CoinFabrik

Censorship-Free Publishing

Introduction

Bitcoin is said to bank the unbanked. Why? In contrast to the traditional bank system, where new clients must provide personal information to open an account, the only requirement to join the Bitcoin financial system is of technical nature: a device to run the software to communicate with the Bitcoin network. No entity decides who is allowed to participate in the network -- an interesting feature also useful for a censorship-free press application.

Blockchains are already used to store *non-financial* data for diverse purposes, e.g. to prove authorship of ideas or to prove the existence of a document. One of the largest files stored successfully into the Bitcoin blockchain is an image of Nelson Mandela. A user managed to insert this photo of about 14 KB. In truly decentralized blockchains, a *valid transaction* -- the standard format of transactions defined in each blockchain -- almost always passes onto the ledger. Hence, not only there is no entity who gives permissions to join the network, there is also no one who might filter out content.

The mechanism to "chain" the blocks of transactions together does not allow to tamper data, once they enter into the blockchain. And, crucial for the avoidance of *single point of failure,* the blockchain is replicated, i.e. nodes run a local copy of it. Therefore, blockchains are not prone to downtimes, or, in the words of a chinese student, "there is no 404 on the blockchain" (source). All this makes blockchain a promising candidate for journalists seeking censorship-free media.

In this article we begin with a short summary of blockchain technology and present a high-level analysis of censorship-resistance in various proposed consensus protocols. In the second section, the publishing process in blockchains is described with a focus on Bitcoin and Ethereum. In section three, security issues about censorship and malleability are addressed. Finally, we compare the available methods for data insertion in Bitcoin and Ethereum by measuring costs and maximum data size per transaction. The article closes with a section summarizing critics and an outlook for future research.

Background

Blockchain is a distributed ledger technology (DLT), i.e. a network protocol to securely communicate state transitions of a shared database. Since there are already many sources explaining blockchains (e.g. [6]), we will just take a look at the concepts relevant to our discussion.

Blockchain as a Secure Transaction Ledger

Blockchains use a peer-to-peer network to communicate new transactions and find a consensus about their validity. Peers of the network are identified by public-private key pairs, where their public key serves as their identity and account address, whereby their private keys are used to identify themselves and authorize transactions. One node per round is elected to propose the next block of transactions for the append-only ledger. The remaining nodes accept or discard the proposal, and update their local copy correspondingly. In Proof of Work (PoW) blockchains, like Bitcoin or Ethereum, validating peers are called *miners*, in Proof of Stake (PoS) or classical Byzantine Fault-Tolerant protocols (BFT) - see below - , they are usually referred to as *validators*. These nodes usually have a full copy of the blockchain in order to validate transactions, they are often referred to as *full nodes*.

In order to protect the network communication from denial of service attacks and spamming, transactions must be of a specific format and satisfy a certain criteria to be gossiped from one node to the other. Furthermore, to protect blockchains from invalid transactions, incentives to behave correctly are set. In blockchains like Ethereum, which allow general-purpose computing, fees must be paid to prevent denial of service attacks. It is very important to take this into account when non-financial data must be passed onto the blockchain. A generic definition of a *standard transaction* can loosely said to be a transaction which:

- is authorized through a digital signature belonging to the account *or* coin owner,
- does not exceed the account balance *or* must be an unspent coin, and
- must be of a *standard transaction format* which depends on the blockchain.

The essential steps a valid transaction must pass - viewed from a high-level perspective - are the following:

I. Validation: Miners or validators collect transactions, validate them and participate in the election process to be the block proposer for this round.
II. Consensus: Miners or validators vote on the block proposal on the basis of the validation of block content and verification of block proposer's winning ticket. There are different ways to introduce a voting mechanism which will be quickly mentioned below.
III. Rewards and Fees: Block proposers whose blocks are accepted by the network, are compensated by a reward and transaction fees. This influences the behaviour of miners and validators in choosing and validating transactions.

Nodes who follow the rules of the protocol are called *honest* nodes. Secure blockchain protocols guarantee that all peers have a *consistent* view of the ledger and that all valid transactions finally enter into the ledger (a property usually called liveness or availability), even in the presence of malicious nodes. This requires at least a certain percentage of honest nodes in the network, usually 51% (in PoW or PoS) or 67% in classical BFT.

The immutability of data is achieved by the use of a *hash function* which links the successive blocks. The link to a new block in the chain ties the content of the block and its timestamp to the block itself not allowing any alteration afterwards. Using the SHA-256 hash function as in Bitcoin or Ethereum, there is no practical threat that permits data to be changed once placed in the blockchain. Threats beyond the security protocol will be addressed in the security section.

Permissioned vs Permissionless Blockchain

In a traditional database system, a *writer* is any entity which has the permission to write content to the database. In DLT this would correspond to a miner or validator that is involved in the consensus protocol and proposes a new block for the distributed ledger. A *reader* in a blockchain system is any user which is not extending the distributed ledger, but participating in either sending a new transaction to the network, or by simply observing the content of the blockchain. There is an important difference between permissionless and permissioned blockchains.

The defining feature of a *permissioned* blockchain is that a central authority (CA) decides and attributes the right to individual peers to participate in the write or read operations of the blockchain. This enables to limit the set of readers and writers. The central entity may change through time, and old authorities may hand their rights to other peers (e.g. Quorum of J.P.Morgan and Multichain). Depending on how the read permissions are set, a permissioned blockchain is called private or public, if reading is only allowed to specific nodes or to everyone.

Permissionless blockchains are defined by the property that any peer can join and leave the network as reader and writer at any time. There is no central entity which manages the membership, or which could ban readers or writers. This implies for a publishing platform that anyone can write and read content which is essential for censorship-resistance.

Which Blockchain for Free Press? -- A High-Level Overview

The two major threats for censorship-free publishing in the permissioned setting are the right settings to participate in the writing process and the small network size. The central authority decides who has the permission to extend and read the shared ledger. This requires a strong trust assumption about the authority. This authority can be a single entity or a consortium (see Hyperledger Fabric or Multichain), but in both cases it is a small group to be trusted not to censor data. On the other hand, the size of the set of validating nodes in classical BFT protocols used in permissioned blockchains does not scale in network size. It is a small number of nodes, e.g. 4-8 in Practical Byzantine Fault-Tolerant protocol (PBFT). This means that in case that these nodes collude, data insertion or transparency cannot be guaranteed.

Although BFT protocols used in permissioned blockchains do not allow a rollback, there is no protection against the threat to go back in the transaction history, if all nodes and the central authority decide to do so in order to change the data. They can enforce this beyond the protocol. It was one of the great achievements of the Bitcoin blockchain to be truly permissionless and to be practical for a big network size, e.g. thousands to millions of nodes, where collusion of a small set of validators can be neglected. As long as the majority of the mining power behaves compliant to the protocol, the security of the blockchain is guaranteed, e.g. valid data enter into the blockchain and are communicated consistently.

Today in practice though, there are mining pools possessing major parts of the hashing power, so a collusion between them could lead to a >51% attack making the security guarantees invalid. The Proof of Stake protocols (PoS) do not depend on external resources (energy in the case of PoW), so they are not prone to the threat of mining pools and are truly decentralized while running on large networks without big computational overhead. Instead of being based on the principle “one cpu, one vote”, the power to vote is determined by “one stake, one vote”. The protocols of Algorand, Avalanche and Cardano prove the viability of this approach. Certainly, unequal distribution of stake incurs other problems.

The scalability problem of Bitcoin blockchain made the community search for alternative consensus protocols. Besides PoS, another prominent proposal is the delegated Proof of Stake (DPoS) consensus, e.g., in EOS, Steemit, or Tron. In these protocols, delegates validate transactions, and can be outvoted, if they behave maliciously. The principle is similar to how a democracy works. Slightly critical in this context is the fact that for short periods, delegates in the committee (21 in EOS case), may reject certain data from entering into the blockchain, as delegates who hinder to insert data, first need to be outvoted.

To sum up, only truly decentralized blockchain protocols can provide guarantees for a censorship-resistant press application. The Bitcoin and Ethereum blockchain are decentralized blockchains with practical limitations. Interesting alternatives are PoS and DPoS blockchains, or protocols even beyond these examples.

Publishing in a Blockchain

How It Works: Inserting Data & “Reading” the Blockchain

Preparation & Broadcast of Non-Financial Transactions: Each blockchain has its specific standard transaction formats. Honest peers do not gossip non-compliant transaction formats to protect the network against Denial of Service attacks. Therefore, to insert non-financial data, it is necessary to do so in a standard-compliant way respecting format and data size, before broadcasting it to the network.

Collection of Transactions and Consensus: Miners or validators collect pending transactions, validate, and pack them into a block. The winner of this round will broadcast the block to the network who decides whether or not to accept the new block and update their local copy. This process depends on the consensus rule of the blockchain design.

Reading the Blockchain: Data stored in the blockchain must be indexed to find them for reading out the content. Large sized files might be split in various transactions, and need to be indexed to link them together. Indexing may be achieved either inside (using metadata stored inside transactions) or outside the blockchain.

Solutions in Bitcoin

Overview

Bitcoin transactions must reference to coins which have been sent previously to a public key address. To authorize a new transaction, the owner of the coin must provide a proof that this coin belongs to her public key address using her private key to generate a cryptographic signature. The relevant information to let the remaining network nodes prove the correctness of the signature is written in the *input scripts* of Bitcoin transactions. Furthermore, transactions specify the receivers of the transfer, again specified by public key addresses which are written into the *output scripts* of the transactions.

Non-financial transactions must appear valid to the Bitcoin miners so that they do not discard them. The conditions for a valid transaction are:

- **Minimum Transfer Amount**: The minimum output value of a transaction is currently about 546 satoshis to be not considered as dust (as of June 7, 2019).
- **Minimum Fees**: The fees are determined by the size of the output and the input script. The current average fee per byte is about 39 Satoshi (cf. https://bitcoinfees.info/, as of June 7, 2019)
- **Maximum Data Size**: The total upper limit of a standard Bitcoin transaction is 100 KB. Input and output scripts may carry specific data of limited size.
- **Unspent Coin**: The input script must reference to an unspent output script.

Transactions that deviate from these rules are considered non-standard and will not be picked by most miners. As we will see below, transactions containing non-financial data may look different from standard transactions. Non-standard transactions may pass to the blockchain anyway, but some with lower probability, and are more critical to future protocol changes.

Data Insertion Methods

Output script:

In [1],[2], the authors identify 5 output script types that are template-compliant which do not involve the input script. Since miners cannot distinguish between legitimate public key address hashes and arbitrary binary data, output scripts can easily be used to insert data indistinguishable to the miners. A disadvantage of the use of output scripts is that users must burn bitcoins as they replace valid receiver addresses with arbitrary data. The following output scripts can be used to insert arbitrary data:

Pay-to-Public-Key (P2PK): Data stored instead of an output of 33 bytes compressed key or 65 bytes uncompressed key together with a non-dust amount of bitcoin to burn.

Pay-to-Public-Key-Hash (P2PKH): Data stored instead of an output public key hash together with a non-dust amount of bitcoin to burn. This allows to store 20 Bytes per output.

OP RETURN: This is a place to store 80 bytes per transaction which is a provably unspendable UTXO that the miners do not need to track.

Multi-Signature: E.g. in case of 1 out of 3 multi-signature script, data can be stored instead of 3 public key hashes, or with 1 real and two unreal signatures in which case the transaction stays spendable (more details in [1]).

Coinbase transaction: Arbitrary data up to 100 bytes can be stored in one transaction per block, but this option is only available to miners.

Input script:

This requires a more sophisticated technique. Input scripts allow bigger size data to be inserted, but must maintain their valid semantics. To achieve this, the input script must refer to a valid output script, e.g. by using a dead branch inserted previously. These transactions are not stored in the list of unspent transaction output set. As described in [1] (see Loc. cit. for more details), there are two special ways to do so:

Pay-to-Script-Hash (P2SH): These data refer to the unspent coin. Data can be stored in the redeem script (limit 520-byte) and/or in the part of the input script followed by the redeem script (limited by the 1650 bytes total limit of the input script). More advanced methods to store data are mentioned in [1]:

- ❏ Data Drop Method: Data get stored in the redeem script.
- ❏ Data Hash Transaction: This uses the script following the redeem script.

Data Reconstruction:

Output scripts in the P2PKH larger than 20 Bytes must be spilt in various output scripts, either within one transaction or, if larger than the maximum size (see the table below), in various transactions. Data need to be linked together either onchain or offchain to allow a reconstruction of datasets stored in the blockchain. One may use the output script to store metadata, e.g. a reference to the transaction ID of the next chunk of data stored in another transaction.

Input scripts may store data using the methods Data Drop and/or Data Hash. As shown in [1], within a single transaction, the maximum file size can be of 96,060 bytes. Files larger than this, require again an indexing of the split data.

Selected Content Insertion Service:

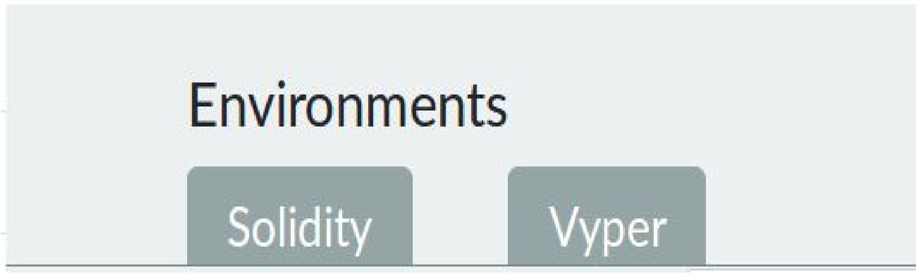
- Apertus: This service allows fragmenting content over multiple transactions using an arbitrary number of P2PKH output scripts. Besides further features, Apertus works also for Litecoin, Dogecoin, and others.

The authors of [1] found that the P2FKH is the method more widely spread, although, it creates the most unspendable UTXO bloat, requires the largest overhead, and is the most expensive. They argue that its popularity can be explained by its simplicity of implementation.

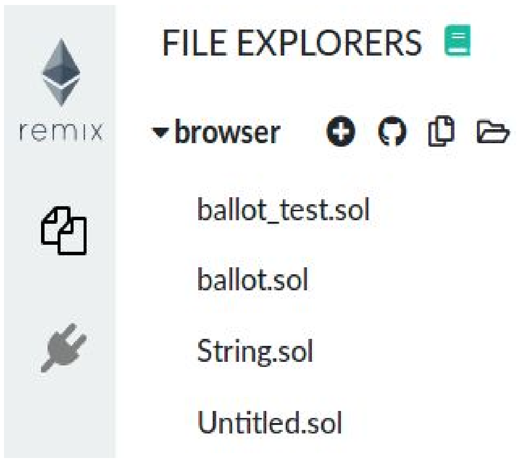
Solutions in Ethereum

This is a step by step guide that shows how to publish a text in the Roptsten testnet of Ethereum. In order to publish in the Mainnet you will need to select it in the first step of the following tutorial.

1. Install MetaMask in your browser and select Ropsten as your network
2. Make sure you have enough Ether in your MetaMask account
3. Deploy a Smart Contract
    3.1. Go to Remix
    3.2. Select the Solidity Environment



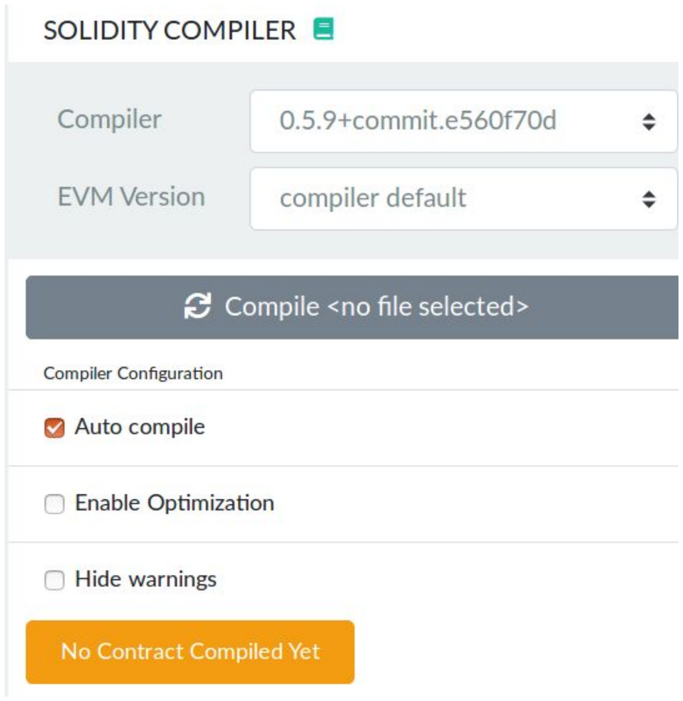
3.3. Create a new File



3.4. Insert the following code

```
pragma solidity ^0.5.9;
contract CoinFabrik {
        event Flag(address indexed _from);
        function() external {emit Flag(msg.sender);}
}
```

3.5. Select the correct Solidity Version (0.5.9) and select your file. After that compile your Smart Contract


SOLIDITY COMPILER

Compiler: 0.5.9+commit.e560f70d

EVM Version: compiler default

Compiler Configuration

[✓] Auto compile

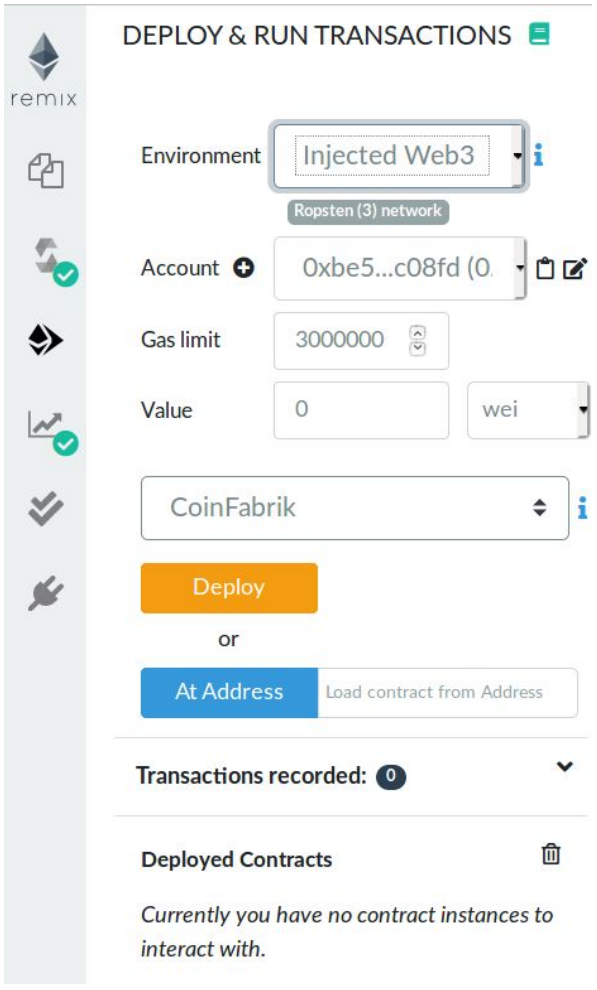
[ ] Enable Optimization

[ ] Hide warnings

Finally, select Auto Compile

3.6. Deploy the contract

3.6.1. In the window below select “Injected Web3” in the environment slot. After that, press Deploy



Environment: Injected Web3
Ropsten (3) network
Account: 0xbe5...c08fd (0.
Gas limit: 3000000
Value: 0 wei

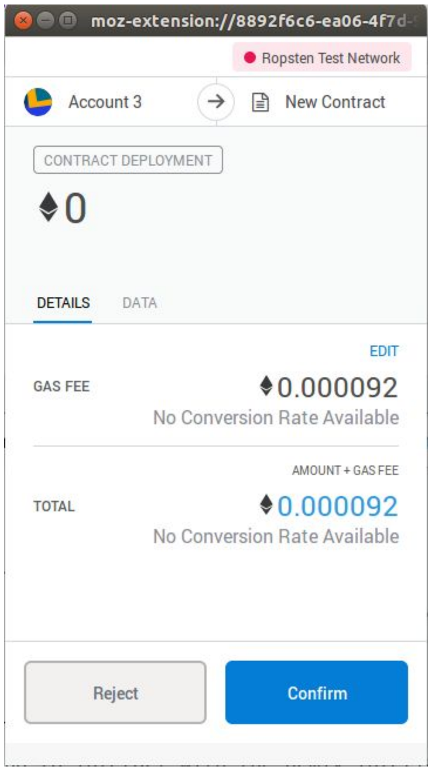
CoinFabrik

Transactions recorded: 0

Deployed Contracts

Currently you have no contract instances to interact with.

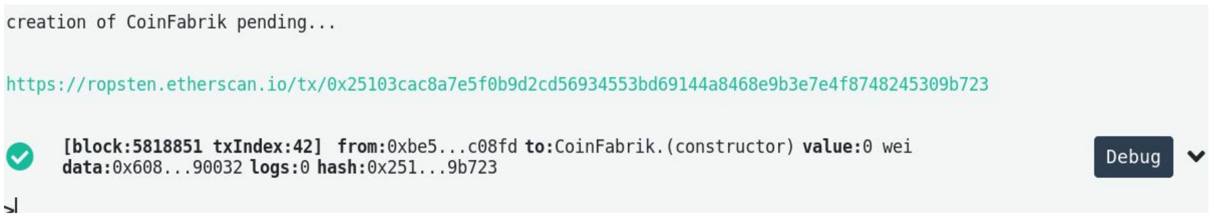
3.6.1.1. Confirm the transaction in MetaMask



3.6.1.2. Wait for the transaction to be confirmed

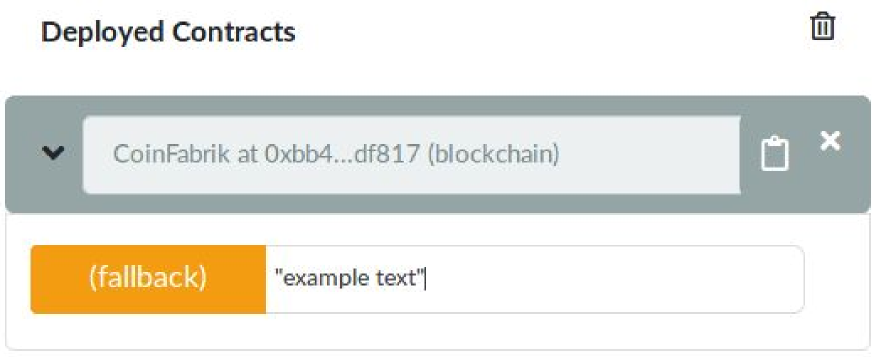
```
creation of CoinFabrik pending...

https://ropsten.etherscan.io/tx/0x25103cac8a7e5f0b9d2cd56934553bd69144a8468e9b3e7e4f8748245309b723
```

You will see what is shown below once the transaction has been confirmed

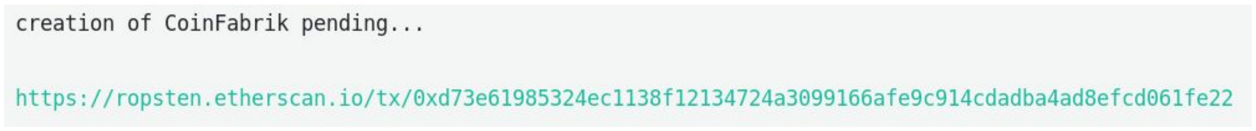


```
creation of CoinFabrik pending...

https://ropsten.etherscan.io/tx/0x25103cac8a7e5f0b9d2cd56934553bd69144a8468e9b3e7e4f8748245309b723

[block:5818851 txIndex:42] from:0xbe5...c08fd to:CoinFabrik.(constructor) value:0 wei
data:0x608...90032 logs:0 hash:0x251...9b723
```

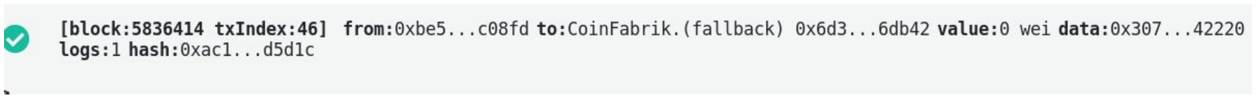
4. Interact with the contract and upload the text
   4.1. First, select the instance of your contract and interact with the fallback function by setting the string you want to upload. Then, press the "(fallback)" button and confirm the transaction



   4.2. Wait until the transaction is confirmed



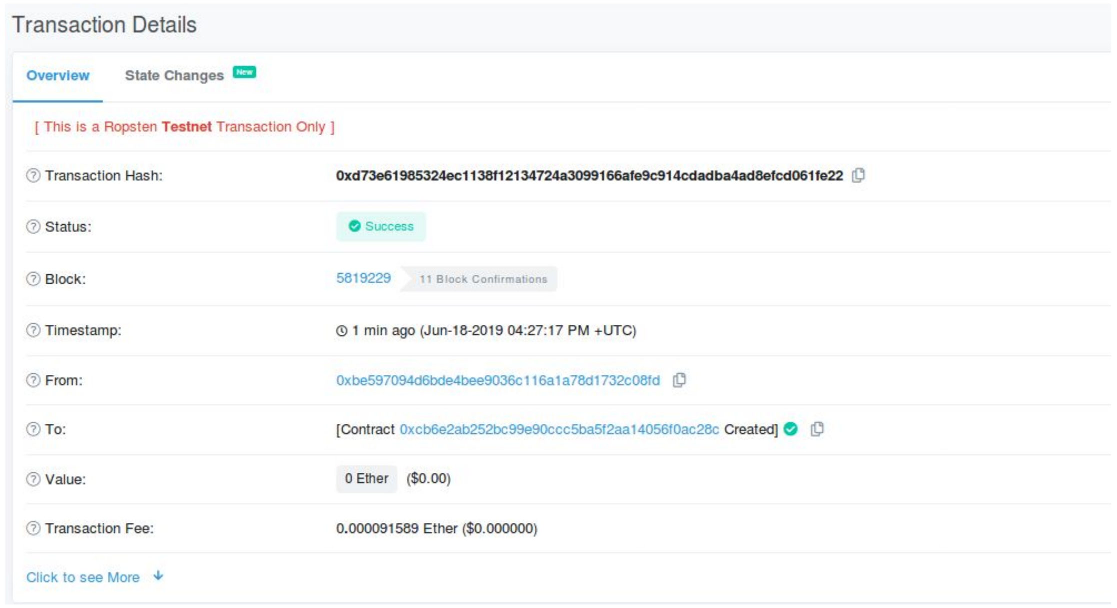
```
creation of CoinFabrik pending...

https://ropsten.etherscan.io/tx/0xd73e61985324ec1138f12134724a3099166afe9c914cdadba4ad8efcd061fe22
```



```
[block:5836414 txIndex:46] from:0xbe5...c08fd to:CoinFabrik.(fallback) 0x6d3...6db42 value:0 wei data:0x307...42220
logs:1 hash:0xac1...d5d1c
```

5. Check the transaction and get your string

5.1. Go to the transaction link and you will see something similar to



Transaction Details

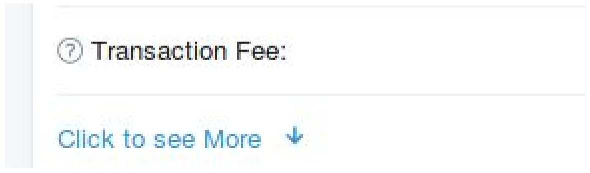
Overview | State Changes New

[This is a Ropsten **Testnet** Transaction Only]

Transaction Hash:	0xd73e61985324ec1138f12134724a3099166afe9c914cdadba4ad8efcd061fe22
Status:	Success
Block:	5819229 11 Block Confirmations
Timestamp:	1 min ago (Jun-18-2019 04:27:17 PM +UTC)
From:	0xbe597094d6bde4bee9036c116a1a78d1732c08fd
To:	[Contract 0xcb6e2ab252bc99e90ccc5ba5f2aa14056f0ac28c Created]
Value:	0 Ether ($0.00)
Transaction Fee:	0.000091589 Ether ($0.000000)

Click to see More

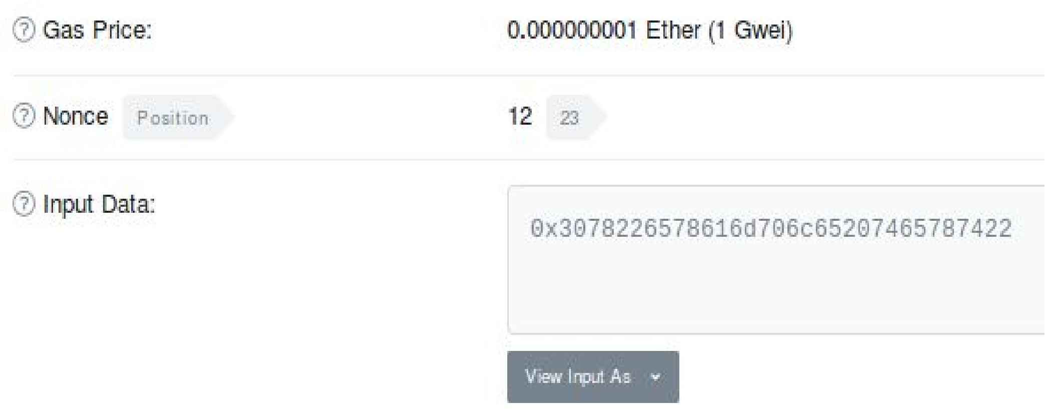
5.2. Press “Click to see More” at the end of the panel



Transaction Fee:

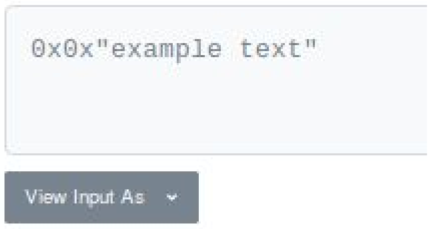
Click to see More

5.3. Go to “View Input As” and select “UTF-8”



Gas Price:	0.000000001 Ether (1 Gwei)
Nonce Position	12 23
Input Data:	0x3078226578616d706c65207465787422

5.4. End up by checking your text


```
0x0x"example text"
```

Security

Secure blockchains always insert valid transactions ("availability") and provide a common view of the actual state of the account balances ("consistency"). But these two properties only hold under some assumptions. In case of PoW, at least 51% of the nodes must be honest and comply with the protocol's rules (not taking selfish-mining into account). Some PoS protocols must assume that 51% of the total stake is in the hands of honest users, others 67%. Classical BFT protocols must assume as well that 67% comply with the rules of the protocol. There must be also cryptographic security and network conditions to be considered, which are not discussed here. In the next two sections we have a closer look at some of the possible practical failures of censorship-resistance and immutability in blockchains.

Censorship-Resistance

Transaction Validation: Transactions can deviate from the approved transaction templates via their output scripts as well as input scripts. In theory, such transactions can carry arbitrarily encoded data chunks. But miners or validators may use detectors to check whether a transaction carries non-financial data and discard those transactions. e.g. in Bitcoin data chunk in the P2PK might be detected, but in P2PKH not, as we already mentioned. In order to protect the blockchain against "pollution" with non-financial data, miners may make use of filters (see [3]) and insert only "standard" outputs. But these changes may incur the risk of false positives which would require a fork of the blockchain. As long as the majority of the miners do not make use of filters, the insertion methods to store non-financial data are applicable in Bitcoin. Furthermore, in the Ethereum network, as a general purpose computing platform, it is even more difficult to think of filters which would not produce false positives.

Incentive Layer: The reward and fee system must be economically aligned to incentivize honest behaviour. Future changes (via forks) of the incentive layer may cause rejection of certain transactions because miners prefer to get the maximum profit. On the other hand, as suggested in [3] to protect the Bitcoin blockchain from insertion of illegal content, minimum fees which increase with the output size can be imposed to disincentivize publishers of inserting non-financial data.

Hard Forks: The liveness of the protocol depends on the size of the network and whether the majority sticks to the protocol rules. Changes of the protocol (forks) may impact the usage of the blockchain for censorship-free publishing (cf. [3]). The condition to have an unchanging blockchain protocol is the existence of a critical mass of network size which is necessary to have safety assurance.

Reading the Blockchain: The only censorship with respect to reading the blockchain is given in the context of permissioned private blockchains. As long as there are many replicas of a permissionless blockchain, there is no threat of censorship of the stored data.

Moreover, some blockchains have special governance rules which allow special members to execute specific actions, e.g. to ban others, as in the case of Steemit (see this report).

Immutability

Block Generation: Transaction malleability refers to a possible change of a transaction during the validation by a miner. When a new transaction is broadcast to the P2P Bitcoin network, it gets passed from node to node, with nodes verifying it and storing it into the mempool of possible transactions to include in a block. An adversarial node may receive a transaction and create a modified version of this transaction to pass along to others in the network. The malleability may impact the input script methods Data Drop and Data Hash Transactions are discussed for the Bitcoin network. We refer to [1], p. 9, for their analysis. In other blockchains, this requires further analysis.

Mutability of the Blockchain: Hard forks may split the blockchain and allow to reverse the transaction history to a previous time. Although those who continue with the old protocol still carry the information, new members may not be attracted to use this chain as there is only a small community left. A typical example of a hard fork is the split of the Ethereum blockchain into Ethereum and Ethereum Classic, while in this case, both blockchains are still alive.

Publishing Requires More Than Blockchain

Threats for censorship-free publishing may occur due to centralized components in the architecture. Although the blockchain is a decentralized peer-to-peer network, one should bear in mind that the communication with the network requires to run a full node or to take advantage of a service, e.g. a web or mobile wallet. The first variant requires to download the entire blockchain which, in the case of Bitcoin, is about 210 Gigabyte and needs several days to be synchronize with the blockchain. The second option is the use of a service who runs a full node. In this last mile, the final end-user may encounter several problems such as failure while interacting with the front end, or even that the service provider denies to accept the article the user wants to publish. The defense strategy against such censorship is to change the service provider or opt for the first solution, leaving the possibility open to always insert non-financial data.

Similarly, centralization applies to “reading” the blockchain. To extract the data from the blockchain requires some more advanced knowledge. These front-end problems may affect journalists who are not experienced with the blockchain technology, and make them depend on services which are not decentralized.

Comparison

Comparison Metrics

In order to compare the different data insertion services on diverse blockchain networks, we consider four basic metrics.

Maximum Data Size: This metric refers to the maximum data size without splitting data into various data chunks with respect to the given method.

Maximum Data Size per Tx: This metric refers to the maximum data size which can be stored inside one transaction while splitting the data into several chunks.

Costs per Byte: This value indicates the price to store 1 Byte in the chosen blockchain. This cost depends on the applied fees, necessary burns, and takes the current price of the native coin into account.

Total Cost: This value indicates the price to store the maximum data size in one transaction.

Evaluation Overview

Cost calculation in Bitcoin:

The cost of data insertion in Bitcoin is composed by the current minimum fee (39 satoshi/byte to be inserted within the next 6 blocks according to bitcoinfees.info as of June 7, 2019), and the minimum non-dust value of 1100 satoshi for P2MS and 546 for other methods requiring burns. Input scripts allow to spend the transaction afterwards. We use the rate 8,101 USD per 1 BTC as of June 7, 2019.

Formula: $Total\ Cost\ =\ (Number\ of\ Bytes\ x\ 39\ Satoshi)\ +\ (Number\ of\ Outputs\ x\ 546/1100\ Satoshi)$

Data sizes of the scripts are shown in table 1 (cf. also table 3 in [1]).

Cost calculation in Ethereum:

Empty fallback function:

If you send the string as a parameter to a fallback function of a smart contract and the smart contract has an empty fallback function, you will have the following formula

Formula:

$$Total\ Cost\ =\ (Number\ of\ Bytes\ x\ \ 68\ Gas\ +\ 21370\ Gas)\ *\ GasValue$$

Fallback function with event saving the address

You can always save the address of the sender in the header of the block by emitting an event in the fallback function of the smart contract with the purpose of indexing the address.

If you do that, you will have the following formula

$$Total\ Cost\ =\ (Number\ of\ Bytes\ x\ \ 68\ Gas\ +\ 22530\ Gas)\ *\ GasValue$$

For the following table we will assume that the Ether price is 242.51 and the gas price is 4 Gwei

Network	**Type**	**Max Size per Tx / Max Data Size**	**Total Cost in BTC / USD**	**Cost per Byte in Satoshi or Gwei/ USD**
Bitcoin	P2PK	85,280 bytes / 65 bytes	0.04606916 / 373.2	54.02 / 0.00437
	P2PKH	58,680 bytes / 20 bytes	0.05499117 / 445.5	93.71 / 0.00759
	OP_Return	80 bytes / 80 bytes	0.00012363 / 1.0	154.5 / 0.0125
	P2MS	92,624 bytes / 195 bytes	0.04918329 / 398.4	53.10 / 0.004301
	P2SH	62,340 bytes / 1,650 bytes	0.072175389 / 584.7	115.78 / 0.009379

	Data Drop	90,099 or 96,060 / 520 bytes	0.03982407 / 322.6	44.2 or 41.6 / 0.0036 or 0.0034
	Data Hash	86,087 or 92,507 / 1,529 bytes	~ 0.03983 / 322. 65	46.27 or 43.06 / 0.0037 or 0.0035
Ethereum	Empty fallback function	117,332 bytes	0.031999784 / 7.76	85480 Gwei per transaction + 272 Gwei per byte / 0.02 USD per transaction + 0.000066 USD per byte
	Fallback Function with Event	117,315 bytes	0.0319998/ 7.76	90120 Gwei per transaction + 272 Gwei per byte / 0.02 USD per transaction + 0.000066 USD per byte

Critics

The authors of [2] and [3] call the attention to the legal problems censorship-free and immutable news platform brings along. Publishing private content may contradict the right to forget, right of privacy (GDPR), and copyright regulations.

On the other hand, a truly censorship-resistant platform can help to bring controversial information to the surface and to start political discussions which might be banned immediately by powerful entities. It is the question of how much freedom is necessary for a society to evolve fairly and to progress into a new state (synthesis), without causing injury to the remaining members of the group.

Besides the legal concerns, there are also technical critics. The storage of fake addresses create irretrievable burn transactions which bloat the ledger size unnecessarily. The miners must permanently track each unspendable UTXO created in this way, producing an extra computing overhead.

Future Investigation

Topics for future research:

- Publishing in other public blockchains: Litecoin (similar to Bitcoin), EOS (Everpedia), etc.
- Publishing in specific blockchains: Permacoin, Arweave.
- Distributed storage + blockchain for data integrity**:** PubliQ, SWARM, Filecoin, Storj, SIA.
- Other peer-to-peer shared storage solutions + comparison with blockchain solutions.
- Accenture. Redactable blockchain.

References

[1] A. Sward, I. Vecna, and F. Stonedahl. Data Insertion in Bitcoin's Blockchain. Ledger Journal, 2018.

[2] R. Matzutt, J. Hiller, M. Henze, J. H. Ziegeldorf, D. Müllmann, O. Hohlfeld, and K. Wehrle. A Quantitative Analysis of the Impact of Arbitrary Blockchain Content on Bitcoin. In Proceedings of the 22nd International Confer-ence on Financial Cryptography and Data Security (FC). Springer, 2018.

[3] R. Matzutt, M. Henze, J. H. Ziegeldorf, J. Hiller, K. Wehrle. Thwarting Unwanted Blockchain Content Insertion. In Proceedings of the First IEEE Workshop on Blockchain Technologies and Applications (BTA), co-located with the IEEE International Conference on Cloud Engineering 2018 (IC2E 2018), IEEE, 2018.

[4] K. Shirriff, Hidden surprises in the Bitcoin blockchain and how they are stored: Nelson Mandela Wikileaks photos and Python software, 2014, [online]. Available: http://www.righto.com/2014/02/ascii-bernanke-wikileaks-photographs.html.

[5] G. Ateniese, B. Magri, D. Venturi, and E. Andrade. Redactable Blockchain – or – Rewriting History in Bitcoin and Friends. In: Proceedings of the 2nd IEEE European Sym-posium on Security and Privacy—EuroS&P 2017.

[6] F. Tschorsch, B. Scheuermann. Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies. In: IEEE Communications Surveys & Tutorials, Volume: 18, Issue: 3, 2016.