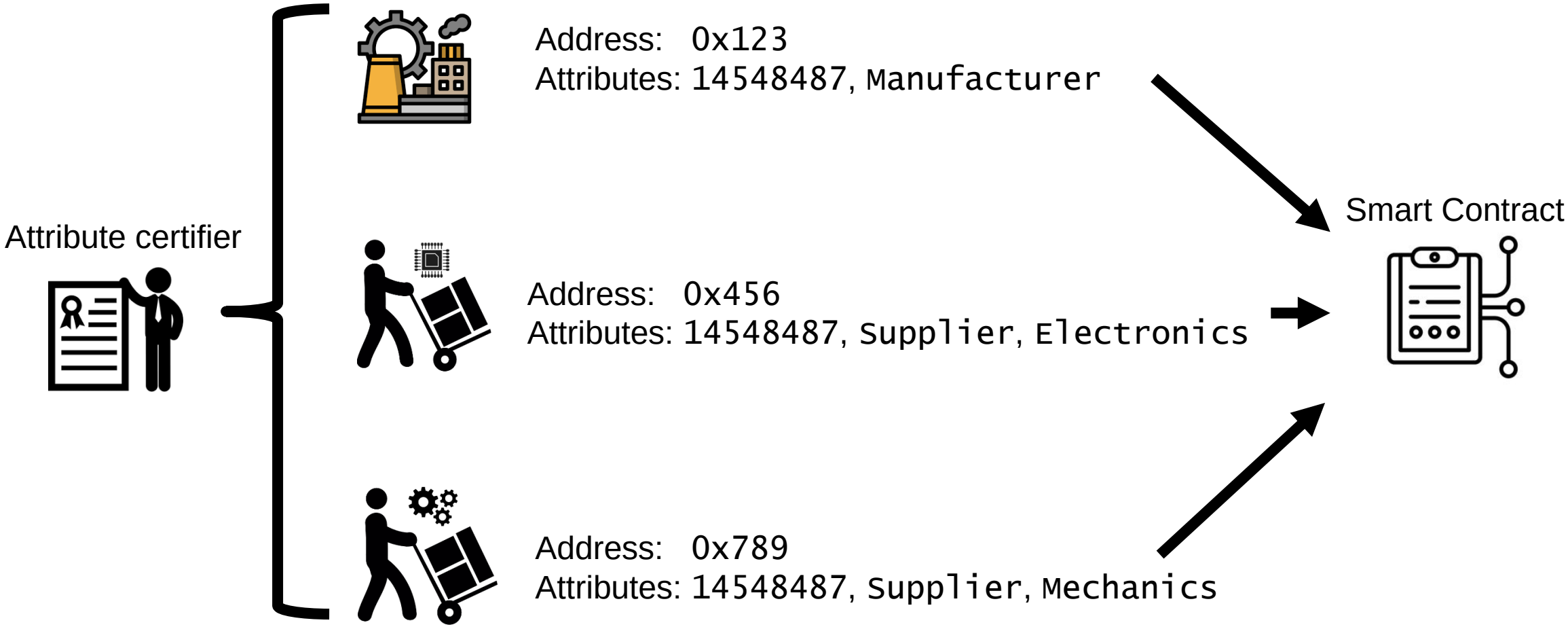
Ciphertext

Secret key

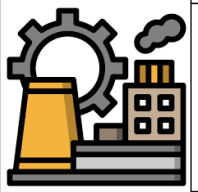
pk

Plaintext

Phase 0: certification

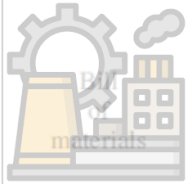


Phase 1: ciphering

Message	Original data
	Manufacturer_company: Beta Address: 82, Beta street E-mail: mnfctr.beta@mail.com
	Frames_quantity: 8 Propeller_quantity: 80 PropellerGuard_quantity: 63 Camera_quantity: 30 Controller_quantity: 4 Amount_paid: $12000
	IMU_quantity: 6 ESC_quantity: 40 Engines_quantity: 9 Batteries_quantity: 25 Amount_paid: $9850

File header	File body (slices)
14548487 and (Manufacturer or (Supplier))	slice_id: 7816105805828306901, metadata: {"c1": [...] 00a0}, cipherText: "oT2W [...] MQ=="
sender: 0x906D [...] Dba8, 14548487 and (Manufacturer or (Supplier and Electronics)) mk: {"beta": "\\u00b2 [...] 00fb}	slice_id: 6847895862959863592, metadata: {"c1": [...] asq2}, cipherText: "AS2w [...] btwd"
14548487 and (Manufacturer or (Supplier and Mechanics))	slice_id: 3147899764966459866, cipherText: "ht3r [...] asf3"

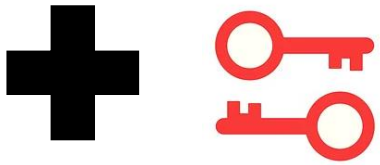
Phase 1: ciphering

Message	Original data	File header	File body (slices)
	Manufacturer_company: Beta Address: 82, Beta street E-mail: mnfctr.beta@mail.com Frames_quantity: 8 Propeller_quantity: 80 PropellerGuard_quantity: 63 Camera_quantity: 30 Controller_quantity: 4 Amount_paid: $12000 IMU_quantity: 6 ESC_quantity: 40 Engines_quantity: 9 Batteries_quantity: 25 Amount_paid: $9850	sender: 0x906D [...] Dba8, message_id: 17071949511205323542, pk: {"g": "\\u0087 [...] 00ca}, mk: {"beta": "\\u00b2 [...] 00fb}	slice_id: 7816105805828306901, hash: 0x953a [...] f8d8, salt: "Zu00 [...] u004", metadata: {"c1": [...] 00a0}, cipherText: "oT2W [...] MQ==" slice_id: 6847895862959863592, hash: 0x12es [...] 1g23, salt: "bw32 [...] b464", metadata: {"c1": [...] asq2}, cipherText: "AS2w [...] btwd" slice_id: 3147899764966459866, hash: 0xj4rs [...] ne3d, salt: "ns1w [...] mey4", metadata: {"c1": [...] 23rs}, cipherText: "ht3r [...] asf3"

Phase 2: key generation

14548487, Manufacturer

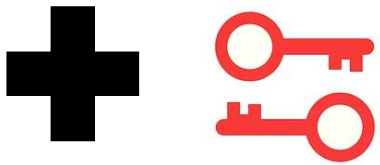
Key pair



D: 2nN6…GCcZ
Dj: 4558…5+Qg
Djp: 8944….5949

14548487, Supplier, Electronics

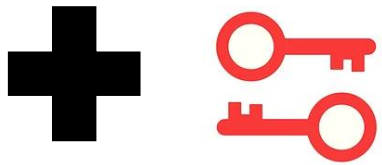
Key pair



D: 1+8Ka…kaUd
Dj: feoH…7393
Djp: bJju…. NIGw

14548487, Supplier, Mechanics

Key pair



D: A9BS…CnoO
Dj: OQEL…1207
Djp: hI2M…. lWBb

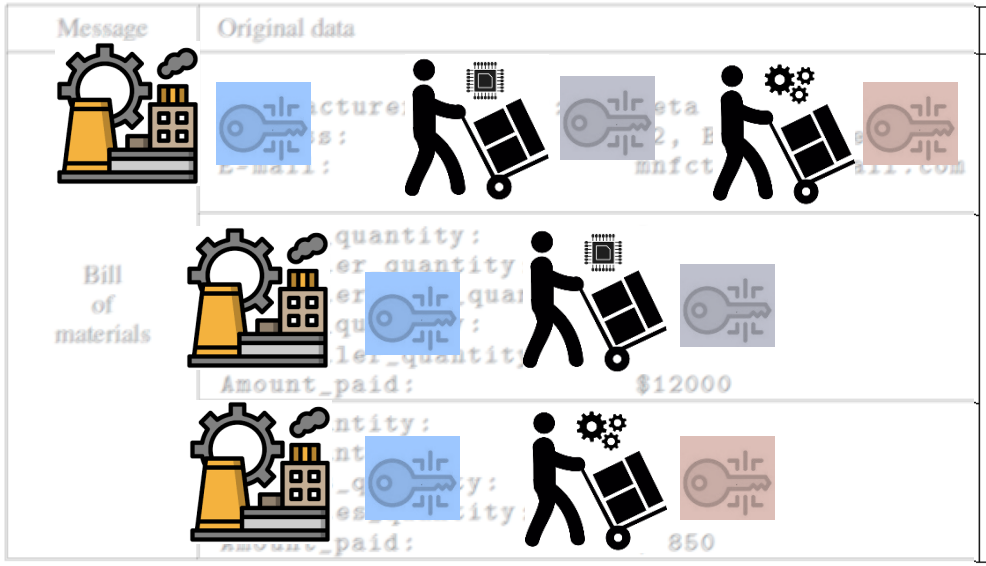
Message policy example

Message	Slice	Policy
Bill of materials	1	14548487 and (Manufacturer or (Supplier))
	2	14548487 and (Manufacturer or (Supplier and Electronics))
	3	14548487 and (Manufacturer or (Supplier and Mechanics))

Process instance (case id)

Attributes

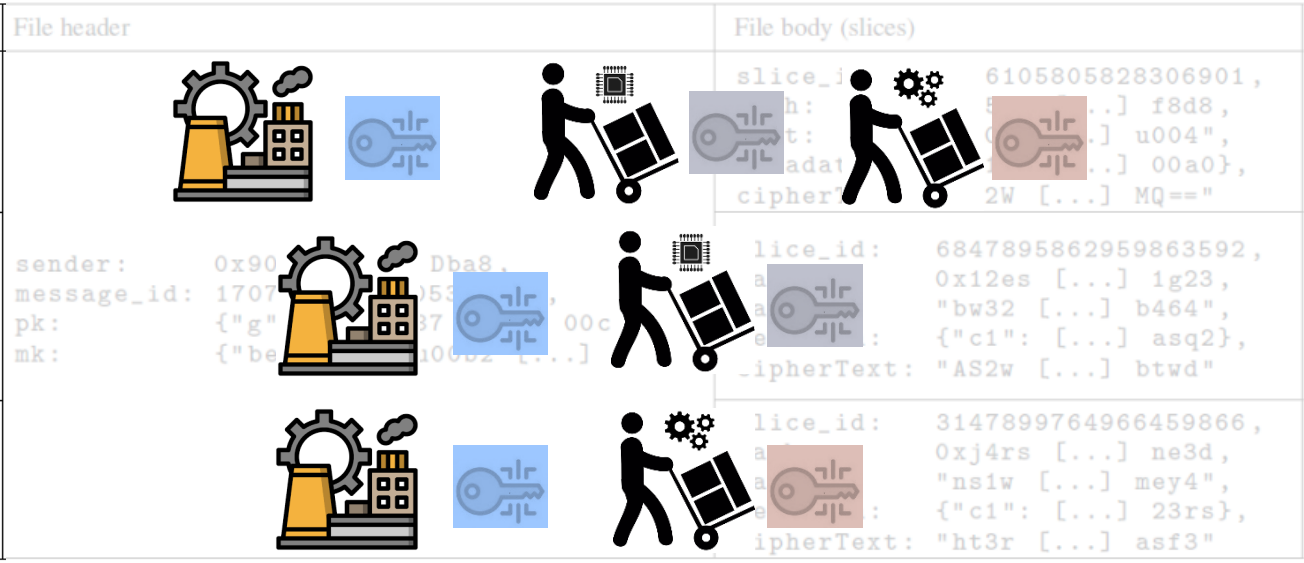
Phase 3: deciphering



File header	File body (slices)
sender: 0x906D [...] Dba8, message_id: 17071949511205323542, pk: {"g": "\\u0087 [...] 00ca}, mk: {"beta": "\\u00b2 [...] 00fb}	slice_id: 7816105805828306901, hash: 0x953a [...] f8d8, salt: "Zu00 [...] u004", metadata: {"c1": [...] 00a0}, cipherText: "oT2W [...] MQ=="
	slice_id: 6847895862959863592, hash: 0x12es [...] 1g23, salt: "bw32 [...] b464", metadata: {"c1": [...] asq2}, cipherText: "AS2w [...] btwd"
	slice_id: 3147899764966459866, hash: 0xj4rs [...] ne3d, salt: "ns1w [...] mey4", metadata: {"c1": [...] 23rs}, cipherText: "ht3r [...] asf3"

Phase 3: deciphering

Message	Original data	
Bill of materials	Manufacturer_company:	Beta
	Address:	82, Beta street
	E-mail:	mnfctr.beta@mail.com
	Frames_quantity:	8
	Propeller_quantity:	80
	PropellerGuard_quantity:	63
	Camera_quantity:	30
	Controller_quantity:	4
	Amount_paid:	$12000
	IMU_quantity:	6
	ESC_quantity:	40
	Engines_quantity:	9
	Batteries_quantity:	25
	Amount_paid:	$9850

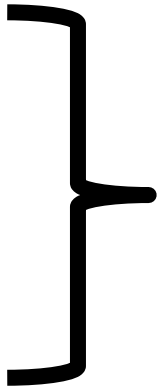


Focus

14548487, Supplier, Mechanics




```
Frames_quantity:          8
Propeller_quantity:       80
PropellerGuard_quantity:  63
Camera_quantity:          30
Controller_quantity:      4
Amount_paid:              $12000
```




14548487 and (Manufacturer or (Supplier and Electronics))





Why?

14548487 and (Manufacturer or (Supplier and Electronics))

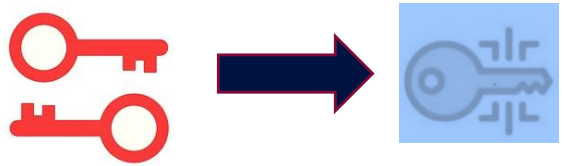





Q&A

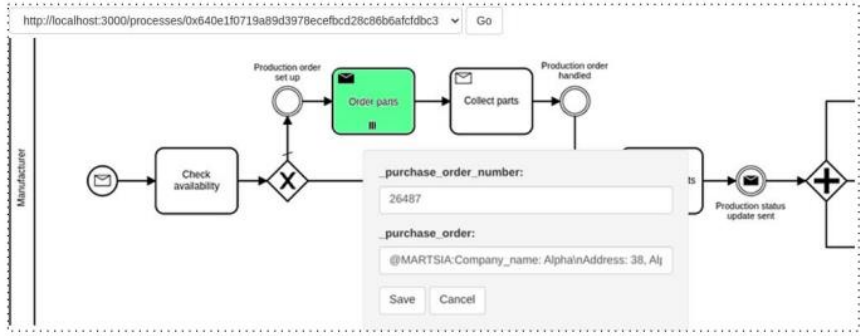
- Why a **certifier**?
  - With **signatures**, you can prove that "**you are you**"
  - Without a **certifier**, you cannot prove that what you say is **true**

- Who forges the **keys**?
  - A delegated key **manager**



- One certifier, one key manager. What about **decentralisation**?
  - Right…

- Can you integrate your technique with a **BPMS**?
  - Not yet but…



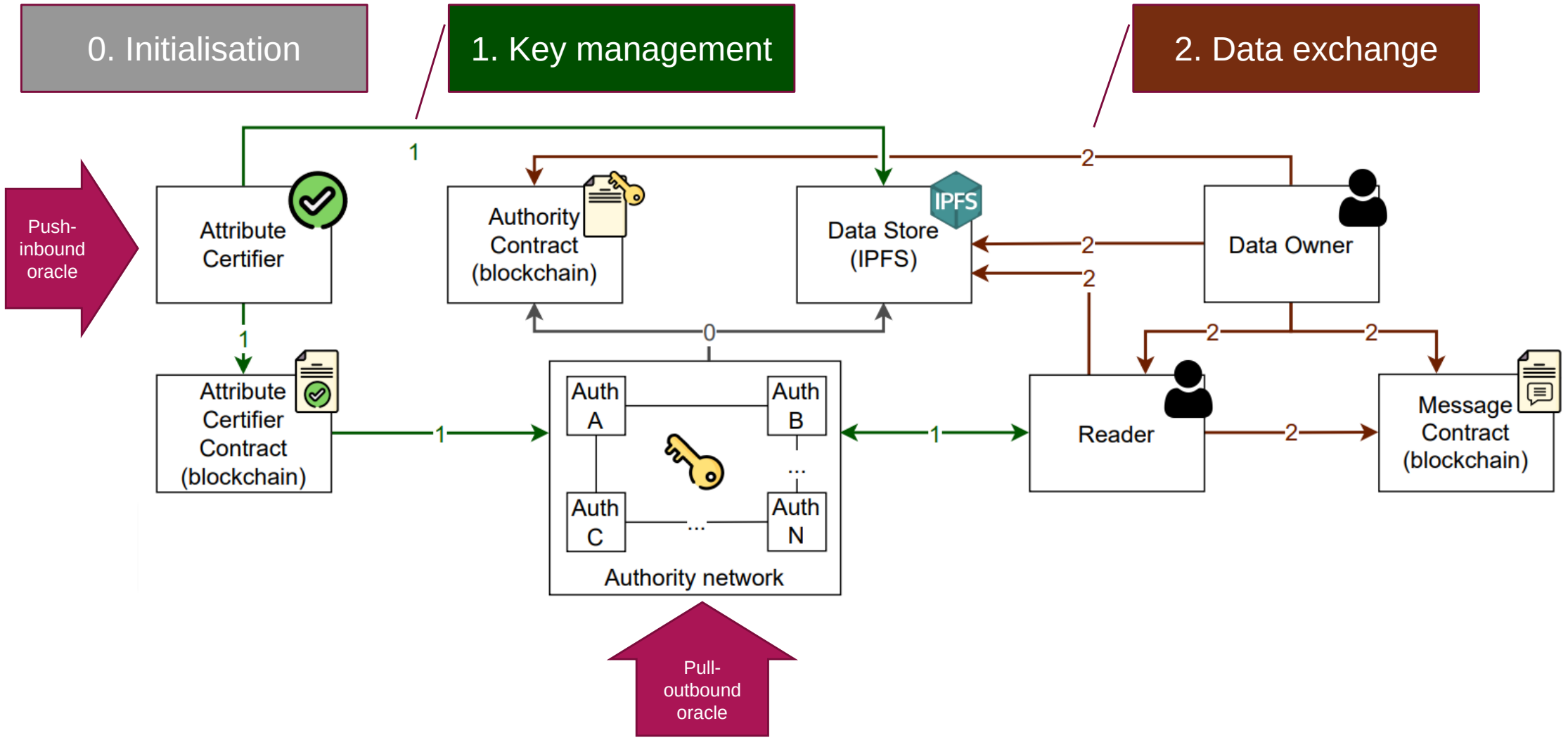
Room for improvement

Multi-Authority Approach to Transaction Systems for Interoperating Applications

	Requirement	CAKE [24]	MARTSIA
R1	Access to parts of messages should be controllable in a fine-grained way (attribute level), while integrity is ensured	✓	✓
R2	Information artifacts should be written in a permanent, tamper-proof and non-repudiable way	✓	✓
R3	The system should be independently auditable with low overhead	✓	✓
R4	The decryption key should only be known to the user who requested it	×	✓
R5	The decryption key should not be generated by a single trusted entity	×	✓
R6	The approach should integrate with control-flow management systems	×	✓

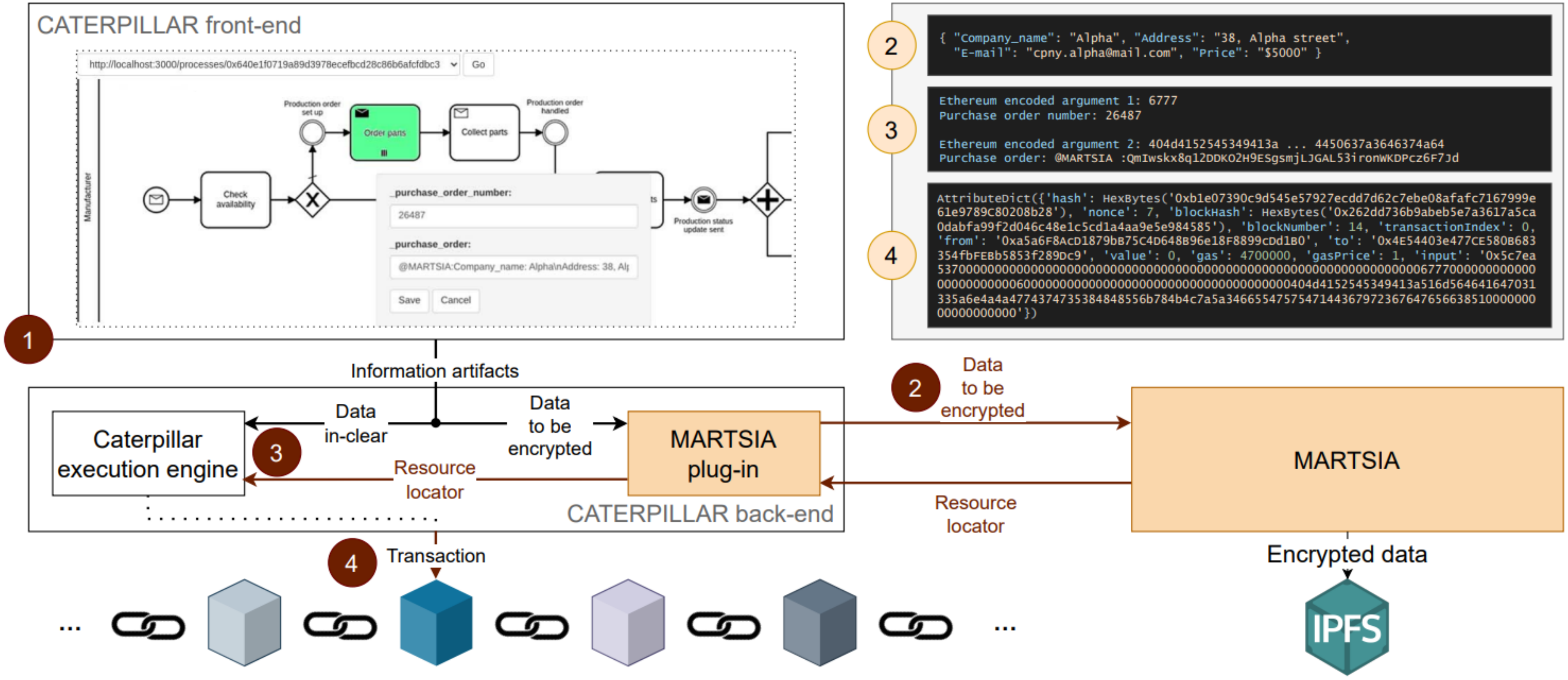
Marangone, D.C., Friolo, Nemmi, Venturi, Weber: MARTSIA: Enabling Data Confidentiality for Blockchain -based Process Execution (2023)

The new architecture: MARTSIA



Marangone, D.C., Friolo, Nemmi, Venturi, Weber: MARTSIA: Enabling Data Confidentiality for Blockchain-based Process Execution (2023)

Integration with Caterpillar



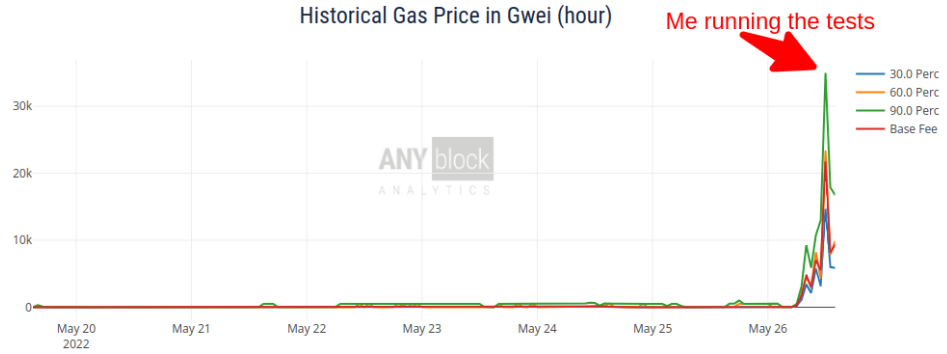
About the costs

	Execution cost $[\text{Gwei} = \text{ETH} \times 10^{-9}]$				
Platform	Contract deployment (1692955 gas units)	Steps **0.1** to **0.5** (476547 gas units)	Step **1.2** (67533 gas units)	Step **2.4** (89772 gas units)	**Avg. latency** [ms]
Sepolia (ETH)	2539432.514	714820.504	101299.501	134658.001	9288.574
Fuji (AVAX)	340498.771	95873.485	13586.538	18060.662	4278.099
Mumbai (MATIC)	1283.163	354.691	50.311	66.012	4944.807
	Off-chain execution time [ms]				
	0.000	2582.471	38.280	158.447	

Why are prices in Gwei?

ETH/EUR exchange

Gas price on the Ropsten testnet



Friday, June 7, 14:00: *Trusted Execution Environment for Decentralized Process Mining* (Goretti, Basile, Barbaro, D.C.)

Open challenges

- Revoke access to data
- Let Smart Contracts use off-chain data via pull-inbound oracles
- Test with real-world multi-party business processes in production
- Extend the policy language with primitives for aggregating and manipulating data
- …

Open challenges at large

Friday, June 7, 14:00: *Trusted Execution Environment for Decentralized Process Mining* (Goretti, Basile, Barbaro, D.C.)

- Strike a balance between “smart-contracting” and off-chain deployment of PAISs
- Define the interplay of Blockchain-as-a-Service for PAISs
- Build a standard communication format for blockchain-based inter-organisational information exchange
- Establish guidelines for the use of blockchain technologies with and within PAISs
- …

Utrecht University

Friday, June 7, 14:00: *Trusted Execution Environment for Decentralized Process Mining* (Goretti, Basile, Barbaro, D.C.)

“I’m still / I’m still / Chaining from the Block”

An Outlook of the Ongoing and Future Relationship between Blockchain Technologies and Process-aware Information Systems

Claudio Di Ciccio | https://diciccio.net/ | c.diciccio@uu.nl
Utrecht University, Netherlands

Picture: https://youtu.be/dly6p4Fu5TE?si=MTqzSi5KGLIZn3sr