Interreg
Euregio Meuse-Rhine
BC4P

EUROPEAN UNION
European Regional
Development Fund

Structure of the Blockchain Development and Database for Pilots

D.T2.1.1

Version	Description	Author	Date
0.1	Initial draft.	Florian Maurer, Lukas Walk (FH-Aachen)	20/01/2023
	Review	Stefano Bromuri (OU)	5/02/2023
0.2	Included review (no citations)	Lukas Walk	6/02/2023
1.0	Citations/References	Lukas Walk	17/02/2023
1.1	Finalization	Florian Maurer	21/02/2023

PXL
LIÈGE université
FH Aachen
University of Applied Sciences
Open Universiteit
howest
Tiorc
ZU YD
CLIMATE CITIES
CAP CONSTRUCTION

Table of Contents

Abstract

This report provides an introduction into the blockchain technology and a description of the basic structure of the blockchain, that is used for all pilots. The focus is on the decentralized database and the back-end programming. Furthermore, it describes the development of the used blockchain technology which is based on an openly available Ethereum-compatible blockchain platform.

The blockchain is used as a decentralized database to store all trades made at the peer-to-peer energy market, that is implemented in this project based on the Grid Singularity framework (gsy-e)[1].

A decentralized blockchain provides traceability, security and other advantages for the stored trade data. These properties are influenced by the type blockchain and the underlaying consensus protocol to synchronize the distributed system.

In this report different options of blockchain technology are explained and it is discussed why a private Ethereum[2]-based blockchain is used for this project.

The available software to deploy a private blockchain used in the project is written in Go. Furthermore, the energy trading platform Grid Singularity and the software based on it is written in Python.

Introduction

The energy market in Europe is a complex system consisting of various participants. The market is regulated by national governments, the European Unions.
The energy sources in Europe consists of various types like fossil fuels, nuclear power and renewable sources like wind and solar.

Particularly these renewable energy sources are an important growing factor inside the energy market.

There are several initiatives in place to incentivize the use of more renewable energy sources and reduce the dependency on fossil fuels.

Blockchain technology has the potential to assist the energy market, because its decentralized peer-to-peer (P2P) network nature could be used for the energy market by enabling the creation of peer-to-peer (P2P) energy trading platforms.[4, 5]

These platforms would allow individuals and businesses to directly buy and sell excess energy that they generate from renewable sources, such as solar panels or wind turbines to other participants of the market for profit. This would increase the incentive to build and use more renewable energy sources. Furthermore, the simplified and higher control over the trading results in the possibility to optimize the monetary gain from using renewable energy sources and energy storage technology.[14,15]

The blockchain can also be used to create smart contracts which allows a higher degree auf automatization[7] inside the energy market.[8,9] Trades can be executed automatically depending on several triggers like reaching specific levels of energy prices. This would result in a more efficient energy market with less need for intermediaries.

Furthermore, the blockchain can be used to create more transparent and secure energy supply chains.[10,11,12,13] This would enable the possibility to track the production and distribution of energy, ensuring that it is being used sustainably and ethically. This can help to build trust in the energy market and increase confidence in the renewable energy sector.

This report focuses on evaluates using blockchain technology to decentralize and enable a peer-to-peer energy market.
In this report, the role of the blockchain inside this project and the blockchain technology in general is explained. Furthermore, the reasons and properties of the used blockchain technology is stated. Finally, the blockchain implementation used for all pilots and its deployment is documented.

The rest of this report is structured as follows:
Section “Blockchain Technology” explains the blockchain technology in general and the solution, that is used inside this project.
Section “Blockchain Implementation and Deployment” describes the deployment of the blockchain used in this project in detail.
Section “Application in P2P simulation” describes the BC4P project as a whole and the role of the blockchain itself inside it.
Finally, section “Conclusion and Future Work” concludes this report.

Blockchain Technology

A blockchain is a distributed ledger, that stores transactions and secures them with the help of cryptographic hashes.[3]
Main focus of this technology is traditionally a heavy decentralization, immutability and censorship resistance of any transactions stored in this ledger.[6]
Furthermore, a blockchain is a peer-to-peer network consisting out of participants with equal rights. That means that everyone participating is usually also involved with securing the network itself.

General Blockchain explanation

A Blockchains is mainly a concatenation of data in a one-way direction. It's structured into blocks which contain the transaction data. Each block links to the previous block via a reference that is the hash of the previous block called parent block. The first block of a blockchain is called genesis block and has no parent block.
The hashes secure the transaction data inside a block and all blocks previous, because the hash of the current block includes the hash of the previous block, too.

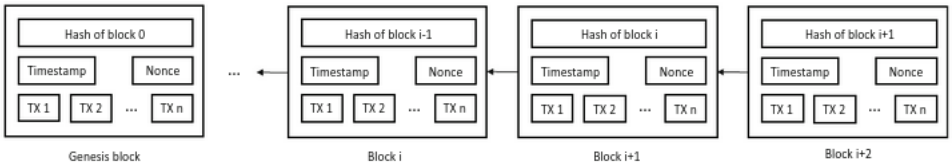

Figure 1: Blockchain structure

Transactions Integrity and Authenticity is secured via asymmetric public-key cryptography. Simplified explained every transaction consists of two or more participants. Funds are saved at different public key addresses. The owner of these public keys is identified by the person who has the corresponding private key to this public key. Only this person is able to access the tokens saved at this address.

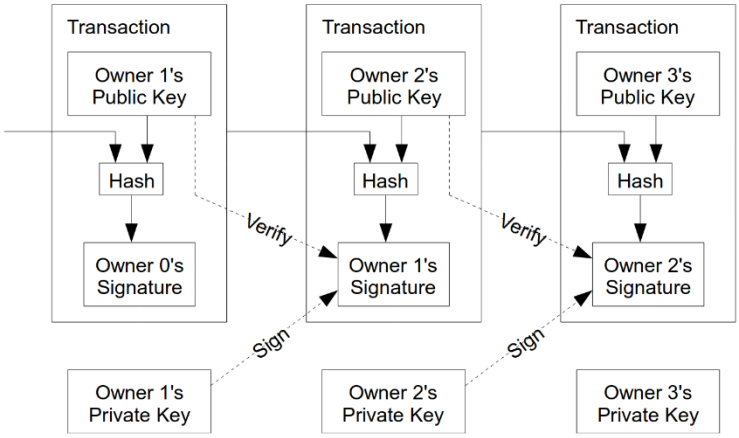

Figure 2: Signing and Verifying Transactions with public key cryptography in Blockchains

The essential part of every blockchain solution in a distributed system is the consensus mechanism.

If multiple participants try to add new data to the blockchain they need to create a new block. In a distributed system this can happen at different participants at the same time. This would result in a conflicting state of the blockchain. To decide which of the added block is the correct one in traditional blockchains a probabilistic consensus mechanism like Proof-of-Work (PoW) [16] or Proof-of-Stake (PoS) [17] is used.

Blockchain Consensus mechanisms

Proof-of-Work (PoW) is a type of consensus mechanism used by many cryptocurrencies, such as Bitcoin, to secure their networks and validate transactions. In a PoW system, miners compete to solve complex mathematical puzzles in order to create new blocks and add them to the blockchain. The first miner to solve the puzzle and add a new block is rewarded with a certain amount of the cryptocurrency.

Proof-of-Stake (PoS) is another type of consensus mechanism that is becoming more popular among cryptocurrencies. In a PoS system, the chance of a miner creating a new block and receiving a reward is proportional to the amount of the cryptocurrency that they hold. This means that the more cryptocurrency a miner holds, the more likely they are to create a new block and receive a reward.

Proof-of-Authority (PoA)[18] is a consensus mechanism that is similar to PoS, but with some key differences. In a PoA system, the miners who create new blocks and validate transactions are pre-selected, rather than being chosen based on the amount of cryptocurrency they hold. This means that the network is controlled by a set of trusted and authoritative nodes, rather than being open to anyone who holds the cryptocurrency. PoA

mimic deterministic consensus mechanisms like Raft[19], because the set of validators is much smaller than in a PoW or PoS system.

Overall, the main difference between PoW, PoS, and PoA is the way that new blocks are created and added to the blockchain. In a PoW system, miners compete to solve complex puzzles, while in a PoS system, the chance of creating a new block is based on the amount of cryptocurrency held. In a PoA system, the miners are pre-selected and not chosen based on the amount of cryptocurrency held.

Decision and Reasons for type of Blockchain used

For the usage in energy markets, anonymous access cannot be provided as this would create blackouts and problems in the grid.

The reason for problems in the grid can occur, because of the oracle problem.[20] In an unrestricted blockchain solution the energy produced or consumed could be different from the trades made to the blockchain. Therefore, the prosumers need to have verified smart meters installed at their locations, that guarantee that the energy amount produced and consumed mirrors the trades made on the blockchain.

The grid operator needs to know which smart meter is installed at which prosumer and which blockchain wallet address is linked to them, so they can punish bad behavior and ensure a stable grid.

This results in the need of a private blockchain solution with restricted access, where the grid operator must have the authority to allow or deny access to individual participants of the system. Furthermore, the grid operator needs to operate the consensus mechanism, because he needs to ensure, that every transaction inside the blockchain is made between valid prosumer addresses.

Therefore, solutions using PoS or PoW cannot be used.
In this project we are therefore using an Ethereum based blockchain with PoA consensus called Clique[26] as this provides the best compatibility with smart contracts and allows us to integrate into existing tools while also providing the demands of control for the correct market behavior in energy systems. In an open PoW blockchain any participant could increase his hashrate and include transactions of non-verified participants.

Clique Consensus Protocol

The Clique Consensus Protocol got developed to be compatible with the normal Ethereum Client. The notable difference to a PoW or PoS protocol is, that there are predefined signers. Only these signers can participate at the consensus protocol and vote new signers.

The public key addresses of the signers are defined inside an authorization list, that is saved inside the genesis block of the blockchain.

The consensus works like Round-robin scheduling. This means, that the signers are defined in an order and the right to generate the next block circulates in this order. The mining itself doesn't require to solve a PoW puzzle. This measure prevents, that a compromised signer has too much influence at the blockchain. It is required, that the signers wait a specific time (the block interval) between signing new blocks and to respect the round robin, so that changes can reach all other signers in the network otherwise it would be possible to create sidechains like in PoW or PoS consensus protocols. Malicious or compromised signers would generate blocks, that would be omitted via generating sidechains. Like in other blockchain consensus protocols the longest chain wins. That means as long as the majority of signers is good-natured the integrity of the blockchain is given.
Signers have the possibility to vote to change the authorization list, so that malicious ones can could be excluded and new ones included.

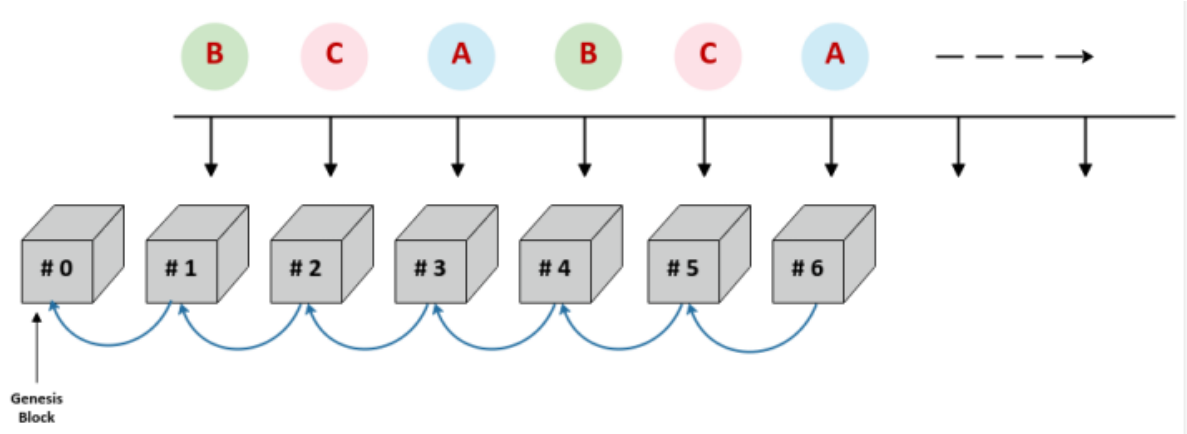

Table 1: Clique consensus protocol: 3 validators Round-robin block generation.[29]

One significant difference is that the number of signers is usually way lower than participants in a PoW or PoS consensus protocol and these signers are usually controlled by a small number of entities. This results in a fast and energy friendly but heavily centralized consensus protocol.

Features and Attributes of the used blockchain system

The use of a private blockchain with PoA consensus results in a heavily centralized blockchain system, that do not have the same features like public blockchains.

Notable missing features are censorship resistance, the immutability of the ledger and the decentralization of the network, that provides a high replication of the database, fail safety and availability of the system.

The use of the private blockchain is needed, because the validity of the energy exchanges needs to be verified by smart meters and the grid safety needs to be ensured by the grid operator. While this prevents the full implementation of the attributes of a public blockchain, attributes like a high decentralization in the blockchain consensus protocol

would contradict the energy grid with centralized authorities in the first place. Attributes like censorship resistance and immutability of the ledger are undesirable, because the system need to mirror real consumption and trades. That means, that bad trades or bad actors needs to be handled.

Furthermore, the Ethereum bases blockchain allows the usage of a standardized smart contract protocol and already widely adapted interfaces.

A big benefit of using a PoA consensus mechanism is, that the energy consumption is not higher than traditional database solutions. A PoW solution on the other hand, would require a significant amount of overhead energy consumption to provide the safety of the public network.

Blockchain implementation and deployment

Blockchain technology is used in cryptocurrencies since the development of Bitcoin in the year 2009.[16] Since then, thousands of cryptocurrencies and other blockchain projects evolved.

In the previous section restrictions and requirements for the blockchain technology, that can be used for the energy market were made. The next step is to get an overview about existing opensource projects, that could fit these requirements.

Comparison of different opensource communities

There are plenty of different available open-source implementations of blockchain technology. In this project support for smart contracts is required, so that execution energy trades can be automated. Furthermore, the option to have complete control over the network is wanted. For development purposes and code safety a wide Userbase for documentation, examples and help and a project in active development is wanted.

In the following table a comparison of open-source communities on GitHub can be seen.

Name	**Stars**	**Watcher**	**Forks**	**Follower**
Bitcoin [21]	67k	3,9k	33k	NA
Ethereum [22]	40k	2,2k	15k	4700
Hyperledger [23]	14k	1k	8k	572
Energy Web [24]	92	21	50	84

Table 1: Size comparison of open-source GitHub communities

Ethereum is the biggest open-source community for a customizable open source blockchain, that has smart contract combability.

Furthermore, using Proof of Authority is possible here and we can deploy our own independent permissioned blockchain.

For the development of the project blockchain and tools a permissioned blockchain with geth, an open-source implementation of a standalone Ethereum client written in go, is set up.

Go-Ethereum

Geth (go-ethereum) is a Go implementation of Ethereum.[25] It is one of original Ethereum implementations and therefore the most battle-hardened and tested client.

It provides the Ethereum Virtual Machine, that can handle transactions, the deployment and execution of smart contracts.

In this project Docker and Docker Compose are used to deploy the different nodes, that power the decentralized network, needed to operate the project blockchain itself.

The nodes required for a stable network consist of a bootnode, an RPC-Node and a set of Validator-Nodes.

The project is available at the Git Repository on GitHub.[28]

Bootnode

A bootnode or bootstrap node sole purpose is to connect new nodes to peers. A new node, that wants to connect to the network, need an endpoint to do this. A bootnode provides this functionality, because its endpoint is provided hardcoded and fixed. From this node the discovery process of other nodes without a fixed endpoint can be started. Geth continuously attempts to connect to a multitude of other nodes inside the P2P network. To not connect to the default Ethereum blockchain we need to specify our own bootnode.

With the following Docker Compose section we deploy a bootnode with the Docker image of the official Golang implementation of the Ethereum protocol.

```
bootnode:
  image: ethereum/client-go:alltools-stable
  ports:
    - "30301:30301"
    - "30303:30303"
  command: [ "bootnode", "-nodekeyhex", "${NODE_PRIVATE}", "-verbosity", "5" ]
```

Figure 4: Screenshot of Docker Compose bootnode section

Validator Nodes

Validator Nodes are full nodes, that participate within the consensus algorithm of this blockchain. The used Proof of Authority consensus protocol is called Clique. The list of authorized signer nodes must be specified among other things inside the genesis block.

This enables the permissioned blockchain, because other nodes, that could connect to the network in the future aren't allowed to participate in the consensus algorithm.

Only the validators specified in the genesis block can provide this functionality. This enables full control over the blockchain for the institution, that deployed it. For this project we use three validators.

With the following Docker Compose section one of the three validators is deployed. The validator needs to have a specific public private key pair, that will be used as verification as participant for the consensus protocol. It is based like the bootnode on the same Ethereum client Docker image.

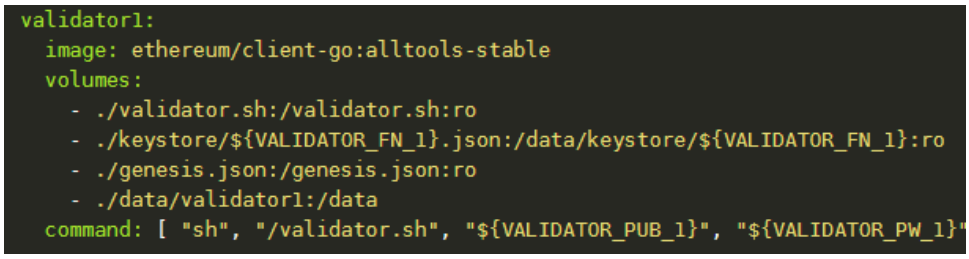

Figure 5: Screenshot of Docker Compose validator section

RPC-Node

The Remote Procedure Protocol or RPC Node allows users to communicate with the blockchain. Usually, it is used to read data from the blockchain and send transactions to the network.
In this project it will be used by the Block explorer, that visualizes the state of the blockchain and by the Grid Singularity Framework to create transactions on the blockchain.
It will be used in applications like MetaMask, too, so that participants in the energy market can see their account status.

The following Docker Compose section shows the deployment of the RPC node, that is based like the Bootnode and validator nodes on the same Ethereum Client Docker image.

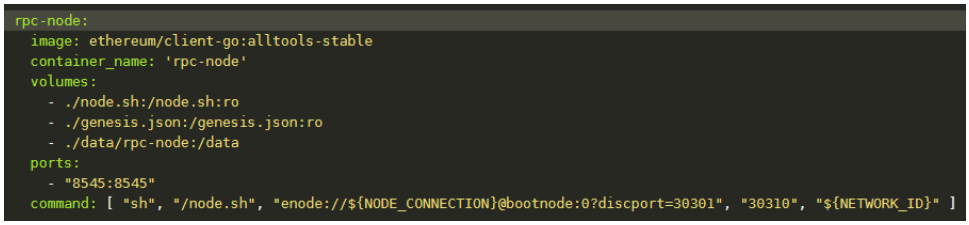

Figure 6: Screenshot of Docker Compose RPC-Node section

Block explorer

The Block Explorer in this project used is an Open-Source solution called Blockscout.[30]
It can be used to view and inspect every transaction and block inside the connected blockchain. It provides a live visualization of new incoming blocks and transactions.

The following screenshot shows, how the overview page of Blockscout looks for the blockchain currently deployed for this project.

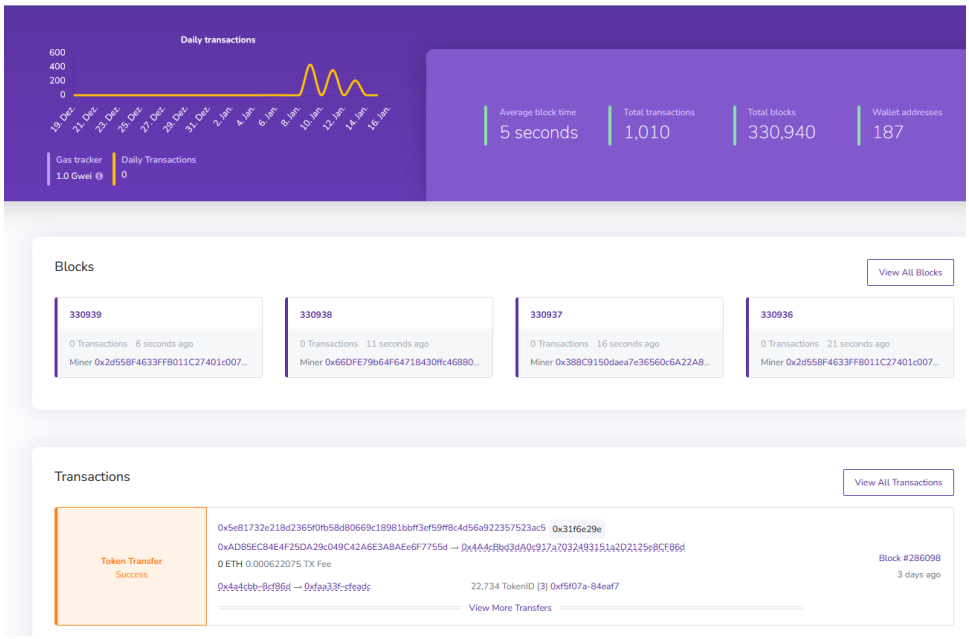

Figure 7: Screenshot of the Blockscout explorer connected to the BC4P cain

The following Docker Compose section shows the Docker Compose section for the deployment of the Blockscout application.
The official Blockscout Docker image is used and the important aspect is, that the RPC-Node must be provided to be able to connect to the blockchain. Furthermore, Blockscout depends on a database, that is used to save the current state of the visualization of the blockchain.

```
blockscout:
  depends_on:
    - blockscout-db
    - rpc-node
  image: blockscout/blockscout:${DOCKER_TAG:-latest}
  restart: always
  container_name: 'blockscout'
  command: bash -c "bin/blockscout eval \"Elixir.Explorer.ReleaseTasks.create_and_migrate()\" && bin/blockscout start"
  env_file:
    -  ./envs/common-blockscout.env
  environment:
      ETHEREUM_JSONRPC_VARIANT: 'geth'
      BLOCK_TRANSFORMER: 'clique'
      ETHEREUM_JSONRPC_HTTP_URL: rpc-node:8545
      DATABASE_URL: postgresql://postgres:@blockscout-db:5432/blockscout?ssl=false
      ECTO_USE_SSL: 'false'
  ports:
    - 4000:4000
```

Figure 8: Screenshot of Docker Compose Blockscout section

For the database used for Blockscout a PostgresSQL database based on an alpine linux image is used.

```
blockscout-db:
  image: postgres:14-alpine
  restart: always
  container_name: 'blockscout-db'
  command: postgres -c 'max_connections=250'
  environment:
      POSTGRES_PASSWORD: ''
      POSTGRES_USER: 'postgres'
      POSTGRES_HOST_AUTH_METHOD: 'trust'
```

Figure 9: Screenshot of Docker Compose Blockscout database section

Connecting to the BC4P Blockchain with MetaMask

MetaMask[31] is a crypto wallet, that supports a lot of different blockchain protocols. One of the supported ones is Ethereum Protocol and therefore we can use it to connect to the blockchain of this project.

To connect to the BC4P blockchain with MetaMask the following information must be specified inside MetaMask.
The important options are the URL of the RPC-Node, that is required to communicate with the blockchain, and the Chain ID, that is used to tell different chains of the same protocol apart from each other.

The following screenshot shows the network configuration in MetaMask:

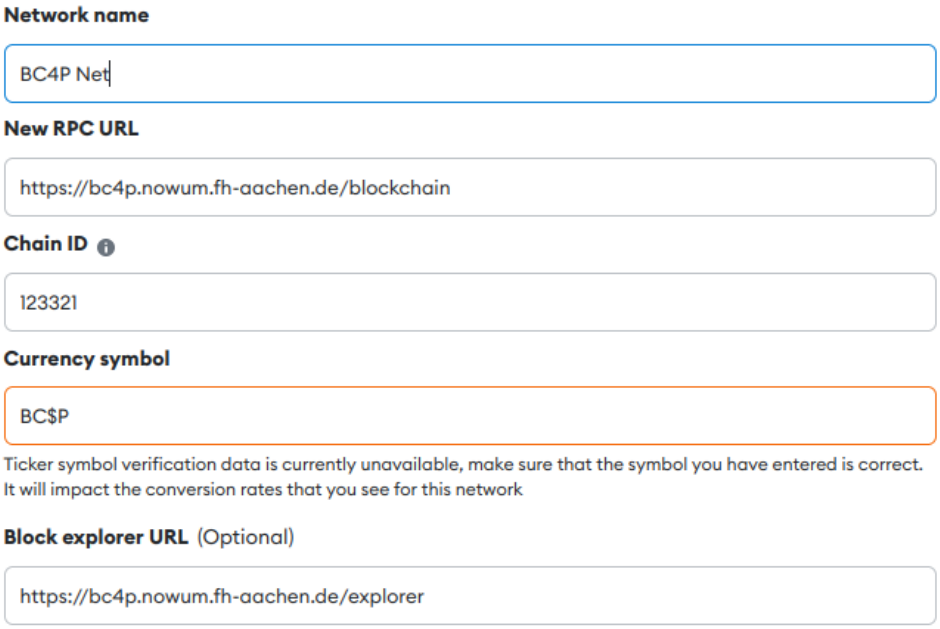

Figure 10: Screenshot of the MetaMask network configuration

Connecting to the BC4P Blockchain with Web3

Another way used inside this project to connect to the blockchain is programmatic via the Web3 Python library.[27] This library makes it possible the interact with blockchains based on etherum. The use of this library is well documented and easy to use.

Basic usage to interact with this is presented in three examples in the following.

The first step is to connect to the BC4P project chain or any other Ethereum based blockchain with the following code snippet:

```
w3 = Web3(Web3.HTTPProvider('https://bc4p.nowum.fh-aachen.de/blockchain'))
```

This creates a HTTP connection to the RPC-Node and enables the possibility to interact with the blockchain.

The entire range of possibilities of this library is well documented.[27]

The usual scenarios are to get the balances for specific addresses or create, sign and send transactions to the network.

Example1: Getting the latest Block

```
w3.eth.get_block('latest')
```

This retrieves the latest block which is available on the blockchain, it can be used to further inspect the blockchain and the transactions on it. A different block can retrieved to for inspection.

Example2: Getting Balance

```
address = '0x2d558F4633FF8011C27401c0070Fd1E981770B94'
w3.eth.get_balance(address)
```

This retrieves the total balance of the given public key/address at the latest block. This is generally calculated through aggregation of all existing transactions which involved the given public key.

Example3: Sending Transaction

While the first 2 examples are very simple requests to the blockchain, sending a transaction consists of multiple steps.

First, we need a nonce, so that replay attacks can be prevented.

Second, the transaction needs to be built. It consists of the nonce, the sender and recipient address, the amount of send funds and the transaction fee.

Third, the transaction must be signed with the private key of the sender.

Lastly, the transaction can be sent to the network and a hash will be returned, that can be used to check the status of the transaction later.

The full code example of the creation of a signed transaction which is send to the blockchain can be seen in figure 11.

```python
#get the nonce.  Prevents one from sending the transaction twice
nonce = web3.eth.getTransactionCount(account_1)

#build a transaction in a dictionary
tx = {
    'nonce': nonce,
    'to': account_2,
    'value': web3.toWei(1, 'ether'),
    'gas': 2000000,
    'gasPrice': web3.toWei('50', 'gwei')
}

#sign the transaction
signed_tx = web3.eth.account.sign_transaction(tx, private_key1)

#send transaction
tx_hash = web3.eth.sendRawTransaction(signed_tx.rawTransaction)

#get transaction hash
print(web3.toHex(tx_hash))
```

Figure 11: Example Code for sending transaction with web3 python

Token-Faucet

For this project a Token Faucet is implemented. It is only used inside the development stage of this project and allows everyone to mint tokens on the blockchain and send them to a given public wallet address.
This is implemented in a docker image which runs a python web server using gunicorn which issues a web3 transaction from a configured faucet wallet address, which contains a lot of preminted coins. The script sends tokens using the private key of one of the validators of the PoA-blockchain to the specified wallet address.
Furthermore, the image provides a small website, that allows users to enter their wallet address and execute the minting script.
The private key of the validator won't be exposed to the public with this solution.
In Figure 12 the Token Faucet website can be seen, which is used to retrieve tokens by inserting the public key of your wallet.

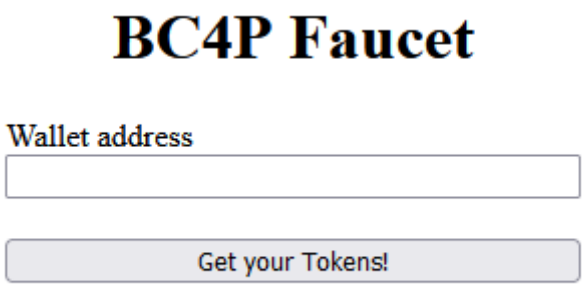

Figure 12: Screenshot of faucet used to transfer initial money for the test network

A REST call to get tokens would look like this:

```
requests.post(' https://bc4p.nowum.fh-aachen.de/faucet/api/add_key',
data= {'public_key': 0x2d558F4633FF8011C27401c0070Fd1E981770B94'})
```

The source code of the faucet is also available on GitHub.[28]

Application in P2P Simulation

The project provides a software stack, so that prosumers can trade their produced and consumed energy directly with each other's.
For this functionality infrastructure represented by the grid singularity framework (GSY-e) is provided. It offers a peer-to-peer energy exchange, which provides the functionality to the prosumers to execute trades.
These trades will be stored with the help of smart contracts inside the blockchain.
The goal and task of the blockchain is to work as a database, that provides makes it possible to trade locally generated energy between prosumers and other participants of the market.

P2P energy trading

The role of the blockchain inside the peer-to-peer (P2P) energy trading environment is to store all trades made between all participants.

The next part of the system is a platform, where participants can find energy trade partners and make trade offers. This P2P energy market needs to match the offers for different time frames and furthermore needs to ensure the stability of the real-world grid.

That includes the balance of production and consumption, the trading of balancing energy and minimizing of grid congestion.

In this project we are using a customized version of the open-source Grid Singularity framework for energy P2P trading, that is connected to the blockchain solution. Inside this framework scenarios for every pilot are defined. These scenarios consist out of information about grid, the producing and the consuming assets of the respecting pilot.

A scenario can describe a simulation of P2P trades over a customized timeframe in the past with data provided and stored inside Influx and PostgreSQL data bases from installed smart meters and other measurement devices at the pilots.

Furthermore, Grid Singularity can be used to provide a real time P2P trading platform for the pilot scenarios. In this case current energy production and consumption data can be provided from the pilot assets to the energy market, where trade offers get collected and matched for the next timeframe.

After matching the trades these get finalized through storing them inside the blockchain solution.

At this point funds get transferred. If the real-world consumption or production doesn’t match the traded amount for the specific timeslot a balancing transaction will be initialized from the P2P market.

Grid Singularity Exchange

Grid Singularity provides a facility to consume, trade, or share energy based on the consumer's preferences of an energy type, location source, price, or trading partner. As a result of Grid Singularity's application interface, households and distributed energy assets are digitally represented by trading agents, and grid operators can integrate to facilitate a bottom-up market design.

Grid singularity exchange enables prosumers and consumers organized in energy communities, with the support of local aggregators, to simulate and implements peer-to-peer and community energy trading through the creation of active local energy markets.

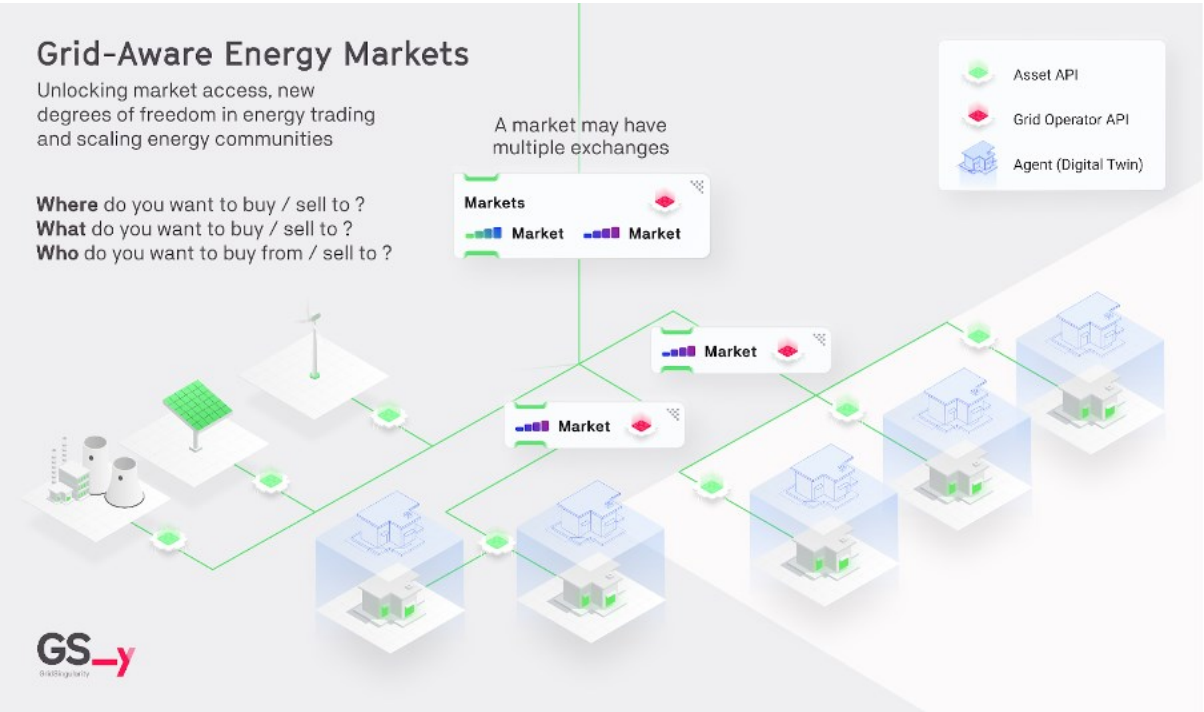

Figure 13: Grid Singularity framework overview of a part of an energy grid connected to the energy market

Conclusion and Future Work

This report aims to investigate the implementation of blockchain technology in the realm of energy trading. Specifically, the target is to test a blockchain solution, for its suitability for usage in energy markets and its limitations. The deployed Ethereum blockchain will be used as base for the smart contract development outlined in the deliverable “WPT2.2 Smart Contract”. Furthermore, the development of the user interfaces in the deliverable “WPT2.3 Development of User Interfaces – Platforms and Apps” connect to this blockchain for visualizations. Additionally, the deliverable “WPT3.1 Digital Identities” describes a digital identity management system based on smart contracts, that are stored inside this blockchain, too.

The project as a whole utilizes Gridsingularity Exchange framework, that provides the infrastructure to execute peer-to-peer trades between prosumers, by simulating energy trading with respect to locality.
The responsibility of the blockchain is to store these trades, but in comparison to traditional crypto currencies the energy market is regulated and not everyone can participate without the connection to the real-world energy grid – as our energy grid is part of critical infrastructure and our society depends on the proper functioning of it.
Furthermore, this grid is operated by grid operators, that must ensure the safety of the net. This means, that trades on the blockchain must mirror real energy consumption, which requires the use of smart meter solutions, that are verified by the energy operator. Otherwise, traders can announce energy consumption or generation which did not happen in the real world.
This results in the need of the verification of the prosumer itself as customer of at the grid operator.
These restrictions make the use of a public blockchain solution impossible.
In addition to that, the high energy consumption of Proof-of-Work algorithms, that are commonly used in public blockchains, would negate all positive impacts of a more decentralized energy production and consumption on the energy usage in total.

Therefore, a permissioned Ethereum-based blockchain solution was implemented and deployed for the research project. The Clique consensus protocol is a Proof of Authority mechanism, that can be operated by the grid operator or another alliance of trusted and verified third parties.
The operational complexity of such a system is much higher than using a centralized database cluster which also hosted in the grid operators responsibility.

Future Work needs to involve ensuring the scalability of the system for a larger scale – as the European energy market provides market access to multiple hundred agents which can trade in real-time at the market. The amount of p2p energy trades generated from

thousands of participants would generate a large amount of data and traffic, that the blockchain solution needs to be able to handle in real time to ensure grid safety.
Yet it still needs further research to work out a scenario where the blockchain solution shows clear benefits in contrast to a central data storage using a database cluster or a data stream solution like Apache Kafka.

References

1. Grid Singularity. https://github.com/gridsingularity. GitHub.

2. Vitalik Buterin. Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. 2014. URL: https://ethereum.org/669c9e2e2027310b6b3cdce6e1c52962/Ethereum_Whitepaper_-_Buterin_2014.pdf

3. Xu, X., Weber, I., Staples, M., Zhu, L., Bosch, J., Bass, L., Pautasso, C., Rimba, P.: A taxonomy of blockchain-based systems for architecture design. In: 2017 IEEE International Conference on Software Architecture (ICSA), pp. 243–252. IEEE (2017)

4. Aitzhan, N.Z., Svetinovic, D.: Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams. IEEE Trans. Depend Secure Comput. **15**(5), 840–852 (2016)

5. Kang, J., Yu, R., Huang, X., Maharjan, S., Zhang, Y., Hossain, E.: Enabling localized peer-to-peer electricity trading among plug-in hybrid electric vehicles using consortium blockchains. IEEE Trans. Ind. Inform. **13**(6), 3154–3164 (2017)

6. Tschorsch, F., Scheuermann, B.: Bitcoin and beyond: a technical survey on decentralized digital currencies. IEEE Commun. Surv. Tutor. **18**(3), 2084–2123 (2016)

7. Mohanta, Bhabendu Kumar, Soumyashree S. Panda, and Debasish Jena. "An overview of smart contract and use cases in blockchain technology." *2018 9th international conference on computing, communication and networking technologies (ICCCNT)*. IEEE, 2018.

8. Kirli, Desen, et al. "Smart contracts in energy systems: A systematic review of fundamental approaches and implementations." *Renewable and Sustainable Energy Reviews* 158 (2022): 112013.

9. Kounelis, Ioannis, et al. "Fostering consumers' energy market through smart contracts." *2017 International Conference in Energy and Sustainability in Small Developing Economies (ES2DE)*. IEEE, 2017.

10. Iqbal, R., Butt, T.A.: Safe farming as a service of blockchain-based supply chain management for improved transparency. Clust. Comput. **23**, 2139–2150 (2020)

11. Korpela, K., Hallikas, J., Dahlberg, T.: Digital supply chain transformation toward blockchain integration. In: Proceedings Of The 50th Hawaii International Conference on System Sciences (2017)

12. Kshetri, N.: Blockchain's roles in meeting key supply chain management objectives. Int. J. Inf. Manag. **39**, 80–89 (2018)

13. Latif, R.M.A., Farhan, M., Rizwan, O., Hussain, M., Jabbar, S., Khalid, S.: Retail level blockchain transformation for product supply chain using truffle development platform. Clust. Comput. (2020). https://doi.org/10.1007/s10586-020-03165-4

14. Gunarathna, Chathuri Lakshika, et al. "Reviewing global peer-to-peer distributed renewable energy trading projects." *Energy Research & Social Science* 89 (2022): 102655.

15. Liu, Jia, Hongxing Yang, and Yuekuan Zhou. "Peer-to-peer energy trading of net-zero energy communities with renewable energy systems integrating hydrogen vehicle storage." *Applied Energy* 298 (2021): 117206.

16. Nakamoto, S. (2008) Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf

17. Saleh, Fahad (2021-03-01). "Blockchain without Waste: Proof-of-Stake". The Review of Financial Studies. **34** (3): 1156–1190. doi:10.1093/rfs/hhaa075. ISSN 0893-9454.

18. Gavin, Wood (November 2015). "PoA Private Chains". GitHub.

19. Diego Ongaro and John Ousterhout. 2014. In search of an understandable consensus algorithm. In Proceedings of the 2014 USENIX conference on USENIX Annual Technical Conference (USENIX ATC'14). USENIX Association, USA, 305–320.

20. Caldarelli, Giulio. "Understanding the blockchain oracle problem: A call for action." *Information* 11.11 (2020): 509.

21. Bitcoin Community. https://github.com/bitcoin. GitHub.

22. Ethereum Community. https://github.com/ethereum/go-ethereum. GitHub

23. Hyperledger Fabric Community. https://github.com/hyperledger/fabric. GitHub

24. EnergyWe Foundation Community. https://github.com/energywebfoundation?q=&type=all&language=&sort=stargazers . GitHub

25. Go-ethereum. https://geth.ethereum.org/

26. Szilágyi, Péter. EIP-225: Clique proof-of-authority consensus protocol. 2017. https://eips.ethereum.org/EIPS/eip-225

27. Web3.py Documentation. https://web3py.readthedocs.io/en/v5/

28. Haas, Maurer. PoA Chain Setup Configuration. https://github.com/BC4P/poa-chain

29. Bhaskar S. Proof of Authority Consensus – Clique. https://www.polarsparc.com/xhtml/Clique.html

30. Blockscout Documentation. https://docs.blockscout.com/

31. MetaMask Documentation. https://docs.metamask.io/guide/