

CONTROL AND AUTHENTICATION OF A FULLY
DECENTRALIZED PAY-PER-USE ENERGY TRADING
PLATFORM

BY

MERRAD YACINE

A thesis submitted in fulfillment of the requirement for the
degree of Doctor of Philosophy in Engineering

Kulliyyah of Engineering
International Islamic University Malaysia

FEBRUARY 2023

ABSTRACT

This research aims to solve some problems related to blockchain peer-to-peer energy trading, which are purchased energy under consumption and transaction authentication and demand response control, which still rely on centralized schemes. The purpose of this thesis is to control and authenticate a fully Decentralized pay-per-use energy trading platform. It presents a blockchain-based peer-to-peer (P2P) energy trading platform where prosumers can trade energy autonomously and without interference from a central authority. Multiple prosumers can collaborate on energy generation to form a single supplier. Customers' electricity consumption is monitored via a smart meter connected to an IoT node connected to a private blockchain network. Smart contracts invoked on the blockchain enable autonomous trading interactions between parties and govern the behavior of accounts within the Ethereum state. The decentralized P2P trading platform uses autonomous usage-based billing and energy routing monitored by a smart contract. A Deep Learning-based Gated Recurrent Unit (GRU) model predicts future consumption based on past data collected on the blockchain. The predictions are then used to set Time of Use (ToU) ranges using the K-means cluster. The data used to train the GRU model is shared among all parties within the network, making the predictions transparent and verifiable. By implementing K-Mean clustering in a smart contract on the blockchain, the set of ToU is independent and unchallengeable. To ensure the validity of the data uploaded to the blockchain, a consensus algorithm is proposed to detect fraudulent nodes, along with a Proof of Location (PoL) to ensure that the data is uploaded by the expected nodes. To address the conflict of interest between prosumers and distribution system operators (DSOs) in decentralized P2P energy trading platforms, where prosumers seek to maximize their profit on the one hand, while DSOs seek optimal power flow (OPF) on the other hand, a novel fully decentralized architecture is proposed for an OPF-based demand response management system that uses smart contracts to force generators into compliance without the need for a central authority or hardware. The study details the proposed platform architecture, operation, and implementation. The results are presented mainly in terms of gas consumption of smart contracts and transaction latency for different loads. The work presented in the thesis is relevant in the sense that we have attempted to address a popular and important contemporary research issue related to blockchain technology. The research work holds great potential for industrial use. In the future era of renewable energy, decentralization of energy trading is a necessity for the future society.

خلاصة البحث

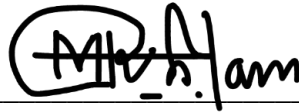
يهدف هذا البحث إلى حل بعض المشكلات المتعلقة بتداول الطاقة من نظير إلى نظير (P2P) على البلوكتشين (blockchain)، والتي تتمثل في عدم الاستهلاك الكلي للطاقة التي تم شراؤها، ومصادقة المعاملات والتحكم في استجابة الطلب، والتي لا تزال تعتمد على طرق مركزية. الغرض من هذه الأطروحة هو التحكم والمصادقة على منصة تداول الطاقة اللامركزية بالكامل للدفع على حسب الاستخدام. إنه يقدم منصة تداول الطاقة من نظير إلى نظير القائمة على بلوكتشين حيث يمكن للمستهلكين تداول الطاقة بشكل مستقل ودون تدخل من أي سلطة مركزية. يمكن للمستهلكين المتعددين التعاون في توليد الطاقة لتشكيل مزود واحد. تتم مراقبة استهلاك الكهرباء للعملاء عبر عداد ذكي متصل بعقدة إنترنت الأشياء المتصلة بشبكة بلوكتشين خاصة. تتيح العقود الذكية والتي يتم استدعاؤها على بلوكتشين تفاعلات تجارية مستقلة بين الأطراف وتحكم سلوك الحسابات داخل حالة إيثيريوم (Ethereum). تستخدم منصة التداول اللامركزية P2P الفوترة المستندة إلى الاستخدام المستقل وتوجيه الطاقة الذي يتم مراقبته بواسطة عقد ذكي. يتنبأ نموذج الوحدة المتواترة (GRU) القائم على التعلم العميق بالاستهلاك المستقبلي بناءً على البيانات السابقة التي تم جمعها على البلوكتشين. تُستخدم التنبؤات بعد ذلك لتعيين نطاقات وقت الاستخدام (ToU) باستخدام تجميع K-mean. تتم مشاركة البيانات المستخدمة لتدريب نموذج GRU بين جميع الأطراف داخل الشبكة، مما يجعل التنبؤات شفافة ويمكن التحقق منها. من خلال تنفيذ تجميع K-Mean في عقد ذكي على بلوكتشين، تكون مجموعة ToU مستقلة وغير قابلة للطعن. لضمان صحة البيانات التي تم تحميلها على بلوكتشين، تم اقتراح خوارزمية إجماع للكشف عن العقد الاحتمالية، إلى جانب إثبات الموقع (PoL) لضمان تحميل البيانات بواسطة العقد المتوقعة. لمعالجة تضارب المصالح بين المستهلكين ومشغلي أنظمة التوزيع (DSOs) في منصات تداول الطاقة اللامركزية P2P، حيث يسعى المستهلكون إلى تعظيم أرباحهم من ناحية، بينما يسعى DSOs للحصول على تدفق الطاقة الأمثل (OPF) من ناحية أخرى، لهذا تم اقتراح بنية لا مركزية بالكامل لنظام إدارة استجابة الطلب المستند إلى OPF الذي يستخدم العقود الذكية لإجبار المولدات على الامتثال دون الحاجة إلى سلطة أو أجهزة مركزية. توضح الدراسة تفاصيل بنية المنصة المقترحة وتشغيلها وتنفيذها. يتم عرض النتائج بشكل رئيسي من حيث استهلاك الغاز للعقود الذكية وزمن انتقال المعاملات للأحمال المختلفة. العمل المقدم في الأطروحة ذو أهمية لأننا حاولنا معالجة قضية بحث معاصرة شائعة وهامة تتعلق بتكنولوجيا البلوكتشين. العمل البحثي يحمل إمكانات كبيرة للاستخدام الصناعي. في عصر الطاقة المتجددة في المستقبل، تعتبر اللامركزية في تجارة الطاقة ضرورة للمجتمع المستقبلي.

APPROVAL PAGE

The thesis of Merrad Yaçine has been approved by the following:



Mohamed Hadi Habaebi
Supervisor



Md Rafiqul Islam
Co-Supervisor



Teddy Surya Gunawan
Co-Supervisor

Mashkuri Yaacob
Internal Examiner

Suhaidi Hassan
External Examiner

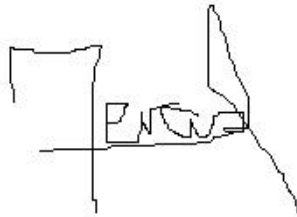
Chairman

DECLARATION

I hereby declare that this thesis is the result of my own investigations, except where otherwise stated. I also declare that it has not been previously or concurrently submitted as a whole or any part of for any other degrees at IIUM or other institutions.

Merrad Yaçine

Signature

A handwritten signature in black ink, appearing to read 'Merrad Yaçine', written over a horizontal line.

Date: 1st February 2023

INTERNATIONAL ISLAMIC UNIVERSITY MALAYSIA

**DECLARATION OF COPYRIGHT AND AFFIRMATION
OF FAIR USE OF UNPUBLISHED RESEARCH**

**CONTROL AND AUTHENTICATION OF A FULLY
DECENTRALIZED PAY-PER-USE ENERGY TRADING
PLATFORM**

I declare that the copyright holders of this thesis are jointly owned by the student and IIUM.

Copyright © 2023 Merrad Yaçine International Islamic University Malaysia. All rights reserved.

No part of this unpublished research may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise without prior written permission of the copyright holder except as provided below

1. Any material contained in or derived from this unpublished research may be used by others in their writing with due acknowledgement.
2. IIUM or its library will have the right to make and transmit copies (print or electronic) for institutional and academic purposes.
3. The IIUM library will have the right to make, store in a retrieved system and supply copies of this unpublished research if requested by other universities and research libraries.

By signing this form, I acknowledged that I have read and understood the IIUM Intellectual Property Right and Commercialization policy.

Affirmed By Merrad Yaçine

Signature



Date: 1st February 2023

ACKNOWLEDGEMENTS

Indeed, all praise is due to Allah, we praise Him, we seek His aid, and we ask for His forgiveness. We seek refuge in Allah from the evil of our actions and from the evil consequences of our actions. Whom Allah guides, no one can misguide, and whom Allah misguides, no one can guide. I bear witness that there is no god worthy of worship except Allah, and I bear witness that Muhammad is the servant and messenger of Allah.

I would like to express my sincere appreciation to my supervisor. This thesis would not have been possible without the kind support of my supervisor: Professor Mohamed Hadi Habaebi.

I thank you, Professor, for your constant advice, patience and energy until the completion of this work. Despite the difficult conditions created by the worldwide pandemic of Covid 19, I would like to thank you for your support and warm friendship, which definitely helped me to complete this work.

Many thanks to my co-supervisors Prof. Dr. Md. Rafiqul Islam and Prof. Dr. Teddy Surya Gunawan for their guidance and support.

To the members of my supervisory committee, thank you for the guidance offered.

I am also very grateful to my entire family, especially my dear mother and grandmother and my father for their love and encouragement. Many thanks to them for always accompanying me during my childhood and growing up, teaching me morals and ethics as precious jewels that will remain forever.

It is also a good opportunity to sincerely thank my dear uncle for his continuous support, both moral and financial.

Finally, I would like to express my best feelings to all my teachers since elementary school and convey my warmest greetings to the academic and administrative staff of IIUM.

A big thank you to all my friends.

TABLE OF CONTENTS

Abstract	ii
Abstract in Arabic	iii
Approval Page	iv
Declaration Page	v
Copyright Page	vi
Acknowledgements.....	vii
List of Tables.....	xi
List of Figures	xiii
List of Abbreviations.....	xvii
List of Symbols.....	xix
CHAPTER ONE: INTRODUCTION	1
1.1 Background.....	1
1.2 Problem statement	5
1.3 Research hypotheses	6
1.4 Research motivation	8
1.5 Research objectives	8
1.6 Research methodology.....	9
1.7 Research scope.....	12
1.8 Thesis Outline.....	12
CHAPTER TWO: LITERATURE REVIEW	15
2.1 Introduction.....	15
2.2 Overview of Blockchain technology	15
2.2.1 Blockchain concept as technology behind decentralized cryptocurrency	16
2.2.2 Blockchain beyond cryptocurrency	18
2.2.3 Blockchain for decentralized applications and software	19
2.2.4 Blockchain in the industrial world	20
2.2.5 Smart contracts.....	23
2.2.6 Blockchain consensus algorithms	26
2.2.6.1 Consensus in permission-less Blockchains	27
2.2.6.2 Consensus in permissioned-blockchains	47
2.2.6.3 Blockchain consensus performance metrics.....	52
2.2.6.4 Drawbacks of current trend in consensus algorithm development	56
2.3 Current P2P energy trading tokens platforms	60

2.3.1	Power Ledger	60
2.3.2	Grid+ (<i>Grid+ Whitepaper</i> , 2021).....	61
2.3.3	LO3 Energy (<i>Microgrid Knowledge</i> , 2021)	62
2.4	PoL to secure blockchain applications	69
2.5	Demand response control in P2P energy trading system	73
CHAPTER THREE: RESEARCH METHODOLOGY		77
3.1	Introduction.....	77
3.2	Research framework.....	77
3.2.1	Proposed pay-per-use blockchain P2P energy trading platform	78
3.2.1.1	Platform design.....	84
3.2.2	Proposed PoL to secure blockchain energy trading transactions.....	114
3.2.2.1	Traditional centralized PoL.....	115
3.2.2.2	PoL decentralized version	123
3.2.3	Proposed, fully decentralized, cost-effective P2P energy demand response management system, with smart contract-based OPF solution	153
3.2.3.1	On-chain based solution for a 3-bus network, using DC-OPF approximation	157
3.2.3.2	Enhanced decentralized OPF solving, generalized for any problem formulated	178
3.3	Experimental setup	190
3.3.1	Proposed pay per use blockchain P2P energy trading platform.....	191
3.3.1.1	Energy onsumption Time Series Forecasting Using GRU	191
3.3.1.2	Definition of ToU Ranges Using On-Chain Clustering	192
3.3.1.3	Modified PoW consensus, adapted for the platform	193
3.3.1.4	Pay per use P2P energy trading platform evaluation	196
3.3.2	Decentralized PoL scheme	197
3.3.3	Smart contract-based OPF system for energy demand response management	198
CHAPTER FOUR: RESULTS AND ANALYSIS		201
4.1	Introduction.....	201
4.2	Pay-per-use P2P energy trading platform	201

4.2.1	ToU ranges using GRU model predictions and K-Mean clustering	201
4.2.1.1	Energy consumption time series prediction with GRU performance.....	201
4.2.1.2	Definition of ToU ranges with on-chain clustering execution cost	202
4.2.2	Modified PoW adapted for the P2P energy trading platform	203
4.2.3	Execution cost and response time of smart contract transactions on the proposed P2P energy trading platform ..	204
4.2.4	Discussion of the results	206
4.3	Proposed decentralized PoL scheme for blockchain energy trading platforms	207
4.3.1	Determination of PoL execution cost and response time	207
4.3.2	Discussion of the results	208
4.4	Decentralized smart contract-based OPF.....	209
4.4.1	Decentralized smart contract-based OPF execution costs ...	209
4.4.2	Discussion of the Results	211
CHAPTER FIVE: CONCLUSION AND RECOMMENDATION		214
5.1	Conclusion	214
5.2	Novelty	216
5.3	Contribution	217
5.4	Recommendation.....	219
REFERENCES		221
LIST OF PUBLICATIONS		245

LIST OF TABLES

2.1	Proof of Stake-based algorithms.	39
2.2	Top 10 Cryptocurrencies by Marketcap and the Corresponding Consensus Protocols Used, 2021.	39
2.3	Hashing rate, mining time and orphan blocks rate for different blockchain platforms.	42
2.4	Comparison of Blockchain consensus protocols for a set of essential Blockchain properties.	46
2.5	Mining Pools Contribution in Bitcoin (From February 1st to February 4th, 2022)	55
2.6	Mining pools contribution in Ethereum, (From February 1st to February 4th, 2022)	56
2.7	Blocks are not filled to their full authorized potential despite still having pending transactions left over, in the Mempool.	59
3.1	Smart contracts' events in the proposed platform.	109
3.2	Strategy played versus gain in the game theory scenario between DSO and non-DSO provers, where $(x_t, y_t) = (x^*, y^*)$.	146
3.3	Strategy played versus gain in the game theory scenario between DSO and non-DSO provers, where $(x_t, y_t) \neq (x^*, y^*)$.	146
3.4	Distance coverage for different wireless communication technologies.	153
3.5	Generators cost coefficients and generation capacity.	166
3.6	Transmission lines parameters.	167

3.7	Hourly load power demand for 24 hours.	176
3.8	3-bus example, DC-OPF solution.	177
3.9	The gain for each player according to his played strategy.	180
3.10	Tthreshold for different sizes of mining sets.	193
4.1	Gas consumption.	202

LIST OF FIGURES

1.1	Blockchain in peer to peer energy trading	5
1.2	Flowchart of the adopted methodology research	11
2.1	The Censor-proof property in blockchain-based apps and software.	20
2.2	Transaction input and output formats in Bitcoin.	25
2.3	National energy use in TW/h. University of Cambridge Bitcoin Electricity Consumption Index (Criddle, 2021).	31
2.4	The most extended chain-win principle in Blockchain.	31
2.5	Selfish mining	33
2.6	Validator voting process in Casper PoS.	36
2.7	Increasing Importance Score in NEM PoI by contributing in the network.	38
2.8	Percentage of Orphaned blocks per day versus block mining power consumption in blockchain-based cryptocurrencies.	43
2.9	PBFT consensus reaching with three lieutenant nodes.	51
2.10	Validators voting in stellar consensus protocol.	52
2.11	Evolution of market shares and domination for the top 25 mining pools.	60
3.1	Residential PV electrical generation potential versus the total national electricity generation capacity in the EU.	82
3.2	The proposed model defining TOU peak and off-peak ranges.	85

3.3	Distribution of the observations in the dataset.	86
3.4	The K-mean algorithm flowchart.	95
3.5	ERD diagram of the proposed platform.	99
3.6	Time dependency of the Provider smart contract state.	103
3.7	The proposed P2P Energy trading sequence diagram.	108
3.8	Interactions between the entities of the proposed P2P trading Platform.	110
3.9	State diagram.	111
3.10	Trilateration technique.	117
3.11	The itinerary followed.	119
3.12	Mapping diagram illustrating how each consumer node is mapped to its own distance computation model in the Anchor server node.	120
3.13	Standard deviation for device 1.	121
3.14	Standard deviation for device 2.	121
3.15	Pathloss exponent for device 1 deduced using best fitting curve.	122
3.16	Pathloss exponent for device 2 deduced using best fitting curve.	122
3.17	Flowchart illustrating the process by which potential prover nodes apply for location verification for a given PoL request from the target node.	129
3.18	Process by which the DSO sends an encrypted number, then reveals it, and how it is verified by the location verifier smart contract.	133
3.19	Random number generation process by the location verifier smart contract.	135

3.20	Game theory scenario in target node positioning between DSO and Non-DSO anchors.	147
3.21	Entity relationship diagram of the location verifier smart contract.	150
3.22	Transmission line model in DC-OPF.	159
3.23	3-bus power network.	166
3.24	Solving DC-OPF problem.	169
3.25	Flowchart illustrating how parties in the smart grid that do not comply with the OPF solution are not rewarded for their energy contribution.	175
3.26	Proposed OPF solution scheme explained using a card betting game.	179
3.27	Transmission line π model.	181
3.28	Entity-relationship diagram for the enhanced OPF solution model.	190
3.29	Threads running concurrently on CPU cores, simulating miners competing to add a new bloc.	195
3.30	Gas consumption for Ethereum transaction execution.	199
3.31	Hourly gas price fluctuation in US Dollars for 15/05/2022.	200
4.1	GRU prediction model performance evolution as the trading rounds go by.	202
4.2	Miners involvement in forks using the proposed consensus protocol.	203
4.3	Transaction's latency with increasing send rates for different numbers of consumer node.	204
4.4	Gas consumption for Provider Smart Contract.	205

4.5	DSO Smart Contract Gas consumption.	205
4.6	Transaction's latency with increasing send rates for different numbers of target node.	208
4.7	Location verifier smart contract gas consumption.	208
4.8	On-chain 3-bus DC-OPF solution, execution cost in US Dollars, according to gas price fluctuation in 15/05/2022.	210
4.9	Execution cost of 14-bus AC-OPF solution, cost in US dollars, depending on gas price fluctuation on 15/05/2022, using the proposed scheme.	210
4.10	Comparison of execution costs between on-chain OPF solution for a 3-bus DC-OPF problem and the solution for a 14-bus AC-OPF problem using the enhanced proposed model.	212
4.11	Hourly minimum wage for different industrialized countries.	213

LIST OF ABBREVIATIONS

Aps	Access Points.
BG	Byzantine Generals.
CRNs	Cognitive Radio Networks.
DER	Distributed Energy Resources.
DLT	Distributed Ledger Technology.
DPoS	Delegated Proof of Stake.
DSO	Distribution System Operator.
ERD	Entity Relationship Diagram.
ETH	Ethers.
FBA	Federated Byzantine Agreement.
GPS	Global Positioning System.
GRU	Gated Recurrent Unit.
HR	Hash Rate.
IDE	Integrated Development Environment.
IoT	Internet of Things.
LBS	Location Based Services.
LP	Location Proof.
MAPE	Mean Absolute Percentage Error.
NEM	National Electricity Market.

NIST	National Institute of Standards and Technology.
OPF	Optimal Power Flow.
P2P	Peer-to-Peer.
PBFT	Practical Byzantine Fault Tolerance.
PL	Proof of Location.
PoB	Proof of Burn.
PoET	Proof of Elapsed Time.
PoI	Proof of Importance.
PoL	Proof of Location.
PoS	Proof of Stake.
PoW	Proof of Work.
PQC	Post-Quantum Cryptography.
RSA	Rivest-Shamir-Adleman.
RSSI	Received Signal Strength Indication.
SCP	Stellar Consensus Protocol.
SHA 256	Secure Hash Algorithm 256-bit.
T&D	Transmission and Distribution.
TEE	Trusted execution environment.
ToU	Time of Use.
UTXO	Unspent Transaction Output.

LIST OF SYMBOLS

CN_i	Consumer node i.
I_i	Current at node i.
K_j^{Pr}	Node j private key.
P_{G_i}	Active power injected at bus i.
P_{L_i}	Power load demand at bus i.
P_{ij}	Active power flowing in transmission line between nodes i and j.
Pl_0	Path-loss at a reference distance of $1m$.
Pl_T	Wi-Fi signal power transition.
$Pol_Req_{i \rightarrow j}$	Target node i requests location proof from anchor node j.
Q_{G_i}	Reactive power injected at bus i.
Q_{ij}	Reactive power flowing in transmission line between nodes i and j.
R_{cst_i}	Independent constant in linear equation.
S_i	Net power injected at bus i.
$S_{i,j}$	Power flow between nodes i and j in a power network.
V_i	Voltage at node i.
W_c	Block mining power consumption.
X_{ij}	Reactance of transmission line between nodes i and j.
δ	Standard deviation (Section 2.1.2 in Chapter 3).
δ_i	Phase voltage angle.

γ	Path-loss exponent (Section 2.1.2 in Chapter 3).
λ	Lagrangian coefficients for different constraint functions (Section 2.1.3 In Chapter 3).
λ	Wave length of the emitted signal (Section 2.1.2 in Chapter 3).
a_i	Bus i generation cost quadratic function coefficients.
b_{ij}	Suceptance of transmission line between nodes i and j.
y_{ij}	Admittance of transmission line between nodes i and j.
y_{sh_i}	Shunt suceptance at bus i.
B	Bus suceptence matrix.
D	Current Mining Difficulty..
Gw	Gigawatt..
L	Lagrange function.
ND	Number of Devices used to mine a block..
P	Wattage of the mining device used.
TAv	Block average mining time.
Y	Bus admittance matrix.

CHAPTER ONE

INTRODUCTION

1.1 Background

Blockchain is the underlying technology of the most potential applications which has been gaining traction in recent times (Salimitari et al., 2020), (Bouraga, 2021), (J. Zhang et al., 2020). Blockchain network is a decentralized, distributed ledger technology where all nodes must agree on the validity of the ledger they are sharing, to ensure that any data included are perfectly valid; then, hence, after being committed to the blockchain, it is impossible to delete, deny or tamper with (Nofer et al., 2017). Blockchain is a transformation of the Byzantine Generals (BG) Problem, where at least two third of the network should be willing to maintain the integrity of the network so that impact of malicious parties can be nullified (Lamport et al., 2019). In order to agree on the new data (blocks to be added to the chain) and ensure trust in the system, the nodes must rely upon a consensus protocol. These algorithms are the backbone of blockchain, which in several ways determine the performance that such a system can achieve. Up to now, increasing attention has been paid with regard to many consensus protocols, and different cryptocurrencies implemented different protocols. Consensus algorithms in blockchain have been subjected to numerous research studies and are still opened to

new ones. Despite the fact that the proposed algorithms endeavored to strengthen some of the existing weaknesses, they will witness new constraints. Some consensus protocols are intensively robust, while others will also lead towards centralization and trust issues whilst blockchain was created to avoid it (Monrat et al., 2019).

In a world where energy technologies and battery storage systems are rapidly improving, and the sharing economy seems to be disrupting every industry, many consumers are asking, why can't we trade energy? Peer-to-peer (P2P) energy trading looks to address just that question by enabling people to buy and sell energy to each other directly.

One of the main hurdles that prevents small-scale producers (prosumers) of renewable energy from engaging in exchange is regulation from the National Electricity Market (NEM). NEM requires that vendors on their market have a generator larger than 5 megawatts, which is equivalent to 5,000 5kW solar systems. If the cost of a 5kW system is approximately USD 10,000, then the cost of a system large enough to meet the NEM's minimum would be USD 50 million. This is significantly more than the vast majority of households can afford, creating a huge barrier to entry for homeowners with a small array of solar panels. However, with P2P energy trading, everyone from a 1.5kW solar system homeowner to the biggest coal-fired power plant can engage in peer-to-peer energy exchange.

On top of lowering the barrier to entry for many consumers, P2P energy trading

would allow users to purchase energy from specific sources. This means that it would be possible to buy solar energy or energy produced specifically by your neighbor. By enabling P2P commerce, prosumers will be able to generate revenue on their excess energy and consumers will be able to obtain transparently-sourced, reliable energy. The hope is that by cutting out the middleman, prices will go down and more people may be incentivized to install solar panels and other types of renewable energy generators. Blockchain in P2P energy trading is illustrated in Figure 1.1 (Saxena, 2019)

There are three main options for P2P energy trading:

- Trading through the power grid: in this option, a consumer remains associated with a central power grid and administer price and volume risk in an independent fashion, by selling or buying energy directly to other parties. The main benefits are the low upfront investment required by the P2P trader, the savings in subscription premiums billed by an energy retailer, the mindfulness and control of the consumption profile (peak shaving), the 100% certainty of purchasing green energy, and finally, the reduced vulnerability to market fluctuations as seen in the fall of 2021.
- Partially independent microgrid: In this constellation, participants build a microgrid that manages part of the total energy demand, but remains connected to the central grid for a remaining portion of the required capacity. An example of this is the Schoonschip Amsterdam microgrid. In this community-managed microgrid, electricity is traded among residents while the community remains connected to the grid for redundancy.

This encourages participants to balance each other's profiles and reduces the need for centralized grid investment. The investment cost for the parties involved is higher than trading over the grid, but the savings are also much higher. One trend is that partially independent microgrids are emerging in industrial clusters. For example, P2P platforms have been established in shipping ports around the world, such as the Port of Rotterdam and the Port of San Diego.

- Fully independent microgrid: In this constellation, multiple participants create a private, fully self-sufficient power grid that is not connected to the central power grid, and the parties transact through a P2P platform. In this case, the investment for the participants in the initiative, as well as the associated savings, are higher than in the other two concepts. However, given the current cost of energy storage and the maturity of microgrid technology, it is unlikely that this concept will catch on. It is more likely that consumers will first explore grid trading and partial independence from the electric grid in the coming years until the technology matures enough for this concept to become mainstream. The industry of P2P trading platforms is quite young but some players have already reached substantial communities of peers. An example of a P2P energy trading platform that can enable the three set-ups is *ENTRNCE* (2019). The platform was introduced in 2017 and currently handles transactions for over 10.000 peers on a daily basis. *ENTRNCE* (2019) facilitates all three set-ups and is an independent and fully automated (electricity) transaction platform for direct transactions between producers and

consumers, regardless of their location.

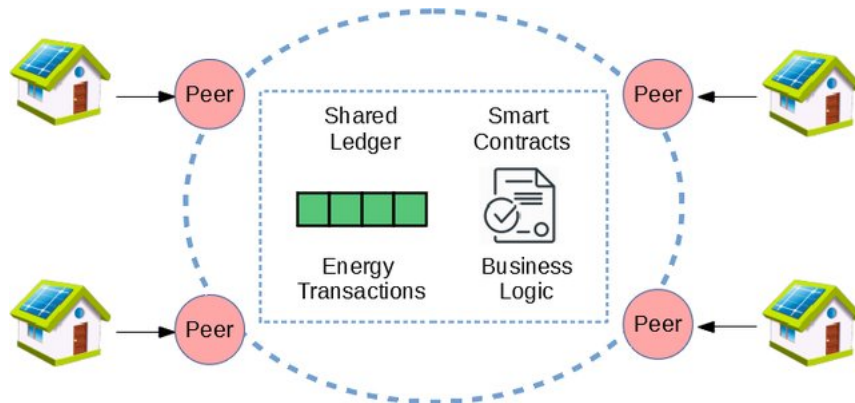


Figure 1.1: Blockchain in peer to peer energy trading

1.2 Problem statement

Based on related work, we can define the following research gaps that this research aims to address. First, purchased energy under consumption. In other words, if the energy is redeemed with the corresponding energy token, consumers will not be reimbursed if they underconsume. Second, transaction centralized authentication in blockchain peer to peer energy trading. Since the identity of the transaction issuer is defined in the software running on IoT nodes and smart meters and is thus easily accessible, private key encryption cannot guarantee the identity of the transaction issuer in such cases. Even if it is closed-source code, it is not secret to the developer of the code, who in turn can share it with others. This makes blockchain-based P2P energy trading platforms vulnerable to identity thieves, who could compromise the reliability of the platform and the execution of transactions. Finally, blockchain P2P energy trading platforms centralized

demand response management. Blockchain P2P energy trading platforms face a conflict of interest between grid operators and prosumers, as prosumers seek to maximize their profits on the one hand, while grid operators seek an optimal OPF operating point on the other. Due to the complexity of formulating and solving OPF problems in the presence of renewable energy sources, researchers have focused on mathematical modeling and effective solution algorithms for such optimization problems. However, the control of power generation according to a defined OPF solution is still based on centralized control and management entities owned by the DSO. This contradicts the philosophy of blockchain applications that do not rely on a central authority. All presented problems are described in detail in the literature review chapter.

1.3 Research hypotheses

Our hypothesis states that a different approach than the established token-based approach for energy trading platforms would solve the problem of energy under-consumption in current P2P energy trading platforms. In this novel approach, prosumers cooperatively feed energy into the grid and establish a private provider. The injected energy is accounted for as the prosumer's energy balance and is maintained and updated accordingly in a smart contract. The prosumer's balance is considered a debt that must be repaid by the DSO. Accordingly, the DSO delivers to the subscribed consumers on behalf of the cooperating prosumers for a continuous period of time. This would allow consumers to be billed according to their energy consumption through a prepaid sys-

tem where they can unsubscribe early and be reimbursed for the energy not consumed. As for the problem of reliability of transactions requesting energy tokens or updating the prosumer's account balance, where identification of the issuer is a challenge, our assumption is that location proof would be an ingenious way to securely and reliably identify the legitimacy of such transactions, as they should be issued by appropriately assigned smart meters installed in a fixed and known location and not be mobile nodes. Moreover, we believe that blockchain-based PoL, if properly designed for the particular application in which they are deployed, i.e., in our case the blockchain P2P energy trading platform, can be immune to fraudulent collusion and can be reliably defined without a mandated central authority. As for the last problem defined in the problem statement, which states that the management and control of energy demand in such a platform still depends on the hardware and software deployed and owned by the DSO, the complexity of the OPF formulation and solution in the smart grid makes its implementation on-chain unrealistic. In this context, we believe that the establishment of a Nash game arbitrated by a decentralized smart contract to create competition among suppliers proposing solutions to the OPF problem, and accordingly rewarding those with the optimal proposals, would significantly reduce the computations on the chain while providing a fully decentralized energy demand management system.

1.4 Research motivation

The integration of blockchain and smart contracts into the smart grid, which is behind the emergence of peer-to-peer energy trading, was primarily intended to create a decentralized platform for active prosumers to trade their energy generated and injected into the grid in a secure and reliable manner while minimizing the DSO's authority over the platform. However, due to security and control issues in such unbundled platforms, the role of the DSO as mandated supervisor and controller is still very much present. In this research, we propose a system in which different prosumers and consumers at the end nodes can trade energy with each other on a pay-per-use basis, where the determination of the legitimacy of the transaction as well as the efficient management of energy demand is done in a decentralized and trustless manner without a mandated central authority.

1.5 Research objectives

This study aims to fill the research gap described in the problem statement through the following specific objectives:

- To develop a novel decentralized P2P trading platform for autonomous, smart contract-monitored, consumption-based billing.
- To enhance energy transaction authentication of the energy trading platform with

a decentralized Proof of Location (PoL).

- To develop a new fully decentralized Optimal Power Flow (OPF) demand response control system for blockchain energy trading platform.

1.6 Research methodology

This research proposes a platform where energy is traded between active prosumers and end-node consumers on a pay-per-use basis. The research also includes the development of a method to verify the legitimacy of transactions using decentralized PoL, and efficient consortium-based demand-side management based on smart contracts. The methodology of this work is explained below:

1. Define all entities that constitute the pay-per-use energy trading platform.
2. Define the interactions between the entities and the respective workflow of the platform.
3. Develop and test the necessary smart contracts for the proposed platform using Solidity on Remix IDE.
4. Deploy the smart contracts on a private blockchain to evaluate and validate the proposed system using a case study.
5. Define all the entities involved in determining the issuance location for energy token requests and energy balance update requests.

6. Define the architecture and workflow for defining the PoL.
7. Develop and test the necessary smart contracts for the proposed PoL calculation using Solidity on Remix IDE.
8. Deploy the smart contracts on a private blockchain to evaluate and validate the proposed system in a comparative case study with a centralized PoL version.
9. Define all the entities involved in the computation of the decentralized OPF solution.
10. Define the architecture and workflow for the OPF solution.
11. Develop and test the necessary smart contracts for the proposed OPF solution computation using Solidity on Remix IDE.
12. Deploy the smart contracts on a private blockchain to evaluate and validate the proposed scheme in a comparative case study with a chain-based OPF solution scheme.

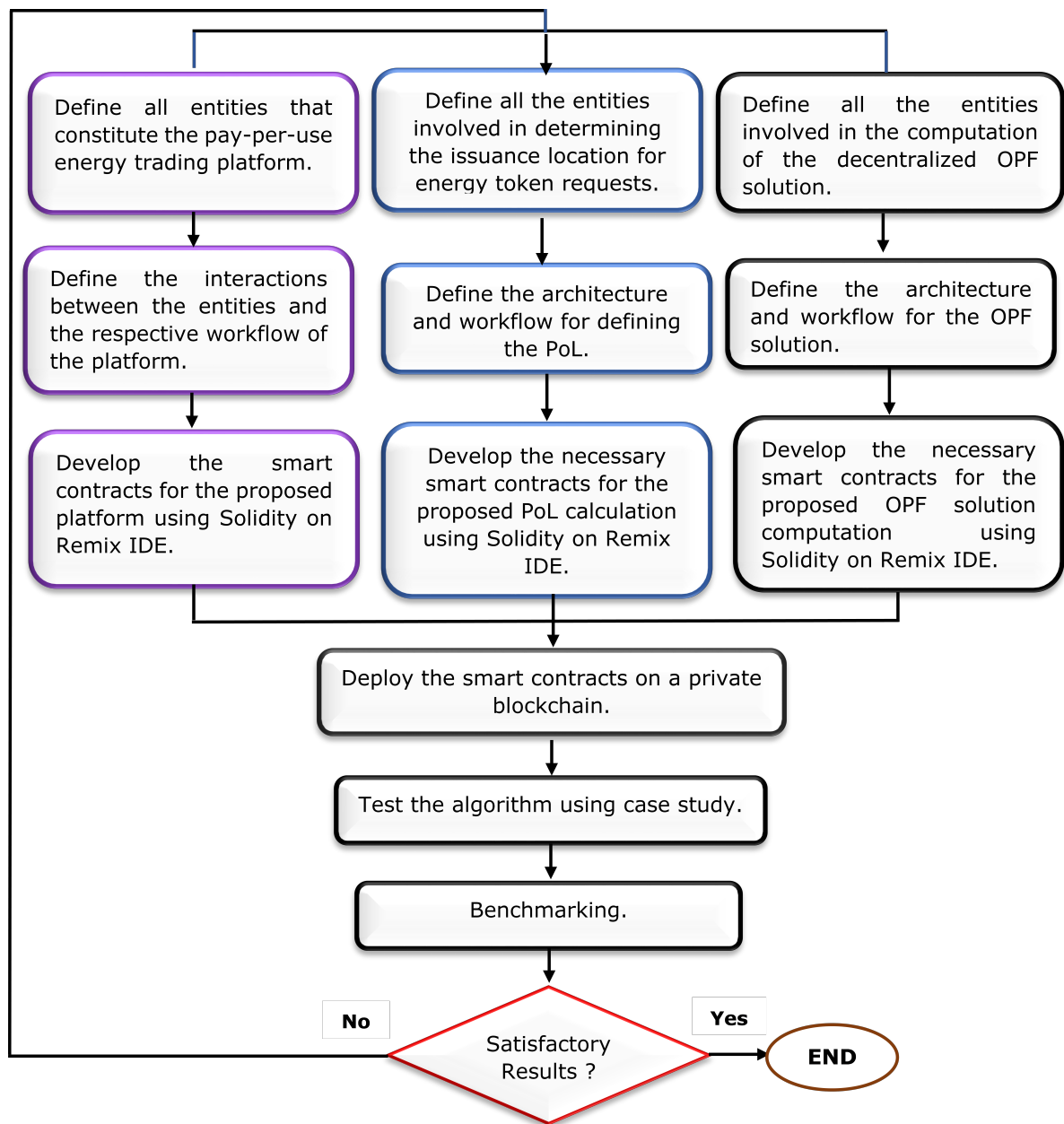


Figure 1.2: Flowchart of the adopted methodology research

1.7 Research scope

The scope of our research focuses on developing algorithms necessary to achieve our defined goals. The primary focus is to ensure that all defined goals are achieved by minimizing the influence of a central authority or trusted parties without compromising the security and reliability of the system, as is the case with blockchain cryptocurrencies, which have now operated securely for more than a decade without oversight by a central authority or privileged credentials. Hence, the scope or work of this thesis encompasses simulation modeling only with actual real-life trace data for testing and validation.

1.8 Thesis Outline

The thesis is organized into five chapters. Chapter Two: This chapter is divided into four subsections. In the first, an overview of the blockchain consensus algorithm is presented to provide a clear understanding of how the blockchain enables the reliability of its public ledger data in a trustless decentralized manner, as this is relevant to our work, which aims to ensure that any proposed solutions do not compromise the decentralization and authority-less nature of the blockchain. The second subsection presents the current paradigm used in blockchain energy trading systems by discussing blockchain energy trading platforms from the literature. It also aims to highlight the drawbacks and limitations faced by such a paradigm, especially the problem of under-consumption of energy. The third subsection highlights the problem of authenticating token claims in

a decentralized platform and shows how a PoL can serve as a secure yet decentralized authentication solution. Blockchain-based PoL solutions are also discussed, highlighting the challenges such systems face in keeping location proofs decentralized without compromising their reliability. The final subsection highlights the difficulty of implementing decentralized demand response in blockchain energy platforms by presenting some of the related work. Chapter Three: This chapter addresses the applied research methodology to achieve our defined research objective. First, the proposed P2P energy trading system with a dynamic pricing mechanism based on machine learning is explained in detail, which aims to solve the problem of buying energy under consumption existing in the current blockchain energy trading platform. The code implementation of such a platform is highlighted by detailing all the entities relevant to the functioning of the platform, their interactions and workflows through pseudocodes. Second, the proposed PoL approach is described and explained how it eliminates all opportunities for fraud and collusion. Its code implementation is also presented through corresponding pseudocodes. Third, the proposed new improved smart contract-based model that can be generalized for any defined OPF problem with effective execution cost is described. Its code implementation is described by expressive pseudocodes. The chapter describes the experimental setup used to evaluate the execution cost and latency of all smart contracts involved. Fourth Chapter: This chapter presents and discusses the results of the evaluation experiments described in Chapter Three. Chapter Five: This chapter summarizes

the research results and presents the contribution and future work.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

This chapter contains a literature review to highlight the research gap that this thesis aims to fill, as stated in the problem statement and research objectives in Chapter 1. It is divided into 4 sections. The first section provides a detailed overview of blockchain technology, including consensus algorithms, which are the core mechanism behind blockchain reliability. The second section explains the principles and architecture of the current token-based energy trading platform and highlights its limitations. The third section looks at PoLs on Blockchain applications, the different approaches, their limitations, and the main challenges in their implementation. The last section discusses approaches to energy demand management in P2P energy trading platforms.

2.2 Overview of Blockchain technology

As its name indicates, blockchain is in simple term, a chain of blocks containing information linked together in a chain-like structure, using cryptography. The term DLT (Distributed Ledger Technology) is also used to refer to blockchain. There have been rapid developments within blockchain and it has become a hot topic right now. To start

with, this section will try to provide a concise overview of this new sort of technology and the widely-known and noteworthy applications of blockchain.

2.2.1 Blockchain concept as technology behind decentralized cryptocurrency

Blockchain was originally created to enable the first decentralized cryptocurrency (Nakamoto, 2008), where there is no need for central authority, such as a bank to validate or deny transactions. Since then, it has seen huge growth to implement any decentralized application leading to various processes and sectors revolutionizing. Blockchain is implemented in a peer-to-peer network with no point of failure (Raval, 2016), (Aste et al., 2017). It merges technologies that have existed for decades into an ingenious manner that no one thought of before. It combines hashing technology, mainly using SHA256 hashing algorithm, digital signature and consensus algorithm (Nofer et al., 2017). After the 2008 financial crisis, considering cryptocurrency, which is the base from where the blockchain originated, any transaction issued must be propagated to all peers in the network, in hopes to be validated via consensus (Nakamoto, 2008), (Biryukov et al., 2014). Transactions are in a form of a data structure containing the issuer and recipient addresses, the number of coins to be transferred as well as the issuer digital signature to tag messages in a way that uniquely identifies the signer (Haber & Stornetta, 1990). When any entity which connects to the blockchain (node) joins the blockchain network and creates an account, the system generates a unique pair of keys using a well-known mathematical model, namely elliptic curve cryptography (SEC, 2000): a

public key, which is publicly known and essential to identify the user's account in a pseudo anonymous fashion, and a private key, which serves to sign the transaction and is kept confidential for authentication and encryption (Blundo et al., 2009). The security of this system comes from the fact that a given transaction can only be issued on behalf of an account if and only if the issuer possesses the particular account's private key (Nakamoto, 2008). Furthermore, blockchain is a tamper proof structure so that whatever data or transaction committed in it, it can never be deleted, denied or modified. All the nodes in the network should have an identical copy of the most up-to-date blockchain ledger. On this basis, blockchain contains the history of all issued transactions and thus, the balance of all accounts can be traced back and known by all the network's nodes. Blockchain, as a tamper proof structure, is based on two principles (Nofer et al., 2017):

- The first principle: blocks are chained together by their hashes. The hash is the difficult mathematical problem the miner has to solve in order to find a block. Typically, SHA256 hashing algorithm is used (Hoy, 2017). This goes according to main principles which are: i) never get the same output given two different inputs. ii) Given a specific output, it is theoretically impossible to get its corresponding input. So, each block is hashed using the hash of the block that precedes and thus, tampering with the chain is detectable, unless hashes are modified accordingly.
- The second principle: Blockchain tamper-proof structure relies on the "the longest chain wins". Consequently, conditions are imposed on the block's hash and the

legibility to add a new block, making it difficult and resource extensive.

Moreover, adding new valid blocks is rewarded. The first account which adds a new eligible block would have his block accepted and would get a reward in coins, what makes adding new blocks like a racing competition. So, meanwhile a fraudulent node is tempted to mess with the current chain, other nodes have already added new valid blocks and created a new longest chain.

2.2.2 Blockchain beyond cryptocurrency

Blockchain technology achieves a key milestone, when Satoshi Nakamoto presented the idea of Bitcoin as a cryptocurrency (Nakamoto, 2008) and many business organizations allow their customers to use Bitcoin as a method of payment. Since, and with the massive surge in popularity of Bitcoin, a bunch of other cryptocurrencies has emerged, including Tether, Tezos, Litecoin, Monero, and Maker (*Coinmap*, 2022). After it has attracted incredible attention by proving its efficiency in being behind several cryptocurrencies which have billions in terms of market capitalization, as well as it has prevailed for more than a decade, blockchain concept has evolved beyond what it was initially meant for. blockchain as the technology behind cryptocurrency has become a more globalized concept to be the technology behind decentralization (Raval, 2016).

2.2.3 Blockchain for decentralized applications and software

Blockchain has been implemented to enable many decentralized applications and software (Raval, 2016). On regular server-client web apps, all pages and software reside on a server or on the cloud infrastructure. They are then delivered to the client who is requesting them. If a client wants to share some data with another node, data are uploaded first to the server and then, it is delivered to the appropriate addressee on the behalf of the sender. Combating this centralization is achievable through decentralization with blockchain where information such as software, data or web pages are shared on the blockchain so that each node holds in its possession its copy of the ledger (Xu et al., 2016). What blockchain decentralized App is offering as better alternative to traditional web Apps is as follows:

- It enables a Censorship free network, as depicted in Fig 2.1; this is because any data shared to blockchain are unchangeable and unerasable. So, for any content to be dismissed, new blocks should be added to the block that precedes the one containing the undesirable content until that particular block would become within a smaller branch and get bypassed (Swan, 2015).
- It is hacking-proof as a copy of the blockchain ledger is held by all nodes so that a hacker must break into half of the nodes in the network at least to crack the system (Verma & Garg, 2017).

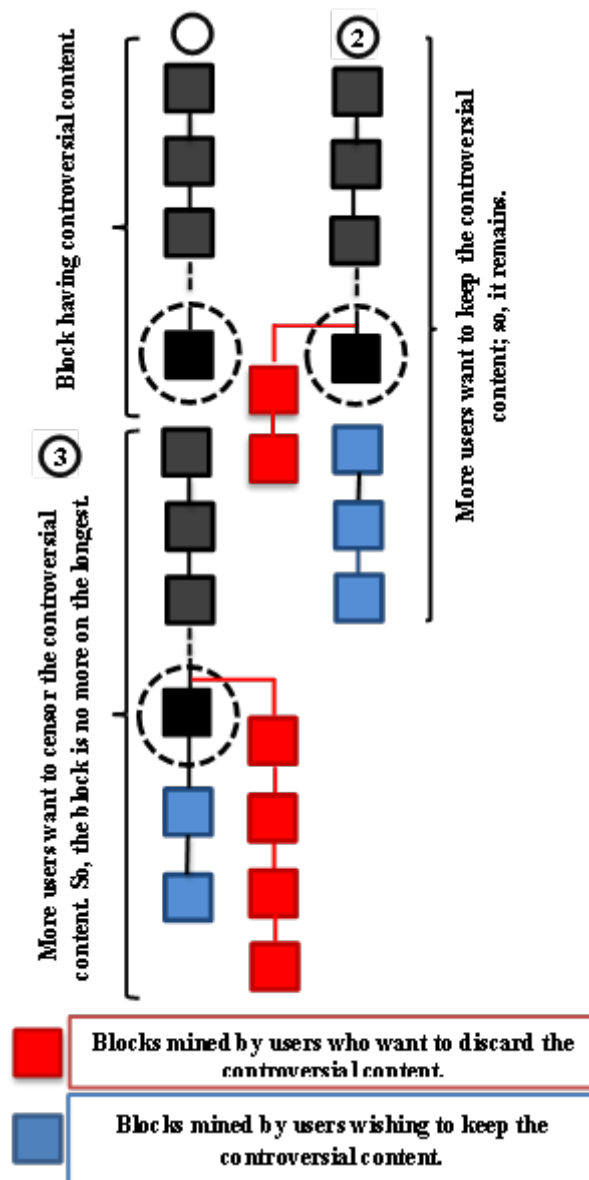


Figure 2.1: The Censor-proof property in blockchain-based apps and software.

2.2.4 Blockchain in the industrial world

The current popularity of Blockchain technology is peaking. It has inspired the broader use of this technology in different applications in the industrial world (Miller, 2018),

not as a mere startup, but rather as an established technology implemented and trusted by pioneer and giant industrial firms. One of the potential applications of this nascent technology is industrial supply chain management (Kouhizadeh & Sarkis, 2018).

Blockchain is now evolving towards permissioned platforms for industrial companies and organizations (Polge et al., 2021). In contrast to open Blockchain, where nodes are pseudo anonymous and the network is open; permissioned blockchain is a closed environment where parties belonging to the same industry or organization want to interact in a decentralized distributed manner rather than a traditional centralized one using a central database server. The incentive behind is instead of relying on a centralized architecture where data integrity depends on some admins having total access to the database and the full ability to tamper with it; using a decentralized architecture, in contrast, will ensure that each node or party within the organization holds a copy of a tamper proof ledger and the ledger validity as well as correctness is agreed upon a consensus reached by all involved parties. In addition, a decentralized architecture is immune against hackers since they have to get access to more than half nodes in the network to be able to corrupt data (Boireau, 2018). In the traditional centralized network architecture, there is rather a single target which is the central database. This enables origin information for goods and furniture and improves logistical transparency and supply chain quality. Indeed, as for the supply chain management scenario, Blockchain is meant to trace back each type of component or raw material that are used

in the manufacturing process of the product (Kouhizadeh & Sarkis, 2018). Therefore, any inoperative piece or material can be traced back to its origin and it is impossible for the responsible party to make it denied, so that any products with defective pieces can be removed from the market. So, why Blockchain in supply chain management rather than traditional logs to a central database? Actually, supply chain may involve different industrial parties from different countries i.e., each entity would have its own database and its exclusive access. So if a defective component shows up, it is not certain that the responsible party wants to pass on data that could make them responsible for the component. Those involved entities have the option to agree to rent a common database hosted in the cloud, but all parties must still have access to it and tamper with the data at will, in addition to the cloud services cost. In fact, such an incident already happened, when Walmart hypermarkets received contaminated pork and mango in 2012, resulting in poisoning several customers. Walmart tried to track back the contaminated product but was unable to trace its provenance and so, he had to remove all the pork and mango stocks from the sale. This caused significant losses and bore them full responsibility for the affected customers (Heineman, 2014). Walmart decided then to use IBM's blockchain solution based on hyper ledger Fabric to keep track their entire purchased product (Mohamed & Al-Jaroodi, 2019) in order to be able to make the defective components and raw materials traceable. Beyond supply-chain management, Blockchain has been implemented in any process that involves different parties working separately,

but the outcomes of their tasks are interdependent and must be synchronized (Mohamed & Al-Jaroodi, 2019). So, whenever any team or individual completes any task, it is logged to the Blockchain. Thus, thanks to some key characteristics of Blockchain, the history and related information on all the achieved tasks could be saved forever and shared between the involved parties in a transparent decentralized way, making tasks' synchronization easier to achieve. When a problem occurs, it is made easy to trace back the cause till the root person supposed to be responsible since some professional faults can have severe consequences (Rizal Batubara et al., 2019).

2.2.5 Smart contracts

We believe that it is unfair to talk about Blockchain without addressing the case of smart contracts. In fact, Blockchain, as a peer-to-peer technology without a centralized system, requires a method to verify and maintain authenticity. This is ensured by smart contracts. These are computer codes that define the terms and conditions of agreements between two or more parties, which must be confirmed by network members (Luu et al., 2016). These digitally written lines of code are published on the blockchain, becoming immutable, indisputable, and visible to all parties on the network (Christidis & Devetsikiotis, 2016). Smart contracts are inspired by the Bitcoin scripting concept. Basically, Bitcoin allows adding a script to any transaction for any related specification or condition (Klomp & Bracciali, 2018). Bitcoin transactions must have a specific format and contain an input and an output (Nakamoto, 2008). Fig 2.2 shows a typical example.

For example, we can say that Alice issues a transaction in which she sends 3 bitcoins to Ali; this transaction has as input 3 bitcoins and as output it mentions that these 3 coins are for Ali; if Ali wants to use these bitcoins and send them to another party, he should issue a transaction with an input that refers to the output of Alice's transaction to show where he got the coins. In the same way, Alice's input in the transaction intended for Ali points to another transaction output from which those coins were transferred to her. This continues until it points to a coin base transaction, which is the transaction that provides the mining reward, and which has no input. This input-output format helps verify any condition or specification set as a script for the coins before they are spent. So, if Alice has added a condition to the transaction that says these coins cannot be spent until two months after the transaction date, Ali needs to reference Alice's transaction output in his transaction input; then anyone can check the predefined conditions for approving that transaction. It will turn out that the coins cannot be issued until two months have passed after the transaction date. So everyone can check whether Ali is already allowed to spend the coins or not, and whether the acceptance of the transaction when he wants to spend them is valid or not.

This idea was then extended from simple script forms to advanced algorithmic codes and computer protocols specific to set more complex conditions in the form of contract codes developed to enable proper, highly automated workflows without central authorities, first introduced by Ethereum and then integrated into mainstream

blockchain platforms (S. Wang, Ouyang, et al., 2019).

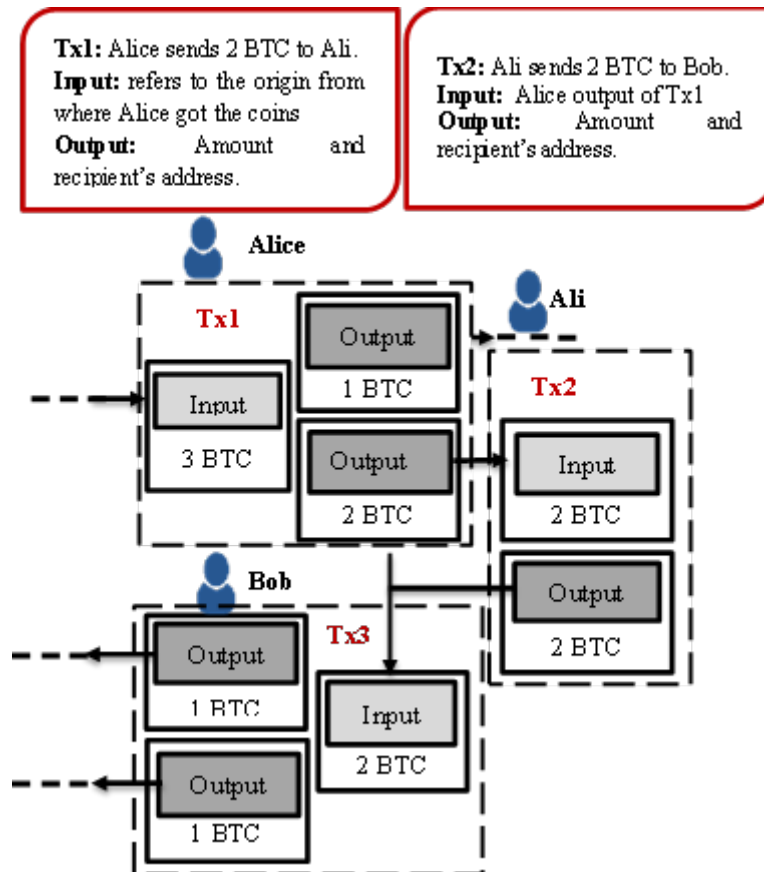


Figure 2.2: Transaction input and output formats in Bitcoin.

Let us give a typical example of using smart contracts to collect automatic payments. Suppose A agrees to pay B a certain amount of money on a certain date in return for any service. This can be done using a blockchain smart contract issued and signed by A or, better, by both parties, since the smart contract supports the signature of multiple wallets (Zhou et al., 2018). Once the condition (the due date) is met, B can claim the money from A by issuing a transaction that refers to the particular contract. Such a

transaction is approved by other parties, who can confirm it according to the contract, of which they all have a copy in their ledger (S. Wang, Ouyang, et al., 2019). This digital transaction approval simplifies the use of blockchain technology in complex business networks.

2.2.6 Blockchain consensus algorithms

First and foremost, consensus is about using protocols that mainly dictate the nodes in the network on how to agree with each other about the ledger that they hold and the order in which the transactions are listed on the newly-mined block. This can be quite a difficult task without the use of consensus algorithms which are meant to serve two prominent requirements in terms of Safety (a new validated block committed to the blockchain is no more changeable) and liveness (Motepalli & Jacobsen, 2022) which means that a honest user will ultimately get approved if he tries to submit a block that is proved in a verifiable manner to be valid. Thus, a good decentralized Blockchain relies on a good consensus algorithm that moderates it by enabling the nodes to achieve an agreement and maintain the Blockchain updated (Elsden et al., 2018). Further, this section aims to provide a comprehensive overview on how consensus is reached in Blockchain, detailing more rigorously those used in Blockchain platforms with the most market capital.

2.2.6.1 Consensus in permission-less Blockchains

Permission-less Blockchains such as Bitcoin (Nakamoto, 2008), Ethereum (Wood et al., 2014), and Monero based-systems do not require pre-established entities to reach decentralized consensus (Labazova et al., 2019). Public nodes can join the Blockchain without needing permission and consensus aims to ensure the validity of transactions and data committed to the blockchain in an environment where users do not know or trust each other (Neudecker & Hartenstein, 2018) . The nature of consensus in such environment is mostly proof-based (Pahlajani et al., 2019) where nodes are competing with each other to earn the right to append a new block to the current chain. In Proof-based consensus, it is compulsory for any node that is proposing a new block to include a proof verifiable by the other nodes, so to ensure that he would be penalized in any way if this block contains invalid data or transaction. But since it costs something significant to the node to propose the new block; moreover, this cost would be wasted in vain if the proposed block is not accepted by most nodes in the network (Suresh et al., 2020).

2.2.6.1.1 Proof-based Consensus Protocols

In this subsection we sketch in short, the basics of the proof-based consensus algorithms. Apart from the original work Proof of Work (PoW), proposed by Nakamoto (Nakamoto, 2008), many other variants have followed, which will be discussed here so to identify the pros and cons that their integration brings to the system.

Proof of Work (PoW)

- PoW is a mathematical puzzle which the network nodes have to solve in order to be able to add a new block to the ledger. The process is called mining and the nodes involved are miners. The puzzle consists of finding a hash for the block to mine with specific numbers of zero at its most right. The number of zeros defines a property known as the mining difficulty. Actually, the block hash is computed by concatenating all the block fields forming a long string that is hashed using the SHA256 algorithm. Hashing the same string using the SHA256 algorithm would always lead to the same result. For that reason, a field that is called the nonce, is used in PoW, which proves the effort invested by a node for getting the right to append his block to the chain. The miner role is to try different values of the nonce until coming up with a value meeting the difficulty condition. This makes the block mining resource and time costly. Indeed, mining a block in Bitcoin costs between 531 to a stunning 26,170 (Young, 2020). Whenever a miner solves the puzzle by guessing a secret value (the nonce field), he adds the block to the chain and broadcasts it to the network. The time when the block was found is the Timestamp. Its chain is accepted to be the real new ledger only if it is the longest one. In other words, whenever a node in the Blockchain network receives a new chain, it first checks its validity by verifying that all the hashes are valid and each block hash is computed using the hash of the block that precedes. The

validity of the transactions in the last block must also be verified and finally, its length must be higher than the one it is currently holding. Getting these conditions met, the node would accept the chain as the new valid ledger. The first node which solves the puzzle and adds the new block to the chain will broadcast the new longest valid chain, and would gain a mining reward, making the process of mining a hard race and lucrative investment at the same time (Nakamoto, 2008). After appending the new block to the Blockchain, the cycle restarts, increasing the size of the Blockchain indefinitely. PoW make it more financially advantageous for the miners to show honest behavior by keeping the system in good order rather than trying to tamper with ledger or include invalid transactions (Gervais et al., 2016). Most famous cryptocurrencies using PoW consensus algorithm, are Bitcoin and Ethereum (Kiayias & Zindros, 2019). PoW algorithm did its proof since it is the oldest existing in Blockchain. It also allowed Bitcoin to prevail as a secure and reliable cryptocurrency ranked number one market capitalization since 2009 till today (*Cap, Today's Cryptocurrency Prices by Market*, 2021). It makes Blockchain tamperproof, strong and secure, and it avoids double spending. However, it has limited throughput estimated to 27 transactions/s due to the limited allowed block size and the time required when mining the block (Georgiadis, 2019). In addition, it has costly mining prize as Blockchain mining is very energy consuming due to the large energy required to supply the extremely

powerful specific hardware (O'Dwyer & Malone, 2014). Fig 2.3, reveals that Bitcoin uses annually more electricity than the whole of Argentina. If Bitcoin was a country, it would be in the top 30 energy users worldwide (Criddle, 2021).

- Because mining a block is linked to the miner's hardware hashing power, this may lead to the monopolization of mining by nodes with enormous hashing power. In fact, the more processing power a miner has, the more monopoly he may have over the network. If a user possesses 51 per cent of the hashing power, this will ruin the Blockchain because nobody would be able to compete with him, and the ledger integrity would be totally at his mercy. He would be able to double spend at will, forking on the block which precedes, accepting or boycotting the transaction he desires. In other words, Blockchain integrity would be destroyed if it undergoes such an attack (Ye et al., 2018). Fig 2.4 demonstrates how fraudulent parties may win the mining race if having sufficient hashing power. Forking is an important concept of the Blockchain that must be clarified. A fork takes place if blocks are mined by two miners at the same time. If such a case occurs, the other miner will have to choose to mine their new blocks on one of the forks, then, it is the longest chain that prevails.

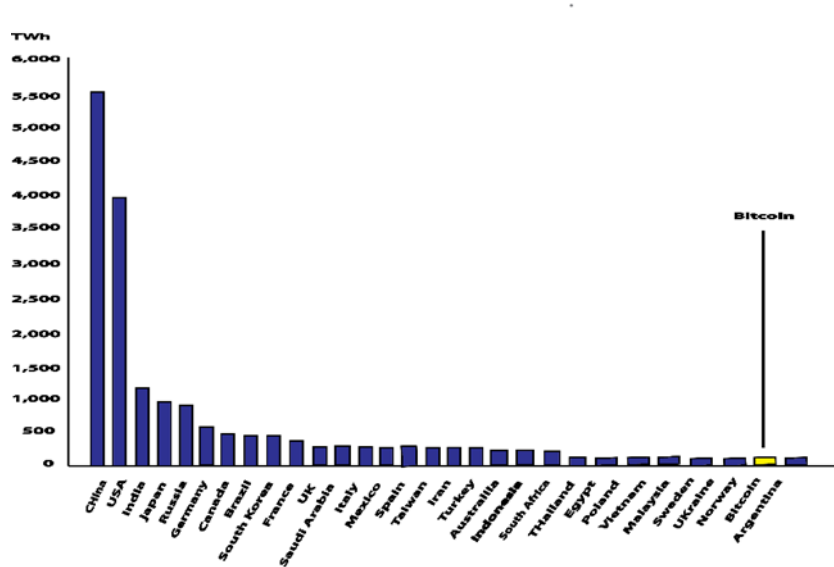


Figure 2.3: National energy use in TWh. University of Cambridge Bitcoin Electricity Consumption Index (Criddle, 2021).

However, forks may be intentional, and users may agree not to mine on a specific block but instead on the previous block. If they succeed mining enough blocks and win the mining race, all the upcoming blocks after the fork would be undone (S. Zhang & Lee, 2019). Forks are illustrated in Fig 2.4.

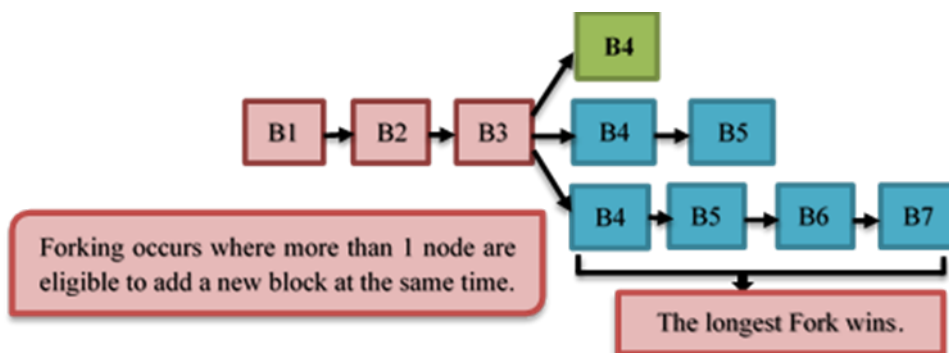


Figure 2.4: The most extended chain-win principle in Blockchain.

As it turns out, there are quite a few other problems with Proof-of-Work if the

hashing power was not evenly split between miners. They are as follows:

- Sybil attack: which consists of faulty or malicious entities which possibly present themselves as multiple identities in order to get control of the system. Since the nodes in PoW-based network are identified using only their public key, it is possible that a user generates as many public keys as he wishes and be potentially able to create endless accounts (O'Dwyer & Malone, 2014).
- Selfish mining: If a party (miner or miner-pool) is sure of being able to mine newly created blocks at least twice faster than any other node in the network, they may exercise what is called selfish mining. Selfish mining occurs when strong nodes can withhold blocks which they create and selectively postpone their broadcast. Then, a fork is deliberately generated
- that would be the longest chain, which possibly force the honest network to discard some of its blocks (Criddle, 2021), (Ye et al., 2018). Thus, selfish mining is the downside of the longest chain win principle illustrated in Fig 2.5.

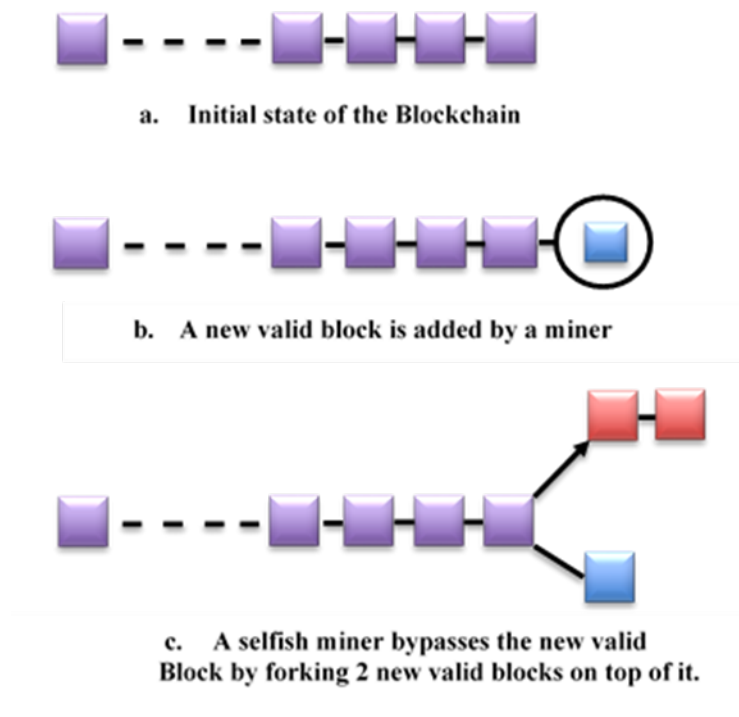


Figure 2.5: Selfish mining

Proof of Stake (PoS) and its variants

Ethereum looked to Proof of Stake as a solution to solve these problems. In opposition to PoW, Proof of Stake algorithm (Vasin, 2014) favors miners with the most hashing power and thus, avoids the network to be monopolized by those having the most powerful hardware and makes the mining process much less energy-consuming in transaction validation and consensus so that the mechanism becomes much greener. In PoS new specific vocabulary is used. Hence, adding a block to the chain is called validating a block. The nodes which are involved in the block are called validators.

a. Proof of Stake

PoS form could be seen on the Nxtcoin platform (Nguyen & Kim, 2018). Overall, Sunny King and Scott Nadal, published their first research on PoS in 2012. (King & Nadal, 2012). Instead, the block validation is done by a set of validators which have staked a sufficient amount of the cryptocurrency in the network. The more stake amount a validator has, the more its chances of validating a new block are. Additionally, the validator with the most considerable staked amount is not automatically chosen to validate a new block. If this is the case, mining a block would be monopolized by those validators. Thus, Blockchain would neither be decentralized nor reliable since the consensus would be taken by only a fixed number of users having the most significant number of stakes in the network. In fact, an element of randomness should be added in the process of choosing the next block validator (Shibata, 2019).

There are many variants of the PoS algorithm (Vasin, 2014). In general, there is a certain threshold of coins any user needs to have staked in the network to be a validator candidate. Staking is done by a defined type of transaction to a defined account where the stakes are locked. Once the owners have validated the block they mined in the network; they can take back the stake amount, but they can no longer participate in the validation process.

b. Chain-based PoS

In essence, it assigns the right to create new blocks to stakeholders in a pseudo random manner (Sameeh, 2016). A case of the chain-based-PoS may be where each validator proposes a new valid block that is then, chosen using a verification algorithm which considers the size of the generated hash and the amount that is staked by the validator. Since the hash computation is unpredictable, it adds an element of randomness to the validator selection. This is seen in NXT cryptocurrency (Hanke et al., 2018). Another variant is the coin age approach which combines how long the stakes have been held as well as their amount. It is the Peercoin's PoS system (Hanke et al., 2018).

c. Casper PoS

Casper Vlad Zamfir is often referred to as the "Front of Casper", the new PoS algorithm proposed for Ethereum (Rosic, 2020). Ethereum has previously officially stated that it plans to move from PoW to Casper PoS soon (A. Jain et al., 2018). However, only one hard fork has been released that implements PoS called GHOST, which has been subject to three fraudulent attacks so far (Schwarz-Schilling et al., 2021). Three attacks on proof-of-stake Ethereum. arXiv preprint arXiv:2110.10086.]. Full consensus conversion has not yet taken place. Currently, there is talk of a merge rather than a full migration, which is announced for the second quarter of 2022. So, what makes Casper different from other PoS protocols.

The motivation behind Casper is to centrally address the "nothing at stake" prob-

lem in the PoS consensus algorithm. Moreover, Casper uses an accountability function protocol that penalizes any malicious behavior or attempted fraud (King & Nadal, 2012), (Buterin & Griffith, 2017). In PoW, mining a new block on both chains forces the node to split his hashing power by two at the risk of losing the mining race and thus wasting energy and hashing power for nothing. In contrast, with PoS and PoI, every time an eligible validator faces a fork, its new block can simply be added to both chains. Consequently, the network ends up with two chains of the same length in which coins can be double spent (W. Li et al., 2017).

What is proposed in Casper, is that to even qualify as a validator, you need to lock up a significant amount of your fund as a stake and even then, you will be rewarded proportional to what you are betting. If a validator acts maliciously in a “nothing at stake manner”, they will immediately be punished, and getting their invested stake slashed and removed. This is depicted in Fig 2.6.

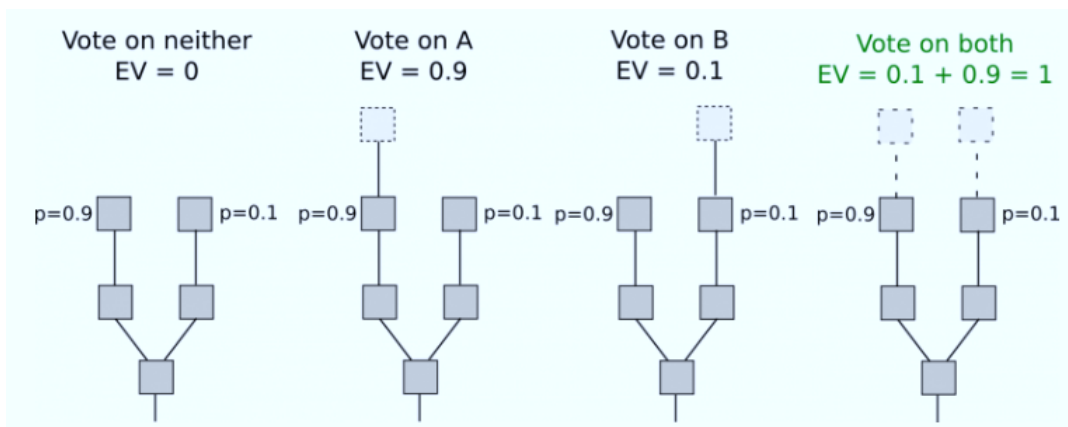


Figure 2.6: Validator voting process in Casper PoS.

d. Delegated PoS (DPoS)

Delegated Proof of Stake is another variant of PoS (Larimer, 2014) where stake holders vote for a delegate to be responsible of adding the new block, instead of voting or adding it themselves. The stake holders possess a number of votes that is proportional to the amount staked in the network. Compared to PoS, DPoS can significantly reduce latency which is very attractive for IoT networks. Tough, it may lead the system to centralization.

e. Proof of Importance (PoI)

Proof of Importance is the consensus protocol used by NEM (Beikverdi, 2015) where each account is allotted an importance score that represents its acquired importance to the NEM. The nodes with the highest importance score have higher opportunities of reaping a block. In the case of NEM, the importance score is computed depending on the amount of vested money which is equivalent to the staked amount. NEM requires a balance of at least 10,000 vested XEM (XEM is the cryptocurrency enabled by NEM) to harvest, which creates bounds on the number of accounts expected in Sybil attacks. In addition, such attacks are moreover unlikely in PoI making transfers decayed over time as only 10% of this amount can be actually vested by the network in 24 hours. Accordingly, it would require ten days to vest 10000 XEM while several weeks are needed to get inflows fully vested into an account. Gathering these countermeasures against Sybil attacks, PoI still suffers from the nothing at a stake problem (Košťál et al.,

2018). In PoW, mining a new block on both chains, forces the node to split his hashing power by two at the risk to lose the mining race and thus, wasting energy and hashing power for nothing. In the opposite, any time a validator is confronted with a fork, he can simply add his new block to both chains. Consequently, the network will end up with two chains with the same length where coins may be double spent (W. Li et al., 2017). Fig 2.7. illustrates score of importance in PoI. Tab 2.1 provides the major pros and cons in Proof of Stake-based algorithms while Tab 2.2 shows the top 10 cryptocurrencies by Marketcap and the corresponding consensus protocols used, for the year 2022.

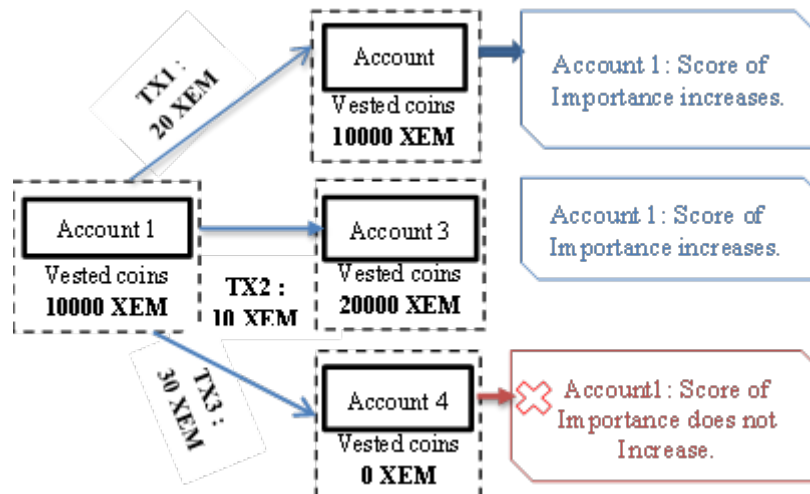


Figure 2.7: Increasing Importance Score in NEM PoI by contributing in the network.

Table 2.1
Proof of Stake-based algorithms.

Strengths	Limitations
<ul style="list-style-type: none"> • A much greener way of adding new block • Risk of centralization reduced since anyone who (a sufficient amount staked in the network allows a potential block validator. • Faster to add a block in PoS than in PoW. • More scalable in terms of number of transactions processed per second. • More resistant to Sybil attacks. 	<ul style="list-style-type: none"> • The nothing at a stake problem. • Not suitable for non-financial decentralized applications.

Table 2.2
Top 10 Cryptocurrencies by Marketcap and the Corresponding Consensus Protocols Used, 2021.

Cryptocurrency	Market capital €	Consensus protocol used
Bitcoin	€564,381,542,293	PoW
Ethereum	€111,207,696,199	PoS
Tether	€18,623,408,862	PoW & PoS
XRP	€11,046,513,838	Ripple
Litecoin	€8,983,001,102	PoW
Cardano	€8,316,233,583	PoS
Polkadot	€7,232,689,163	PoW
Bitcoin Cash	€6,769,504,611	PoW
Stellar	€6,263,250,767	Stellar
Tezos	€1,619,850,592	PoS

2.2.6.1.2 The mining power consumption/nothing at a stake trade-off

As seen, PoW comes with a huge burden of mining power consumption. PoS has been proposed as a greener consensus protocol. However, reducing mining power consumption comes with another compromise. Actually, forks in Blockchain are meant to be the

shortest possible because multiple valid chains of the same length would lead to nodes with different balances and, thus, double spending would be possible. Unintended forks are not documented. Yet, the length of unintended forks can be deduced from the number of orphaned blocks in the blockchain. The longest a fork is, the more blocks are subjected to be orphaned since a single branch would prevail at the end. Tab 2.3 and graph in Fig 2.8 show the number of orphaned blocks, versus the mining power consumption in different blockchain-based cryptocurrencies. As illustrated in Tab 2.3, the average block mining power consumption is a function of:

- The Wattage of the mining device used, (P).
- The number of devices used to mine a block, (ND).
- The block average mining time, (TAv).

Furthermore, it is assumed that all networks are using an Antminer mining device, 14 Th/s, of 1320 Watts (*ANTMINER S9i*, 2021). Number of devices used to mine a block in the desired time is given in Eq 2.1 as:

$$ND = HR(Th/s)/14(Th/s) \quad (2.1)$$

where HR is the network hash rate and 14 Th/s is the hashing power of a single mining device used.

Moreover, block mining power consumption (W_c) is computed for each blockchain network according to the expression in Eq 2.2:

$$W_c(KWh) = P(KW) \times T_{Av}(h) \times ND \quad (2.2)$$

Documentation regarding the network hashing rate was not available for Stratis, Navcoin, and Spectrecoin cryptocurrencies. The references (cryptoID, 2021) (cryptoID, 2021) (*SpectreCoin*, 2020) have been used to get the corresponding current mining difficulty(D). Then, we were able to deduce the network hashing rate according to the formula in Eq 2.3, (X. Zhang et al., 2018):

$$HR = D \times 2^{32} \times T_{Av}(s) \quad (2.3)$$

Table 2.3
Hashing rate, mining time and orphan blocks rate for different blockchain platforms.

Blockchain-based cryptocurrency	Hashing rate (Jan 2020,T Hash/s)	Daily Average orphaned blocks *	Daily Average mined blocks *	Daily Average Percentage of Orphaned blocks	One block's mining Average Time (s) *	Average mining power consumption per block (Antminer 14 Thash/s) *
Bitcoin	146M *	0.1	158	0.06	584	2233104,76
Ethereum	348 *	717	6371	10	13	0.12
DigiByte	84K *	6	5770	0.08	15	3.3
Litecoin	280 *	149	0.2	0.03	578	1.09
Dash	5K *	1	549	0.18	157	72.06
Syscoin	21M *	2	1372	0.14	62	34100
Zchah	0,06 *	3	286	1	302	0.5×10^{-3}
Blackcoin	21 *	50	1296	3.85	66	36.3×10^{-3}
Stratis	4 *	70	1264	5.24	68	7.12×10^{-3}
Navcoin	78, 67 *	80	2828	2.75	30	61.81×10^{-3}
Spectre-coin	80 *	60	1361	4.22	63	0.132

The data in Tab 2.3 were taken from the following sources, which are marked accordingly with '*' in each table row, (d. Kwaasteniet, 2021), (d. Kwaasteniet, 2021), (d. Kwaasteniet, 2021), (*Ethereum Hashrate*, 2021), (*Total Hash Rate (TH/s)*, 2021), (*Ethereum Hashrate*, 2021),(*DigiByte Hashrate Chart*, 2021), (*litecoin/hashrate chart*, 2021), (*dash/hashrate chart*, 2021), (*syscoin/hashrate-chart*, 2021), (*Firo (Zcoin) Hashrate*, 2021), (*bitinfoCharts/blackcoin*, 2021), (*Blockchain Explorers*, 2021), (cryptoID, 2021), (*SpectreCoin*, 2020).

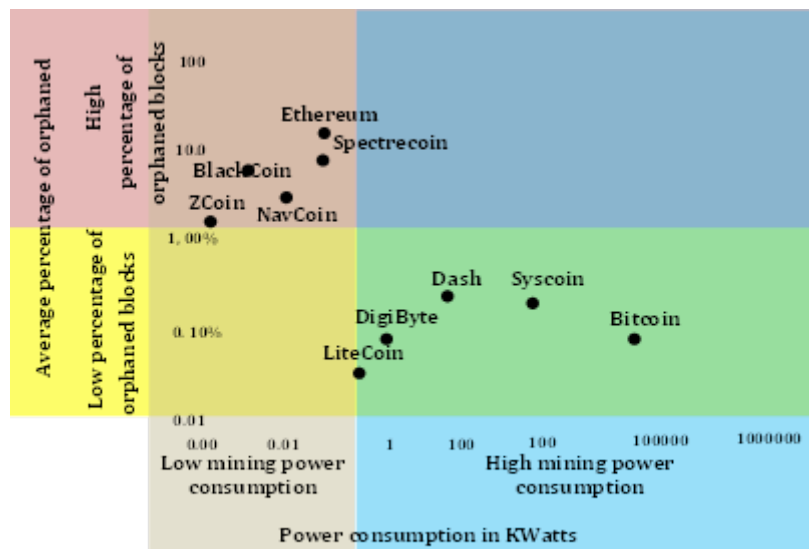


Figure 2.8: Percentage of Orphaned blocks per day versus block mining power consumption in blockchain-based cryptocurrencies.

2.2.6.1.3 Hybrid PoS and PoW algorithms

As made clear so far, PoS and PoW algorithms are compromised. On the one hand, PoW makes it aching and hard to mine a block. This improves the reliability and robustness of the chain. However, the inherent flaws of power wastage and the high computational

power requirement would potentially favor the risk of monopolization and makes the system not ecofriendly. On the other hand, PoS is much greener in terms of energy consumption and may avoid monopolizing the network. But, adding the blocks with certain ease may generate the nothing at a stake problem which may cause potential coin double-spending. Consequently, it can be implied that a mixture of PoW and PoS would be an optimized solution for reaching consensus in Blockchain. In such a scenario, the number of coins the node has staked in the network or its importance score would determine the difficulty of the PoW puzzle to solve. The higher your importance score is, the easier puzzle you have to solve. This concept has been introduced for the first time in a paper by King and Nadal (King & Nadal, 2012). However, even though it looks promising, the idea did not get much interest and rare are the blockchain platforms adopting it.

2.2.6.1.4 Other proof-based consensus algorithms

In addition to PoW and PoS, there are other kinds of proof-based consensus algorithms.

Main of these will be included below.

a. Proof of Burn

Instead of staking coins that may be taken back by stakeholders, in case the validated block is correct and the miners do not wish to be part of the validation process anymore, validators are required to burn coins by sending them to a blocked account making them

irretrievable forever. The more coins burned by a node, the higher its chances to mine a new block, (Karantias et al., 2020). Proof of Burn PoB can be seen in a cryptocurrency called Slimcoin.

b. Proof of Elapsed Time

In Proof of Elapsed Time (PoET) (*Architecture Overview. What is the Sawtooth Lake Distributed Ledger?*, 2015), nodes have to identify themselves before joining the network. Blocks are not accepted if mined prior to a minimum defined threshold time. PoET needs lower power consumption but exhibits high latency, making it particularly interesting for limited-resources, not time constrained IoT applications (Salimitari & Chatterjee, 2018).

c. Proof of Capacity/Space

Similar to PoW, Proof of Capacity/Space needs miners to solve a puzzle but, instead of hashing power, it relies on the hard disk capacity of the miners. Thus, it is significantly less energy-consuming but, creating new blocks is of high latency. This method is not that appropriate for limited-storage capacity devices, (X. Cai et al., 2019).

Table 2.4
Comparison of Blockchain consensus protocols for a set of essential Blockchain properties.

Consensus Protocol	Blockchain	Echo friendly	Tolerated malicious power	Data model	Development language	Execution	Examples
PoW	Public	No	25% computing power	Transact-based, Account-based	Golang, C++, Solidity, Serpent, LLL	Native, EVM	Bitcoin, Litecoin, ZCash-Ethereum
PoS	Public	No	50% Stake	Account-based	Michaelson	Native	Peercoin, Tezos, Tendermint
PoI	Public, Private	Yes	Unknown	Transact-based, Account-based	Java	NEM	XEM
PoA	Public	Partial	50% of on-line stake	Account-based	Solidity, Java, Python	EVM, Dockers	Parity
PoB	Public	No	25% computing power	Transact-based, Account-based	Golang, C++, Solidity, Serpent, LLL	Native, EVM	Slimcoin
PoEt	Public	Yes	Unknown	Key-Value	Python	Native	Sawtooth, Lake
PoC	Public	No	Unknown	Key-Value	Unknown	Unknown	File Share
DPoS	Public	Partial	< 51% Validators	Transact-based, Account-based	No scripting	Native	Bitshares

2.2.6.2 Consensus in permissioned-blockchains

In permissioned-blockchains, all the nodes theoretically know each other in the network and the number of the interacting ones is limited. Thus, consensus used in such platforms is much laxer and much less burdening compared to the ones used with permission-less blockchains (X. Zhang et al., 2018).

2.2.6.2.1 Voting-based Consensus Protocols

Contrarily to proof-based consensus algorithm, where nodes can leave and rejoin the network at will, in the voting-based consensus algorithm, the nodes inside the verifying network should be known and adjustable. Voting-based consensus are also used in some blockchain open networks using the Federated Byzantine Agreement (FBA), used in Ripple and Stellar cryptocurrencies (Mazieres, 2015). Voting based consensus could be categorized into two types: i- Crash fault tolerance-based consensus and ii- Byzantine fault tolerance-based consensus.

Crash or network fault-tolerant consensus

This category permits to reach consensus in a distributed, decentralized network, which is crash and network fault-tolerant but, does not support Byzantine fault-tolerance. This means that nodes can come up with a common consensus by using this kind of algorithm in such networks, even though some other nodes or network connections are technically faulty. However, consensus cannot be achieved with nodes behaving maliciously.

a. Paxos

As a consensus protocol that is used in distributed systems (Lamport, 2019), Paxos is an analogy to the parliament voting in an ancient Greek Paxos Island, where there are three agents: they are proposers of the new law, acceptors who vote for the law, and finally learners who are in charge of writing the new law and its saving. Each time a node wants to add a new block in the Blockchain so that all nodes update their ledgers, some chosen validators that play the acceptors' role, vote either to accept or reject the new block. If most acceptors vote for acceptance, all the other nodes (learners) update their ledger and add the proposed block. Fault tolerance is achieved given that several acceptors vote for the proposal, so even if one is faulty, the consensus can still be reached unless more than $N/2-1$ acceptors in the network are faulty. Let us mention that N is the total number of acceptors in the network). However, Paxos does not tolerate Byzantine faults and can operate only if all nodes ensure mutual trust. In the case of smart contract execution, this is implemented as a state machine and needs only one node machine to be executed. Then, if accepted, the state machine is replicated in all the nodes so, they can add the new contract execution state to their ledger.

b. Raft

This algorithm has been proposed as an alternative to Paxos (Lamport, 2019). The name Raft refers to an analogy where it was used to escape the Paxos Island. Raft offers solutions to potential issues in Paxos. This can be summarized as follows:

It happens that more than one node suggest a proposal simultaneously, however, a single proposal must be agreed (voted) by acceptors while the others are denied and may be proposed again for upcoming voting rounds. Considering a Raft cluster, verifying nodes can belong to three different servers' roles so that at any given time each server is either leader, follower, or candidate. At the beginning of each term, all nodes in the network start as followers. In order to propose a new log to be replicated in the network, the node will candidate to become leader and asks for votes; each node can only vote one time per term. Thus, a candidate wins an election if it gets votes from most of the servers in the whole cluster for the same term. Then, the leader's proposal is replicated by all nodes in the network. Additionally, whenever a node is elected as leader, a term-time is set out. In the case the term-time elapses with no leader; the node is considered to be faulty, so, a new term begins when a new leader is elected. Raft's performance is almost similar with others consensus algorithms such as Paxos. The most important performance indicator concerns the minimal number of messages Raft uses when an established leader is replicating new log entries. Moreover, it is open to possible further improvements (Dib et al., 2018).

Byzantine Fault-Tolerant consensus

There is a more elaborated consensus in this second type of voting-based consensus, namely, Byzantine fault-tolerant, which means consensus can be accomplished on the distributed network, even though some nodes behave maliciously or are technically

faulty (Sousa et al., 2018).

a. Practical Byzantine Fault Tolerance (PBFT)

Proposed by Castro and Liskov (Castro et al., 1999), PBFT is the most commonly used consensus for permissioned-blockchain (Sukhwani et al., 2017). It has been implemented by Tendermint blockchain, IBM's Open chain, Iris DB and Hyperledger. Reaching consensus in PBFT involves three types of agents (client nodes, commanders and lieutenants) and goes through three phases. Firstly, in the Pre-prepare phase, the client sends a request proposing the new block to the commander that is a defined node, assigned as commander in the network, so it would be verified by lieutenant nodes (also assigned nodes). Then, if this request is valid, a Prepare notification is sent to the commander who is supposed to receive f prepare notifications among $2*f + 1$ possibilities, in the network. Finally, during the Commit phase, the request gets validated as a new log to be added to the ledger. In the example illustrated in Fig 2.9, Sami is the commander and there are three lieutenant nodes. Accordingly, the network can tolerate only one faulty or malicious node which is Nadir in the example. The commander receives the client request and sends a Pre-prepare request to the lieutenants to check its validity. He then receives two out of three Prepare notifications considering the last node as faulty or malicious. Since two third of the lieutenants are honest nodes and agreed on the validity of the client request, the log is then validated and propagated in the network to be added to the ledger.

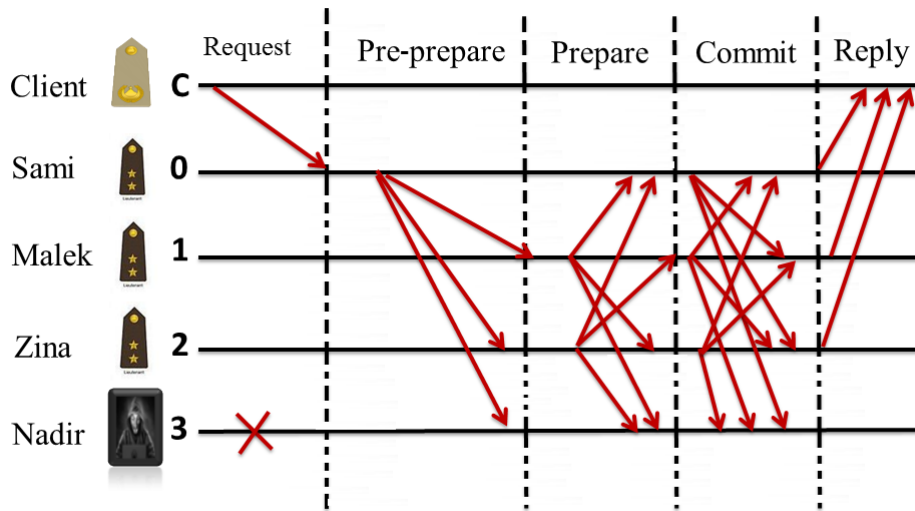


Figure 2.9: PBFT consensus reaching with three lieutenant nodes.

b. Stellar consensus protocol (SCP)

Stellar consensus algorithm is the permission-less blockchain version of the Practical Byzantine Fault Tolerant consensus (PBFT). It is more of a decentralized alternative to PBFT as it is open to the public and allows everyone to participate in the consensus protocol. It uses what is known by the FBA.

PBFT was intended to be a consensus for permissioned blockchain, where a set of validators is predefined in the network, most of the time by the company or organization using the blockchain platform. In Stellar, validators are not predefined but are rather selected in a more decentralized fashion.

This algorithm owes its strength to quorum slices which consist of a list of trusted validators that each node in the network proposes.

Once all nodes in the network are within a quorum slice, validators are those nodes inside the most overlapping slices; in other words, validators are voted out by a group of network nodes as a whole rather than being predefined by a central authority (Mazieres, 2015) as shown in Fig 2.10.



Figure 2.10: Validators voting in stellar consensus protocol.

2.2.6.3 Blockchain consensus performance metrics

Blockchain limits human error and provides more cost and time-effective management by removing intermediaries. It is relevant to note that the performance metrics in Blockchain are different from those in traditional networks. This is because the main concern in Blockchain networks is about reliability of the shared ledger. Throughput is a key performance pointer in conventional networks, although, in Blockchain networks, many consensus algorithms purposely limit the throughput of the blocks committed to

the Blockchain. The main performance metric, aimed to be reached by all Blockchains' consensus remains the integrity of the network and annihilation of malicious attempts such as double spending and tampering with the ledger. Nevertheless, the cost at which this integrity is attained is also a targeted performance endpoint. Meanwhile, researchers aim to achieve it, with the least possible burden without compromising the decentralized aspect of the Blockchain. Another important feature of performance to be considered concerns the mining fairness in the network. Blockchain is a technology for decentralization. However, the more mining new blocks are monopolized by a certain number of nodes, the less decentralized platforms become, in which the worst-case scenario may be the 51% attack that may ruin the integrity of the Blockchain. To capture the fairness of the network (Sediq et al., 2013), a mining fairness index (FI) is defined to demonstrate to what extent all nodes are taking part in the mining process. Intuitively we expect that in an absolute fair network of n nodes, each node must mine $(100/n) \%$ of the total blocks. Given a user number i has allocated an amount of resources, referred to as x_i , Fairness can be computed using Jains Fairness Index formula in Eq 2.4 that is applicable to any resource sharing or allocation problem (R. Jain et al., 1998).

$$FI(x_1, x_2, \dots, x_n) = \frac{[\sum_{i=1}^n x_i]^2}{\sum_{i=1}^n x_i^2} \quad (2.4)$$

The Fairness Index lies between 0 and 1. Consequently, the more (FI) tends to zero, the fairer is the mining process in the network, whereas $FI=1$ corresponds to a

totally fair system where all contenders have been allocated resources equally. In the case of mining, the contenders are the miners competing to add new blocks, while each miner resource allocation is in fact the percentage of block mined by the contender in question.

Motivated by this important metric, we can now look at concrete examples of Bitcoin and Ethereum Blockchain to assess the mining fairness accordingly.

Because in Bitcoin and Ethereum, nodes are pseudo-anonymous, it turns out to be difficult to assess the percentage of blocks mined by a single node, as a single node may be using several different public addresses. However, due to the enormous hashing power required to mine a block, it is doubtful that a single private individual possesses such hashing power. For that reason, there is what is known as mining pools. These are a set of miners combining and joining their hashing power to mine new blocks. The mining reward is then divided in proportion to the node hashing power contribution. In Bitcoin and Ethereum, most mining pools are identifiable and their mining contribution. Then, it is possible to compute the mining fairness with respect to mining pools and their mining contribution, as shown in Tab 2.5 (*Hashrate Distribution*, 2021).

Table 2.5
Mining Pools Contribution in Bitcoin (From February 1st to February 4th, 2022)

Mining pool	Mining pools contribution in Bitcoin (1/02/2022- 04/02/2022)
Unknown	37.39%
F2pool	13.77%
Poolin	12.16%
Antpool	9.66%
Huobi.Pool	8.58%
ViaBTC	6.08%
SluchPool	5.37%
BTC.TOP	2.33%
EMCD pool	1.07%
Novablock	1.07%
OKEXPoool	0.89%
WAYI.CN	0.89%
BTC.com	0.54%
SBI Crypto	0.18%

According to the fairness expression stated in Eq 2.4, the mining Fairness Index of Bitcoin, BTC_FI is

$$BTC_{FI} = 0.36 \quad (2.5)$$

Which means that is unfair to 64 % of the miners. Tab 2.6 also, illustrates mining pools contribution in Ethereum, (From February 1st To February 4th, 2022) (*Bitcoin Average Transactions Per Block*, 2021).

Table 2.6
Mining pools contribution in Ethereum, (From February 1st to February 4th, 2022)

Mining pool	Mining pools contribution in Ethereum (1/02/2022- 04/02/2022)
Ethermine	27%
F2pool	18%
SparkPool	16%
Nanopool	12%
MiningPoolHub	10%
Others	17%

For more simplicity, we considered “others” in Tab 2.5 to be a single mining pool. In such a case, the Fairness coefficient $Ether_{FI}$ for Ethereum would be:

$$Ether_{FI} = 0.90 \quad (2.6)$$

Which means that is unfair to 10 % of the miners. In the same manner, the mining coefficient can be computed for all other Blockchain platforms.

2.2.6.4 Drawbacks of current trend in consensus algorithm development

In reality, blockchain is considered as a fast-paced topic and numerous consensus have been proposed in the few past period. New consensus protocols come into view regularly, and so do the improvements, making the challenges of blockchains’ consensus applicability continual. As for realistic consensus, these are definitely not ideal (flawless). Indeed, they are always subjected to trade-offs which concern the performance, security, and scalability efficiency. Prioritizing a specific feature over another is made

in regards to the specific purpose it is serving. More precisely, when a new consensus algorithm is proposed, it favors miners or validators with specific attributes either it is the activity in the network, the amount of staked money or the hardware capacity. Whenever a new consensus occasionally emerges, it only changes the favorite mining attributes. Regarding the situation with such an approach, those attributes do not differ from circumstances; whatever they are, the same miners with the same attributes are always the more advantageous. As a trend, this would open the rift to risks such as the 51% attack where limited parties can monopolize the system. This became a possible scenario with the existence of mining pools. Fig 2.11 shows that this occurred where some mining pools reach an alarming hashing rate, where for a limited short time, some have even monopolized nearly 51% of the network hashing power. Moreover, those nodes with the most favorite attributes to be eligible to add a new block can abuse this power given the continual possibility to add new blocks to the ledger and include transactions they desire. They can boycott some less powerful users' transactions and never commit them to the ledger. In such a scenario, the less favorite nodes would see themselves powerless and obliged to leave the network anyway. However, a new approach, if attempted, may lead to better performance. The main changes consist of making the favorite miners attributes varying according to circumstances, or favoring less powerful nodes only if they can prove being a victim of abuse such as boycotts. Actually, negligence in committing the valid transaction is a concrete concern in most of blockchain

platforms. In the case of Bitcoin where the maximum block size is 1 megabyte, given the daily average number of transactions per block and the average transaction size per day, one can deduce the average number of transactions in a 1-megabyte block.

From Tab 2.7 [91], (*Bitcoin Block Time Chart*, 2021), it can be seen that even though there are still pending transactions left over, in the Mempool, blocks are not filled to their full authorized potential. This is because if a transaction is left over too long without being confirmed, it is pushed away from Mempool by new transactions coming in. In forum discussions for Bitcoin crypto users [93], (*The Bitcoin Forum Index*, 2021), some users complain about their frustration regarding some of their transactions that have been pending for days, awaiting confirmation. Confronted to this problem, however, the only solution being proposed is most the time to raise the transaction fees to attract miners.

To sum up, the more randomly miners are eligible to mine a block, the fewer the odds for a given party to monopolize the mining process. Similarly, if few favorite nodes face any kind of abuse or bullying by powerful nodes, consensus should enable them to confront it, giving a proof of abuse.

Table 2.7

Blocks are not filled to their full authorized potential despite still having pending transactions left over, in the Mempool.

Date	24/04/2022	05/05/2022	30/05/2022	06/06/2022
Average transaction size (Bytes)	308	479	713	396
Average number of transactions per block	1709	1973	1248	1235
Number of transactions if block size were 1MB	3246	2087	1402	2525
Size of Pending transactions in the Memopool (Kilo-bytes)	80	30	20	0.5

Fig 2.11 shows the market share of each mining pool as the size of the circle over time. The color scale encodes the mining power domination in the Bitcoin network from no domination (0% mining power) in yellow, to complete domination (>50%) in red (Tovanich et al., 2021).

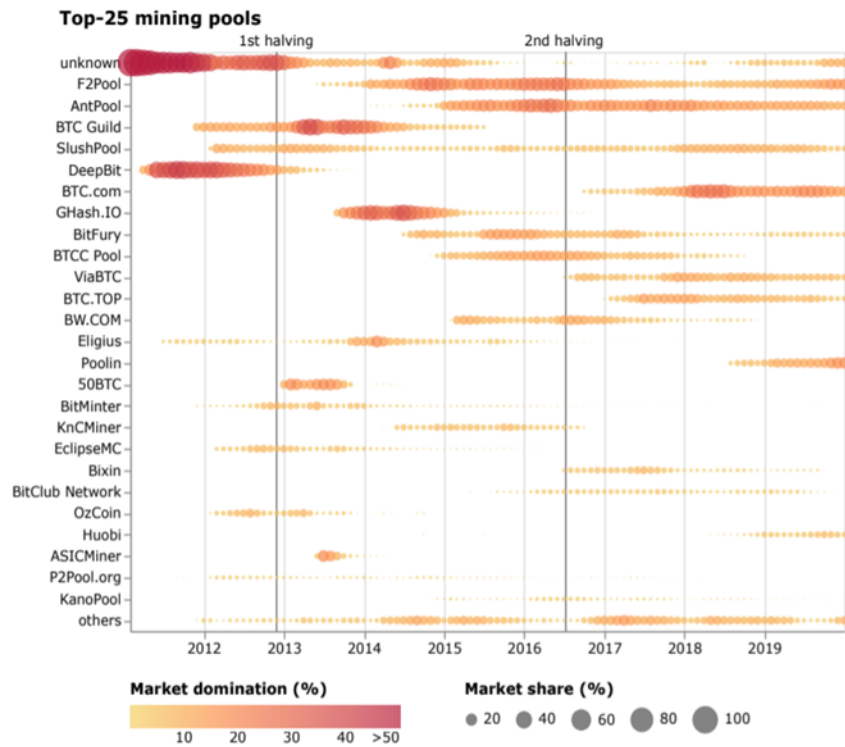


Figure 2.11: Evolution of market shares and domination for the top 25 mining pools.

2.3 Current P2P energy trading tokens platforms

While there are many companies and projects working to make P2P energy trading a mainstream reality, there are three that I would like to highlight — Power Ledger, Grid+ and LO3 Energy.

2.3.1 Power Ledger

(*Power Ledger Whitepaper, 2021*)

Power Ledger aims to bring P2P photovoltaic (solar) energy trading to the world

through the use of easy-to-install hardware, a simple smartphone app, and software based on the Ethereum blockchain. Power Ledger's platform enables fast, low-cost, secure and transparent transactions between multiple parties and the development of dynamic trading ecosystems. The app is easy to use for both prosumers and consumers and enables low-cost, autonomous distributed trading.

The Perth-based company has raised 34 million through its token generation event, AUD, and has already conducted tests in small communities across Australia. It has partnered with companies such as Origin (one of Australia's largest energy providers), Thai government-backed renewable energy developer BCPG, and most recently U.S. nonprofit energy company Helpanswers.

2.3.2 Grid+ (*Grid+ Whitepaper, 2021*)

It's a startup from New York-based blockchain development studio ConsenSys that's trying to disrupt the traditional energy sector from a slightly different angle. They want to remove the middleman from the equation by buying electricity wholesale and selling it to their customers, avoiding the 100 percent markup that utility retailers often make. By using P2P energy technology and "smart meters," Grid+ believes it will be able to better estimate consumers' future demand so that energy can be purchased in advance at a lower price. More accurate data will also allow prosumers to better decide when it is best to sell their energy to Grid+ and could encourage more people to invest in battery storage solutions. In the short term, Grid+'s solution is not quite P2P yet, but in the long

term it should enable P2P capabilities.

Another energy solutions token based on Ethereum's ERC-20 protocol, Grid+ raised more than \$30 million through its token sale last fall. While the company still has a long way to go before it is expected to launch in Texas, it is well positioned to use blockchain technology to disrupt the energy industry.

2.3.3 LO3 Energy (*Microgrid Knowledge, 2021*)

What sets LO3 Energy apart from other projects is that they are innovating in the P2P energy space from a community building perspective. Coupled with Siemens' microgrid software, LO3's blockchain program allows members of the Brooklyn community to buy and sell solar energy. Like other P2P energy trading projects, Brooklyn Microgrid offers economic and protective benefits, but it adds an additional layer of community orientation. The project aims not only to create an example of how P2P energy trading can succeed in the real world, but also to build a strong community behind it.

On a larger scale, LO3 aims to revolutionize the future of energy through a project called Exergy. Exergy aims to bring the technology (the Exergy-compatible meter and mobile app) and spirit that make Brooklyn Microgrid successful to the rest of the world. LO3 aims to create democratized energy marketplaces around the world and turn many of the people who were once consumers into the new energy producers. LO3 opened its ERC-20-compatible XRG token sale to accredited retail investors in late November, and more accredited public pre-sale and general sale rounds will follow. In

December, Siemens invested an undisclosed amount in the company, strengthening its existing relationship with LO3.

The peer-reviewed related studies may be broken mainly into two parts. The first part concerns blockchain backgrounds as a revolutionary technology that has shaped the future of banking and IoT, with the appearance of smart contracts and several distributed consensus protocols and beyond. In this regard, the advent of blockchain technology has been outlined for the first time in 1991 by Stuart Haber and W. Scott Stornetta, to implement a system that inhibits any form of tampering with document timestamps. Moreover, it has allowed digital information to be recorded, distributed, and securely exchanged. However, this would not be detailed more in this Chapter since a plethora of substantial works is generously provided in the literature (Carson et al., 2018; Khan et al., 2021). The second wedge, for its part, gives a fuller insight into the most important features utilizing blockchain technology for the energy sector (Thukral, 2021). The threat of rising energy bills is of significant concern for businesses, worldwide. P2P-based energy trading represents a challenge that can mitigate the specific electricity consumption pricing at a given site. Meanwhile, renewable and distributed energy resources are definitely at the core of any business sustainability schedule to ensure high investment return prospects (B. Wang et al., 2021; Lu et al., 2019; Jenkins et al., 2009). Peer-to-peer energy trading gives prosumers a real alternative for their electricity supply as well as the choice to decide whom they sell it to. Prosumers are required to communi-

cate with each other for negotiating energy prices and payment transactions. Exporting back to the grid excess energy, for a small charge rate has become old-fashioned as modern and future-oriented energy systems allow clients to manage and control profitably the way their resources are distributed using microgrids and how this can affect their electricity bills. Recorded P2P energy trade goes back to the year 2016, in Brooklyn, New York, when Ethereum blockchain was used to sell extra generation of a few kilowatt-hours from a resident's solar panels energy to his neighbor. Several residential trials have followed since, whereas some companies are rushing in the P2P trade across the world. Global enterprise companies including LO3, SonnenFlat, Power Ledger, Grid+, Suncontract, and Eemnes Energie have tangible use cases for blockchain technology using platforms that have drastically changed and revolutionized the way energy is shared and distributed (Ahl et al., 2019). In addition to sustainability, suitability and profitability also play a decisive role in any new solution, which represents an active topic of research.

In what follows, we will be highlighting the major role of blockchain in the use of energy trading (S. Wang, Taha, et al., 2019; Z. Li et al., 2019; Noor et al., 2018). P2P trading being a new trend in technology, requires overwhelming adoption by a large mass to claim success. This is achieved only if the needs and the possibilities of the individual are seriously taken into account. Using blockchain combined with smart contracts for reliable tamper-proof transactions are catching particular interest among

the researchers' community and industry. As far as smarter and stronger feasible technologies with enhanced capacity and flexibility for power transmission and distribution (T & D) systems are concerned (ISGAN, 2022), the blockchain-based technology can be identified in two main classes of particular interest (Pop et al., 2018); the first class, referred to as Electrical energy trading, comprises principally P2P energy sales where today's energy systems are changing in ways that make old strategies obsolete (Park & Yong, 2017; Jogunola et al., 2017; Di Silvestre et al., 2020). The second class can be observed through the role of Certification of Renewable Energy emissions, which enables the prosumer in an aggregation program to be granted recognition as a participant. Additionally, Demand Response tracking, where the blockchain records the amount of green power that is injected into the grid, manages the remuneration mechanisms and ensures transparency. Moreover, this would allow the operating point of the power system to be continuously adjusted.

A thorough review that elucidates what blockchain can do for power grids is found in (Foti & Vavalis, 2021). The authors have considered an in-depth survey of the use of blockchain-based technology in the energy sector in general and the power grid especially, through a systematic categorization according to the field of activity, regardless whether this is coming from start-up companies or big technology firms. Meanwhile, a comprehensive review (Thukral, 2021) analyzed the current trends in the energy market and how blockchain technology can contribute to this line. The authors

in (Siano, 2014) have reviewed the demand response in smart electricity grids equipped with renewable energy sources. A critical survey was carried out in (Pinson et al., 2014) that spotlights the benefits and challenges of electrical demand response. Furthermore, the work in (Dalhues et al., 2019) has shown how flexibility and forecasting contribute to reducing the need for grid extension, but this would require strengthened monitoring of the distribution grids. In other works, such as in (Song et al., 2021), the authors have proposed a smart contract-based P2P energy trading system with a dynamic pricing model. In reality, several industries are investing in blockchain technology, albeit the expectations of researchers are very high, indicating the continual challenges of blockchains' applicability, in the energy sector particularly. According to the literature reviewed in the context of blockchain-enabled energy trading, it is noted that most of the p2p blockchain energy trading proposed platforms are based on energy tokenization. Energy asset-based tokens are created upon a defined energy amount generation. Created tokens can be traded, such as brikCoin (*Whitepaper BRIKCOIN*, 2017), or redeemed as energy, such as WePower (*WePower White Paper*, 2016). It is noticed through the review literature that in most of the proposed P2P energy platforms, the trading takes part between the prosumers and consumers. Prosumers' energy is injected into the power grid, which is metered and tokenized. Tokens are created using an asset-based token protocol such as the ERC20 protocol in Ethereum. A token smart contract is created with a certain amount of total supplied tokens, which are set to have

an abstracted value in terms of energy. Energy aggregated to the utility power grid (by the prosumers) is redeemed in the form of tokens, which are transferred to them. Tokens can then be traded according to their value in the market (Bitquery, 2022a,b,c). Accordingly, the initial smart contract defining the energy token is set with a total supply of tokens, which means, if new tokens need to be created, then, a new smart contract needs to be deployed with additional supply, which is very costly. The gas cost to deploy the EWT energy token is 1,274,198 with a cost of \$171.62 (Team, 2022). Moreover, transferring tokens is costly as well. In addressing such gaps, we wish to consider in our work that prosumers cooperate as a single provider and the energy they aggregate to the utility grid is tracked in a smart contract deployed by the DSO. This amount of energy is mapped to the issuer prosumer's account as its energy balance. Instead of a token transfer, prosumers' energy balance is updated by a mere contract call that updates a mapping, rather than a costlier financial token transfer. The energy balance of the prosumers represents the amount of energy the DSO owes them. Then, consumers can buy energy from prosumers. If the prosumer (seller) has enough energy balance to supply the demand, the payment is made from the consumer (buyer) to the seller set of prosumers. Consequently, the energy being sold is transferred from the DSO to the consumer as a debt that is paid back to the respective prosumer. His energy balance is then updated accordingly (Muzumdar et al., 2021; Gai et al., 2019). Mainstream token-based P2P energy trading platform bears another main drawback. In fact, in such a

situation, the client does not pay per use for the energy he purchased. After the energy token is bought, it can be then redeemed as energy. However, the respective corresponding energy amount is transferred to the client and he is charged accordingly, regardless of to which extent the purchased energy is consumed. Additionally, most of the time, purchased energy is not consumed to the fullest (Devine et al., 2019). It is as worth mentioning that some substantial research papers have been also conducted using smart contract-based electricity consumption billing system, outside the context of P2P energy trading, where household electricity bills' computation and payment are monitored by a smart contract in a traditional centralized DSO–client architecture (Hlaing & Nyaung, 2019; Gür et al., 2019; Aiman et al., 2018).

This paradigm suffers from two drawbacks that we aim to address:

- Purchased energy under consumption: When the energy is redeemed using its corresponding energy token, consumers are not refunded in case of under-consumption. This issue can be deduced from the results presented in (Devine et al., 2019), where the author laid out the purchased energy versus its consumption using his proposed energy token, namely ForDelToks. It is seen that in most cases, the energy purchased is not fully consumed. Similarly, another work presented in (Cali & Fifield, 2019) also highlights this issue. As a matter of fact, in the result presented by the author in his proposed token-based energy trading platform, it can be noticed that the seller tokenized surplus energy is mostly under-consumed by

its respective buyer. However, the author proposes that the excess energy bought be accumulated to the surplus energy generated and tokenized, when it reaches the threshold of 150 watts. This solution is not appropriate where the power is purchased by a pure consumer. In such a case, unused energy purchased is wasted.

- Token handling gas cost: Another concern in token-based energy trading systems, is the high gas cost of token manipulation to both deploy and transfer (Castellanos et al., 2017).

In this direction, we propose the present study to tackle the token manipulation gas cost overhead using an energy balance-based trading scheme, as well as a pay-per-use smart meter-based energy trading, as a solution to the highlighted problem under consumption.

2.4 PoL to secure blockchain applications

In recent years, the 'smart city' revolution has paved the way for innovative solutions to improve living standards, centered on the Internet of Things, leading to real-time interactions (Asuquo et al., 2018). Indeed, the rise of wireless devices has been facilitated by the emergence of a new class of applications known as Location Based Services (LBS). Location Based Services is a growing technology that focuses on providing GIS and spatial data through mobile and field devices. Location-based apps and services have gained popularity in recent years due to the variety of useful services they provide.

These include activity tracking apps, location-based services, cognitive radio networks (CRNs), and location-based access control systems, which are used in both established and emerging domains. Navigation software, social networking services, location-based advertising, and tracking systems are all examples of location-based services (Raper et al., 2007; Gupta & Rao, 2017). These location-based applications/services may also be used in other access control systems that require a proximity sensor, such as in a critical healthcare system or for entertainment purposes, where these systems may only provide content to users who are in a particular area of interest while charging users who are in a different location (Saroju & Wolman, 2009).

For many current applications, it is critical that users prove their location to the services. Users may misrepresent their location in order to receive rewards (Z. Zhang et al., 2013; Pham et al., 2015). To confirm user legitimacy, transparency and control over the use and sharing of location data are required. Depending on the system architecture, the mechanisms of Location Proof (LP) are divided into two groups: centralized and decentralized. The most recent centralized mechanism was proposed by Javali et al. (Javali et al., 2016) proposed. In their paper, the authors propose a novel technique to create a location proof for mobile users and to verify the location information by application services. Their approach takes advantage of the unique properties of the WiFi signal and uses a fuzzy vault technique that is potentially secure. However, their work does not consider scenarios in which Access Points APs collaborate with adversaries to

create a fake location proof, or services that deny access to honest users while granting illegal access to dishonest users. In addition, the proposed architecture uses a number of entities, which makes it costly to implement. In decentralized architectures, there is no need to use trusted access points to generate LP. Therefore, these architectures tend to be less costly to develop than centralized mechanisms. In decentralized systems, on the other hand, architectural design ingenuity is more challenging to ensure the reliability of the required LPs. Several researchers have studied and proposed various techniques for securing LPs in smart systems, all of which use blockchain smart contracts that have become essential for today's decentralized applications. The initial decentralized architectures presented in the literature face a number of security and privacy issues, such as target-target collusions (terrorist scams) (Zhu & Cao, 2011; Boureau et al., 2015; Boureau & Vaudenay, 2015), target-prover collusions (Gambs et al., 2014), and site privacy threats (Kounadi et al., 2018). In (Nosouhi et al., 2020), the authors address the threat of target-target collusions by proposing a fixed-time frame for the transmission of Proof of Location (PL). Although the ingenuity of this approach, which has definitely reduced the possibility of target-target collusion, as shown by the results obtained by the authors and presented in the paper, such an approach is not infallible, since the execution time of smart contract transactions depends entirely on the time it takes to mine them in a block, which is directly related to the difficulty of block mining in PoW consensus-based blockchains.

It is worth noting that reducing the mining difficulty can lead to gaps in the reliability and security of the blockchain. In (Corp, 2021), it is proposed that location claims are curated by tokens. The location claim is subject to a vote that decides the validity of the PL claim, and if the claim has more rejecting nodes than approving nodes, the claimant loses its deployed tokens. This approach also relies on the integrity of the voters, which can be damaged especially if the destination has financial significance. To ensure the reliability of blockchain-based PoL, we believe it must be designed specifically for the application in which it will be used. In our work, we presented a secure yet decentralized PoL designed specifically for blockchain energy trading platforms. In doing so, we used Game Theory applicable to blockchain energy trading and decentralized random selection based on smart contracts.

The biggest challenges to decentralized PoL are malicious collusion, which can be divided into the following categories:

- Target-target collusion: it occurs when two malicious target nodes collude with each other at different sites. A target node may send a signed PoL request to the colluding node, which then forwards it to the assigned provers on its behalf. The LP calculated by the provers would be based on the location of the collaborating node and would be appropriate for the second target node.
- Prover-prover collusion: In this situation, a dishonest prover conspires with a remote malicious witness and provides him with a fake location proof. This form of

malicious collaboration is most difficult to handle in degenerate untrusted positioning. Most implemented proof of location are centralized, i.e., there are defined anchor nodes that are trusted as location verifiers. There is a growing interest in implementing blockchain-based decentralized location proofs in a trustless architecture. However, as mentioned in (Nosouhi et al., 2020), the main challenge in blockchain-based decentralized proof of location is the prover-prover collusion. What we propose is a fully trustless Blockchain-based proof of location that is specifically designed for Blockchain-based energy application and robust against prover-prover collusion.

2.5 Demand response control in P2P energy trading system

The current tremendous expansion of distributed energy sources and the democratization of electricity generation through conventional and renewable sources such as wind turbines, solar energy, and advances in energy storage capacity (Z. Wang et al., 2021; Sotkiewicz & Vignolo, 2007), combined with the advancement of digital communication and control technologies, have enabled the transition from traditional grid systems to what is known as a smart grid, in which traditional consumers have evolved into proactive prosumers that can join a grid branch of the larger distribution network and contribute to energy demand management within the grid they have joined, either with traditional fuel-based energy generators or renewable sources (Rinaldi et al., 2019; Wu & Conejo, 2022). In addition, the integration of blockchain into the smart grid

and the functionalities of smart contracts enabled the emergence of peer-to-peer energy trading (Merrad et al., 2022; Junlakarn et al., 2022), where prosumers can trade their energy generated and injected into the grid in a decentralized manner. It should be noted that the reduction of influence and control by a central authority is paramount in all blockchain-based applications (Shabani, 2019). Similarly, the decentralization of Blockchain-based P2P energy trading platforms and the safe reduction of the authority of the DSO is an aimed target (Han et al., 2020; Gajić et al., 2022). Since prosumers mainly use renewable energy sources, energy and climate policies promote and support such energy platforms to meet both increasing energy demand and sustainable climate goals (“Community energy strategy: Full report”, Dec. 2014; Afzal et al., 2020). On the other hand, end users can reduce their electricity costs and gain lucrative benefits, making such platforms a win-win scenario. In P2P energy trading platforms, prosumers are rewarded accordingly for their delivered energy through crypto-tokens, which are either fungible or non-fungible (Karandikar et al., 2021) and delivered through appropriately designed smart contracts. In an integrated system, the utility has complete control over the location and connection of distributed energy resources (DER) and will seek to optimally manage demand, while in unbundled systems, each prosumer seeks its own benefit and its goal is to maximize its lucrative profit, according to market rules. Thus, from this perspective, a prosumer that joins a grid and the grid operator can potentially have conflicting goals. The DSO analyzes grid operations and grid investments in

terms of peak power flows, while the prosumer sees its revenue in terms of aggregated energy exchange. Moreover, the flexibility of the smart grid, where new prosumers can constantly join or leave the grid, poses a challenge to the effective implementation of demand response control and management systems, which require less flexible deployment of hardware (Jayachandran et al., 2022). Moreover, a demand response control and management system deployed by DSOs would not be considered a decentralized solution and would run counter to the consortium-based decision-making principle of blockchain applications. One of the most widely used demand response solutions is the OPF based control system. The OPF solution aims to achieve a steady-state operating point in the grid that reduces power generation costs while satisfying demand and operating constraints (Z. Wang & Anderson, 2021). The approach to optimize the power injection within grids considering thresholds is generally challenging because it is a nonlinear and non-convex optimization problem. With renewable energy sources in the grid, the OPF problem becomes even more complicated in both formulation and solution because the generation capacities of the renewable energy sources, which are part of the constraints of the problem, are unpredictable (Riaz et al., 2021). Therefore, many researchers have addressed the formulation and solution of the OPF problem for hybrid grids and presented a number of ingenious models (Nusair & Alasali, 2020; Hassan et al., 2021). Most of the works have focused on presenting realistic mathematical models or algorithms to solve the OPF problem for hybrid power sources. However, beyond

that, little work has been done on how the computed OPF solution can be used for decentralized power generation control in smart grids. So far, the established scheme is to use control units (hardware and software), which does not go in the direction that aims to minimize DSO authority.

CHAPTER THREE

RESEARCH METHODOLOGY

3.1 Introduction

In this chapter, we will explain the general inspiration and approach in developing the fully decentralized pay-per-use P2P energy trading platform using proof-of-location identification and based on the consortium for optimal demand response management.

The culmination of the study is the development of a decentralized solution where no central authority controls the flow of electricity. It is a cost-effective solution considering the cost of providing the hardware and the wages of the staff for continuous maintenance and monitoring. This study also presents a blockchain-based demand-side management for smart grids and pay-per-use energy trading platforms, which uses an optimal power flow control based on smart contracts, with no central authority in charge of controlling the generation of the different energy sources in the grid and no control hardware deployed by the distribution system operator.

3.2 Research framework

This section first describes the general framework and methods for implementing the proposed blockchain pay-per-use P2P energy trading platform, which is our first ob-

jective. It describes how the transition from a token-based trading architecture to a balance-based architecture was accomplished, allowing energy buyers to be billed based on consumption and solving the problem of non-consumption of purchased energy in token-based platforms. All the steps that make this new paradigm possible are described. Second, it explains how to secure prosumers' energy contribution through a novel, fully decentralized PoL specifically designed for blockchain energy applications, which is our second goal. Finally, this section also explains how energy demand is managed in a fully decentralized manner to ensure that greedy prosumers do not incessantly oversupply the network just to maximize their profit, which is the third objective.

3.2.1 Proposed pay-per-use blockchain P2P energy trading platform

Here, we are presenting a P2P energy trading platform where energy can be traded in a decentralized manner on an Ethereum private blockchain. The energy traded on the platform is electrical energy, thus the two terms energy and electricity as well as client and consumer will be used interchangeably. The key features of the Proposed P2P energy trading system are:

- **Dynamic Time of Use pricing:** This would maximize profit and enable consumers to get an appropriate and accurate demand response. The tariffs are proposed ahead of each trading period, based on energy consumption predictions using machine learning techniques (S. Wang, Taha, et al., 2019). The tariffs are then imple-

mented in the smart contract to enable autonomous billing. A new smart contract is created ahead of each upcoming trading round, with updated tariffs based on the trading round electricity consumption predictions. Moreover, the tariffs are made known immediately after each trading round and their validity can be verified by all participants, since each one would be having his copy of the shared ledger containing the training set data. Indeed, each participant is aware of the state history representing the energy consumption data that is used in the prediction process. Based on the predictions, ToU ranges are defined using the K-mean clustering algorithm, implemented on a smart contract, and executed on the blockchain. This ensures that the ToU ranges are reliable and bias-free, since they are defined by an independent autonomous smart third party.

- **Prevention against fraudulent behavior:** Considering energy is traded online, it is critical to ensure that the same energy is not sold more than once. Clients must as well upload valid data of their energy consumption. To achieve such a purpose, we propose in the present study, that the DSO plays the role of a privileged participant, which is allowed certain credentials, making our proposed solution autonomic in nature instead of fully autonomous. First, only approved nodes can join the trading platform as consumers. These nodes need to be certified by the DSO before being able to subscribe to any trading round. Any uncertified accounts have their subscription requests automatically denied. The DSO per-

forms random checks on consumer nodes to detect any fraudulent or technically faulty ones. The DSO is the only agent given the credential to decertify accounts accordingly. Moreover, a customized consensus algorithm is proposed to spot suspicious nodes. Furthermore, a PoL protocol is implemented to confirm the aggregated data source.

- Automatic and autonomic running: Ethereum blockchain is used to implement the trading procedure through smart contracts, ensuring the forcible payment when demand is supplied, the forcible balance return in case of premature unsubscription, as well as the independent smart contract-based billing.

One can identify a few challenges in developing such a platform. The first challenge is to be able to accurately bill the consumers for the energy they consumed. To achieve this, prosumers need to be the only supplier from the start to the end of the trading round. Trading rounds have a certain duration, each one is administrated by a smart contract that sets the electricity tariffs. For this purpose, prosumers and micro-grids' owners collaborate to form a single provider, which would be a full-time supplier to the subscribed customers for an entire trading period. The second challenge is concerned with the prosumer's power generation, which is based on renewable energy and low voltage energy sources. It is a concern that it can be not sufficient to supply for consumers on a continuous period. Yet, this could be seen as a fake problem. Actually, in a research report, published in May 2017, by the European Commission (Dos

Santos Silva, 2017), they provide the total potential capacity of residential solar PVs, based on the proportion of available free space that can be used for the PVs installation. A graphical representation of residential PV electrical generation potential compared to the total national electricity generation capacity in the EU is illustrated in Fig 3.1 (Dos Santos Silva, 2017; *Poland - Country Commercial Guide*, 2019; *National statistics, Digest of UK Energy Statistics (DUKES): electricity*, 2021; *Installed capacity for electricity plants in France in 2020, by energy source (in megawatts)*, 2020; *Portugal Europe RE SP*, 2022; Institut Fraunhofer pour les Systèmes Energétiques Solaires ISE à Fribourg , 2021; *Energy in Finland*, 2020; Fernández, 2021; *Greece Europe RE SP*, 2022; Alves, 2019). It can be seen that, if exploited to its fullest, energy generated by households using only PV solar panels can be tremendously significant. Hence, residential electricity generation has great potential and in some countries such as the UK, and Belgium, it can reach 50% of the total energy being generated by the respective national provider.

Further studies also showed that, if prosumers formed renewable energy cooperatives, the amount of energy generation capacity would strongly resonate. In this vision, extensive research that has been conducted in (Bauwens et al., 2016) revealed data in favor of our claim. It is stated, that 15,000 households own shares in wind electricity generation cooperatives, which consist of 40% of the electricity generated by wind, in Denmark. Germany as well played an essential role in the development of the RE market within Europe. In this regard, community ownership is estimated to have a share

of almost 20% of the total onshore wind power.

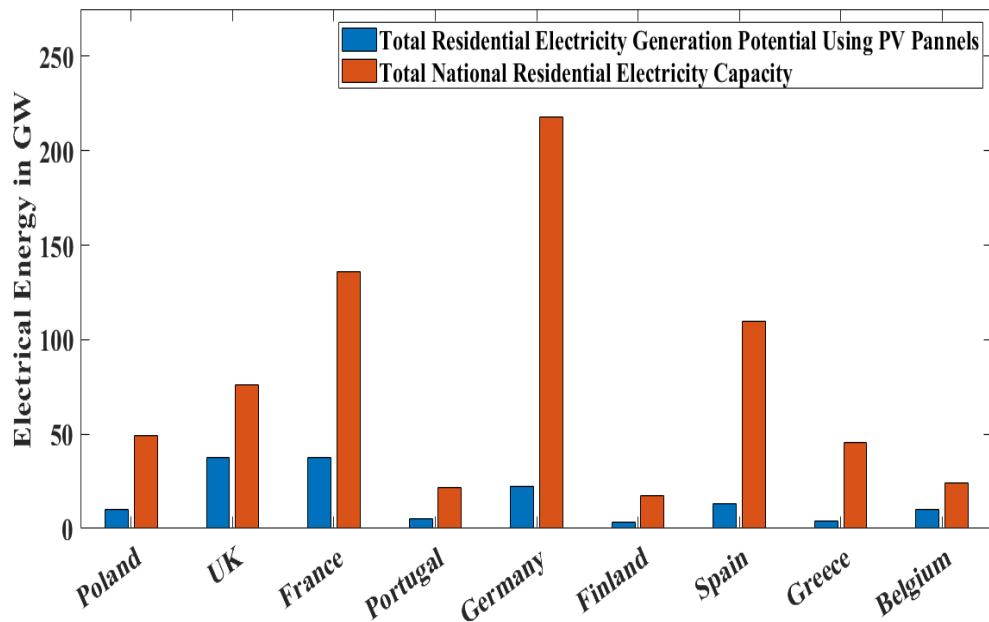


Figure 3.1: Residential PV electrical generation potential versus the total national electricity generation capacity in the EU.

Another aspect of developing a smart contract for energy trade is local legislation and the legal framework of the country where it aspires to be implemented. How each participant is considered from a legal point of view is inferred according to his role and his interactions with the rest of the participants. This implies a set of rules and regulations that they have to abide by. For example, in Germany, if a prosumer is considered a business, he has to contribute to the balancing of the grid (the German Energy Industry Act) (Hasse et al., 2016) and add a withdrawal policy, according to the Consumer Rights Act in the EU (Legislation.gov.uk, 2022). These regulations must be implemented and enforced by the smart contract to ensure that each participant complies with the laws

to which they are bound. A typical example is the Enerchain project (Merz, 2020), which is an energy trading blockchain for peer-to-peer transactions, specially designed to be compliant with the EU's Regulation on Wholesale Energy Market Integrity and Transparency (REMIT). On the other hand, some countries' legal systems, obstruct the decentralized energy P2P market. For example, according to the EU Renewable Energy Directive (EUR-Lex, 2019), a "renewables self-consumer" consumes local energy that is generated behind the meter. Similarly, the Dutch law requires a provider certification for the sale of power to the electricity grid (Butenko, 2016). In such cases, the legislative framework should be amended accordingly. In this direction, one can cite the BEST (Blockchain-based decentralized energy market design and management structures) project in Germany, which is working to establish a bidding system for the open-source electricity market supported by the German Federal Ministry for Economic Affairs and Energy (Hasse et al., 2016) (*Blockchain for a smart energy market: BEST*, 2022; Kirli et al., 2022).

On the other hand, reliability is also given priority as the third challenge. Ensuring the electricity consumption data committed by the consumers be secure, remains of considerable importance. In the following subsection, the design of the proposed platform and its architecture is vividly explained, so that the approach that is adopted in our research to solve this challenge is made clearer.

3.2.1.1 Platform design

In this part, the design of the proposed P2P energy trading platform is delineated. It first details the proposed model defining the ToU ranges using GRU time-series predictions and K-mean clustering. Then, the key entities that are involved are described, their roles underlined, and how they possibly interact with each other, as well as the set of rules governing these interactions. This section also presents the entities (Smart Contracts) that ensure that all interactions are bound by the agreed-upon rules and that each interaction results in the appropriate consensual outcome.

3.2.1.1.1 ToU Ranges Using GRU Model Predictions and K-Mean Clustering

The aim is to dynamically set the ToU peak and off-peak ranges. Based on past consumers' electricity consumption data available on the shared ledger blockchain, future electricity consumption predictions are then made, using the GRU model, subsequently, the ToU ranges for the next trading round are set by performing K-mean clustering on the blockchain as illustrated in Fig 3.2. The purpose behind it is to have a transparent, decentralized dynamic demand response.

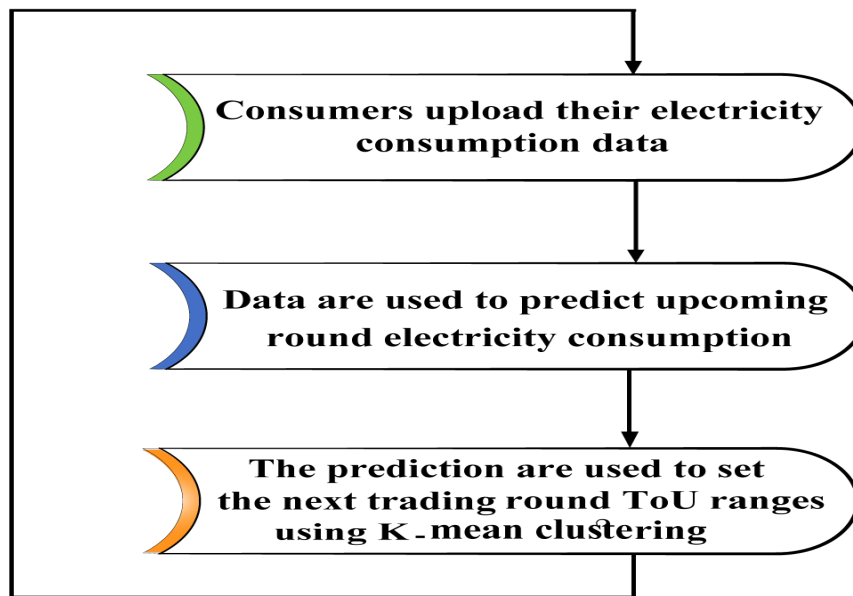


Figure 3.2: The proposed model defining TOU peak and off-peak ranges.

Energy Consumption Time Series Forecasting Using GRU

Our forecasting model is using an open-source dataset from PJM, in Megawatts (*Over 10 years of hourly energy consumption data from PJM in Megawatts, 2022*). The data consist of hourly records of electricity consumption in the USA for a period of time ranging from 31 December 2004 to 1 February 2018. They are structured into double columns of 121,274 values with no empty values. The energy consumption histogram provided in Fig 3.3 shows a normal distribution and hence, normalization of the dataset is not required.

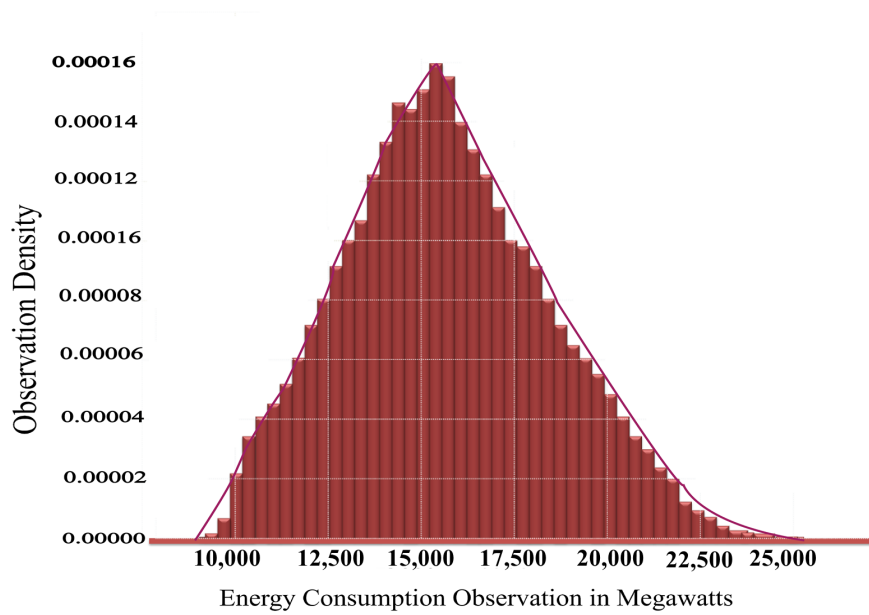


Figure 3.3: Distribution of the observations in the dataset.

The model is trained by utilizing the GRU network. Numerous research works including (BİŞKİN & ÇİFCİ, 2019; Tokgöz & Ünal, 2018) have successfully applied the GRU network for predicting electricity consumption. The aforementioned studies have investigated the performance of several machine learning predictive models to forecast electricity consumption. Results have shown that the GRU-based method achieves more accurate predictions, making it the reason behind our choice.

GRU is part of a special model of a recurrent neural network that seeks to use connections through a sequence of nodes to perform machine learning tasks associated with memory and clustering. It was developed in 2014 by (Cho et al., 2014) to solve the vanishing gradient problem that occurs in a normal recurrent neural network. GRU can also be considered a long short-term memory (LSTM) with a forgetting

gate,(Gers et al., 2000) but has fewer parameters than LSTM because it lacks an output gate.(Britz, 2015) GRU’s performance on certain tasks of polyphonic music modeling, speech signal modeling, and natural language processing was found to be similar to that of LSTM.(Ravanelli et al., 2018)(Su & Kuo, 2019) GRUs have been shown to perform better on certain smaller and less frequent data sets.(Chung et al., 2014)(Gruber & Jockisch, 2020)

There are several variations of the Full Gated Unit, where gating is done using the previous hidden state and bias in various combinations, as well as a simplified form, the Minimal Gated Unit.(Chung et al., 2014) The operator \odot denotes the Hadamard product in the following.

Fully gated unit

Initially, for $t=0$, the output vector is $g_0 = 0$

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z) \quad (3.1)$$

$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r) \quad (3.2)$$

$$\hat{h}_t = \phi_h(W_h x_t + U_h (r_t \odot h_{t-1}) + b_h) \quad (3.3)$$

$$h_t = z_t \odot \hat{h}_t + (1 - z_t) \odot h_{t-1} \quad (3.4)$$

Variables:

- x_t : input vector
- h_t : output vector
- \hat{h}_t : candidate activation vector
- z_t : update gate vector
- r_t reset gate vector
- W, U and b : parameter matrices and vector

Activation functions

- σ_g : The original is a sigmoid function
- ϕ_h : The original is a hyperbolic tangent

Alternative activation functions are possible, provided that $\sigma_g(x) \in [0, 1]$. Alternate forms can be created by changing z_t and r_t (Dey & Salem, 2017)

- Type 1, each gate depends only on the previous hidden state and the bias.

$$z_t = \sigma_g(U_z h_{t-1} + b_z) \quad (3.5)$$

$$r_t = \sigma_g(U_r h_{t-1} + b_r) \quad (3.6)$$

- Type 2, each gate depends only on the previous hidden state .

$$z_t = \sigma_g(U_z h_{t-1}) \quad (3.7)$$

$$r_t = \sigma_g(U_r h_{t-1}) \quad (3.8)$$

- Type 3, each gate is computed using only the bias.

$$z_t = \sigma_g(b_z) \quad (3.9)$$

$$r_t = \sigma_g(b_r) \quad (3.10)$$

Minimal gated unit

The minimal gated unit is similar to the fully gated unit, except the update and reset gate vector is merged into a forget gate. This also implies that the equation for the output vector must be changed:(Heck & Salem, 2017)

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \quad (3.11)$$

$$\hat{h}_t = \phi_h(W_h x_t + U_h (f_t \odot h_{t-1}) + b_h) \quad (3.12)$$

$$h_t = (1 - f_t)_{t-1} + f_t \odot \hat{h}_t \quad (3.13)$$

Variables

- x_t : input vector
- h_t : output vector
- \hat{h}_t : candidate activation vector
- f_t : forget vector
- W, U and b : parameter matrices and vector

Content-Adaptive Recurrent Unit

Content Adaptive Recurrent Unit (CARU) is a variant of GRU, introduced in 2020 by Ka-Hou Chan et al. (Chan et al., 2020). The CARU contains the update gate like GRU, but introduces a content-adaptive gate instead of the reset gate. CARU is designed to alleviate the long-term dependence problem of RNN models. It was found to have a slight performance improvement on NLP tasks and also has fewer parameters than the GRU. (Ke & Chan, 2021)

In the following equations, the lowercase variables represent vectors and $[W; B]$ denote the training parameters, which are linear layers consisting of weights and biases, respectively. Initially, for $t = 0$, CARU directly returns $h^{(1)} \leftarrow W_{vn}v^{(0)} + B_{vn}$; Next, for

$t > 0$, a complete architecture is given by:

$$x^{(t)} = W_{vn}v^{(t)} + B_{vn} \quad (3.14)$$

$$n^{(t)} = \phi((W_{hn}h^{(t)} + B_{hn}) + x^{(t)}) \quad (3.15)$$

$$z^{(t)} = \sigma(W_{hz}h^{(t)} + B_{hz} + W_{vz}v^{(t)} + B_{vz}) \quad (3.16)$$

$$l^{(t)} = \sigma(x^{(t)}) \odot z^{(t)} \quad (3.17)$$

$$h^{(t+1)} = (1 - l^{(t)}) \odot h^{(t)} + l^{(t)} \odot n^{(t)} \quad (3.18)$$

Note that it has $t \leftarrow t + 1$ at the end of each recurrent loop. The operator \odot denotes the Hadamard product, σ and ϕ denotes the activation function of sigmoid and hyperbolic tangent, respectively.

Variables

- $x^{(t)}$: It first projects the current word $v^{(t)}$ into $x^{(t)}$ as the input feature. This result would be used in the next hidden state and passed to the proposed content-adaptive gate.
- $n^{(t)}$: Compare to GRU, the reset gate has been taken out. It just combines the parameters related to $h^{(t)}$ and $x^{(t)}$ to produce a new hidden state $n^{(t)}$.
- $z^{(t)}$: It is the same as the update gate in GRU and is used to the transition of the hidden state.

- $l^{(t)}$: There is a Hadamard operator to combine the update gate with the weight of current feature. This gate had been named as content-adaptive gate, which will influence the amount of gradual transition, rather than diluting the current hidden state.
- $h^{(t+1)}$: The next hidden state is combined with $h^{(t)}$ and $n^{(t)}$.

Data Flow

Another feature of CARU is the weighting of hidden states according to the current word and the introduction of content-adaptive gates, instead of using reset gates to alleviate the dependence on long-term content. There are three trunks of data flow to be processed by CARU:

- content-state: It produces a new hidden state $n^{(t)}$ achieved by a linear layer, this part is equivalent to simple RNN networks.
- word-weight: It produces the weight $\sigma(x^{(t)})$ of the current word, it has the capability like a GRU reset gate but is only based on the current word instead of the entire content. More specifically, it can be considered as the tagging task that connects the relation between the weight and parts-of-speech.
- content-weight: It produces the weight $z^{(t)}$ of the current content, the form is the same as a GRU update gate but with the purpose to overcome the long-term dependence.

In contrast to GRU, CARU does not intend to process those data flow, instead dispatch the word-weight to the content-adaptive gate and multiplies it with the content-weight. In this way, the content-adaptive gate considers of both the word and the content.

Definition of ToU Ranges

What is proposed, is that the energy ToU ranges and tariffs are defined ahead of each trading round; the tariffs are then defined accordingly in the smart contract arbitrating the interactions between the provider and the set of consumer clients. The novelty is that the clustering is implemented and performed on a smart contract. Interestingly, smart contracts are seen not to be suitable for implementing heavy computational algorithms due to code execution cost, which is abstracted in terms of gas. However, implementing a clustering algorithm to define ToU ranges on a smart contract is interesting in the sense that this would permit further decentralization, where ToU ranges are defined by a smart independent entity, ensuring their integrity and transparency. For this purpose, the K-mean clustering algorithm was implemented with $K = 2$ for two clusters, which are the electricity demand peak and off-peak, on a Solidity smart contract. It was compiled and tested on a Remix IDE platform to define ToU ranges on both predicted and real electricity consumption, for each round. The smart contract execution cost was also assessed and benchmarked to the execution cost of other decentralized applications' smart contracts presented in other published works. The aim is to assess if performing K-mean clustering on a smart contract is realistic in terms of gas consumption. The K-

mean algorithm flowchart is illustrated in Fig 3.4.

So, the clustering is performed on a set of points within coordinates (x,y), where x designates the time of the day, ranging from 00:00 h to 23:00 h and y is the mean average of the electricity consumed on a particular hour of the day, during 100 days. For each hour of the day, the corresponding y value is computed from both predicted and real consumption data for each trading round according to the pseudocode in Algorithm 3.1.

Once the arrays X and Y are both defined, the clustering is performed according to pseudocodes in Algorithms 3.2 and 3.3.

Algorithm 3.1 Hourly mean average electricity consumption for one trading round

```
1:  $Sum \leftarrow 0$ 
2:  $size \leftarrow 0$ 
3: for  $i \leftarrow 0$  to 24 do
4:   for  $j \leftarrow i$  to 100 do
5:      $Sum \leftarrow dataset[j]$ 
6:      $j \leftarrow j + 24$ 
7:      $size \leftarrow size + 1$ 
8:   end for
9:    $y[i] \leftarrow sum/size$ 
10: end for
```

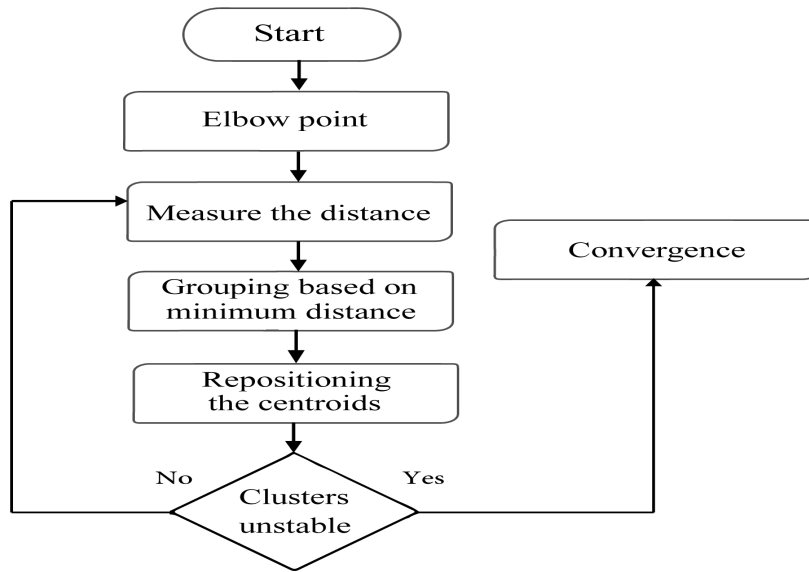


Figure 3.4: The K-mean algorithm flowchart.

The functions in Algorithms 3.2 and 3.3 are internal and can only be called in the main function, which is made public. In Algorithm 3.2, peaks and off-peak clusters are set for each hour of the day, according to the distance of the points from the centroids, passed as parameters. The function of Algorithm 3.3 calculates the new centroid for the next iteration.

Algorithm 3.2 SetCluster(centroid C1,centroid C2)

```
1: for  $i \leftarrow 0$  to  $size(X)$  do  
2:    $d1 \leftarrow \sqrt{X[i] - C1.x^2 + Y[i] - C1.y^2}$   
3:    $d2 \leftarrow \sqrt{X[i] - C2.x^2 + Y[i] - C2.y^2}$   
4:   if  $d1 \leq d2$  then  
5:     clusters.push(1)  
6:   else  
7:     clusters.push(2)  
8:   end if  
9: end for  
10: return clusters
```

To compute the clusters, X and Y arrays are defined in the smart contract as well as the two initial centroids. The initial off-peak cluster centroid is set to be the point of the set with the lowest consumption, whereas the initial peak cluster centroid is set to be the point in the dataset with the highest electricity consumption. The main function defined in Algorithm 3.4 invokes the functions of Algorithms 3.2 and 3.3 through a loop until convergence according to the flowchart in Fig 3.4.

Algorithm 3.3 SetNewCentroids

```
1:  $sumX1 \leftarrow 0$ 
2:  $sumX2 \leftarrow 0$ 
3:  $sumY1 \leftarrow 0$ 
4:  $sumY2 \leftarrow 0$ 
5:  $TempArray \leftarrow 0$ 
6:  $Cluster1Size \leftarrow 0$ 
7:  $Cluster2Size \leftarrow 0$ 
8: Centroid NewC2
9: for  $i \leftarrow 0$  to 24 do
10:   if  $clusters[i] = 1$  then
11:      $sumX1 \leftarrow X[i]$ 
12:      $sumY1 \leftarrow Y[i]$ 
13:      $Cluster1Size \leftarrow Cluster1Size + 1$ 
14:   else if  $clusters[i] = 2$  then
15:      $sumX2 \leftarrow X[i]$ 
16:      $sumY2 \leftarrow Y[i]$ 
17:      $Cluster2Size \leftarrow Cluster2Size + 1$ 
18:   end if
19: end for
20:  $NewC1 \leftarrow \left\{ \frac{sumX1}{Cluster1Size}, \frac{sumY1}{Cluster1Size} \right\}$ 
21:  $NewC2 \leftarrow \left\{ \frac{sumX2}{Cluster2Size}, \frac{sumY2}{Cluster2Size} \right\}$ 
22:  $tempArray.push(NewC1)$ 
23:  $tempArray.push(NewC2)$ 
24: return  $TempArray$ 
```

Algorithm 3.4 Main

```
1: TempArray1 ← SetCluster(InitC1,InitC2)
2: TempArray2 ← []
3: NewC1 ← GetNewCentroids()[0]
4: NewC2 ← GetNewCentroids()[1]
5: Exit ← False
6: while !Exit do
7:   TempArray2 ← SetCluster(InitC1,InitC2)
8:   if TempArray2 ≠ TempArray1 then
9:     Exit ← true
10:  else
11:    TempArray1 ← TempArray2
12:    NewC1 ← GetNewCentroids()[0]
13:    NewC2 ← GetNewCentroids()[1]
14:    TempArray2 ← SetCluster(NewC1,NewC2)
15:  end if
16: end while
```

3.2.1.1.2 Trading Platform Interacting Entities

The Provider: which represents the set of prosumers cooperating in generating energy that is injected into the power grid owned by the DSO, and transferred to the requesting consumers, on demand.

The Consumer: being the entity that purchases energy by subscribing to the provider through a smart contract for a certain duration of time called a trading round, prepays for his consumption and is supplied until exhaustion of the prepaid amount, or billed according to his energy consumption in case of unsubscribing prematurely.

The DSO: also referred to as a utility owner company. The abstraction of the interacting entities is illustrated using the Entity Relationship Diagram ERD in Fig 3.5. However, these entities are not stored in a traditional database, but rather, they are rep-

resented by the state of the smart contract in which they are defined. The state of the smart contract is modified upon a successful call of its functions and is stored in the blockchain shared ledger. Each entity is assigned to its corresponding field through an appropriate mapping.

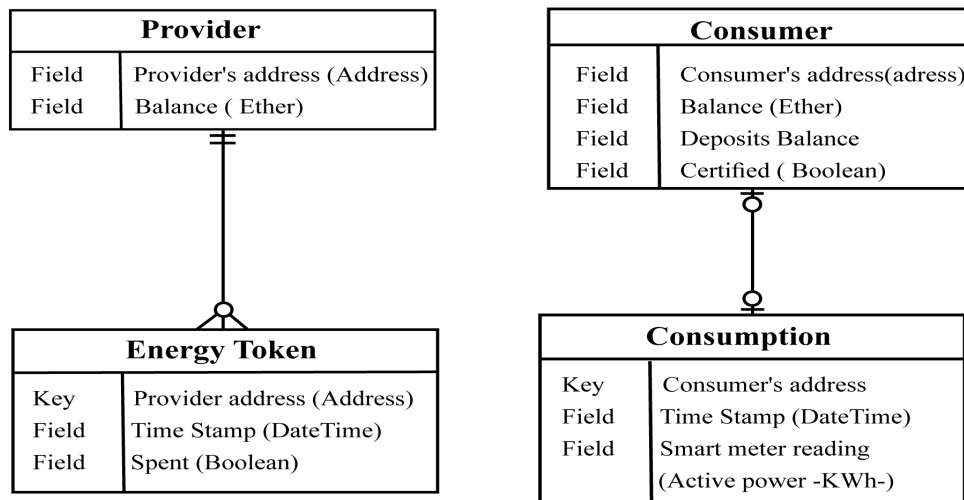


Figure 3.5: ERD diagram of the proposed platform.

Smart contract 1 (DSO): As its name indicates, DSO smart contract is deployed and owned by the DSO. It stores the list of certified consumer nodes as well as the energy balance of the providers. Only the DSO has the credentials to add, remove or update the certified nodes list. Providers' balance is updated by calling the appropriate smart contract function upon a new aggregation of a defined amount of energy or its consumption. The provider's energy balance is abstracted using a simple structure representing a defined amount of energy called an energy token. New instances of this data structure are mapped to prosumers accordingly upon the appropriate aggregation

of energy. Consumed tokens are set to spent state. The global variables for the DSO smart contract are defined in Algorithm 3.5.

Algorithm 3.5 DSO Smart Contract: Global Variables

```

1: ProviderSmartContractAddresses : address[]    ▷ /* Addresses of all providers'
   deployed smart contracts */
2: DsoAddress : address                          ▷ /*DSO account's address*/
3: Certified : Mapping (address: bool)          ▷ /* Check if an account is certified or
   not. By default, accounts are not certified. Certifying an account is restricted to the
   DSO */
4: Structure
5: TimeStamp : uint256
6: Spent : Bool
7: EndStructure
8: ProviderEnergyTokens : Mapping (address: EnergyToken[])    ▷ Keep track of
   EnergyToken instances owned by each Provider account

```

The energy balance of the prosumer is defined by the number of unspent token instances and is publicly accessible by the function defined in Algorithm 3.6.

Algorithm 3.6 DSO Smart Contract: ProviderEnergyBalance

```

1:    ▷ /* This function returns the energy balance of a given address' account. */
2: Input: addr : address
3: Output: uint256
4: CountOfUnspentEnergyToken : uint256,
5: CountOfUnspentEnergyToken ←
6: for i ← 0 to ProviderEnergyTokens[addr].Length do
7:   if !ProviderEnergyTokens[addr][i].spent then
8:     CountOfUnspentEnergyToken ← CountOfUnspentEnergyToken + 1
9:   end if
10: end for
11: Return: CountOfUnspentEnergyToken

```

Updating the provider's balance per consumed or redeemed token is done exclusively through the provider's smart contract, by calling the DSO smart contract functions

defined in Algorithms 3.7 and 3.8, respectively.

Algorithm 3.7 DSO Smart Contract: UpdateProviderBalance1

```
1: Comment/* This function is called to consume N tokens, it takes two inputs, an integer N and an address addr. It sets N unspent tokens from the tokens owned by addr to spent. It is a restricted function which can only be called by the providers' deployed smart contracts */
2: Input: addr : address, N : uint256
3: i ← 0
4: k ← 0
5: while i ≤ N do
6:   if !ProviderEnergyTokens[addr][k].spent then
7:     ProviderEnergyTokens[addr][k].spent ← true
8:     i ← i + 1
9:   end if
10:  k ← k + 1
11: end while
```

Energy tokens have been used to represent the providers' energy balance instead of a simple numerical abstraction, to keep track of the history of the providers' contributions to the utility grid. The relevance of keeping track of prosumers' energy contributions in a P2P energy trading platform can be found in (Amanbek et al., 2018).

Algorithm 3.8 DSO Smart Contract: UpdateProviderBalance2

```
1: ▷ /* This function  
   is called to redeem N tokens, it takes two inputs, an integer N and an address addr.  
   It sets N spent tokens from the tokens owned by addr to unspent. It is a restricted  
   function which can only be called by the providers' deployed smart contracts */  
2: Input: addr : address, N : uint256  
3:  $i \leftarrow 0$   
4:  $k \leftarrow 0$   
5: while  $i \leq N$  do  
6:   if ProviderEnergyTokens[addr][k].spent then  
7:     ProviderEnergyTokens[addr][k].spent  $\leftarrow$  false  
8:      $i \leftarrow i + 1$   
9:   end if  
10:   $k \leftarrow k + 1$   
11: end while
```

Smart contract 2 (Provider): The contract is deployed by the provider ahead of each trading round; it contains the ToU pricing for electricity consumption. Once deployed, consumers have to subscribe to the trading round during a defined period of time, by depositing money corresponding to the amount of electricity they wish to consume. The reason a deadline for subscription is defined, is that receiving a new subscription at any time can prolong the trading round indefinitely, which is not following the proposed scheme since electricity pricing is set according to prediction based on data collected during trading rounds. Each time a new subscription occurs, the smart contract checks the subscribing consumer account by calling Smart Contract 1; in case this is later certified and the provider still has enough energy balance to supply the new coming demand, the provider balance is updated accordingly, otherwise the subscription is denied. The smart contract stops accepting upcoming subscription requests if the

subscription-defined time has elapsed, or if the provider has exhausted all his energy balance. The trading round then starts; it ends when all subscribing consumer deposit balance in the smart contract is zero. This time-dependent smart contract state is illustrated in Fig 3.6. Consumers' electricity consumption data are uploaded to the smart contract. It is used to charge consumers as well as being added to the energy consumption dataset serving to train the prediction model as discussed in Section 3.2.1.1.1 above.

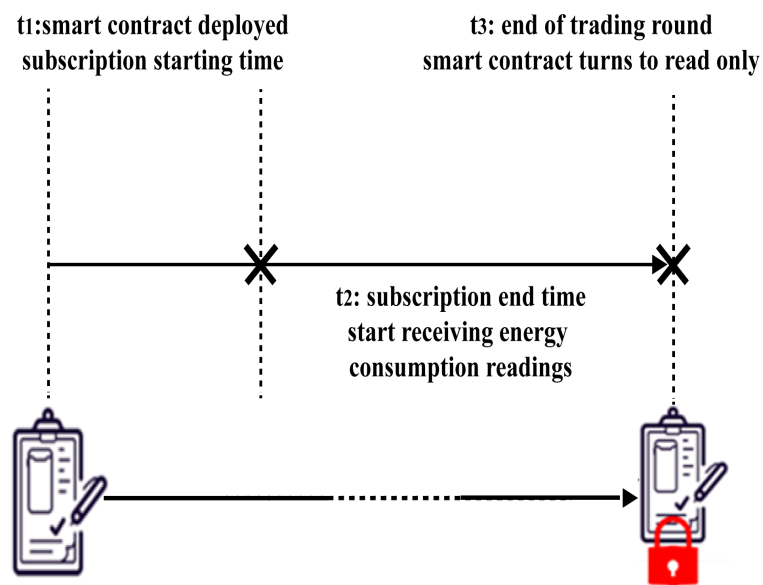


Figure 3.6: Time dependency of the Provider smart contract state.

To test the deployed smart contract, one Ether is set to correspond to 100 kWh. Although this is not realistic, it is simpler for the sake of testing. However, other security tokens, and even stable coins linked to the country's actual currency asset, can serve as a payment method rather than Ether cryptocurrency, which has a very fluctuating value. The global smart contract variables and events are presented below in Algorithms 3.9

and 3.10, respectively.

Algorithm 3.9 Provider Smart Contract: Global Variables

- 1: *Deposits* : *Mapping* (address:uint256) ▷ Keep track of each consumer deposit balance
 - 2: *Consumption* : *Mapping* (address:uint256) ▷ Keep track of each consumer total consumption
 - 3: *EnergyRequested* : *Mapping* (address:uint256) ▷ Access each consumer requested energy
 - 4: *SubscriptionTime* : uint256 ▷ Defined time span to accept new subscriptions after contract deployment
 - 5: *DsoSmartContract* : address ▷ DSO smart contract address
 - 6: *Subscribers* : address[] ▷ Keep track of subscribers' accounts
 - 7: *DeploymentTime*: uint256 ▷ This variable is set to contract deployment time, it is set inside the smart contract constructor called when contract is deployed
 - 8: *EnergyTarif*: uint256
-

Algorithm 3.10 Provider Smart Contract Events

- 1: *event* : *SendEnergy(addressindexed)* ▷ Notify the DSO to supply energy to the address passed through the event
 - 2: *event* : *eventSubscriptionTimeStart()* ▷ Notify consumers that subscription is now open after the smart contracts have been deployed
 - 3: *event* : *StartTradingRound()* ▷ Notify all parties that the trading round has started
 - 4: *event* : *eventBalanceExhausted(address)* ▷ Notify the DSO to stop supplying energy to the address passed through the event
-

Algorithm 3.11 defines the function consumers call to upload their energy consumption data through their certified nodes.

Algorithm 3.12 defines the function called by consumers to subscribe to the trading round.

Algorithm 3.13 defines the function called by consumers to unsubscribe prematurely and exit the trading cycle.

Algorithm 3.11 send_consumption_data

```
1: Input:  $c : uint256$ 
2:  $consumptionBalance : uint256$ 
3:  $consumption[msg.sender] \leftarrow consumption[msg.sender] + c$ 
4:  $consumptionBalance \leftarrow consumption[msg.sender] \times energy\_tarif$ 
5:  $deposits[msg.sender] \leftarrow deposits[msg.sender] - consumptionbalance$ 
6: if  $deposits[msg.sender] \leq 0$  then
7:    $emitBalanceExhausted(msg.sender)$ 
8: end if
```

Algorithm 3.12 Provider's Smart Contract: Subscribe

```
1:  $Time : uint256$ 
2:  $TotalEnergyRequested : uint256$ 
3:  $ProviderEnergyBlance : uint256$ 
4:  $time \leftarrow block.timestamp$ 
5:  $TotalEnergyRequested \leftarrow TotalEnergyRequested + \frac{msg.value}{10^{18}}$ 
6:  $EnergyRequested[msg.sender] \leftarrow msg.value$ 
7:  $ProviderEnergyBlance \leftarrow CallDSOContract1(addressDsoContract, addressProvider)$ 
8:  $\triangleright /*$  Call DSO smart contract ProviderEnergyBalance function, to get Provider's
   energy balance  $*/$ 
9:  $deposits[msg.sender] \leftarrow deposits[msg.sender] - consumptionbalance$ 
10: if  $TotalEnergyRequested > ProviderEnergyBlance \vee TimeSubscriptionTime \leq$ 
     $Time - Deployment \vee !CallDsoContract2(addressDsoContract, msg.sender)$ 
    then
11:    $Revert()$ 
12:   for  $i \leftarrow 0$  to  $Subscribers.Length$  do
13:      $emit SendEnergy(Subscribers[i])$ 
14:   end for
15:    $emit StartTradingRound()$ 
16: else
17:    $Deposits[msg.sender] \leftarrow Deposits[msg.sender] + msg.value$ 
18:    $subscribers.push(msg.sender)$ 
19:    $CallDsoContract3(addressDsoContract, addressProvider, \frac{msg.value}{(10^{18})})$   $\triangleright /*$  Call
    DSO smart contract UpdateProviderBalance1 function to update the provider's en-
    ergy balance according to the subscribers' energy demand that needs to be supplied
    to them  $*/$ 
20: end if
```

Algorithm 3.13 Provider Smart Contract: Unsubscribe

```
1: BillAmount : uint256
2: TotalEnergyRequested : uint256
3: BillAmount  $\leftarrow$  consumption[msg.sender] * EnergyTarif
4: Provider.Transfer(BillAmount)
5: msg.sender.Transfer(deposits[msg.sender] – billamount)
6: Deposits[msg.sender]  $\leftarrow$  0
7: EnergyRequested[msg.sender]  $\leftarrow$  msg.value
8:    $\triangleright$  // Call DSO smart contract UpdateProviderBalance2 function to re-update the
        provider's energy balance accordingly after consumer unsubscribing:
9: CallDSOContract1(addressDsoContract, addressProvider, deposits[msg.sender] –
   billamount / 1018)
10: emit BalanceExhausted(msg.sender)
```

3.2.1.1.3 Agents' Interactions and Workflow in the Proposed Platform

Different agents interact with the pair of smart contracts, according to the defined credentials granted. On the one hand, the provider, consumer, and DSO interact with the smart contract by calling authorized functions. On the other hand, smart contracts also interact with the rest of the platform agents through events, as shown in Tab 3.1. The event trigger is defined within the smart contract. Agents subscribe to a particular event through their back-end application and get systematically notified whenever the event is fired during the smart contract execution, with parameters being passed by the smart contract to the subscribing application. Moreover, an event handler is set within the agents' applications to respond accordingly to the particular event and process the parameters passed along. The pseudocode of the functions in the provider's smart contract is given below. Along with these functions, calls to functions of the DSO smart contract are also defined within the provider's smart contract as procedures; they take

as parameters the DSO smart contract address beside the parameter to pass to the DSO smart contract function to call.

Moreover, the interaction and workflow between the platform agents are highlighted in Fig 3.7 and Fig 3.8. The respective responses are given in chronological order in Fig 3.9. When the trading round starts, all subscribers' energy demand is supplied until prepaid amount exhaustion. In the case of a client's premature un-subscription, he is charged according to his actual consumption and the change from his prepaid deposit is returned to him. The provider's balance is updated as well, by setting back n spent tokens to unspent, where n represents the number of tokens corresponding to the change amount returned to the consumer.

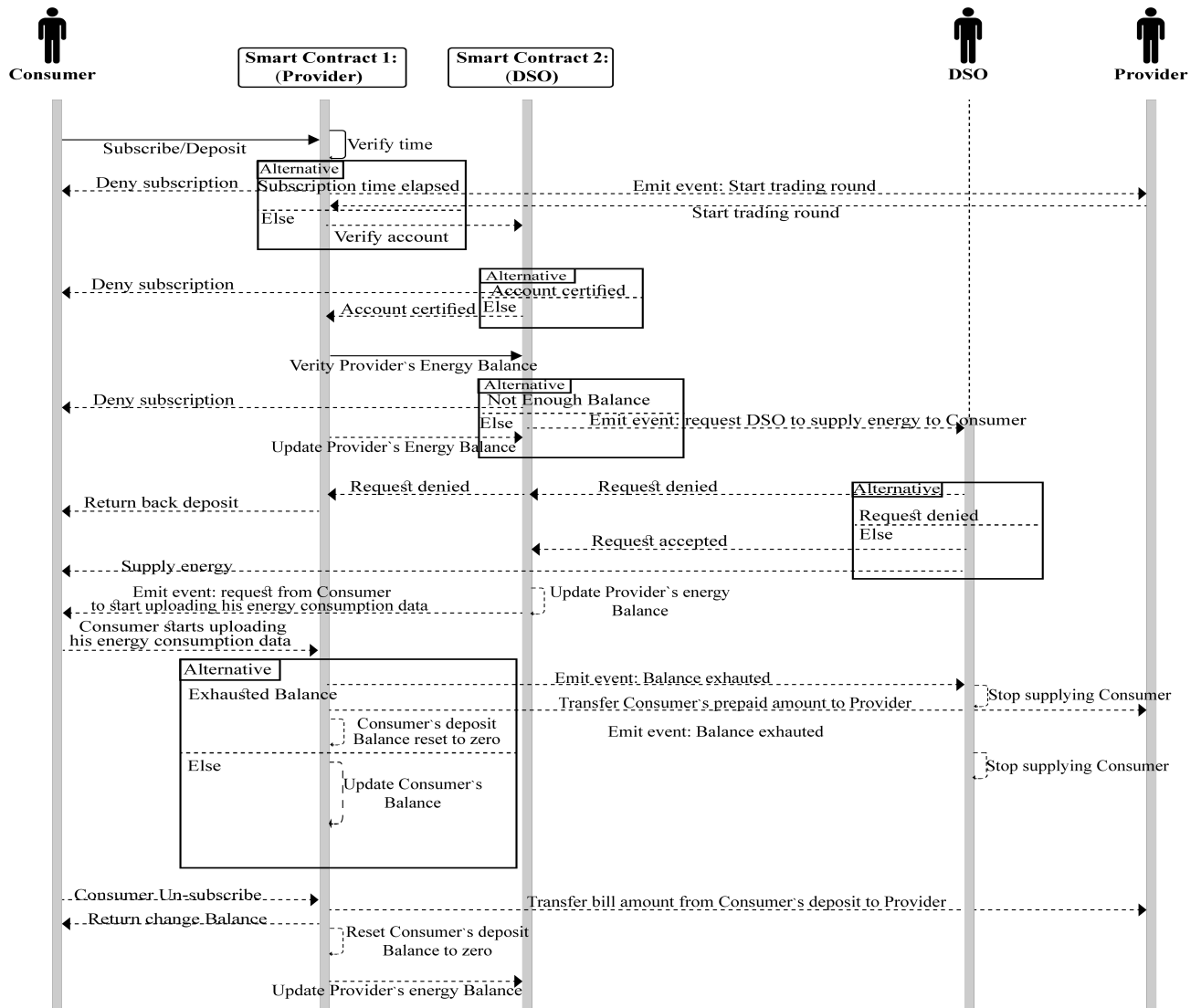


Figure 3.7: The proposed P2P Energy trading sequence diagram.

Table 3.1
Smart contracts' events in the proposed platform.

Event	Source of Emission	Event Trigger	Event Subscribers	Event Handler
Event 1:SubscriptionTimeStarted ()	Smart Contract 1	Smart Contract 1 is deployed	Consumer	Get notified by subscription starting time.
Event 2:Subscription-TimeEnded ()	Smart Contract 1	Subscription time elapses	Consumer	Get notified that subscription time has ended.
Event 3:Transfer-Energy (Address add)	Smart Contract 2	Provider energy balance is verified if enough	DSO	Supply energy to the consumer.
Event 4:Start-TradingRound ()	Smart Contract 1	Smart Contract 2 confirms that the provider has enough energy balance	Consumer	The Consumer starts uploading his energy consumption readings to Smart Contract 1.
Event 5:Balance-Exhausted	Smart Contract	Consumer unsubscribes, or his balance is exhausted	DSO	Stop supplying the consumer.

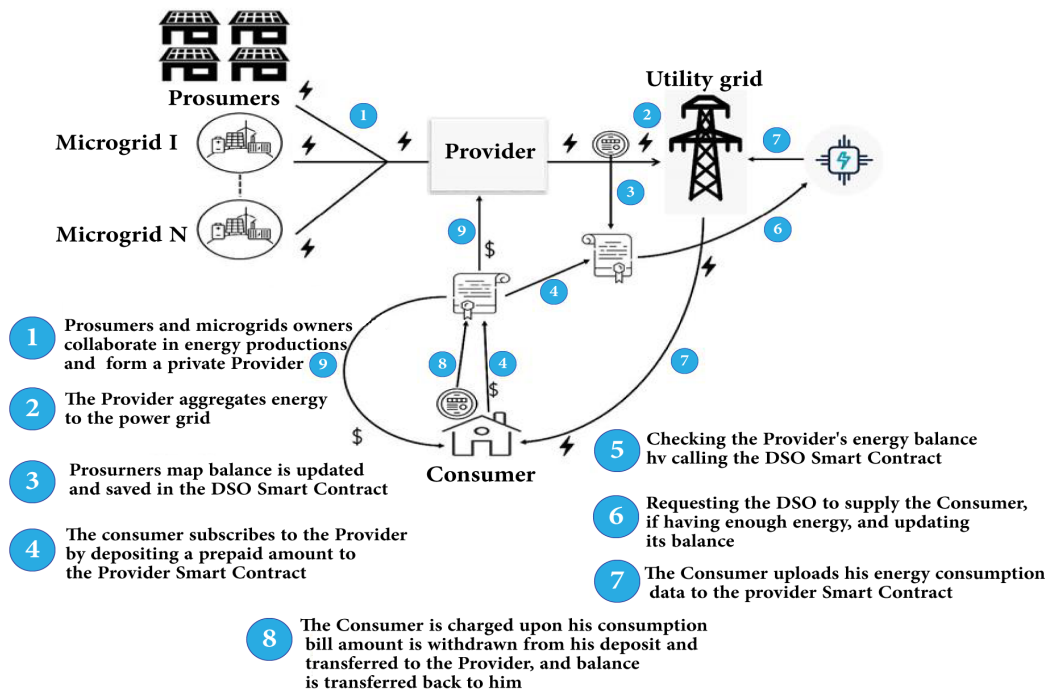


Figure 3.8: Interactions between the entities of the proposed P2P trading Platform.

3.2.1.1.4 The State Diagram Representation

Each time a smart contract is called, it is executed on the Ethereum virtual machine and its new state is stored on the blockchain. Thus, is suitable to represent the smart contract as a finite state machine as depicted in the diagram of Fig 3.9. Inputs represent the different function calls that modify the state of the smart contract.

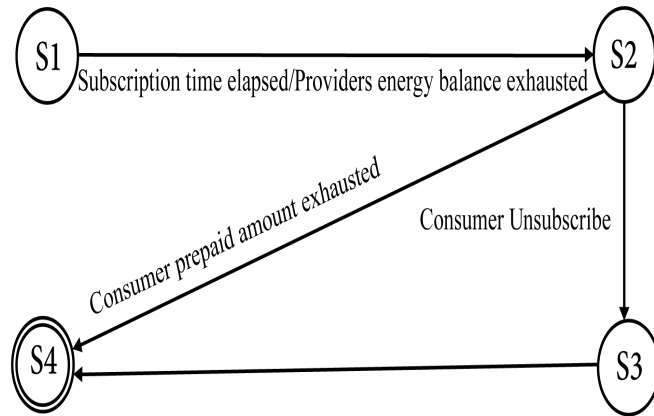


Figure 3.9: State diagram.

3.2.1.1.5 Modified PoW consensus adopted to the platform

Blockchain is principally a distributed architecture with no central authority, the validity of the shared ledger is agreed upon consensus held between nodes in the network. Thus, consensus protocol stands as a key feature in such an architecture-based application. It ensures the integrity of the data, which is uploaded to the blockchain without a central trusted authority. Ethereum private blockchain proposes two options for consensus protocols. Ethash, which is a Proof of Work-based protocol and Clique Proof-of-Authority consensus protocol. Moreover, in this work, we preferred to use Ethash to keep the platform decentralized, since PoA tends toward centralization by having a set of predefined validators. What is suggested is to use Ethash with a fixed difficulty and no mining rewards. All consumer nodes in the network are potential miners. All miner nodes would be having the same hashing power, thus, if a node attempts to tamper with the ledger, it would have to find new valid hashes for all the blocks preceding the blocks tampered

with, and will be certainly leftover in the mining race. The only scenario where a node can successfully be fast enough to tamper with the already committed ledger is to use hardware having higher hashing power, which is supposedly not allowed, given that the nodes' hardware need to be certified. For that purpose, the proposed consensus algorithm for this architecture is that blocks are not accepted if the time difference between the last block mining time and the time of proposing a new valid block is less than a defined threshold.

To implement the proposed consensus protocol on an Ethereum private blockchain, the go-Ethereum source code needs to be slightly overwritten. This can be done by performing minor additions to the source-code in the file: "<https://github.com/ethereum/go-ethereum/blob/master/consensus/ethash/consensus.go>" accessed on 26 March 2022. The following Algorithm 3.15 is a code snippet with the original line members from go-ethereum source code on Github; the applied additions are highlighted.

Algorithm 3.15 Modifications performed on PoW consensus in Go-Ethereum

// Various error messages to mark blocks invalid. These should be private to prevent engine specific errors from being referenced in the remainder of the codebase, inherently breaking if the engine is swapped out. Please put common error types into the consensus package.

```
var (  
    errOlderBlockTime = errors.New("timestamp older than parent")  
    errTooManyUncles = errors.New("too many uncles")  
    errDuplicateUncle = errors.New("duplicate uncle")  
    errUncleIsAncestor = errors.New("uncle is ancestor")  
    errDanglingUncle = errors.New("uncle's parent is not ancestor")  
    errInvalidDifficulty = errors.New("non-positive difficulty")  
    errInvalidMixDigest = errors.New("invalid mix digest")  
    errInvalidPoW = errors.New("invalid proof-of-work")  
    errInvalidMininegTime = errors.New("invalid mining time")  
  
    func (ethash *Ethash) verifyHeader(chain consensus.ChainHeaderReader, header,  
    parent *types.Header, uncle bool, seal bool, unixNow int64) error {  
        // Ensure that the header's extra-data section is of a reasonable size  
        if uint64(len(header.Extra)) > params.MaximumExtraDataSize {  
            return fmt.Errorf("extra-data too long: %d > %d", len(header.Extra),  
            params.MaximumExtraDataSize)  
        }  
        // Verify the header's timestamp  
        if header.Time > uint64(unixNow+allowedFutureBlockTimeSeconds) {  
            return consensus.ErrFutureBlock  
        }  
    }  
    if header.Time <= parent.Time {  
        return errOlderBlockTime  
    }  
    if header.Time - parent.Time < Tthreshold {  
        return errInvalidMininegTime  
    }  
}
```

3.2.2 Proposed PoL to secure blockchain energy trading transactions

When the blockchain first appeared, it served as the foundation for cryptocurrency applications. Financial transactions were recorded in the public ledger. The validity of data uploaded to the blockchain depended on the validity of the financial transaction (no double spending). This was ensured by models such as the UTXO (Unspent Transaction Output) model in Bitcoin, and the balance-based account model in Ethereum, which work by making each account's transaction history accessible and thus the validity of a new transaction can be verified.

Key pair cryptography and digital signatures are used to verify the identity of the issuer of the transaction. This worked well when blockchain was only meant for financial cryptocurrency transactions, and transactions were issued by mindful human beings who personally cared about maintaining the confidentiality of their secret key. However, with the advent of smart contracts, blockchain has evolved beyond its original application in finance and now enables decentralized applications in a variety of areas, particularly the Internet of Things (IoT), where systems are automated and enabled through the decentralized execution of smart contracts without the need for a server or centralized cloud entity.

In such scenarios, IoT nodes can invoke smart contract functions which are blockchain transactions, based on defined code implemented on the IoT nodes. Since the identity of the transaction issuer is defined in the software running on the IoT node,

and is thus easily accessible, private key encryption cannot guarantee the identity of the transaction issuer in such cases. Even if it is closed-source code, it is not secret to the developer of the code, who, in turn, can share it with others. Although misuse of an IoT or smart node identity is not always mandatory for some IoT applications, it is so for others. A typical example is the Peer to Peer (P2P) energy trading system, where prosumers are rewarded with tokens for the energy they generate and feed into the grid. An assigned smart meter measures and monitors the energy fed into the grid and is responsible for claiming a token on behalf of the prosumer. Such architectures have already been proposed in (Todorean et al., 2021; Munoz et al., 2022; Buccafurri et al., 2021). In such a situation, it is tempting for prosumers to abuse the identity of their assigned smart meter to claim an undeserved energy token. Therefore, it is critical to include a verifiable identity attribute that is difficult for others to guess. Since smart meter nodes are permanently deployed, a location trace may be sufficient to identify the origin of the transaction. In this section, we describe a decentralized PoL, suitable for blockchain-based energy trading.

3.2.2.1 Traditional centralized PoL

Consumer node deployment and setting need to be supervised and certified by the DSO so that the nodes are allowed to join the trading network. Consumer node transactions are identified by their digital signature. However, since consumers are billed according to the consumption data uploaded by their respective assigned nodes, which are set

to upload the consumer's consumption reading from the respective smart meter to the smart contract, it is crucial to ensure that the data uploaded is indeed coming from the certified nodes and not from any malicious clone or identity thief who could gain access to the consumer node private key. What is proposed is to add a proof of location (PoL) that must be included in every transaction issued by the consumer node to prove the transaction is coming from the known location of the certified node. Outdoor localization mainly uses Global Positioning System (GPS) as the main localization technique. However, GPS is not suitable for trustless decentralized scenarios. What is suggested is to use trilateration to localize the consumer nodes. Trilateration is a technique for node localization that determines object position, knowing its distance from at least 3 reference points.

Considering 2-dimensional trilateration, a 2-dimensional frame needs to be set with an origin $O(0,0)$ and three reference points, R_{p1} , R_{p2} and R_{p3} with known coordinates (x_1, y_1) , (x_2, y_2) and (x_3, y_3) , knowing the distances d_1, d_2 and d_3 between a target point $T_p(x_T, y_T)$ and R_{p1} , R_{p2} and R_{p3} , respectively, the couple (x_T, y_T) can be deduced by solving the set of three expressions of Eq. 3.19, as illustrated in Fig 3.10.

$$\begin{aligned}
 d_1^2 &= (x_t - x_1)^2 + (y_t - y_1)^2 \\
 d_2^2 &= (x_t - x_2)^2 + (y_t - y_2)^2 \\
 d_3^2 &= (x_t - x_3)^2 + (y_t - y_3)^2
 \end{aligned} \tag{3.19}$$

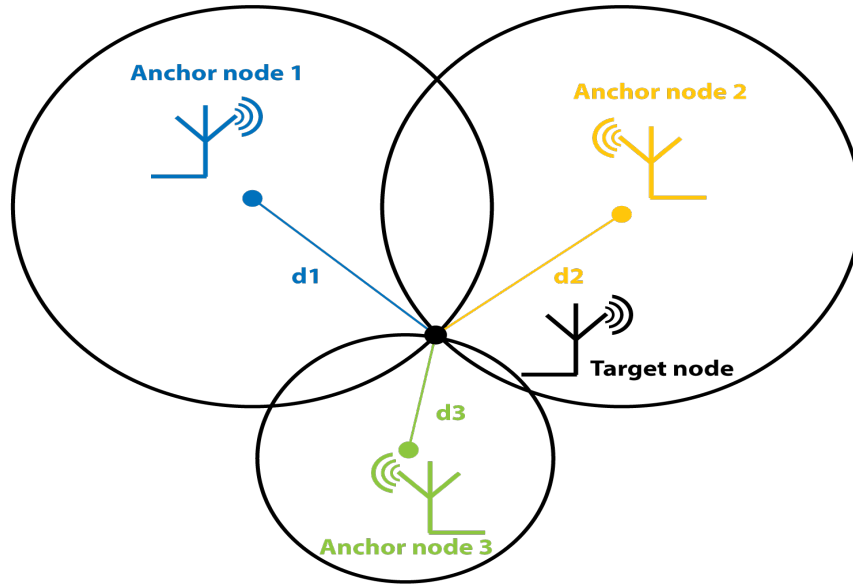


Figure 3.10: Trilateration technique.

In the proposed PoL model, three anchor nodes are deployed to cover a defined area with a defined number of consumer nodes. Consumer nodes send a PoL request to the 3 anchor nodes; each anchor node computes the distance separating him from the consumer node, then sends back the measured distance signed using his private key K_j^{Pr} . Consumer nodes need to include the three signed locations they received from the 3 anchors as a PoL. Given a defined geographical location with a set of n Consumer nodes $CN_i = CN_0, CN_1, \dots, CN_n$ and a set of 3 anchor nodes $An_j = An_0, An_1, An_3$ each consumer node i should send a PoL request $Pol_Req_{i \rightarrow j}$ to the anchor j and receive Pol

response $Pol_Res_{j \rightarrow i}$, accordingly, as in Eq. 3.20.

$$Pol_{Res_{j \rightarrow i}} : \left\{ \begin{array}{c} \text{Distance between } An_j \text{ and } CN_i \\ \text{Timestamp} \end{array} \right\} K_j^{Pr} \quad (3.20)$$

There are two main techniques to measure the distance between a reference anchor and a target node, which use either the Time of Arrival (ToA) or the Received Signal Strength Indicator (RSSI). Although assessing the most suitable distance measurement technique is not within the scope of our study, we found some reference papers proposing RSSI techniques for outdoor localization as in (Anagnostopoulos & Kalousis, 2019; Ingabire et al., 2021; Goldoni et al., 2019; Barai et al., 2020) and we aimed to present a blueprint for Wi-Fi RSSI-based node localization, that potentially can be used for PoL in blockchain-based decentralized P2P energy trading. Given the RSSI of a received signal, the distance between the transmitter and the receiver nodes can be deduced according to the expression in Eq. 3.21.

$$d = 10^{-(RSS - P_t + Pl_0 - \delta)/10\gamma} \quad (3.21)$$

where Pl_0 is the path loss at a reference distance of (1 m) measured using the expression in Eq. 3.22, λ is the wave length of the emitted signal and γ is the path loss exponent

and depends on the environment.

$$Pl_0 = 20 \log_{10} \left(\frac{4\pi}{\lambda} \right) \quad (3.22)$$

δ is the standard deviation and depends on the hardware and antenna used as well as on the noise. We tried to define the location of 2 smartphones through Wi-Fi RSSI. RSSI values of a hotel Wi-Fi access point were measured in an urban area of Kuala Lumpur using two different smartphones, a SAMSUNG GALAXY S30 and an OPPO F11. RSSI readings were recorded each 2-meter interval walking from the hotel main entrance following the itinerary illustrated in Fig 3.11 for 50 times, to deduce the standard deviation model:



Figure 3.11: The itinerary followed.

Since there is no standard industrial norm for computing RSSI, its measurement

for the same distance is different in the two phone devices. In such an architecture with no unified protocol in RSSI measurement, the anchor node needs to have a reference database with a $d(\text{RSSI})$ model for each device. We kept the same distance function as in Equation (1) with personalized λ and δ and for each device as shown in Fig 3.12.

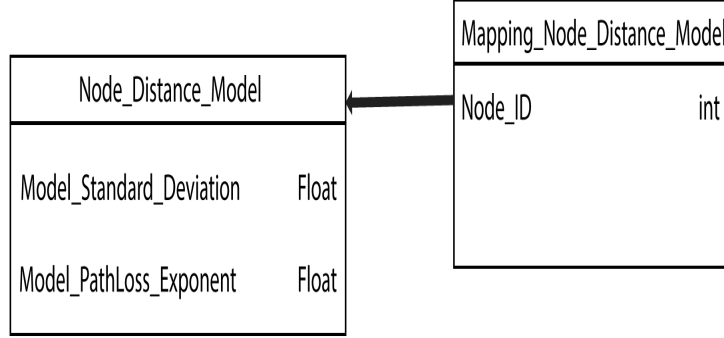


Figure 3.12: Mapping diagram illustrating how each consumer node is mapped to its own distance computation model in the Anchor server node.

The Wi-Fi access point is of 5 GHz with a wavelength of 6 cm, the PoL is deduced according to Equation (3), the access point Wi-Fi signal power transition is $P_T = 25$ dBm. The measured RSSI corresponding to each distance in the entire 50 testing sets was used to compute the distance standard deviation for each device. In our case, the RSSI in a given position varies randomly over time from one test set to another, the values that are measured for each distance in each training set are provided in Equation (7). The corresponding results are reported in Fig 3.13 and Fig 3.14.

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^{N=50} \text{RSSI}_i - u} \quad (3.23)$$

where, $u = \frac{1}{N} \sum_{i=1}^{N=50} RSSI_i$

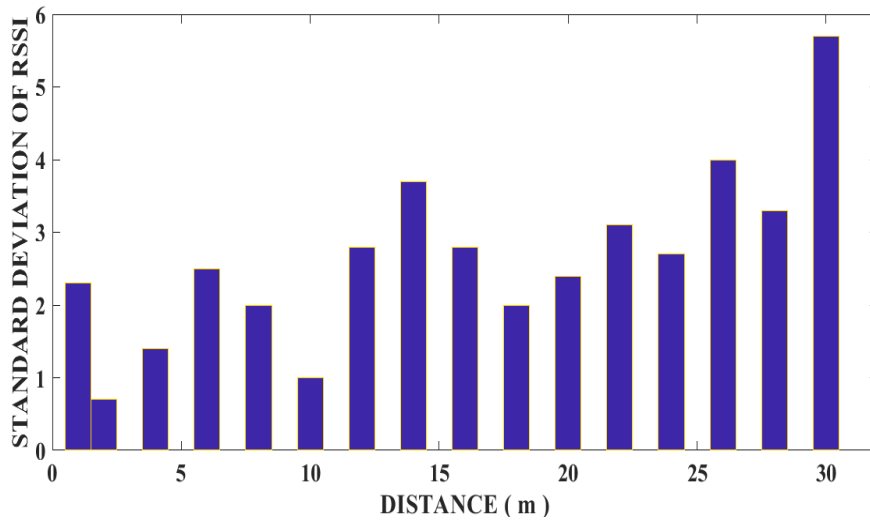


Figure 3.13: Standard deviation for device 1.

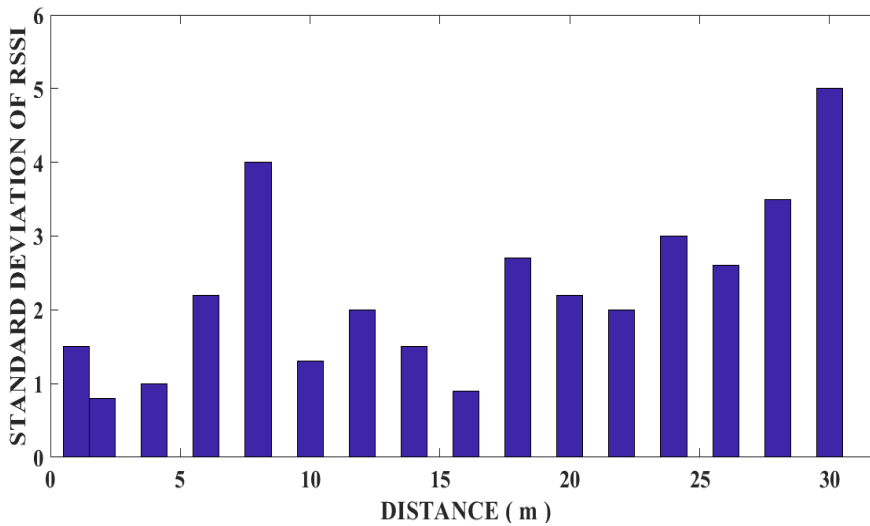


Figure 3.14: Standard deviation for device 2.

The mean standard deviation for device 1 is 2.2625, whereas it is 2.6500 for device 2. By fixing δ for each device, the next step is to deduce γ for each one of them using best fitting curve method between the set of distances d_1, d_2, \dots, d_{15} and the set of

corresponding RSSI mean averages $RSSI_{av0}, RSSI_{av1}, \dots, RSSI_{av15}$, using Equation (5).

This is depicted in Figures 3.15 and Fig 3.16.

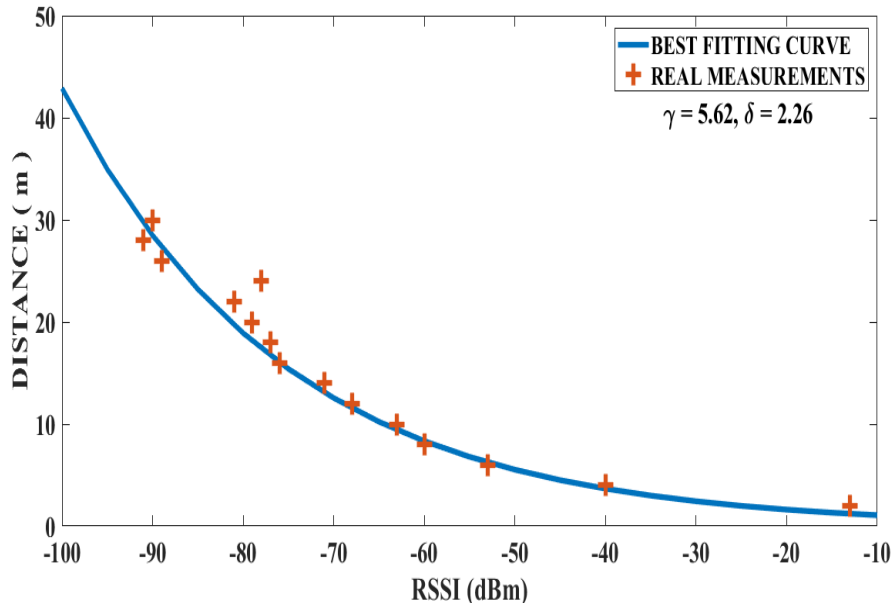


Figure 3.15: Pathloss exponent for device 1 deduced using best fitting curve.

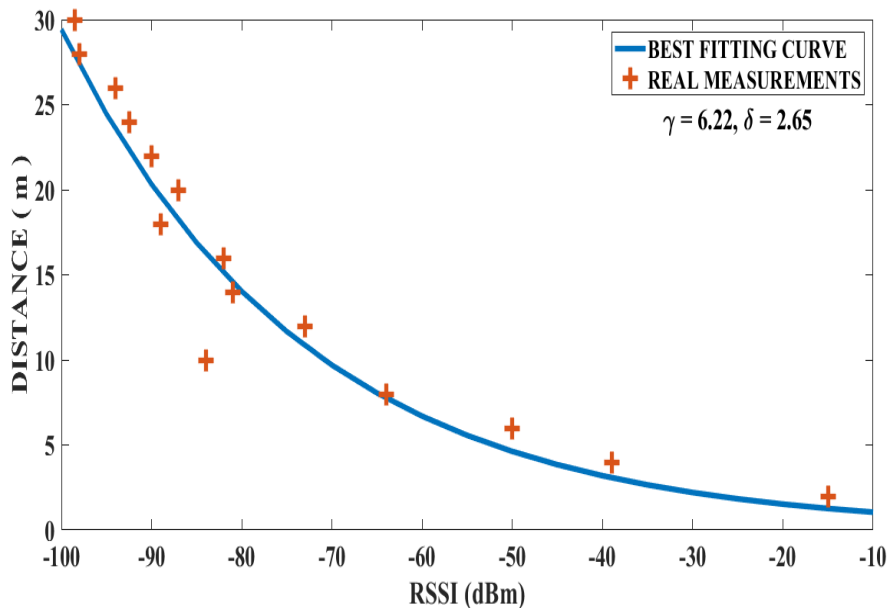


Figure 3.16: Pathloss exponent for device 2 deduced using best fitting curve.

3.2.2.2 PoL decentralized version

PoL is significant in all applications when services can be claimed according to the origin of the request location. This is the case in blockchain-based energy applications, as tokens are delivered only if requested by the corresponding certified smart meter deployed at a specific fixed and known location. The PoL scheme is detailed in full in this subsection. First, all entities involved in the system architecture are introduced. Next, the model architecture and the PoL setting procedure are explained. Finally, we describe how all the security vulnerabilities in setting a reliable PoL are closed by the threat and trust model of the proposed design.

Entities

In the proposed system architecture, there are four distinct entities:

- The target nodes: These are the nodes that need their locations confirmed before their transactions may be recorded in the blockchain's public ledger. In the context of the proposed P2P energy trading system, the target nodes are prosumer nodes, which are set and certified to request tokens upon the appropriate energy aggregation to the grid.
- Location prover nodes: These are end nodes that compete for a financial reward by calculating the location of a target node. To be a potential candidate for the location proving process, prover nodes must stake a certain amount of tokens in

a specified smart contract. If a prover's claimed location turns out to be false following verification, the prover forfeits his or her wager. If, following verification, the claim is found to be correct, the prover is rewarded with tokens proportionate to the amount of money wagered. This turns the target location claim into a Token Curated Registry in which curators (prover nodes) bet financially on the validity of their claim (target node's location).

- The verifier: This is the entity in charge of verifying the target node's location, which is provided by the assigned group of provers. The verification process is performed by an intelligent independent entity, i.e., a smart contract, which is tasked with rewarding trustworthy LPs and penalizing fraudulent ones in the proposed system. A smart contract's verification ensures that it is decentralized and free of bias, collusion, or corruption. The prover nodes in the proposed scheme use the Difference Time of Arrival (TDoA) algorithm to calculate the distance between them and the target node, rather than the specific location of the target node. Trilateration, which is implemented in the location verifier smart device, is used to determine the exact location.
- Sink node: It collects and transmits to the location verifier smart contract the distances estimated by the selected set of prover nodes that separate them from the target node in question. The sink node is used to transfer locations calculated by different provers in a single transaction, rather than many transactions.

PoL Primary Model Architecture

a. Definition of Geographic Ranges for Prover and Target Nodes

As the location of the target node is determined by trilateration based on the time difference of the arrival of a transmitted RF signal, both the target and its assigned prover nodes must be in close proximity to ensure the accuracy of the calculated position. For this purpose, appropriate geographic square zones are defined that cover the entire area of interest and have no overlap. The geographic areas, as well as the coordinates where the target nodes should be located, are defined in the location verifier smart contract. Each target node i , denoted TN_i , is an object consisting of: a location Li represented by two coordinates (x_i, y_i) , and a particular geographic area Gi , where: $\forall x, y \in [a, b], G(x, y) = x \times y$, where a and b are the area boundaries, and a is a set of potential prover nodes and selected prover nodes. Each node can apply to be an anchor in a permission-free manner by submitting its exact location coordinates to the location verifier smart contract and is accordingly connected to the corresponding target nodes via a mapping. Even if the location of a particular target node in the verifier smart contract is calculated relative to the provers' claimed position, prover nodes in the proposed system cannot afford to lie about their position and would have to constantly update it in the location verifier smart contract. Any misstatement of position would adversely affect to the prover in question. This will become clearer as the paper progresses.

b. Location Verification Process

The target nodes of the proposed P2P energy trading system are smart meter nodes configured to monitor prosumers' energy contributions to the power grid, requesting tokens when the associated prosumer delivers the corresponding energy. Because the private key needed to sign transactions is contained in the source code running on these nodes, anyone who gains access to the node's source code can access the private key. This makes the system vulnerable to identity theft, especially by the consumer to whom the smart meter is assigned. Consequently, transactions generated by the aforementioned target nodes must include a PoL to interact with the smart contract responsible for granting tokens to prosumers in accordance to their energy aggregation. Although location is not an absolute proof of identity, a PoL provides more security alternatives to the system.

Once a suspected fraudster linked with a particular target node is identified, proof of its dishonest act can be found in the public ledger, which contains transactions associated with the transmission of data to the blockchain that is incompatible with the smart meter readings in question. A target node sends a request to the location verifier smart contract before sending the aforementioned transactions to the corresponding smart contract. The location verifier smart contract then sends an event to notify the potential registered prover nodes in the same area as the target node in question. To be considered for participation in the target node's location verification process and potentially obtain a financial reward, interested prover nodes must make a deposit of at least

a specific threshold. It is proposed to set a time limit for the submission of an application for a prover node and to stop accepting applications after the time limit expires. In such an application, a timer function with a specified time limit and a callback function are usually used. The callback function is automatically called when the selected time period ends, and the timer is triggered each time it is called. However, in most cases, implementing timers in a smart contract requires the use of an oracle, which means that the timers are implemented off-chain. The function to trigger the smart contract's timer is implemented on-chain. When the set time expires, an event is triggered to notify the associated off-chain timer application, which in turn invokes the callback function in the smart contract.

However, as there is no callback function in our context, the desired on-chain timer application is applicable. When a target node makes a new request for a location trace, the timer starts counting. This also triggers an event that notifies registered provers in the same geographic area that they can apply to be PoL verifier nodes for the target node's specific request. As the smart contract may receive overlapping or concurrent PoL requests from different target nodes, each target node has its own assigned timer. New prover applications are not accepted after the timer expires. The pseudocodes of Algorithm 3.16 and the flowchart in Figure 3.17 show how this is implemented in the location verifier smart contract.

Algorithm 3.16 Location Verifier Smart Contract: RequestPoL, Apply/Deposit

```
1: Address: TargetNode
2: Uint: DepositThreshold
3: Uint: TimerDeadLine
4: Mapping: Address  $\rightarrow$  Address[: TargetNodePotentialProvers
5: Mapping: Address  $\rightarrow$  Uint: PoLRequestId
6: Mapping: Address  $\rightarrow$  Address[: TargetNodeCandidateProvers
7: Mapping: Address  $\rightarrow$  Uint: ProverDepositBalance
8: Mapping: Address  $\rightarrow$  Uint: Timer
9: Procedure RequestPoL
10: Timer(msg.sender)  $\leftarrow$  Time.Now()
11: PoL(msg.sender)  $\leftarrow$  Time.Now()
12: emit event: ProversApplicationOpen
13: End Procedure
14: Procedure Payable Apply/Deposit(TargetNode)
15: Timer  $\leftarrow$  Timer(TargetNode)
16: If msg.value  $\geq$  DepositThreshold Then
17:   Revert()
18: End If
19: If Time.now() - Timer  $\leq$  TimerDeadLine Then
20:   TargetNodeCandidateProvers(TargetNode).push(msg.sender)
21:   ProverDepositBalance(msg.sender)  $\leftarrow$  msg.value
22: Else
23:   Emit Event: NotifyTargetNode(TargetNodeCandidateProvers)
24: End If
25: End Procedure
```

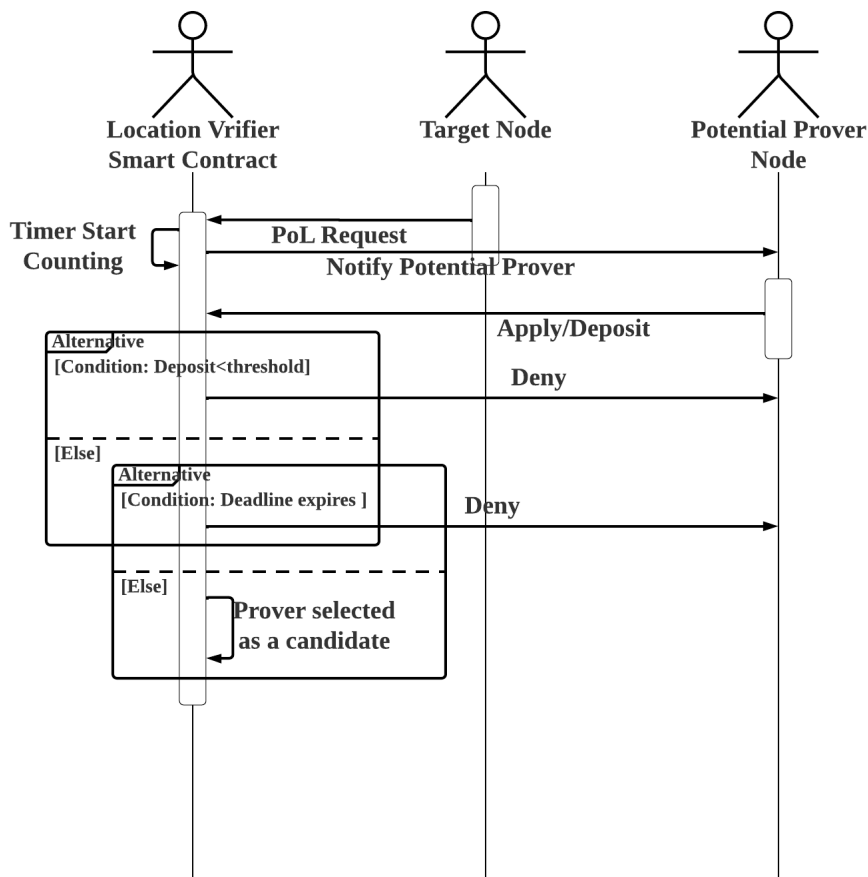


Figure 3.17: Flowchart illustrating the process by which potential prover nodes apply for location verification for a given PoL request from the target node.

The random selection of provers among the candidates begins whenever a new prover application is denied.

c. Provers Random Selection

The random selection of the provers reduces the likelihood of collusion by reducing the probability that verifiers know each other. However, it must be based on an unpredictable random number generated by the location verifier smart contract. The generation of a random number by a smart contract, in contrast, is less obvious. There

are only three approaches for generating random numbers with a smart contract. The first approach is to use the current block hash, block hash size, or current difficulty, which is determined by the block mining throughput. However, miners have a high degree of control over all of these parameters. In fact, it is the miners who calculate the block hash, and they might collude to mine blocks with a desired throughput, making this approach risky and the generated random number predictable. The second option is to have the random number generated outside of the smart contract by an oracle. This approach, however, is purely centralized and goes against the spirit of decentralization of the blockchain. It keeps a loophole open for bribed, corrupt central entities, especially if the generation of the random number has a financial incentive, as in our case. The last and most popular option is to use oracle-based verifiable random functions (VRF). The argument of the VRF oracle is that it cannot predict when the smart contract in question will request a random number. Therefore, it uses the timestamp of the request to generate a random number using an unpredictable but verifiable random function, which it then sends to the smart contract along with the proof of random generation. The VRF is made up of three different functions. They are, respectively: $Generate(x) = (P^{Key}, S^{Key}); F(x, S^{Key}) = y; PROOF(y, P^{Key})$. The VRF must satisfy three conditions, namely:

- Uniquenes: $\exists (x_1, y_1), (x_2, y_2) : PROOF(y_1, P^{Key1}) = PROOF(y_2, P^{Key2})$.
- Pseudo-randomness: Given a set of n inputs $S_{input} = x_1, \dots, x_n$ and their respective

set of output $S_{output} = y_1, \dots, y_n$, there is no pattern linking outputs together.

- Provability:

The first function generates a Public/Secret key P^{Key} and S^{Key} based on the seed input x . The second function generates a random output from the seed input and S^{Key} ; the third function is a proof of the correctness of $F(x, S^{Key})$, which can be verified using the output y and P^{Key} . Although VRF is a proven random number generation method in today's smart contracts (Shu & Lei, 2021; Emmanuel & Nimmy Chacko, 2020), it is not appropriate for our application because it also relies on a smart contract-generated seed that is not random and can be manipulated. Although the three basic random number generation methods did not seem suitable for our proposed scheme, a thorough review of the literature led us to a model that did. The authors in (M. Du et al., 2019) presented a roulette game implemented with a smart contract. Players wager a certain number of tokens on a number and win if their chosen number comes up in the smart contract's lucky draw. A game owner sells the tokens to the players and takes them back if the players lose, similar to casino roulette.

As the game owner and the player are counter-parties who would never collude, the game owner selects a random number, encrypts it, and sends it first to the smart contract in charge of the draw; after both the game owner's and the player's numbers are committed to the blockchain's public ledger, the game owner decrypts the number. Using the owner's public key, the smart contract checks whether the revealed number

matches the encrypted number. If it does, it generates a random number using both the player and owner numbers. The owner starts first to ensure that he cannot manipulate the final result, and neither can the player, as the owner's number is revealed only after the player has committed his number to the blockchain. This scheme ensures fully decentralized random number generation in an application with counter-parties (cannot collude). The DSO is the entity in the proposed architecture for which the exact locations of the target nodes are important.

As the DSO provides the tokens and grants them to the target nodes through energy aggregation, other nodes might be tempted to falsify the computed location to obtain the tokens without satisfying the agreed conditions. Because the tokens are used to pay the DSO for power consumption, the reliability of the system is of paramount importance to the DSO. Thus, the proposed system is similar to a roulette game where the DSO is the owner of the game and the prover nodes are the players. For this reason, we chose to use the same random number generation as implemented in (M. Du et al., 2019). Having its public key stored in the smart contract, the DSO starts the process by sending an encrypted number of its choice. Only after the DSO transaction is committed to the blockchain ledger can the prover node candidates send their own number to the blockchain, and the prover node candidates agree on the sink node that transmits their picked number with their respective signature. However, because identification in a permissionless blockchain environment is pseudo-anonymous, DSO can introduce

covert nodes that use unknown public keys to manipulate the numerical sequence of prover candidates. For this reason, the sink node responsible for transmitting the prover number sequence should be an identified non-DSO node, such as a registered prosumer. The sink node would be the last prover to add a number to the sequence of numbers transmitted to the smart contract. After the numbers are committed to the blockchain and the corresponding signatures are verified, the DSO reveals its number, which is verified by the smart contract. If it matches the encrypted message, both the DSO and prover numbers are used by the smart contract to randomize the prover. See Figure 3.18 for how the encrypted DSO number is revealed and verified.

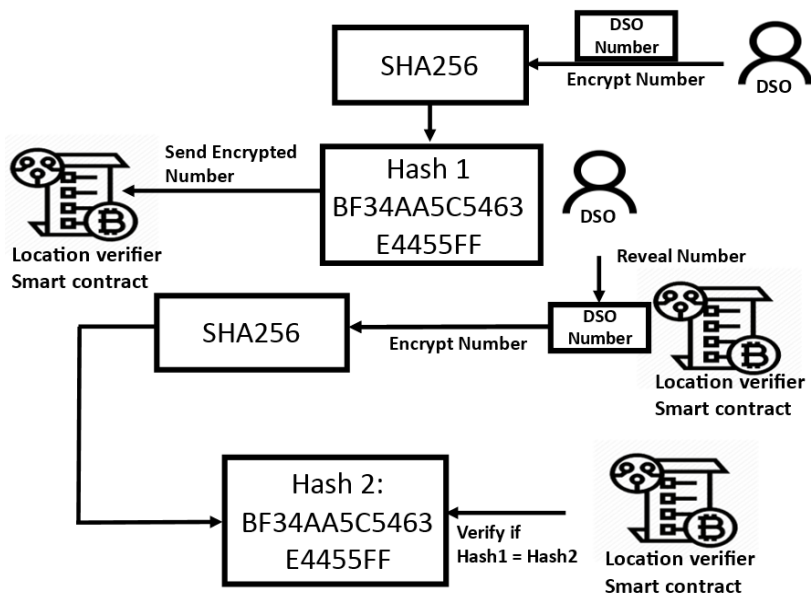


Figure 3.18: Process by which the DSO sends an encrypted number, then reveals it, and how it is verified by the location verifier smart contract.

The functions of the smart contract involved in random number generation are explained using the pseudocode of Algorithm 3.17 and the flowchart in Figure 3.19.

Algorithm 3.17 Location Verifier Smart Contract: SendDsoEncryptedNumber, SendProversNumber, DsoRevealNumber

```
1: Address: DSO
2: Mapping: Address → Uint: DsoSeedNumber
3: Mapping: Address → Uint: NonDsoSeedNumber
4: Mapping: Address → Bytes32: EncryptedDsoNumber
5: Procedure DsoSendEncryptedNumber(Bytes32 EncryptedNumber, Address
   TargetNode)
6: EncryptedDsoNumber(TargetNode)← EncryptedNumber
7: Bool DsoNumberCommitted← True
8: emit event: NotifyProversToSendTheirNumber
9: End Procedure
10: Procedure SendProversNumber(Uint Number, Address TargetNode)
11: If DsoNumberCommitted = True Then
12:   NonDsoSeedNumber(TargetNode)← Number
13:   Bool ProversNumberCommitted← True
14:   Emit Event: NotifyDsoToRevealNumber
15: Else
16:   Revert()
17: End If
18: End Procedure
19: Procedure DsoRevealNumber(Uint Number, Address TargetNode)
20:   If ProversNumberCommitted = True Then
21:     EncryptedDsoNumber ← EncryptedDsoNumber(TargetNode)
22:     c ← keccak256(abi.encode(Number))
23:     If c == EncryptedDsoNumber then
24:       DsoSeedNumber(TargetNode)← Number
25:     Else:
26:       Revert( )
27:     End If
28:   Else
29:     Revert()
30:   End If
31: End Procedure
```

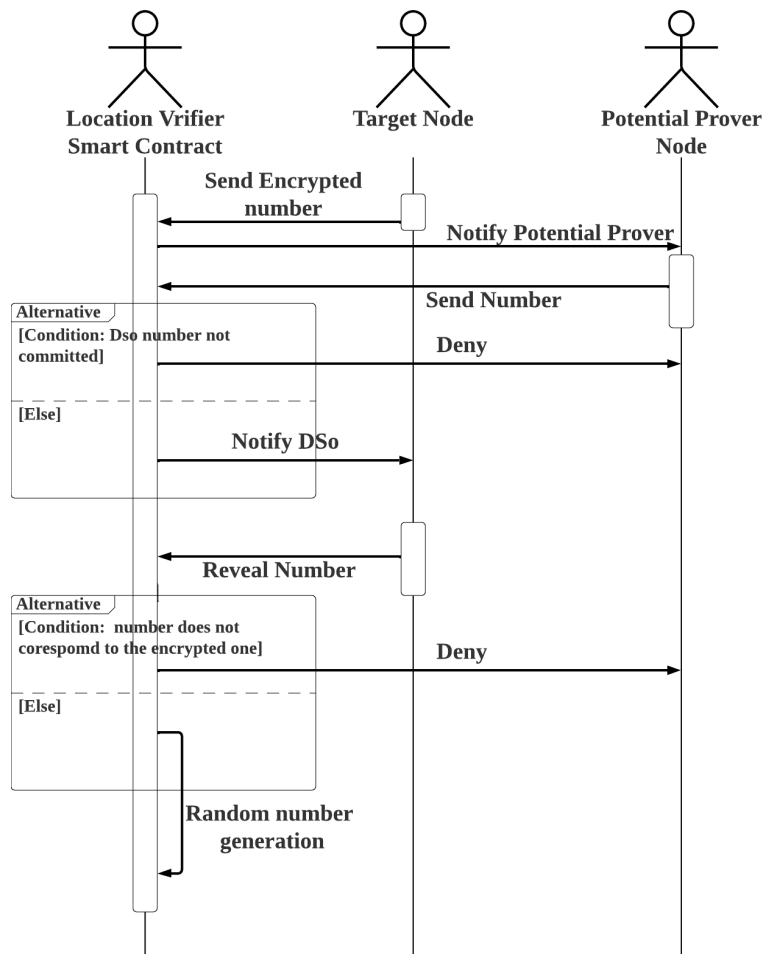


Figure 3.19: Random number generation process by the location verifier smart contract.

Random number generation is used for random selection of the prover. The function in question is described in pseudocode in Algorithm 3.18.

Algorithm 3.18 Location Verifier Smart Contract: SelectProvers

```
1: Mapping: Address → Address[]: TargetNodeCandidateProvers
2: Mapping: Address → Address[]: TargetNodeSelectedProvers
3: Mapping: Address → Uint: DsoSeedNumber
4: Mapping: Address ← Uint: NonDsoSeedNumber
5: Procedure SelectProvers(Address TargetNode)
6:   Uint: DsoNumber ← DsoSeedNumber(TargetNode)
7:   Uint: ProversNumber ← NonDsoSeedNumber(TargetNode)
8:   i ← 1
9:   while i ≤ TotalNumberOfSelectedProvers
10:    k ← keccak256(abi.encode(DsoNumber, ProversNumber, i)) % 6
11:    Prover ← TargetNodeCandidateProvers(TargetNode)[k]
12:    if Prover ∉ TargetNodeSelectedProvers(TargetNode) then
13:      TargetNodeSelectedProvers(TargetNode).push(Prover)
14:    End If
15:    i → i + 1
16:  End While
17:  Emit Event:   NotifyTargetNodeOfItsSelectedProvers(TargetNodeSelected-
18:    Provers(TargetNode))
19: End Procedure
```

d. Target Node Position Determination

The location verifier contract notifies the target node of its assigned prover after the prover nodes have been assigned. The target node, in turn, sends a RF message to each prover assigned to it, and each prover replies with the distance separating it from the target node calculated using TDoA. Before sending it back to the target node, each prover node signs the distance it has calculated. The location verifier smart contract verifies the signed distances received by the target node from the provers. Trilateration is used in TDoA localization to calculate the position of the target node based on the distances between the target node and at least three anchor nodes with known positions.

However, three distances from three separate anchor nodes are sufficient to position the target node by solving the formula for the set of three equations, given a target node with coordinates (x_T, y_T) and three anchor nodes, where the coordinates of each anchor i are (x_i, y_i) and d_i is the distance between the target node and each anchor i .

$$\begin{cases} d_1^2 = (x_T - x_1)^2 + (y_T - y_1)^2 \\ d_2^2 = (x_T - x_2)^2 + (y_T - y_2)^2 \\ d_3^2 = (x_T - x_3)^2 + (y_T - y_3)^2 \end{cases} \quad (3.24)$$

As there are six selected anchor nodes, the target positioning can be done twice. After the six calculated distances are sent to the smart contract for location verification, the smart contract randomly selects two groups of three anchors to derive the target position by trilateration using the two randomly selected anchor groups. The DSO and the prover nodes are involved in the random selection based on the same random number generation discussed earlier. Following the pseudocode described in Algorithm 3.19, the two positions calculated using the selected anchor node groups are determined.

Algorithm 3.19 Location Verifier Smart Contract: ComputetTargetNodePosition

```
1: Structure: Position
2:   Uint: x
3:   Uint: y
4: End Structure
5: Structure: ComputedDistance
6:   Address: Prover
7:   Uint: d
8: End Structure
9: Structure: PositionEquationParameters
10:  Uint: AnchorNodeXPosition
11:  Uint: AnchorNodeYPosition
12:  Uint: Distance
13: End Structure
14: Mapping: Address → Position: NodePosition
15: Mapping: Address → ComputedDistance[6] TargetNodeComputedDistances
16: Procedure ComputeTargetNodePosition(Address TargetNode) Returns Position
    P1, P2
17: PositionEquationParameters: DistanceEquation
18: PositionEquationParameters[2][3]: DistanceEquations
19: Uint[]: AlreadyPicked
20:  $i \leftarrow 1$ 
21:  $\text{nonce} \leftarrow 1$ 
22: While  $i \leq 2$ 
23:    $j \leftarrow 1$ 
24:   While  $j \leq 3$ 
25:      $k \leftarrow \text{keccak256}(\text{abi.encode}(\text{DsoNumber2}, \text{ProversNumber2}, \text{nonce})) \% 6$ 
26:     If  $k \notin \text{AlreadyPicked}$ 
27:        $\text{AlreadyPicked.push}(k)$ 
28:        $c \leftarrow \text{TargetNodeComputedDistance}(\text{TargetNode})[k]$ 
29:        $\text{DistanceEquation.Distance} \leftarrow c.d$ 
30:        $x \leftarrow \text{NodePosition}(c.\text{Prover}).x$ 
31:        $\text{DistanceEquation.TargetNodeXPosition} \leftarrow x$ 
32:        $y \leftarrow \text{NodePosition}(c.\text{Prover}).y$ 
33:        $\text{DistanceEquation.TargetNodeYPosition} \rightarrow y$ 
34:        $j \leftarrow j + 1$ 
35:        $\text{nonce} \leftarrow \text{nonce} + 1$ 
36:        $\text{DistanceEquations}[i-1][j-1].\text{push}(\text{DistanceEquation})$ 
```

```
37: Else
38:   nonce  $\leftarrow$  nonce+1
39: End If
40: End While
41: i  $\leftarrow$  i + 1
42: End While
43: Position P1, P2
44: P1  $\rightarrow$  solve(DistanceEquations[0])
45: P2  $\rightarrow$  solve(DistanceEquations[1])
46: Return P1,P2
47: End Procedure
```

Provers are rewarded in proportion to their stake if the two calculated points match; otherwise, they lose their stake. The calculated position of the target node is then compared to its legitimate position. The target node is an identity thief if the two positions do not match; otherwise, the transaction is accepted. Assume that the six selected provers have no prior knowledge of each other and do not collude, and each computes the distance between itself and the target node. If one of the nodes delivers an incorrect distance, the two locations estimated by the smart contract verifier will have different coordinates, causing all verifiers to lose their bet. Therefore, if the prover wants to participate in location verification, the specified coordinates of the prover nodes in the smart contract must be correct and they must invest in the implementation of the agreed network communication protocol and media standard required for TDoA positioning. However, regardless of the anchor-node combination used, trilateration positioning will always result in the same derived point if all six prover nodes decide to collude and agree on a specific target position, and if they all send the distance that separates them

from the agreed incorrect target position. As a result, this scheme is ineffective against prover node collusion. In the next section, we will look at how game theory is used to effectively address challenges. To solve the three expressions of Eq. 3.24, we can expand the squares in each expression as follows:

$$\begin{cases} x_T^2 - 2x_1x_T + x_1^2 + y_T^2 - 2y_1y_T + y_1^2 = d_1^2 \\ x_T^2 - 2x_2x_T + x_2^2 + y_T^2 - 2y_2y_T + y_2^2 = d_2^2 \\ x_T^2 - 2x_3x_T + x_3^2 + y_T^2 - 2y_3y_T + y_3^2 = d_3^2 \end{cases} \quad (3.25)$$

The second expression is then subtracted from the first, and the third expression is subtracted from the second in Eq. 3.25, yielding a system of two equations with two unknowns of the type:

$$\begin{cases} Ax_T + By_T = C \\ Dx_T + Ey_T = F \end{cases} \quad (3.26)$$

whose solution is given in Eq. 3.27:

$$\begin{cases} x_T = \frac{CE - FB}{EA - BD} \\ y_T = \frac{CD - AF}{BD - AE} \end{cases} \quad (3.27)$$

where:

$$\left\{ \begin{array}{l} A = -2x_1 + 2x_2 \\ B = -2y_1 + 2y_2 \\ C = d_1^2 - d_2^2 - x_1^2 + x_2^2 - y_1^2 + y_2^2 \\ D = -2x_2 + 2x_3 \\ E = -2y_2 + 2y_3 \\ F = d_2^2 - d_3^2 - x_2^2 + x_3^2 - y_2^2 + y_3^2 \end{array} \right. \quad (3.28)$$

Enhanced Anti Collusion PoL Using Game Theory

In blockchain-based energy scenarios, there are two parties with two conflicting goals. The DSO might be tempted to misrepresent a location by claiming that the request did not originate from the smart meter node location in order to deny the prosumer a legitimate token claim. The prosumer, in turn, might be tempted to falsely claim that its request originated from the smart meter location in order to illegitimately claim an energy token. This leads to a possible game-theoretic scenario, explained below. Suppose a target node Tn with coordinates (x_t, y_t) requests PoL, where its legitimate position as deployed and set by the DSO is (x^*, y^*) , and the position of the destination node calculated by TDoA trilateration is (x_c, y_c) . In the proposed system, the involved parties can be classified into three categories:

- The parties for whom it is harmful if $(x_t, y_t) \neq (x^*, y^*) \& (x_c, y_c) = (x^*, y^*)$, but beneficial for them if $(x_t, y_t) = (x^*, y^*) \& (x_c, y_c) \neq (x^*, y^*)$, i.e., the DSO or other

partners, because in the latter situation they could reject a legitimate prosumer's request for a token and enjoy the free energy aggregation. However, if $(x_t, y_t) \neq (x^*, y^*)$, and $(x_c, y_c) = (x^*, y^*)$, they could deliver an energy token to a non-deserving node.

- Parties that are harmed when $(x_t, y_t) = (x^*, y^*) \& (x_c, y_c) \neq (x_t, y_t)$; these are the prosumer nodes and their partners. They might be only tempted to maliciously seek the position (x^*, y^*) , to obtain a free energy token without having to supply the energy countervalue.
- Neutral parties who participate only for the financial reward.

Because the system is permissionless, all parties can apply to be location provers. It is proposed that N DSO prover nodes be defined in the smart contract, identified by their respective addresses. For prover applications, both DSO and non-DSO prover nodes are given fair slots, i.e., instead of a time-limited application, applications are open until a certain number m of applicants is reached, where $m/2$ applications are allowed for non-DSO nodes and $m/2$ for DSO nodes. Similar to the PoL scheme above, six examiners are randomly selected from the set of m , but there must be three DSO prover nodes and three non-DSO prover nodes. The function responsible for the random selection of examiners in this extended scheme is shown in the pseudocode of Algorithm 3.20. The distances asserted by the checkers must be sent to the smart con-

tract in encrypted form. They can only be decrypted after they have all already been submitted to the blockchain.

Algorithm 3.20 SelectRandomProvers

```

1: Uint: NumberofProverApplicants
2: Mapping: Address  $\rightarrow$  Uint: DsoSeedNumber
3: Mapping: Address  $\rightarrow$  Uint: NonDsoSeedNumber
4: Address[NumberofProverApplicants/2]: NonDsoProverNodes
5: Address[NumberofProverApplicants/2]: DsoProverNodes
6: Procedure ToggleBool(Bool b)
7:   If b = True Then
8:     b  $\leftarrow$  False
9:   Else
10:    b  $\leftarrow$  True
11:  End Procedure
12: Procedure SelectRandomProvers(Address TargetNode)
13:   Bool DsoProverTurn  $\leftarrow$  True
14:   i  $\leftarrow$  1
15:   While i  $\leq$  6
16:     DsoNumber2  $\leftarrow$  DsoSeedNumber(TargetNode)
17:     ProversNumber2  $\leftarrow$  NonDsoSeedNumber(TargetNode)
18:     k  $\leftarrow$  keccak256(abi.encode(DsoNumber2, ProversNumber2, i)) % (m/2)
19:     If DsoProverTurn = True Then
20:       Prover  $\leftarrow$  DsoProverNodes[k]
21:       If Prover  $\notin$  TargetNodeSelectedProvers(TargetNode) then
22:         TargetNodeSelectedProvers(TargetNode).push(Prover)
23:       End If
24:       ToggleBool(DsoProverTurn)
25:     Else
26:       Prover  $\leftarrow$  NonDsoProverNodes[k]
27:       If Prover  $\notin$  TargetNodeSelectedProvers(TargetNode) then
28:         TargetNodeSelectedProvers(TargetNode).push(Prover)
29:       End If
30:       ToggleBool(DsoProverTurn)
31:     End If
32:     i  $\leftarrow$  i + 1
33:   End While
34: End Procedure

```

After the six encrypted distances are committed, they are decrypted and output by the respective prover nodes. Two target position points are derived by the location verifier smart contract. One is calculated using the three distances given by the DSO prover nodes, and the second is calculated using the distances given by the non-DSO prover nodes, according to the pseudocode in Algorithm 3.21. Note that the only case in which DSO prover nodes can be tempted to lie is when the target coordinates $(x_c, y_c) \neq (x^*, y^*)$, if the target coordinates claimed by the DSO nodes $(x_c, y_c) = (x^*, y^*)$, this must be true, whereas non-DSO prover nodes can only be tempted to lie about the position of the target node by claiming $(x_c, y_c) = (x^*, y^*)$. If the target coordinates computed by the non-DSO nodes are $(x_c, y_c) \neq (x^*, y^*)$, this must be true. This allows game theory to be applied to this system, as shown in Figure 3.20.

We assume that the target positions calculated by the smart contract based on the distances of the DSO and non-DSO providers are (x_d, y_d) and (x_p, y_p) , respectively. We can discern for both DSO and non-DSO prover nodes in which they Certainly Tell the Truth (CTT) or Certainty Lie (CL), or situations in which no judgement can be made but At Least One Certainly Lies (ALOC). These situations are presented in Equations 3.29 and 3.30 for DSO and non-DSO prover nodes, respectively.

Algorithm 3.21 Location Verifier Smart Contract: ComputeTargetNodePosition

```
1: Structure: Position
2:   Uint: x
3:   Uint: y
4: End Structure
5: Structure: PositionEquationParameters
6:   Uint: AnchorNodeXPosition
7:   Uint: AnchorNodeYPosition
8:   Uint: Distance
9: End Structure
10: Mapping: Address  $\rightarrow$  Position: NodePosition
11: Mapping: Address  $\rightarrow$  Uint[6]: TargetNodeComputedDistances /*The 3 first
    distances corresponds to distances computed by DSO anchors, the rest by
    Non DSO anchors*/
12: Procedure ComputeTargetNodePosition(Address TargetNode) Returns
    Position P1, P2
13:   PositionEquationParameters DistanceEquation
14:   PositionEquationParameters[2][3] DistanceEquations
15:    $i \leftarrow 1$ 
16:   While  $i \leq 2$ 
17:      $j \leftarrow 1$ 
18:     While  $j \leq 3$ 
19:        $d \leftarrow$  TargetNodeComputedDistances(TargetNode)[ $j-1$ ]
20:       DistanceEquation.Distance  $\leftarrow$  d
21:       Prover  $\leftarrow$  TargetNodeSelectedProvers(TargetNode)[ $j-1$ ]
22:        $x \leftarrow$  NodePosition(Prover)
23:       DistanceEquation.TargetNodeXPosition  $\leftarrow$  x
24:        $y \leftarrow$  NodePosition(Prover).y
25:       DistanceEquation.TargetNodeYPosition  $\leftarrow$  y
26:       DistanceEquations[ $i-1$ ][ $j-1$ ].push(DistanceEquation)
27:        $j \leftarrow j + 1$ 
28:     End While
29:      $i \leftarrow i + 1$ 
30:   End While
31:   Position P1, P2
32:   P1  $\rightarrow$  solve(DistanceEquations[0])
33:   P2  $\rightarrow$  solve(DistanceEquations[1])
34:   Return P1,P2
35: End Procedure
```

$$\text{DSO prover node: } \left\{ \begin{array}{l} \text{CTT: } (x_d, y_d) = (x^*, y^*) \parallel (x_d, y_d) = (x_p, y_p) \neq (x^*, y^*) \\ \text{CL: } (x_d, y_d) \neq (x_p, y_p) \& (x_p, y_p) \neq (x^*, y^*) \\ \text{ALOC: } (x_d, y_d) = (x^*, y^*) \& (x_p, y_p) \neq (x^*, y^*) \end{array} \right. \quad (3.29)$$

$$\text{Non-DSO prover node: } \left\{ \begin{array}{l} \text{CTT: } (x_p, y_p) \neq (x^*, y^*) \parallel (x_p, y_p) = (x_d, y_d) = (x^*, y^*) \\ \text{ALOC: } (x_p, y_p) \neq (x^*, y^*) \& (x_d, y_d) = (x^*, y^*) \end{array} \right. \quad (3.30)$$

Tables 3.2 and 3.3 illustrate gains for both DSO and non-DSO provers' according to their adopted game strategy (Lie/Tell the truth).

Table 3.2

Strategy played versus gain in the game theory scenario between DSO and non-DSO provers, where $(x_t, y_t) = (x^*, y^*)$.

		DSO Prover Node	Lie:	Truth:
Non-DSO Prover Node			$(x_c, y_c) \neq (x^*, y^*)$	$(x_c, y_c) = (x^*, y^*)$
	Truth: $(x_c, y_c) = (x^*, y^*)$		(-10, -10)	(10, 10)
	Lie (declared without computing): $(x_c, y_c) = (x^*, y^*)$		(-10, -10)	(10, 10)

Table 3.3

Strategy played versus gain in the game theory scenario between DSO and non-DSO provers, where $(x_t, y_t) \neq (x^*, y^*)$.

		DSO Prover Node	Truth:	Lie:
Non-DSO Prover Node			$(x_c, y_c) \neq (x^*, y^*)$ & $(x_c, y_c) = (x_t, y_t)$	$(x_c, y_c) \neq (x^*, y^*)$ & $(x_c, y_c) \neq (x_t, y_t)$
	Lie: $(x_c, y_c) = (x^*, y^*)$		(-10, -10)	(10, 10)
	Truth: $(x_c, y_c) \neq (x^*, y^*)$ & $(x_c, y_c) = (x_t, y_t)$		(20, 20)	(30, -30)

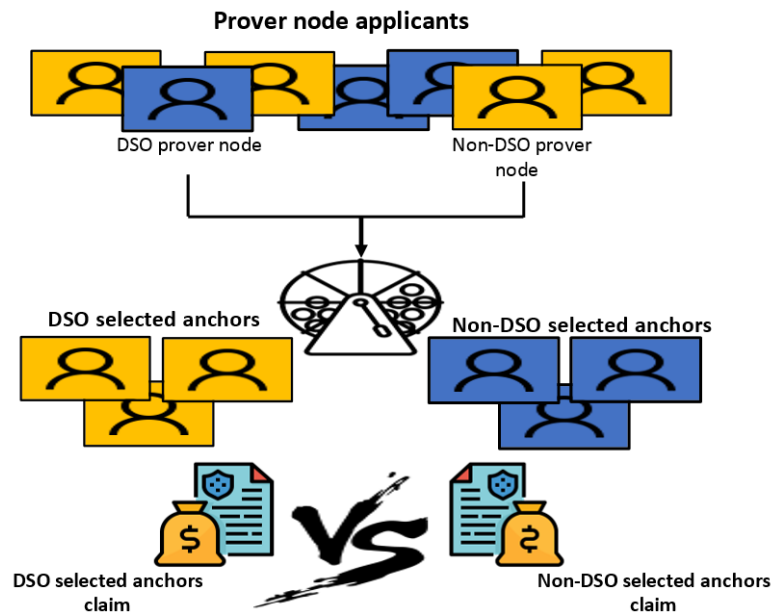


Figure 3.20: Game theory scenario in target node positioning between DSO and Non-DSO anchors.

To calculate the Nash equilibrium, we calculate the total gain for both entities in the two situations described in Tables 3.2 and 3.3—when the entity lies and when it is truthful. For the non-DSO prover node, the total gain is 50 when it is truthful and -20 when it lies. For the DSO-prover node, the total gain is 30 if it is truthful and -60 if it lies. Although the non-DSO prover node has an advantage in this game, the Nash equilibrium for both parties is to be truthful. The proposed PoL procedure is secure and safe against prover-target collusion. As for target-target collusion, it is not a threat to the proposed P2P application, as any target-target collusion is irrelevant if it does not usurp the respective legitimate position of the smart meter.

After P1 and P2 are calculated by the location verifier smart contract correspond-

ing to target position claimed by the DSO and non-DSO sets of anchors, respectively, the PoL for the particular target node request is set according to the pseudocode in Algorithm 3.22. After the PoL is computed, it must be included as a parameter in the *ClaimToken* function noted in Algorithm 3.16. To verify the validity of the PoL passed by the caller in the *ClaimToken* function, the ERC20 smart contract delivering the energy token checks the validity of the asserted PoL by first verifying that it was issued by the location verifier smart contract by calling a getter function that accesses the key-value mapping linking the destination node addresses to their most recently computed PoL. The latter must be identical to the one passed in the *ClaimToken* function call. The ERC20 smart contract that issues the energy tokens must also verify that the PoL Id matches that of the *ClaimToken* call. Both Ids are associated with the respective function caller by mapping. It should be noted that both Ids are constrained and can only be incremented when the appropriate function is called. Because there must be a PoL for each token claim, the two Ids must be identical. How the PoL is checked when a token is claimed is shown in the pseudocode in Algorithm 3.23.

Algorithm 3.22 Location Verifier Smart Contract: SetPoL, GetNodePol

```
1: Structure: PoL
2:   Uint: PoLRequestId
3:   Position: P
4: End Structure
5: Mapping: Address  $\rightarrow$  PoL: TargetNodePoL
6: Procedure SetPol(PcomputedByDsoAnchors, PcomputedByNonDsoAnchors)
7:   PoL ProofOfLocation
8:   P1  $\leftarrow$  PcomputedByDsoAnchors
9:   P2  $\leftarrow$  PcomputedByNonDsoAnchors
10:  P*  $\leftarrow$  NodePosition(TargetNode)
11:  If P1=P2 Then
12:    ProofOfLocation.P $\leftarrow$  P1
13:    ProofOfLocation.PoLRequestId $\leftarrow$  RequestId(TargetNode)
14:  ElseIf P1 != P2 Then
15:    If P1 != P* & P2 != P* Then
16:      ProofOfLocation.P $\leftarrow$  P2
17:      ProofOfLocation.PoLRequestId $\leftarrow$  RequestId(TargetNode)
18:    Else
19:      Revert() /* The proof of location is not considered and need to be recomputed*/
20:    End If
21:  End If
22: End Procedure
23: Procedure GetNodePol(Address Addr) Returns(PoL)
24:   Return TargetNodePoL(Addr)
25: End Procedure
```

Algorithm 3.23 ERC20 Smart contract: Verify

```

1: Address: LocationVerifierSmartContractAddress
2: Mapping: Address → Position: TargetNodeDefinedPosition
3: Procedure Verify(Pol L) Returns (Bool)
4:  $c \leftarrow$  CallLocationVerifierSmartContract(LocationVerifierSmartContractAddress)
5:  $l \leftarrow$  c.GetNodePol(msg.sender)
6:  $Id \leftarrow$  RequestId(msg.sender)
7:  $N \leftarrow$  TargetNodeDefinedPosition(msg.sender)
8: if ( $L = l$  &  $l.RequestId = Id$  &  $L.Position = N$ ) Then
9:   Return true
10: Else
11:   Return false
12: End If
13: End Procedure

```

The entity diagram relationship in the proposed PoL scheme using the location verifier smart contract is described in Figure 3.21.

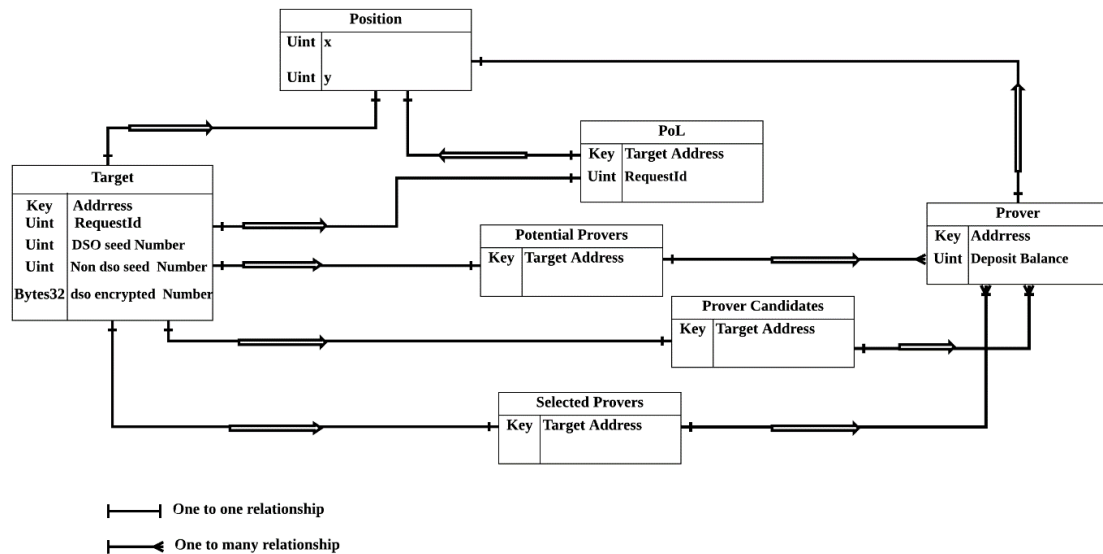


Figure 3.21: Entity relationship diagram of the location verifier smart contract.

Time Difference of Arrival Time difference of arrival (TDoA), also known as multilateration, is a proven method for positioning RF transmitters. Using three or

more anchor receivers, TDoA positions a signal source based on the difference in arrival times at the receivers. Before a round of TDoA positioning begins, both the target and its assigned anchor nodes must establish time synchronization. Here, $t_0 = 0$ refers to the exact time at which the target sends a RF message to the respective anchor nodes. The distance between the anchor and the RF sender anchor node is calculated based on the time it takes for the RF message to reach the anchor node, as given in Eq. 3.31:

$$distance = SpeedOfLight \times time \quad (3.31)$$

The time delay is determined using the cross-correlation in the time domain between the received signals at two anchors, which peaks when $t = \tau$, where τ is the time delay between the two signals. Given a set of n received signals $R_i(t)$, each received at anchor i where $0 < i \leq n$; Given a pair of sampled received signals $R_1(n), R_2(n)$. Time delay at which the two signals are received is computed by time domain cross-correlation and is found according to Eq. 3.32:

$$\begin{cases} corr(m) = \sum_{m=0}^{N-1} R_1(m)R_2(m+n) \\ corr(\tau_{1,2}) = Max(\sum_{m=0}^{N-1} R_1(m)R_2(m+n)) \end{cases} \quad (3.32)$$

where $\tau_{1,2} = \tau_1 - \tau_2$

Thus: $c\tau_{12} = c\tau_1 - c\tau_2 = d_1 - d_2$; where d_1 is the target distance from anchor 1, and d_2 is its distance from anchor 2. The hyperbola function of the distances between

anchor nodes 1 and 2 and the target is given in Eq. 3.33:

$$\begin{cases} d_1 = \sqrt{(x_T - x_1)^2 + (y_T - y_1)^2} \\ d_2 = \sqrt{(x_T - x_2)^2 + (y_T - y_2)^2} \end{cases} \quad (3.33)$$

However, d_1 and d_2 are both unknown, only $d_1 - d_2$ is known, which was derived according to Eq. 3.34:

$$d_1 - d_2 = \sqrt{(x_T - x_1)^2 + (y_T - y_1)^2} - \sqrt{(x_T - x_2)^2 + (y_T - y_2)^2} \quad (3.34)$$

To solve (x_T, y_T) , two more equations are needed. Thus, we need two more pairs of anchors and the time lag between their received signals. Therefore, this approach requires six anchor nodes to position the target node. However, there are other numerical methods that can be used to calculate the target position with fewer anchor nodes, such as in (Phruksahiran & Michanan, 2021).

The TDoA positioning accuracy is not really affected by the distance between the target node and the anchors, as TDoA is the positioning technique implemented by GPS localization and uses satellite anchors that are thousands of kilometers away from the target nodes. TDoA accuracy is mainly influenced by the quality of correlation, which depends on several factors:

- The nature of the received RF signal (especially its bandwidth);

- The different characteristics of the receivers;
- The different propagation paths between the transmitter and the receivers;
- The correlation method used.

However, the range between anchors and targets must be consistent with the wireless communication protocol (WCP) used. Table 3.4 shows the range in meters of different WCPs that allow precise TDoA localization (*LoRa Alliance TM Strategy Committee, LoRaWAN Geolocation Whitepaper, 2021*).

Table 3.4
Distance coverage for different wireless communication technologies.

Wifi	Bluetooth	LoRaWan	ZigBee
20–50 m	1–7 m	1000–2000 m	20–100 m

3.2.3 Proposed, fully decentralized, cost-effective P2P energy demand response management system, with smart contract-based OPF solution

The current tremendous expansion of distributed energy sources and the democratization of electricity generation through conventional and renewable sources such as wind turbines, solar energy, and advances in energy storage capacity (Z. Wang et al., 2021; Sotkiewicz & Vignolo, 2007), combined with the advancement of digital communication and control technologies, have enabled the transition from traditional grid systems to what is known as a smart grid, in which traditional consumers have evolved into proactive prosumers that can join a grid branch of the larger distribution network

and contribute to energy demand management within the grid they have joined, either with traditional fuel-based energy generators or renewable sources (Rinaldi et al., 2019; Wu & Conejo, 2022). In addition, the integration of blockchain into the smart grid and the functionalities of smart contracts enabled the emergence of peer-to-peer energy trading (Merrad et al., 2022; Junlakarn et al., 2022), where prosumers can trade their energy generated and injected into the grid in a decentralized manner. It should be noted that the reduction of influence and control by a central authority is paramount in all blockchain-based applications (Shabani, 2019). Similarly, the decentralization of Blockchain-based P2P energy trading platforms and the safe reduction of the authority of the DSO is an aimed target (Han et al., 2020; Gajić et al., 2022). Since prosumers mainly use renewable energy sources, energy and climate policies promote and support such energy platforms to meet both increasing energy demand and sustainable climate goals (“Community energy strategy: Full report”, Dec. 2014; Afzal et al., 2020). On the other hand, end users can reduce their electricity costs and gain lucrative benefits, making such platforms a win-win scenario. In P2P energy trading platforms, prosumers are rewarded accordingly for their delivered energy through crypto-tokens, which are either fungible or non-fungible (Karandikar et al., 2021) and delivered through appropriately designed smart contracts. In an integrated system, the utility has complete control over the location and connection of distributed energy resources (DER) and will seek to optimally manage demand, while in unbundled systems, each prosumer seeks

its own benefit and its goal is to maximize its lucrative profit, according to market rules. Thus, from this perspective, a prosumer that joins a grid and the grid operator can potentially have conflicting goals. The DSO analyzes grid operations and grid investments in terms of peak power flows, while the prosumer sees its revenue in terms of aggregated energy exchange. Moreover, the flexibility of the smart grid, where new prosumers can constantly join or leave the grid, poses a challenge to the effective implementation of demand response control and management systems, which require less flexible deployment of hardware (Jayachandran et al., 2022). Moreover, a demand response control and management system deployed by DSOs would not be considered a decentralized solution and would run counter to the consortium-based decision-making principle of blockchain applications. One of the most widely used demand response solutions is the OPF based control system. The OPF solution aims to achieve a steady-state operating point in the grid that reduces power generation costs while satisfying demand and operating constraints (Z. Wang & Anderson, 2021). The approach to optimize the power injection within grids considering thresholds is generally challenging because it is a nonlinear and non-convex optimization problem. With renewable energy sources in the grid, the OPF problem becomes even more complicated in both formulation and solution because the generation capacities of the renewable energy sources, which are part of the constraints of the problem, are unpredictable (Riaz et al., 2021). Therefore, many researchers have addressed the formulation and solution of the OPF problem

for hybrid grids and presented a number of ingenious models (Nusair & Alasali, 2020; Hassan et al., 2021). Most of the works have focused on presenting realistic mathematical models or algorithms to solve the OPF problem for hybrid power sources. However, beyond that, little work has been done on how the computed OPF solution can be used for decentralized power generation control in smart grids. So far, the established scheme is to use control units (hardware and software), which does not go in the direction that aims to minimize DSO authority. A novel decentralized, transparent, and secure OPF model is used to locally coordinate power generation in distributed networks while taking into account network constraints, without the need for centralized control unit. The research describes a detailed approach for implementing the decentralized OPF on a private blockchain smart contracts platform that enables an immutable and access-controlled transaction system for tokenized power assets. The model solves the OPF problem of a given grid where all constraints and fixed parameters are set within an immutable and autonomous smart contract. The only variable parameter is the load demand, which can be updated in the smart contract by a load monitoring unit, either periodically (Jamil & Mittal, 2020) or in real time (Vale et al., 2021) or even using short-term load forecasting (Ahmad et al., 2019), which would allow the smart contract to operate without any required outside interaction. The smart contract solves the OPF problem for a local network. The OPF solution would be computed by a decentralized, unbiased smart entity and would thus be unchallengeable. The solution would be stored

in the blockchain public ledger, making it safe from tampering. Prosumers would only receive the energy tokens they are entitled to if they comply with the OPF solution of the smart contract.

The structure of the article can be outlined as follows. Section 2 explores the feasibility of an on-chain solution to the OPF problem through a case evaluation of the execution cost of an on-chain solution to an OPF model for a 3-bus network using the DC-OPF linear approximation. Section 3 describes the proposed new improved smart contract based model that can be generalized for any defined OPF problem with effective execution cost.

3.2.3.1 On-chain based solution for a 3-bus network, using DC-OPF approximation

Smart contracts are not suitable for computationally intensive problems because the execution cost of transactions is proportional to their processing complexity (G. Wang et al., 2020; Y. Du et al., 2021). Moreover, complex code can lead to potential gaps in the source code that can be exploited by hackers to intentionally bug the system (Xing et al., 2020). This is fatal for the blockchain application that is tailored to the smart contract in question, as flaws in smart contracts cannot be repaired since the source code cannot be modified once it is deployed. In this section, we attempt to develop and evaluate the execution cost of an on-chain OPF solution for a simple 3-bus network. In doing so, we use a simplified approach to the OPF problem, called the DC-OPF (Ergun et al.,

2019) model. The goal is to evaluate how realistic it is to implement an on-chain OPF solution by comparing the execution cost for a much simpler version. Thus, if the cost for a simple version is already too high, the approach is not suitable for more complex models.

3.2.3.1.1 DC-OPF model

Even though the model is called DC, it does not refer to DC high voltage generators power flow through transmission lines. Actually, the model is indeed an attempt to give a linear approximation to non-linear AC power network equations. In fact, power flow between nodes i and j in a power network is represented by Eq. 3.35.

$$S_{i,j} = V_i^2 y_{ij}^* - V_i V_j^* y_{ij}, \quad (3.35)$$

where V_i and V_j are the voltage at nodes i and j , respectively and y_{ij} is the line admittance and. They are all complex quantities, the $(^*)$ operator refers to the conjugate of a complex entity. DC-OPF considers line flow constraints, as opposed to the Economic Dispatch, which is the simplest approximation to the OPF problem and which only considers the grid as only a single transmission line with no constraints on power flow between two nodes. how transmission line is modelled in DC-OPF approximation is illustrated in Fig 3.22.

The first step in linearizing the power flow equations is to consider only the series

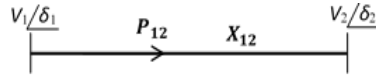


Figure 3.22: Transmission line model in DC-OPF.

reactance of the line and neglect its ohmic resistances in the transmission line model π described in Section 3. Accordingly, the line active power flow between point 1 and 2 is given by Eq. 3.36 as in Fig. 3.22, where $V_1, V_2, \delta_1, \delta_2$ are the voltage magnitudes and angles for points 1 and 2, respectively and X_{12} is the line reactance:

$$P_{12} = V_1 V_2 \frac{\sin(\delta_1 - \delta_2)}{X_{12}}, \quad (3.36)$$

As one can see, Eq. 3.36 is obviously nonlinear; to linearize it, the following assumptions are made: Voltages are nominal constants, where:

$$V_1 = V_2 = 1 \text{ Per Unit (p.u.); } \sin(\delta_1 - \delta_2) = \delta_1 - \delta_2$$

So, we obtain Eq. 3.37:

$$P_{12} = V_1 V_2 \frac{(\delta_1 - \delta_2)}{X_{12}} = (\delta_1 - \delta_2) * b_{ij} \quad (3.37)$$

These assumptions are only possible for networks that are not heavily loaded. The more heavily loaded the network, the more important are the phase angles at the nodes of the network buses. The larger the angle t , the less accurate the approximation $\sin(t) = t$.

Thus, the active power flow on the line is related to the voltage phase angles δ and the line susceptance b_{ij} . The power injected at a given generator bus i , connected to

n other buses via n transmission lines, and its relation to the reactive power flow of the transmission lines is described in Eq. 3.38.

$$P_i = \sum_{\substack{j=1 \\ j \neq n}}^n P_{ij} = \sum_{\substack{j=1 \\ j \neq i}}^n b_{ij}(\delta_i - \delta_j); i > 1; i \neq j, \quad (3.38)$$

The matrix notation for Eq. 3.38 is described in Eq. 3.39.

$$P = B\delta = \begin{bmatrix} P_1 \\ \cdot \\ \cdot \\ \cdot \\ P_n \end{bmatrix} = \begin{bmatrix} b_{12} + \dots + b_{1n} & \dots & -b_{1n} \\ \vdots & \ddots & \vdots \\ -b_{n1} & \dots & b_{n1} + \dots + b_{nn-1} \end{bmatrix} \begin{bmatrix} \delta_1 \\ \cdot \\ \cdot \\ \cdot \\ \delta_n \end{bmatrix}, \quad (3.39)$$

B is called the bus susceptance matrix for the power network of n connected buses. B is used to calculate the DC-OPF. Note that all scalar elements of the diagonal matrix $B_{ii}, i \in [1, n]$ are calculated by formula in Eq. 3.40.

$$B_{ii} = \sum_{\substack{j=1 \\ j \neq n}}^n b_{ij} \quad (3.40)$$

As for non diagonal elements of B where $i \neq j$, they are either null if there is no line between nodes i and j , otherwise $B_{ij} = -b_{ij}$

The OPF objective function consists of minimizing the generators' power injec-

tion cost, subject to the voltage angles of the lines and the maximum power capacity of the generators and transmission lines constraints while fulfilling load power demand. Having defined cost functions $C(P_G)$ for all generators in the network and the equation for power flow along each transmission line that defines the relationship between power injections and voltage angles, deduced through the bus susceptance matrix, the OPF problem in a network with n generation buses can be formulated as in Eq. 3.41:

$$F(P_G) = \sum_{i=0}^n C(P_{Gi}), \quad (3.41)$$

Eq. 3.41 is subject to the constraints defined in Eq. 3.42.

$$\left\{ \begin{array}{l} B\delta = P_G - P_D, \\ H_1(P_{ij}) = \frac{1}{x_{ij}}(\delta_i - \delta_j) \leq P_{ij}^{max}, \\ H_2(P_{ij}) = \frac{1}{x_{ij}}(\delta_i - \delta_j) \geq -P_{ij}^{max}, \\ G_1(P_i) = P_{Gi} \leq P_G^{max}, \\ G_2(P_i) = P_{Gi} \geq P_G^{min}, \end{array} \right. \quad (3.42)$$

Generator cost functions define the relationship between power generated and cost. A study of how generator cost functions are derived can be found in (Durvasulu &

Hansen, 2018). However, the most commonly used type of generator cost function is of quadratic form, as formulated in Eq. 3.43:

$$C_i(P_i) = \alpha_i P_i^2 + \beta_i P_i + \gamma_i, \quad (3.43)$$

$\alpha_i, \beta_i, \gamma_i$ are scalar constants. The equation of the bus susceptance matrix represents equality constraints for the OPF objective function, whereas the transmission lines and the maximum power capacity of the generators are the inequality constraints for the problem. The Lagrangian multipliers and the Hessian matrix derived from the Lagrangian function are used to solve the OPF objective function defined by the DC-OPF model for power networks. A reference bus with a fixed voltage phase angle $\delta_{ref} = 0$ must be defined for this purpose. Eq. 3.44 defines the Lagrangian of the DC model OPF problem in a network of N_b generation buses and M transmission lines.

$$\begin{aligned} L = & \sum_{i=1}^{N_b} C(P_{Gi}) + \lambda_i (B\delta - P_G - P_D) + \sum_{i=1}^{N_b} \mu_i G_1(P_{Gi}) - \sum_{i=1}^{N_b} \mu'_i G_2(P_{Gi}) \\ & + \sum_{k=1}^M \beta_k H_1(P_{Gm}) - \sum_{k=1}^M \beta'_k H_2(P_{Gm}), \end{aligned} \quad (3.44)$$

Considering only the problem equality constraints, the set of linear equations derived from the Lagrangian function in Eq. 3.44 is as defined in Eq. 3.45.

$$\begin{cases} \frac{dL}{dP_{G1}} = 0; \dots \frac{dL}{dP_{Gnb}} = 0, \\ \frac{dL}{d\lambda_1} = 0; \dots \frac{dL}{d\lambda_{nb}} = 0, \\ \frac{dL}{d\delta_1} = 0; \dots \frac{dL}{d\delta_{nb}} = 0, \end{cases} \quad (3.45)$$

Inequality constraints need to fulfill the Karush Khun Tucker conditions (Gordon & Tibshirani, 2012). This means that for every inequality constraint defined, there are two possibilities. Either the inequality constraint is satisfied at the boundary condition and actually functions as an equality constraint, and its Lagrange multiplier is nonzero. If the inequality condition is inactive, it does not matter; its Lagrange multiplier is zero. In Eq. 3.44, we have $(2N_b + 2M)$ inequality constraints with two possibilities each, thus, allowing $2^{(2N_b+2M)}$. Accordingly, for each combination, the active constraint is considered as an equality constraint and Eq. 3.45 is updated accordingly. The solution corresponding to a particular combination must satisfy all system constraints and all computed Lagrange multipliers must be greater than or equal to zero. Otherwise, the respective solution would be considered invalid.

3.2.3.1.2 Implementation for a 3-bus DC-OPF problem

To control a power grid in a decentralized manner using a smart contract, a smart contract must be developed and deployed for a specific grid. First, a reference bus must be selected. To ensure decentralization and fairness the selection for the reference bus should be cyclic. However to simplify the implementation we fixed the reference bus to

bus 1 for the network illustrated in Fig. 3.23. In a power system, all parameters are fixed except the load, which varies. The load can be monitored in real time or periodically and updated to the controlling smart contract or can even be pre-implemented in the smart contract using forecasted data. The implementation was done using hourly updating of the load. Off-chain load monitoring unit calls the smart contract periodically to update the load. Before the load is changed, the most recently computed OPF solution is shared on the ledger bidding each generator to its optimal power flow generation requirements. The OPF solution is linked to its corresponding date in *Hour/Day/Month/Year* format. The date is updated on-chain internally in the smart contract each time the load monitoring unit updates the network load demand in a hourly basis, and serves as a non-contestable timestamp issued by an impartial party. How this is implemented in the concerned smart contract is illustrated in pseudocode in algorithm 3.24.

Algorithm 3.24 OPF Solver Smart contract: Update DC-OPF Buses Load Demand.

```
1: Fixed  $Pl_1$ 
2: Fixed  $Pl_2$ 
3: Fixed  $Pl_3$ 
4: Structure: Date
5:   Uint: Hour
6:   Uint: Day
7:   Uint: Month
8:   Uint: Year
9: End Structure
10: Date: CurrentDate
11: Address: LoadMonitoringUnit
12: Constructor()
13:   CurrentDate.Hour  $\leftarrow$  hh/* set to contract deployment hour */
14:   CurrentDate.Day  $\leftarrow$  dd/* set to contract deployment day */
15:   CurrentDate.Day  $\leftarrow$  mm/* set to contract deployment month */
16:   CurrentDate.Day  $\leftarrow$  yy/* set to contract deployment year */
17: End Constructor
18: Procedure UpdateCurrentDate()
19:   CurrentDate.Hour  $\leftarrow$  (CurrentDate.Hour + 1)% 24
20:   If CurrentDate.Hour == 0 Then
21:     CurrentDate.Day  $\leftarrow$  (CurrentDate.Day + 1)% 7
22:     If CurrentDate.Day==0 Then
23:       CurrentDate.Month  $\leftarrow$  CurrentDate.(Month+1)% 12
24:       If CurrentDate.Month==0Then
25:         CurrentDate.Year  $\leftarrow$  CurrentDate.Year + 1
26:       End If
27:     End If
28:   End If
29: End Procedure
30: Procedure UpdateBusesLoadDemand(Fixed P11, Fixed P12, Fixed P13)
31:    $Pl_1 \leftarrow$  P11
32:    $Pl_1 \leftarrow$  P12
33:    $Pl_1 \leftarrow$  P13
34:   UpdateCurrentDate()
35: End Procedure
```

The example aimed to be solved is a 3 buses network with a load and generation

line on each bus connected in a ring topology, as shown in Fig. 3.23. The generation cost function for all generators in the network, is of quadratic form as in Eq. 3.43. The

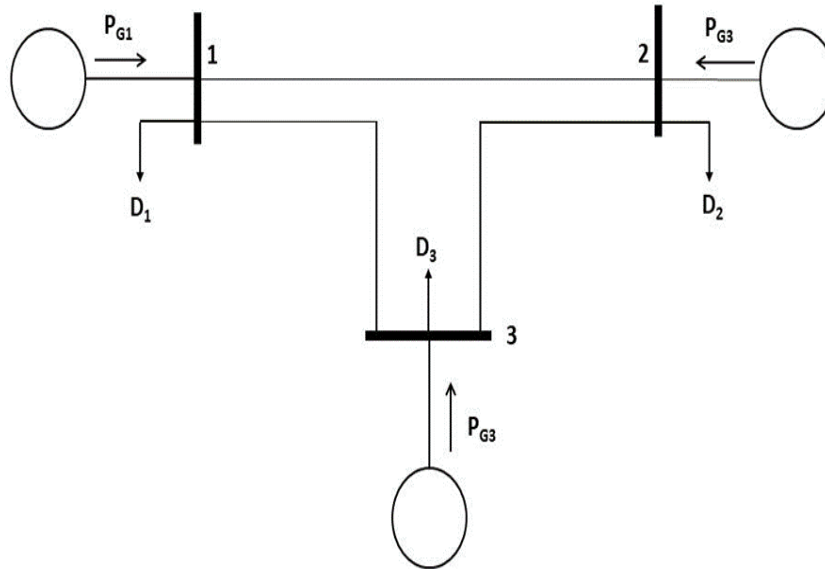


Figure 3.23: 3-bus power network.

generation cost functions' parameter for each generator and their generation capacity, as well as the network parameters are defined in Tables 3.5 and 3.6, respectively.

Table 3.5
Generators cost coefficients and generation capacity.

Unit	Bus	Cost coefficients			P_{gmin}	P_{gmax}
		a	b	c		
G_3	3	118.8206	37.8896	0.01433	5.0	20.0
G_2	2	218.3350	18.1000	0.00612	10.0	150.0
G_1	1	142.7348	10.6940	0.00463	20.0	200.0

Table 3.6
Transmission lines parameters.

Line	Resistance	Reactance(P.u.)	Flow Limit (MW)
1-2	0.0	0.20	55.0
1-3	0.0	0.40	55.0
2-3	0.0	0.25	55.0

The network bus susceptance matrix B is defined in Eq. 3.46.

$$\begin{pmatrix} B_{\times 11} & B_{\times 12} & B_{\times 13} \\ B_{\times 21} & B_{\times 22} & B_{\times 23} \\ B_{\times 31} & B_{\times 32} & B_{\times 33} \end{pmatrix} = \begin{pmatrix} 1x_{12} + 1x_{13} & -1x_{12} & 1x_{13} \\ -1x_{12} & 1x_{12} + 1x_{23} & -1x_{23} \\ -1x_{13} & -1x_{23} & 1x_{13} + 1x_{23} \end{pmatrix} \quad (3.46)$$

The Lagrange function for the system is given in Eq. 3.47

$$L = \sum_{i=1}^{Nb} (a_i + b_i P_{Gi} + c_i P_{Gi}^2) + \sum_{i=1}^{Nb} \lambda_i \left(\sum_{j=1}^{Nb} \beta_{ij} \gamma_j - P_{Gi} + P_{ij} \right) + \sum_{k=1}^M \mu_k (P_{Gi} - P_G^{max}) \quad (3.47)$$

From Eq. 3.47 we deduce Eq. 3.48, where the partial derivative of the Lagrangian function with respect to each variable is null.

$$\begin{cases} \frac{dL}{d\mu_1} = 0; \dots \frac{dL}{d\mu_{nb}} = 0, \\ \frac{dL}{d\mu'_1} = 0; \dots \frac{dL}{d\mu'_{nb}} = 0, \\ \frac{dL}{d\beta_1} = 0; \dots \frac{dL}{d\beta_{nb}} = 0, \\ \frac{dL}{d\beta'_1} = 0; \dots \frac{dL}{d\beta'_{nb}} = 0, \end{cases} \quad (3.48)$$

From Eq. 3.48, we derive 24 linear equations with 24 variables. Since the load on each bus is the only variable parameter in the network, the goal is to come up with a system of linear equations as described in the matrix described in Eq. 3.49 where the coefficients are functions of the buses load power demand.

$$\begin{bmatrix} [ccc|c]Coef_{00}(P_{l1},P_{l2},P_{l3}) & \cdots & Coef_{024}(P_{l1},P_{l2},P_{l3}) & Cst_1 \\ \vdots & \ddots & \vdots & \vdots \\ Coef_{240}(P_{l1},P_{l2},P_{l3}) & \cdots & Coef_{2424}(P_{l1},P_{l2},P_{l3}) & Cst_{24} \end{bmatrix}, \quad (3.49)$$

P_{l1}, P_{l2}, P_{l3} are then sent to the smart contract every hour and the coefficients of the matrix are set accordingly in the smart contract. As explained earlier, each Lagrange multiplier can assume one of two states with respect to the inequality constraints, either as an equality constraint or as neglected. Since there are 12 inequality constraints, this leads to $2^{12} = 5096$ possibilities. However, the system has a unique solution. The system of linear equations in Eq. 3.49 is solved for each possibility by looping through them until coming up with a valid solution as described in the flowchart in Fig. 3.24. Solutions for which a multiplier is negative or which do not satisfy the system constraints are not considered valid.

To solve each of the possible systems of linear equations, each matrix must be row-reduced so that all diagonal coefficients $Coef_{ii}$ are ones and while the rest are zeros

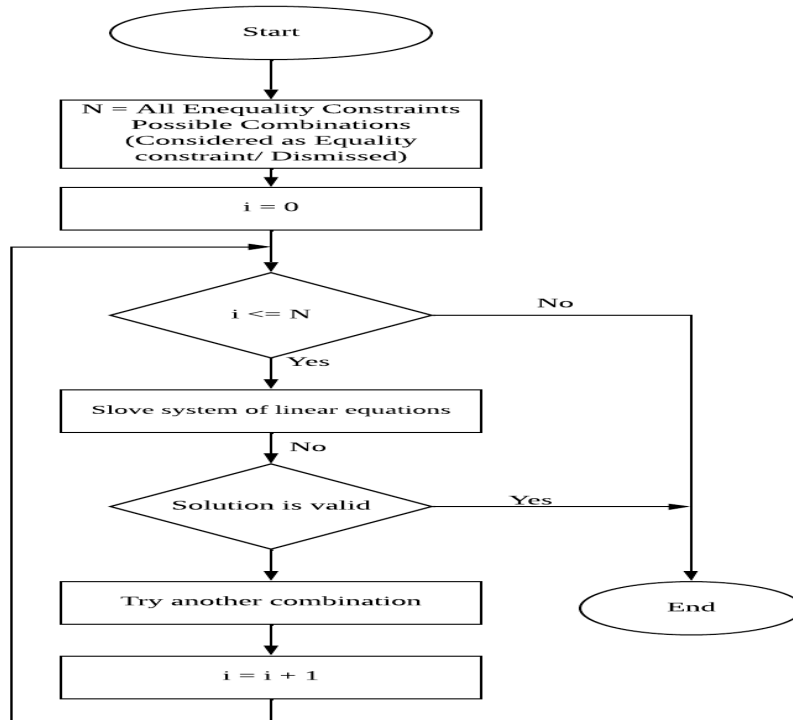


Figure 3.24: Solving DC-OPF problem.

to obtain a matrix of the form of Eq. 3.50.

$$\begin{bmatrix} 1 & \cdots & 0 & R_{cst1} \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & R_{cst24} \end{bmatrix}, \quad (3.50)$$

The solution to the linear system of equations, which is the solution to our OPF objective function, is then $S = \{S_1, \dots, S_{23}\}$, which represents the P_G at each generator as well as the angular displacement at each end of the transmission line, as well as all the Lagrange multipliers, simply saying $S_i = R_{cst_i}$. The main function, which loops through

all the linear systems of equations corresponding to each of the possible combinations of inequality constraints, is described in the pseudocode in Algorithm 3.25.

Algorithm 3.25 OPF Solver Smart contract: Solving DC-OPF Matrix (Main Function)

```

1: Procedure UpdateMatrixCoeff(Fixed[][] Coeff)
2:   UInt[12] Sum /* a local array of 12 uint elements, where all indexes are initialized
   to 0 */
3:   UInt i ← 0
4:   UInt j ← 0
5:   While !(∀ s in Sum, s==1) Do
6:     While i ≤ 11 Do
7:       If Sum[i]==0
8:         DeleteRow(i+12, Coeff) /* Delete row corresponding to the inequality
           constraint multiplier that is not considered */
9:       End If
10:      While j ≤ 12 Do /* Set to zero columns corresponding to the inequality
           constraint multiplier that are not considered */
11:        Coeff[i][j+11] ← Coeff[i][j+11] * Sum[i]
12:        j ← j + 1
13:      End While
14:      i ← i + 1
15:    End While
16:    Increment(Sum)
17:    Bool Solved ← Solve(Coeff)
18:    If Solved Then
19:      Break /* Break from the loop, system already solved */
20:    End If
21:  End While
22: End Procedure

```

The function internally calls auxiliary functions that are used to update the matrix coefficient of the linear system of equations that corresponds to the chosen combination of inequality constraints. These are described in the pseudocode in Algorithm 3.26. The Lagrange multipliers for inequality constraints correspond to the elements in the columns from 12 to 24 in the DC-OPF matrix defined in the smart contract and the

rows from 12 to 24 corresponding to the partial Lagrange derivative with respect to the multipliers for inequality constraints.

Algorithm 3.26 OPF Solver Smart contract: Update DC-OPF Matrix Coefficients (Helper Functions)

```

1: Fixed[][]: OPFmatrixCoeff
2: /* The following helper function used to increment a twelve bits binary number,
   representing all the inequality constraints possibilities, either considered as equality
   constraint or not considered */
3: Procedure Increment(Uint sum[])
4:   Uint j ← 1
5:   Uint carry ← 0
6:   sum[0] ← (sum[0]+1)% 2
7:   carry ← (sum[0]+1)/2
8:   While carry !=0 || j==11 Do
9:     sum[j] ← (sum[j]+carry)% 2
10:    carry ← (sum[j]+carry)/2
11:    j ← j + 1
12:   End While
13: End Procedure
14: /* The following helper function used to delete a row from a matrix given its index,
   it serves to delete rows corresponding to inequality constraints multipliers that are
   not considered */
15: Procedure DeleteRow(Uint index, Fixed[][] Matrix)
16:   Uint i ← 0
17:   Uint j ← 0
18:   While i ≤ Matrix.length() Do
19:     i ← i + 1
20:     While j ≤ Matrix[0].length() Do
21:       Matrix[i][j] ← Matrix[i+1][j]
22:       j ← j + 1
23:     End While
24:     Matrix.pop()
25:   End While
26: End Procedure

```

The *solve()* function, called at each iteration of the loop in the main function, solves the system of linear equations passed to it as a matrix, using an auxiliary function

that row-reduces the passed matrix. This function is described in Algorithm 3.27, while the *solve()* function is described in Algorithm 3.28. After the system is solved, an updated solution corresponding to a given total load power demand is stored in the OPF smart contract and is thus available in the public ledger of the blockchain. The solution is linked in a structure to the corresponding date of the load. When a smart meter associated with a particular generator on the grid requests an energy token from the smart contract assigned to reward prosumers, the smart contract must first verify that the generation matches the particular OPF solution by invoking the OPF Solver smart contract. If the generation does not match the most recent OPF solution, the corresponding token reward will not be granted. In such a situation, the respective prosumer or generator owner would see their energy contribution wasted. Prosumers and microgrid owners are thus forced to comply with the OPF solution without the need for a central authority or deployed hardware. This is illustrated in Fig. 3.25.

Algorithm 3.27 OPF Solver Smart contract: Solve a system of linear equations using row-reduction algorithm.

```
1: Procedure (Fixed[[]] Coeff)
2:   Uint N ← B.length
3:   Uint i ← 0
4:   Uint j ← 0
5:   Fixed[N][N-1] A /* Variables coefficient matrix*/
6:   Fixed[N] B /* Constants vector*/
7:   While i ≤ N Do
8:     B[i] ← Coeff[i][N]
9:     While j ≤ N-1 Do
10:      A[i][j] ← Coeff[i][j]
11:      j ← j + 1
12:    End While
13:    i ← i + 1
14:  End While
15:  i ← 0
16:  While i ≤ N Do
    /* Find pivot */
17:    Uint max ← i
18:    Uint k ← i+1
19:    While i ≤ N Do
20:      If abs(A[k][i]) ≤ A[max][i]
21:        max = k
22:      End If
    /* swap row in matrix */
23:    A[i] ← A[max]
24:    A[max] ← temp
    /* swap corresponding values in constants matrix B */
25:    Fixed t ← B[i]
26:    B[i] ← A[max]
27:    B[max] ← temp
    /* pivot within A and B */
28:    j ← i+1
```

```

29: While  $j \leq N$  Do
30:   Fixed factor  $\leftarrow A[j][i]/A[i][i]$ 
31:    $B[j] \leftarrow B[j] - (\text{Factor} * B[i])$ 
32:    $n \leftarrow i$ 
33:   While  $n \leq N$  Do
34:      $A[j][n] \leftarrow A[j][n] - (\text{Factor} * A[i][n])$ 
35:      $n \leftarrow n+1$ 
36:   End While
37:    $j \leftarrow j+1$ 
38: End While
39:    $k \leftarrow k + 1$ 
40: End While
41:  $i \leftarrow i + 1$ 
42: End While
43: End Procedure

```

Algorithm 3.28 OPF Solver Smart contract: Solve row reduced DC-OPF system of linear equations.

```

1: Structure: Solution
2: Fixed: VoltageMagnitude
3: Fixed: VoltagePhaseAngle
4: Fixed: TimeStamp
5: End Structure
6: Address[3]: GenerationBuses
7: Mapping: Address $\rightarrow$  Solution: DC_OPF_Solution
8: Procedure Solve(Fixed[[ $\square$ ] A, Fixed[ $\square$ ] B) Returns (Bool)
9:   Uint N  $\leftarrow$  B.length
10:  Bool SystemSolved  $\leftarrow$  False
11:  Fixed [N] Solution
12:   $i \leftarrow N-1$ 
13:  While  $i \geq N$ 
14:    Fixedsolution  $\leftarrow$  0
15:     $j \leftarrow i+1$ 
16:    While  $j \leq N$ 
17:      sum  $\leftarrow$  sum + ( $A[i][j] * \text{Solution}[j]$ )
18:       $j \leftarrow i+1$ 
19:    End While
20:    solution[i]  $\leftarrow$  ( $B[i]-\text{sum}$ )/ $A[i][i]$ 
21:     $i \leftarrow i-1$ 
22:  End While

```

```

23: If (In solution, all Lagrange multipliers are positive and all constraints
    are satisfied) Then
24:   Solution S
25:   SystemSolved  $\leftarrow$  True
26:    $i \leftarrow 0$ 
27:   While  $i \leq 3$ 
28:     S.VoltageMagnitude  $\leftarrow$  solution[i]
29:     S.VoltagePhaseAngle  $\leftarrow$  solution[i+3]
30:     S.TimeStamp  $\leftarrow$  CurrentDate
31:      $i \leftarrow i+1$ 
32:   End While
33: End If
34: Return SystemSolved
35: End Procedure

```

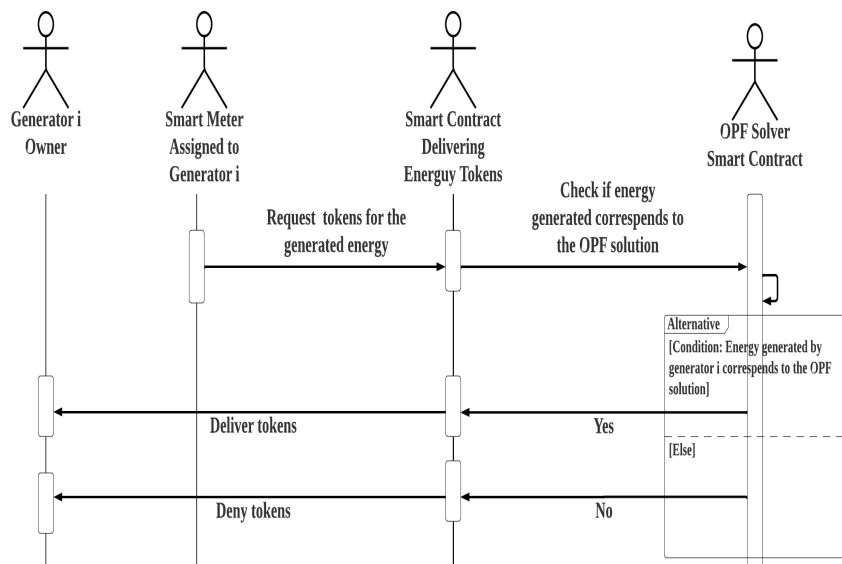


Figure 3.25: Flowchart illustrating how parties in the smart grid that do not comply with the OPF solution are not rewarded for their energy contribution.

After the system is solved, an updated solution corresponding to a given total load power demand is stored in the OPF smart contract and is thus available in the public ledger of the blockchain. The solution is linked in a structure to the corresponding date

of the load. When a smart meter associated with a particular generator on the grid requests an energy token from the supplying smart contract, the smart contract must first verify that the generation matches the particular OPF solution by invoking the OPF Solver smart contract. If the generation does not match the most recent OPF solution, the corresponding token reward will not be granted. In such a situation, the respective prosumer or generator owner would see their energy contribution wasted. Prosumers and microgrid owners are thus forced to comply with the OPF solution without the need for a central authority or deployed hardware. This is illustrated in Fig. 3.25. The used example was solved for a daily (24 hour) load distribution using day-ahead market load forecasting presented in (Shahidehpour et al., 2002) and depicted in Tab 3.7.

Table 3.7
Hourly load power demand for 24 hours.

Hr	P_D (MW)			Hr	P_D (MW)		
	D_1	$D_2&D_3$	Total		D_1	$D_2&D_3$	Total
1	132.66	44.22	221.1	13	153.06	51.02	255.1
2	122.4	40.8	204.0	14	149.64	49.88	249.4
3	115.62	38.54	192.7	15	147.96	49.32	246.6
4	112.2	37.4	187.0	16	147.96	49.32	246.6
5	108.84	36.28	181.4	17	154.74	51.58	257.9
6	110.52	36.84	184.2	18	170.04	56.68	283.4
7	112.2	37.4	187.0	19	163.26	54.42	272.1
8	119.04	39.68	198.4	20	161.52	53.84	269.2
9	136.02	45.34	226.7	21	159.84	53.28	266.4
10	149.64	49.88	249.4	22	165.42	52.14	260.7
11	153.06	51.02	255.1	23	147.96	49.32	246.6
12	154.74	51.58	257.9	24	137.76	45.92	229.6

The DC-OPF solution for the formulated problem, for each hour of the day is

presented in Tab 3.8. It is to note that only in Tab 3.8 the (*) operator is use to denote an optimal entity and not the conjugate of complex one.

Table 3.8
3-bus example, DC-OPF solution.

Hour	P_{G1}^*	P_{G2}^*	P_{G3}^*	δ_2^*	δ_3^*
01	200.0	16.1	5.0	-0.0799	-0.1095
02	189.0	10.0	5.0	-0.0808	-0.1048
03	177.7	10.0	5.0	-0.0752	-0.0979
04	172.0	10.0	5.0	-0.0724	-0.0944
05	166.4	10.0	5.0	-0.0696	-0.0910
06	169.2	10.0	5.0	-0.0710	-0.0927
07	172.0	10.0	5.0	-0.0724	-0.0944
08	183.4	10.0	5.0	-0.0780	-0.1014
09	200.0	21.7	5.0	-0.0741	-0.1077
10	200.0	44.4	5.0	-0.0506	-0.1002
11	200.0	50.1	5.0	-0.0447	-0.0983
12	200.0	52.9	5.0	-0.0418	-0.0974
13	200.0	50.1	5.0	-0.0447	-0.0983
14	200.0	44.4	5.0	-0.0506	-0.1002
15	200.0	41.6	5.0	-0.0535	-0.1011
16	200.0	41.6	5.0	-0.0535	-0.1011
17	200.0	52.9	5.0	-0.0418	-0.0974
18	200.0	78.4	5.0	-0.0154	-0.0890
19	200.0	67.1	5.0	-0.0271	-0.0927
20	200.0	64.2	5.0	-0.0301	-0.0937
21	200.0	61.4	5.0	-0.0330	-0.0946
22	200.0	55.7	5.0	-0.0389	-0.0965
23	200.0	41.6	5.0	-0.0535	-0.1011
24	200.0	24.6	5.0	-0.0711	-0.1067

3.2.3.2 Enhanced decentralized OPF solving, generalized for any problem formulated

As can be seen from the evaluation performed in the previous section, which showed very high execution costs for a simplified version of the OPF problem for a simple network, averaging about \$9 per hour, such an approach is not conceivable for a much more complex problem such as the AC-OPF problem, which is non-convex and non-linear, and even less so for an even more complex problem that considers renewable energy sources with unpredictable generation capacity. What is proposed is that optimal solution is proposed by prover nodes in a decentralized and permission-less manner. Interested provers, can apply by depositing a defined amount of money into the smart contract, they are then notified by smart contract through an event when the buses loads are updated by the assigned unit. Provers solve the OPF problem off-chain, then send the solution they calculated which represents the optimal combination of net power generated at each bus. Initially, the provers need to send their calculated solution in an encrypted form. They can only reveal their proposed solution after all eligible candidates have successfully committed their encrypted solution. After all proposed solution are revealed and verified if they do correspond to the committed sent encryption by the smart contract, the smart contract then verifies if each proposed solution satisfies the problem constraints, then only the solutions giving the cheapest generation cost are considered. To clarify more, we assume M provers have successfully staked the defined

required amount A to be eligible to propose a solution to the OPF problem. If $N, N \leq M$ provers form the M candidates have proposed identical solutions satisfying the problem constraints and which is less expensive in term of generation cost than the solutions proposed by the rest of the $M - N$ provers, then the N provers will gain $\frac{(M-N)*A}{N}$ as reward, where the rest of $M - N$ provers loses their staked money, similar to a card betting game as illustrated in Fig. 3.26. In case the M total number of provers all propose an identical solution, it is adopted to control the generators if satisfying the problem constraints, in such a scenario all the provers gain a symbolic reward to encourage them to continue solving the OPF problem.

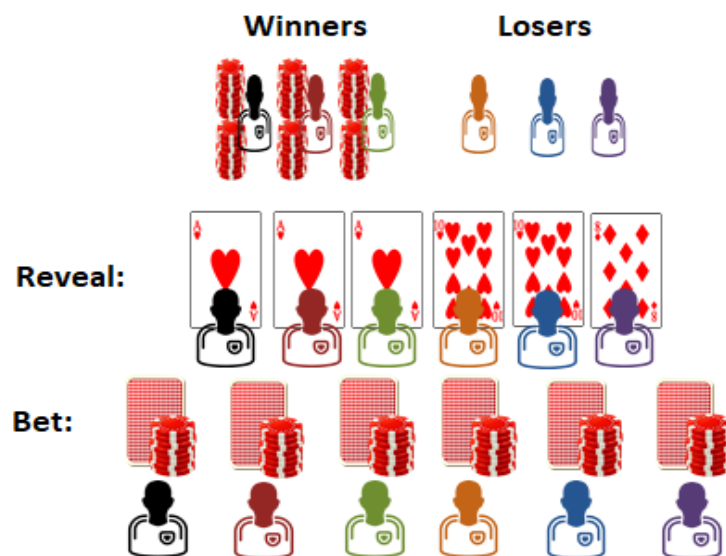


Figure 3.26: Proposed OPF solution scheme explained using a card betting game.

Since provers proposed solution are only revealed after they are all committed to the blockchain, this give rise to a Nash game scenario. To illustrate the game scenario

applicable to the proposed platform, we assume two prover candidates proposing an OPF solution. The illustration with only two candidates can be extrapolated to n provers. In this game scenario, the two provers conspire to both propose identical non-optimal solution and gain the symbolic reward so that there is no loser. The two provers have two game strategies they can play. Either comply to the conspiracy and commit the agreed upon solution, or betray the second player and propose an optimal solution. We define the symbolic gain if both payers propose identical solution to be 5, whereas the staked amount by each player to be 50. The gain for each player according to his played strategy is illustrated in Tab 3.9.

Table 3.9
The gain for each player according to his played strategy.

Player1 \ Player2	Propose optimal solution (Betray)	Collude
Propose optimal solution (Betray)	(+5,+5)	(+50,-50)
Collude	(-50,+50)	(+5,+5)

From Tab 3.9, it can be seen the total gain for each player when proposing an optimal solution is +55, whereas it is -45 if proposing a non-optimal one, which makes the proposed scheme immune against collusion. Such a system is suitable for any defined and formulated OPF problem where the OPF is calculated by impartial third parties who solve the OPF problem for remunerative purposes. Provers invest in developing the solution algorithms and performing the computations to find the most optimal solution and win the jackpot, similar to how miners invest in hashing power to be the first to solve

the mining puzzle and win the mining reward. Since the OPF is computed off-chain and only the proposed solutions are compared and verified on-chain, this proposed system implies that much less heavy computation is performed on-chain without compromising the decentralisation of the system.

3.2.3.2.1 Implementation for a 14-bus AC-OPF problem

In the AC model, transmission lines are represented by the π model. This model is the most commonly used representation of transmission lines in power networks and is adopted by most AC-OPF simulation platforms, such as Matlab. The π model for the transmission line between two nodes i and j is shown in Fig. 3.27. In the π model, a

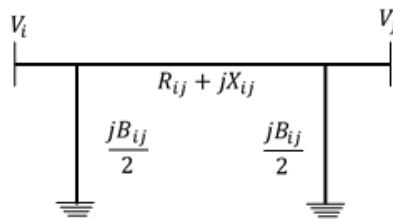


Figure 3.27: Transmission line π model.

transmission line consists of a series impedance $R_{ij} + jX_{ij}$ connected at the nodes i and j , and two equal shunt susceptances $y_{sh_i} = y_{sh_j} = j\frac{B_{ij}}{2}$. The line is mostly defined by its series admittance $y_{ij} = \frac{1}{R_{ij} + jX_{ij}}$ rather than its series impedance. Accordingly, as in the DC model, the OPF objective function aims to minimize the generation cost function $F(P_G)$, where $F(P_G) = \sum_{i=0}^n C(P_G)$. Thus, the objective function of the model AC-OPF

is $MinF(P_G)$ Subjected to constraints defined in Eq. 3.51 to Eq. 3.56

$$\mathbf{AC\ flow} \quad \bar{S}_G - \bar{S}_L = diag(\bar{V})\bar{Y}_{bus}^*\bar{V}^* \quad (3.51)$$

$$|\bar{Y}_{line,i \rightarrow j}\bar{V}| \leq I_{line,max},$$

$$\mathbf{Line\ Current} \quad |\bar{Y}_{line,i \rightarrow j}\bar{V}| \leq I_{line,max} \quad (3.52)$$

$$|\bar{Y}_{line,j \rightarrow i}\bar{V}| \leq I_{line,max},$$

$$\mathbf{Gen.\ Active\ Power} \quad 0 \leq P_G - P_{G,max}, \quad (3.53)$$

$$\mathbf{Gen.\ Reactive\ Power} \quad -Q_{G,max} \leq Q_G - Q_{G,max}, \quad (3.54)$$

$$\mathbf{Voltage\ Magnitude} \quad V_{min} \leq V \leq V_{max}, \quad (3.55)$$

$$\mathbf{Voltage\ Angle} \quad \delta_{min} \leq \delta \leq \delta_{max}, \quad (3.56)$$

The (*) used operator denotes the conjugate of a complex entity, the bar (–) used above an entity is to express that it is a complex one. *min,max* indices are used to denote the maximum allowed value for an entity. In a network of N buses and M transmission lines, V is a vector of size N of voltages at each bus of the network. $diag()$ operator a vector to diagonal matrix converter. In our case $diag(V)$ returns an $N \times N$ diagonal matrix, where all non-diagonal elements starting from left-upper to right-

bottom are zeros and the diagonal elements are successively elements of the vector V . I_{line} is a vector of size M of the currents at each transmission of the network. Y_{line} is an $M \times N$ matrix corresponding to the line admittance matrix of the network, it links the bus voltages vector V to the lines current flow vector I_{line} , where $I_{line} = VY_{line}$. Because, the current flowing from a node i to a node j is different from the one flowing from node j to node i , there must be two different line admittance matrices Y_{line_1} , and Y_{line_2} . accordingly there are two I_{line} vectors, $I_{line_1} = \{I_{1 \rightarrow 2}, I_{2 \rightarrow 3}, \dots, I_{M \rightarrow M}\}$, $I_{line_2} = \{I_{1 \leftarrow 2}, I_{2 \leftarrow 3}, \dots, I_{M \leftarrow M}\}$, where $I_{line_1} = VY_{line_1}$ and $I_{line_2} = VY_{line_2}$. A row k in the line admittance matrix corresponds to the transmission line with current I_k , which is the k^{th} element in the linked line current flow vector I_{line} . given that row k corresponds to line $i \rightarrow j$, the i^{th} element in the row $Y_{line_{ki}} = y_{sh_i} + y_{ij}$, whereas the j^{th} element in the row $Y_{line_{kj}} = -y_{ij}$, the rest of the row elements are all null. Y_{bus} is an $N \times N$ matrix, representing the bus admittance matrix. Diagonal element $Y_{bus_{ii}}$ of the matrix corresponding to a row i , are the sum of the shunt susceptance at bus i and the sum of all series admittance of all the lines connected to bus i . For non diagonal elements $Y_{bus_{ij}, i \neq j}$, they are either null if there is no connecting line between bus i and bus j , otherwise $Y_{bus_{ij}} = -y_{ij}$. S represents a vector of size N of the total net apparent power at each bus of the network. given $Y_{line, i \rightarrow j}$ matrix, $S = S_G - S_L$, where S_G and S_L are both vectors of size N . S_G is the total net generation power vector and S_L is the net load demand power vector. S has two components. A real component representing

the active power flow $P_G - P_L = \text{Re}(S_G - S_L)$, and an imaginary one representing the reactive power flow $Q_G - Q_L = \text{Im}(S_G - S_L)$. P_G, Q_G, P_L, Q_L , are the vectors of size N of the active and reactive power generation at each bus and the active and reactive load demand power at each bus as well respectively. Both active and reactive power at given node i are represented in Eq. 3.57, with respect to both bus active and reactive load demand power respectively.

$$\begin{cases} P_{Gi} = \sum_{j=1}^N |V_i||V_j|(G_{ij}\cos(\theta_k - \theta_j) + B_{kj}\sin(\theta_i - \theta_j)) + P_{Li}, \\ Q_{Gi} = \sum_{j=1}^N |V_i||V_j|(G_{ij}\sin(\theta_k - \theta_j) - B_{kj}\cos(\theta_i - \theta_j)) + Q_{Li} \end{cases} \quad (3.57)$$

G_{ij} and B_{ij} represent the real and imaginary parts of Y_{ij} in the admittance matrix Y that should be preset in the smart contract. Therefore, for each bus, provers must specify the voltage magnitude and phase at the bus, and the voltage magnitudes and phases at each bus connected to that bus. P_{Gi} and Q_{Gi} are accordingly calculated within the Smart Contract using sin and cos functions available in the Smart Contract at (Karapetsas, 2020). Thus provers should only upload for each bus the respective bus voltages magnitudes and phase angles corresponding to the bus both power active and reactive generation in Eq. 3.57. The pseudocode in algorithm 3.29 illustrates how interested prover candidates apply to solve the OPF problem by depositing the required amount. As demonstrated applications are no more accepted once the preset number of maximum applications is reached.

Algorithm 3.29 OPF Smart Contract: Apply

```
1: Uint: SmartContractBalance
2: Uint: DepositRequiredAmount
3: Uint: TotalNumberOfPermittedProvers
4: Uint: NumberOfProvers  $\leftarrow$  0
5: Address:[] OPFproverCandidates
6: Mapping: Address  $\rightarrow$  Uint: ProversBalance
7: Procedure Payable Apply()
8: Uint: NumberOfProvers  $\leftarrow$  NumberOfProvers + 1
9: If msg.value  $\geq$  DepositRequiredAmount
10:   Revert()
11: Else If NumberOfProvers  $\leq$  TotalNumberOfPermittedProvers
12:   ProverBlance[msg.sender]  $\leftarrow$  ProverBlance[msg.sender] + msg.value
13:   OPFproverCandidates.Push(msg.value)
14:   SmartContractBalance  $\leftarrow$  SmartContractBalance+msg.value
15: Else
16:   Revert()
17: End If
18: End Procedure
```

Before provers start uploading their proposed solution to the smart contract, the network buses' load demand is updated by the assigned unit on an hourly basis, the smart contract date is also updated on the same occasion, similar to the model described in the previous section. This is illustrated in pseudocode in algorithm 3.30. It is to note that in order not make the pseudocodes excessively long, only active generation and demand power are considered in the presented pseudocodes. However, the system constraints related to buses reactive power are considered during the implementation used to evaluate the execution cost of the model.

Algorithm 3.30 OPF Smart Contract: UpdateBusesLoadPowerDemand

```
1: Address: LoadMonitoringUnit
2: Fixed[14]: BusesLoadPowerDemand
3: Fixed[14][14]: BusAdmittanceMatrixRealComponents
4: Fixed[14][14]: BusAdmittanceMatrixImagComponents
5: Procedure UpdateBusesLoadPowerDemand(Fixed[14] BusesLoadP)
6: If msg.sender != LoadMonitoringUnit Then
7:   Revert()
8: Else
9:   For all L in BusesLoadP Do
10:
11:     BusesLoadPowerDemand.Push(L)
12:   End For
13:   UpdateCurrentDate()
14:   Emit Event: NotifyProverCandidates()
15: End If
16: End Procedure
```

As explained earlier, the provers first upload an encrypted solution, which they cannot reveal until all provers have uploaded an encrypted solution. Once this is done, the smart contract sends out an event to notify the provers to reveal their solution, which is then checked to see if it matches the corresponding encrypted message sent earlier. This is explained in pseudocode in Algorithm After each prover discloses their proposed solution, it is checked to see if it satisfies the problem constraints. If it does, the corresponding generation cost is calculated and assigned to the respective prover. This is described in the pseudocode of the algorithm 3.31.

Algorithm 3.31 OPF Smart Contract: Send encrypted solution/ Reveal solution.

```
1: Structure: BusGeneratedPowerFlowEquationParameters
2: Fixed[] ConnectedBusesPhaseAngles
3: Fixed[] ConnectedBusesVoltageMagnitude
4: Fixed BusPhaseAngle
5: Fixed BusVoltageMagnitude
6: End Structure:
7: Uint EncryptedSolutionsCommitted  $\leftarrow$  0
8: Uint SolutionsRevealed  $\leftarrow$  0
9: Mapping: Address  $\rightarrow$  BusGeneratedPowerFlowEquationParameters[]:
   ProverProposedSolution
10: Fixed[14][14]: BusAdmittanceMatrixImagComponents
11: Mapping: Address  $\rightarrow$  Bytes32: ProverEncryptedSolution
12: Procedure SendEncryptedSolution(Bytes32 EncryptedSolution)
13: If ProversBalance[msg.sender]  $\lesseqgtr$  DepositRequiredAmount Then
14:   Revert()
15: Else If EncryptedSolutionCommitted = TotalNumberOfPermittedProvers
16:   Emit Event: NotifyProversToRevealSolution
17: Else
18:   ProverEncryptedSolution(msg.sender)  $\leftarrow$  EncryptedSolution
19:   EncryptedSolutionsCommitted  $\leftarrow$  EncryptedSolutionsCommitted + 1
20: End If
21: End Procedure
22: Procedure Reveal-OPF-Solution(BusGeneratedPowerFlowEquationParameters[]
   Eq)
23: Uint a  $\leftarrow$  EncryptedSolutionsCommitted
24: Uint b  $\leftarrow$  TotalNumberOfPermittedProvers
25: Uint c  $\leftarrow$  ProversBalance[msg.sender]
26: Uint d  $\leftarrow$  DepositRequiredAmount
27: If a  $\lesseqgtr$  b  $\parallel$  c  $\lesseqgtr$  d Then
28:   Revert()
29: Else If SolutionsRevealed  $\geq$  b
30:   ProversSolutionsCost()
31:   RewardOPFsolvers()
32: Else
33:   Bytes32 c  $\leftarrow$  ProverEncryptedSolution(msg.sender)
34:   Bytes32 d  $\leftarrow$  keccak256(abi.encode(Eq))
35:   If c  $\neq$  d Then
36:     Revert()
37:   Else
```

```
38:   ProverProposedSolution(msg.sender) ← Eq
39:   SolutionsRevealed ← SolutionsRevealed + 1
40:   End If
41: End If
42: End Procedure
```

After each prover discloses their proposed solution, it is checked to see if it satisfies the problem constraints. If it does, the corresponding generation cost is calculated and assigned to the respective prover. This is described in the pseudocode of the algorithm 3.32.

Algorithm 3.32 OPF Smart Contract: ProversSolutionCost.

```
1: Mapping: Address → Bool: ProverSolutionIsOptimal
2: Mapping: Address → Fixed: ProposedSolutionCost
3: Procedure ProverSolutionsCost()
4:   Fixed: GenerationCost ← 0
5:   For Each P in OPFproverCandidates Do
6:     Uint: i ← 0
7:     Uint: j ← 0
8:     Fixed  $\theta_i$  ← ProverProposedSolution[i].BusPhaseAngle
9:     Fixed  $V_i$  ← ProverProposedSolution[i].BusVoltageMagnitude
10:    Fixed[]  $\theta_j$  ← ProverProposedSolution[i].ConnectedBusesPhaseAngles
11:    Fixed[]  $V_j$  ← ProverProposedSolution[i].ConnectedBusesVoltageMagnitudes
12:    Fixed  $Pg_i$  ← 0
13:    Fixed  $G_{ij}$  ← BusAdmittancesMatrixReal[i][j]
14:    Fixed  $B_{ij}$  ← BusAdmittancesMatrixImag[i][j]
15:    Fixed  $Pd_i$  ← BusesLoadPowerDemand[i]
16:    While i ≤ 14 Do
17:      While j ≤ 14 Do
18:         $Pg_i$  ←  $Pg_i + (V_j * V_j(G_{ij} * \cos(\theta_i - \theta_j) - B_{ij}\sin(\theta_i - \theta_j)) + Pd_i)$ 
19:        GenerationCost ← GenerationCost + GenerationCostFunction( $Pg_i$ )
20:        j ← j + 1
21:      End While
22:      i ← i + 1
23:    End While
24:    ProposedSolutionCost(P) ← GenerationCost
25:  End For
26: End Procedure
```

Finally, all valid solutions are compared based on their calculated generation costs, and then the prover that proposed the optimal solution is rewarded, as described in the pseudocode in the algorithm 3.32. The entity relationship diagram for the proposed smart contract-based solution is shown in the following Fig. 3.28.

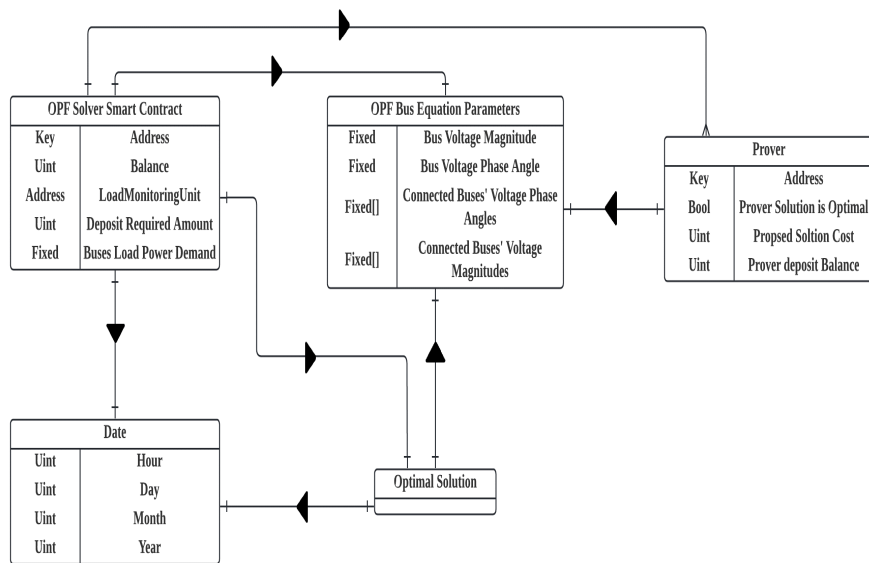


Figure 3.28: Entity-relationship diagram for the enhanced OPF solution model.

3.3 Experimental setup

All smart contracts involved in the proposed platform were implemented on the Remix IDE development tool and developed using the Solidity smart contract programming language. The Remix IDE platform was also the place where the functioning of the smart contracts was tested and the execution cost of the smart functions was evaluated. To evaluate the time response of the various smart contract functions, a private Ethereum blockchain was placed on a machine with 4 cores CPU: Intel(R) Core(TM) i5-9300H CPU @ 2.40 GHz and a GPU: NVIDIA GeForce GTX 1650 with Max-Q design. The smart contracts were implemented on the private blockchain and interaction with them was done via Node.js, a server-side JavaScript framework, and Web3.js, a JavaScript

framework that allows Node.js applications to interact with a local or remote Ethereum node via HTTP, IPC, or WebSocket. To simulate different nodes invoking the smart contract simultaneously, different Ethereum nodes were created on the private blockchain. All created nodes run simultaneously on the same machine on different threads using the worker, *hreadslibraryformulti – threadedprogrammingonNode.js*.

3.3.1 Proposed pay per use blockchain P2P energy trading platform

The proposed pay-per-use blockchain P2p energy trading platform consists of several components, which we will discuss separately

3.3.1.1 Energy onsumption Time Series Forecasting Using GRU

The implementation of the standard GRU prediction model is conducted under the Keras framework with Tensorflow2 as the back end, using Python language. The prediction of the electricity consumption in the next defined time steps should only use the historical electricity consumption data and the date-time as predicting features. Despite the availability of other datasets containing other metrics such as consumers' household temperature and humidity, we stick only to date and time as extracted features for prediction. The reason behind it is that the prediction is meant to be performed by the electricity provider who is not supposed to have access to such private data of his clients. The predictions should solely be based on the electricity consumption data uploaded to the blockchain by the respective clients. During each new trading round,

the GRU model adds the previously collected electricity consumption data to its training set. The model is evaluated by its accuracy. As a consequence, the more rounds go by, the more data available to train the model and thus the more precise the prediction should be. To assess that, we defined the trading round to be 100 days, i.e., we would be using 2400 rows to forecast the next 2400 rows. Then the number of rows $R(n)$ that is used to predict the upcoming 2400 rows in the n -th round is defined according to the expression in Eq. 3.58:

$$R(n) = 2400.n, n > 0 \quad (3.58)$$

The performance of the model is assessed in terms of the Mean Average Percentage Error (MAPE) given in Eq. 3.59, and the results are depicted in Fig ??.

$$MAPE = \frac{100}{n} \sum_{i=1}^n \left| \frac{Y_i - P_i}{Y_i} \right| \quad (3.59)$$

where Y_i is the i -th real electricity consumption reading and P_i is the i -th reading of the predicted electricity consumption, in a test set of size n .

3.3.1.2 Definition of ToU Ranges Using On-Chain Clustering

The K-mean smart contract was evaluated only in terms of execution cost, which is the main problem in implementing the machine learning protocol on the blockchain. In (Niya et al., 2018), a decentralized P2P buying and renting application based on

blockchain was presented. This was used to make a comparison with the implemented K-mean clustering smart contract, both in provisioning and invoking gas consumption, to compare it with an established application and evaluate its feasibility.

3.3.1.3 Modified PoW consensus, adapted for the platform

To define the threshold time, a PoW algorithm is developed in Java and run on a four (4) cores CPU: Intel(R) Core(TM) i5-9300H CPU @ 2.40 GHz and a GPU: NVIDIA GeForce GTX 1650 with Max-Q Design. The mining function is called in a loop for several iterations. In each iteration, the mining time is pushed in an array then the median of the set is deduced. $M = M_0, \dots, M_k$ is the set of mining times where the mining function has been called for k times with a fixed difficulty and a random string of fixed size to be hashed. $T_{\text{threshold}}$ represents the minimum threshold time for accepting a new block. It is defined to be the median of the set M . $T_{\text{threshold}}$ is computed for several sets M of different sizes. $T_{\text{threshold}}$ is picked when convergence is met, as illustrated in Tab 3.10.

Table 3.10
 $T_{\text{threshold}}$ for different sizes of mining sets.

Set Size	Median of the Set [ms]
50	69
100	49
150	39
200	39

According to Tab 3.10, the deduced $T_{\text{threshold}}$ is 39 ms. To test the proposed

mining protocol and assess its performance, it is executed on 10 different threads running concurrently on the same machine, as depicted in Fig 3.29. Each thread represents a miner. Fraudulent miners' nodes are represented by threads running mining protocol with less difficulty, to act as nodes having more hashing power. In the testing experiment, the certified miners are represented by threads mining with difficulty 5, whereas fraudulent miners are represented by threads mining with difficulty 4. The experiment consists of running the threads for 100 rounds corresponding to 100 blocks. The number of fraudulent miner nodes goes from a single node to 6 nodes representing more than 50% of the network. Threads with the shortest execution time in a round represent nodes successfully mining the new block for that particular round. If thread execution time is less than the accepted minimum threshold, its execution time is set to be $T_{\text{threshold}}$.

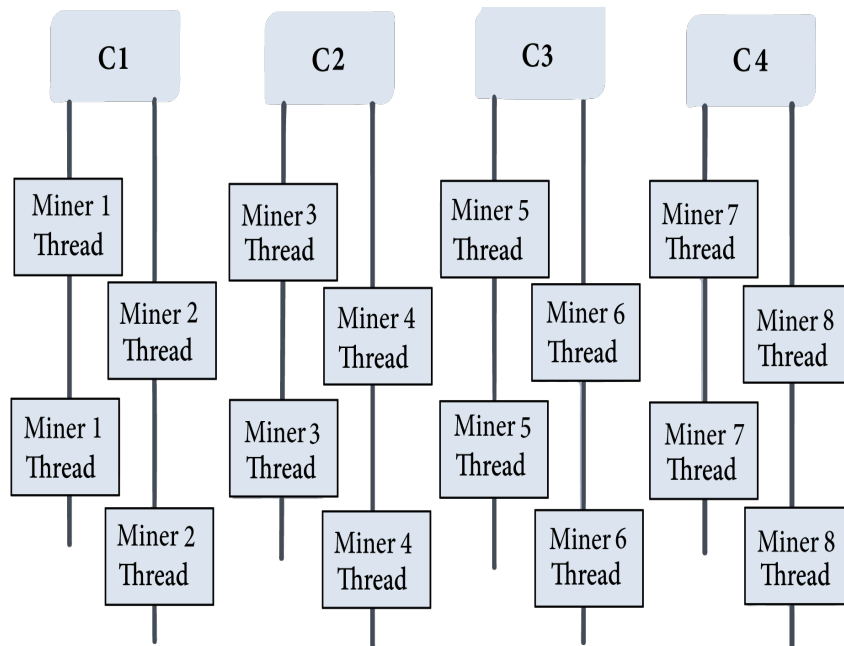


Figure 3.29: Threads running concurrently on CPU cores, simulating miners competing to add a new bloc.

Miners for each round are defined according to the pseudocode of Algorithm 3.33.

The obtained results showed that in each mining round there are always at least two successful miners, which corresponds to a fork situation. This is expected, since all miners are supposed to have the same hashing power. Results in Fig 4.2 describe miner involvement in forks given a different number of fraudulent nodes.

Algorithm 3.33 Miner threads.

```
1: for  $i \leftarrow 0$  to  $NumberOf(threads)$  do
2:   if  $thread.ExecutionTime \leq Threshold$  then
3:      $t \leftarrow Ttreshold$ 
4:   else
5:      $t \leftarrow thread.ExecutionTime$ 
6:   end if
7: end for
8:  $Min \leftarrow Math.min(array)$ 
9: for  $i \leftarrow 0$  to  $Size(array)$  do
10:  if  $array[i] \leq Min$  then
11:     $MinerIndexes.push(i)$ 
12:  end if
13: end for
14: return  $MinerIndexes$ 
```

Fraudulent nodes are threads given less difficulty than the rest, to emulate nodes that have higher hashing power than allowed.

3.3.1.4 Pay per use P2P energy trading platform evaluation

To evaluate the performance of the proposed platform, a private Ethereum blockchain was set up on a machine with 4 cores CPU: Intel(R) Core(TM) i5-9300H CPU @ 2.40 GHz and a GPU: NVIDIA GeForce GTX 1650 with Max-Q design. It is worth mentioning that the purpose of the experiment is not to verify the platform's operation, as this has already been done during the development phase on Remix IDE. We decided to focus on evaluating the performance as the number of subscribed consumers and the rate of sent consumption data increases. The experiment was conducted for 3 cases with corresponding 10, 20 and 40 subscribed nodes. For each case, the sending rate was in-

creased in a step of 10 transactions per second, starting from 10 transactions per second up to 100 transactions per second. For each of these cases, blockchain nodes are created, running on different ports and connected to each other as peers. The created nodes collectively represent subscribing consumers uploading their electricity consumption data to the provider's smart contract. The metric evaluated is the transaction latency, i.e., the time in seconds between the issuance of the transaction by the node invoking the smart contract and its transmission to the general ledger. The execution cost of the SMART contracts' functions was also evaluated. Gas consumption for deploying the proposed smart contracts was benchmarked against the one for deploying Wepower smart contract, updating prosumer energy balance gas consumption was also benchmarked against gas consumption for transferring Wepower tokens. Wepower smart contract solidity source code can be found in [[//github.com/WePowerNetwork/wepower-contracts](https://github.com/WePowerNetwork/wepower-contracts)], its respective execution costs were also assessed on Remix IDE on the same machine as well.

3.3.2 Decentralized PoL scheme

A private Ethereum blockchain was run on a machine with 4 cores CPU: Intel(R) Core (TM) i5-9300H CPU @ 2.40GHz and a GPU: NVIDIA GeForce GTX 1650 with Max-Q design to evaluate the performance of the proposed platform. It is worth noting that the goal of the experiment is not to verify the functionality of the platform, as this has already been done during the development phase with Remix IDE. However, we

decided to focus on evaluating performance as we increase the number of requests to blockchain nodes and their sending rate, as well as the gas cost of executing smart contract functions. Accordingly, blockchain nodes are created to run on different ports and are interconnected, adding other nodes as peers. Each created node, in its entirety, represents a target node that requests PoL with a set of subscribed provers that apply to be location verifiers for that target node request. Using a node-js application and the web3 library, each blockchain node is configured to invoke a set of transactions. The flowchart in Fig 4.2 shows the transaction sequence used, where transactions are not issued until the smart contract is notified of the corresponding event. The number of network nodes is specified in a genesis.json file, along with the Etash PoW mining difficulty. The selected difficulty was 0x04 in hexadecimal, which corresponds to an average mining time of 30 ms on the deployed machine. The metric evaluated is transaction latency, which is the time in seconds between the issuance of a transaction by the node invoking the smart contract and its delivery to the ledger.

3.3.3 Smart contract-based OPF system for energy demand response management

Performing smart contract transactions in an Ethereum network requires a gas fee to be paid for each transaction. Gas refers to an abstract unit referring to the amount of Ethers (ETH) that need to be paid in accordance to the computational effort required to execute certain operations on the Ethereum network, this is because Ethereum transaction

execution requires computational resource. Thus issuing a transaction requires a fee to be paid proportional to the transaction execution complexity. Another concept that is relevant to Ethereum transactions is the gas limit. When developing an Ethereum application that uses smart contracts, the cost of executing transactions is estimated using development tools such as Remix IDE and then a gas limit is set. Because smart contracts are deployed on the blockchain, they are immutable. Therefore, errors in smart contracts that have already been deployed cannot be fixed. Setting a gas limit prevents the smart contract from entering infinite loops that could be triggered by hackers who could exploit loopholes in the smart contract source code. Gas cost and gas limits in Ethereum applications is illustrated in the flowchart in Fig 3.30. The gas cost of the

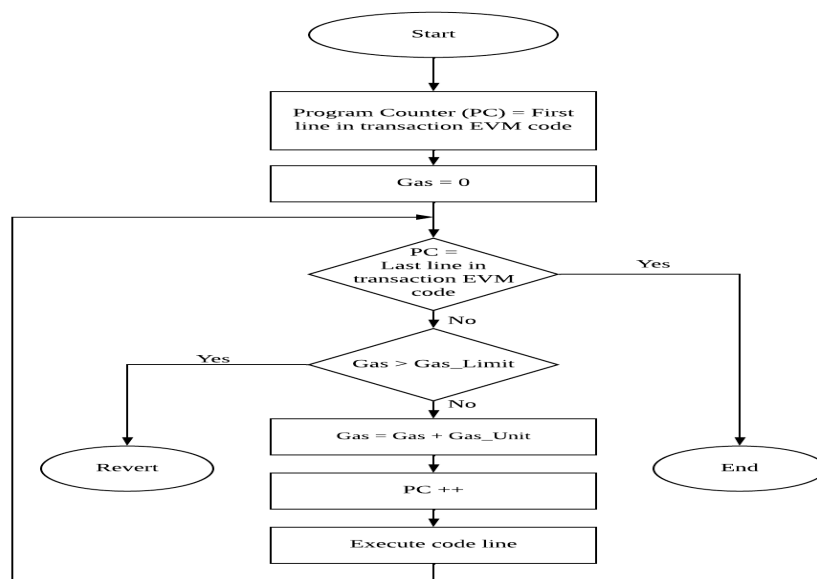


Figure 3.30: Gas consumption for Ethereum transaction execution.

on-chain DC-OPF solution using the proposed system was evaluated using Remix IDE

for the defined 3-bus problem. The gas cost for execution was evaluated according to the value of the gas unit on 15/05/2020, which was 27.95 Gw as in (Ycharts, 2020), and the value of the Ethereum unit in dollars for the same day (Ycharts, 2020). The hourly fluctuation of the gas price in dollars for the day is shown in Fig 3.31.

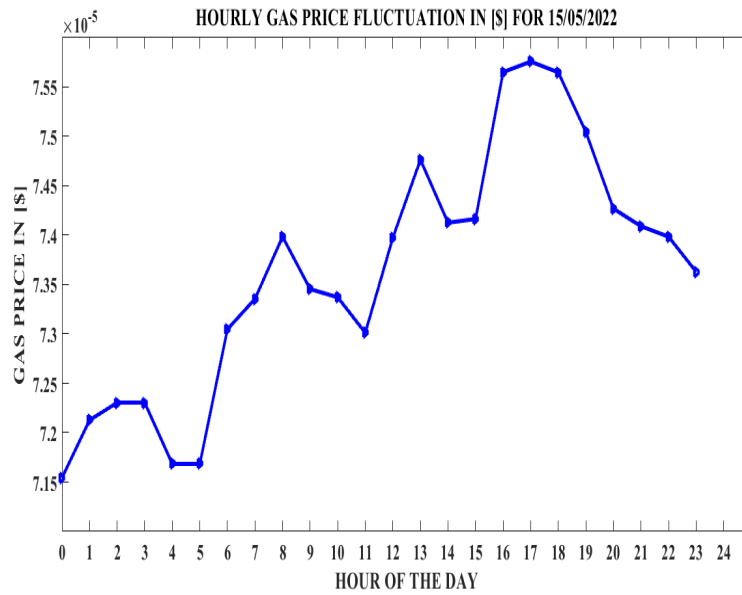


Figure 3.31: Hourly gas price fluctuation in US Dollars for 15/05/2022.

As for the enhanced proposed scheme, it was used to solve the AC-OPF problem for a 14-bus network. All network parameters can be found in (?). 3 evaluations were performed with 20, 50 and 100 provers and a single optimal solution was proposed.

CHAPTER FOUR

RESULTS AND ANALYSIS

4.1 Introduction

In order to properly conclude this study, the results of the experiments described in Chapter 3 must be analyzed to test our hypothesis for answering the research questions defined in the problem statement and to determine the extent to which we are or are not compromising other performance metrics in achieving our research goals.

4.2 Pay-per-use P2P energy trading platform

4.2.1 ToU ranges using GRU model predictions and K-Mean clustering

4.2.1.1 Energy consumption time series prediction with GRU performance

The performance of the GRU prediction model with respect to the evolution of MAPE over the course of the trading rounds and as the training set increases is shown in Fig 4.1

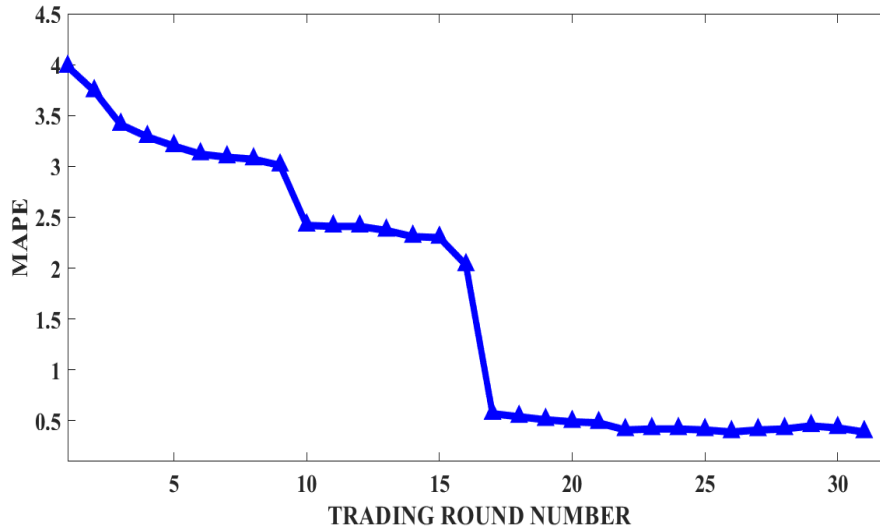


Figure 4.1: GRU prediction model performance evolution as the trading rounds go by.

4.2.1.2 Definition of ToU ranges with on-chain clustering execution cost

In (Niya et al., 2018), a decentralized P2P purchase and rental application based on blockchain was presented. This was used to make a comparison with the implemented K-mean clustering smart contract, both in provisioning and calling gas consumption.

The comparison is described in Tab 4.1

Table 4.1
Gas consumption.

	Deploy	Call contract function
K-mean smart contract	801911	1334160
Smart contract in (Niya et al., 2018)	760,550	82,721

4.2.2 Modified PoW adapted for the P2P energy trading platform

The results in Fig 4.2 describe the participation of miners in forks at different number of fraudulent nodes.

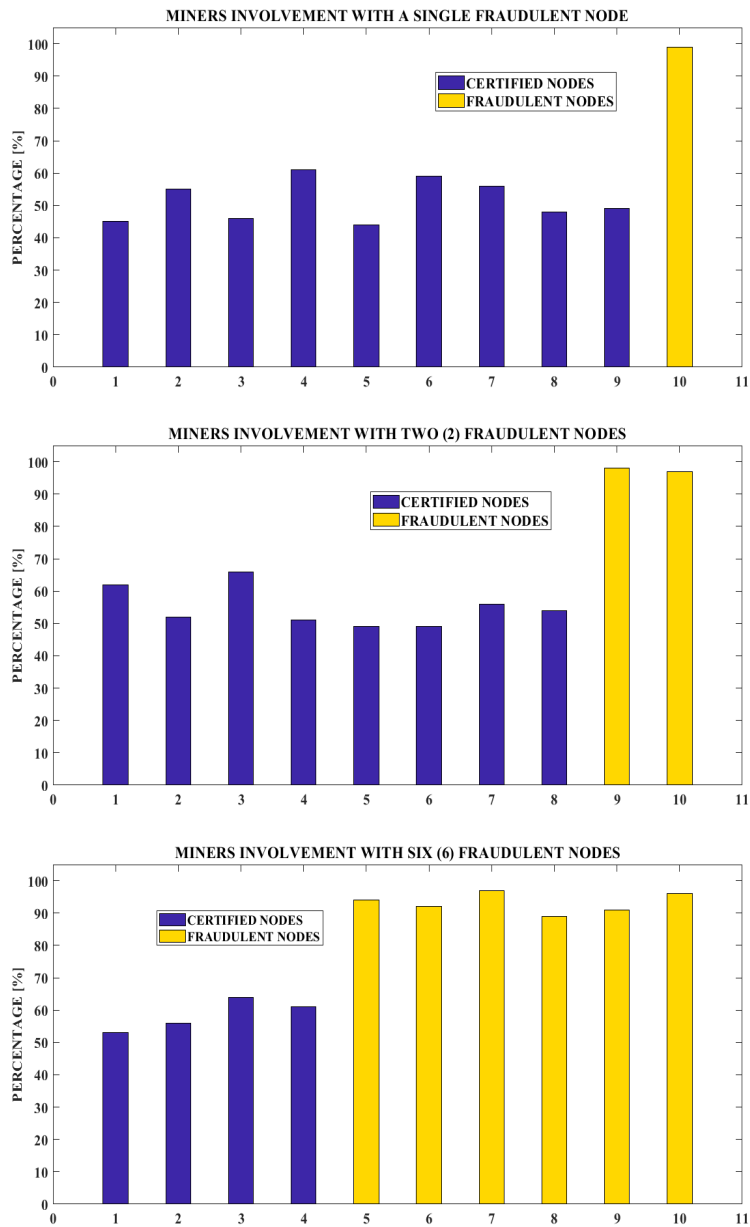


Figure 4.2: Miners involvement in forks using the proposed consensus protocol.

4.2.3 Execution cost and response time of smart contract transactions on the proposed P2P energy trading platform

The latency of transactions with increasing sending rates for different number of consumer nodes defined in the experimental setup described in Chapter 3 is shown in Fig 4.3. The execution costs of the two smart contracts functions involved are shown in Fig 4.4 and Fig 4.5.

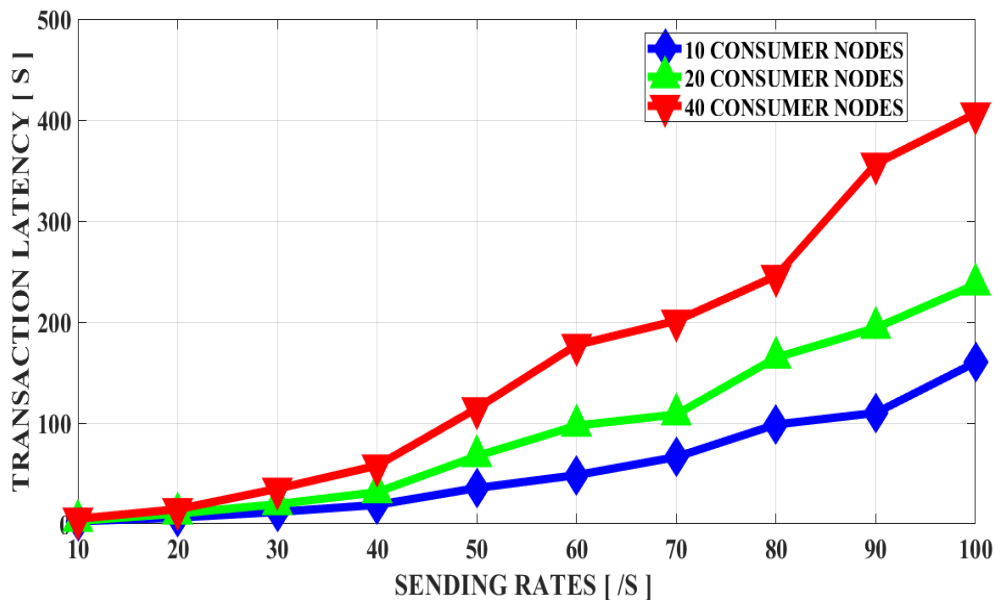


Figure 4.3: Transaction's latency with increasing send rates for different numbers of consumer node.

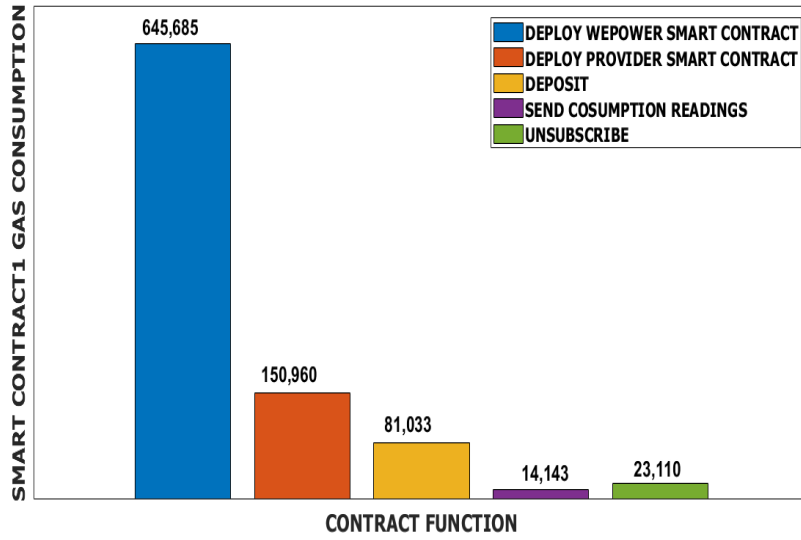


Figure 4.4: Gas consumption for Provider Smart Contract.

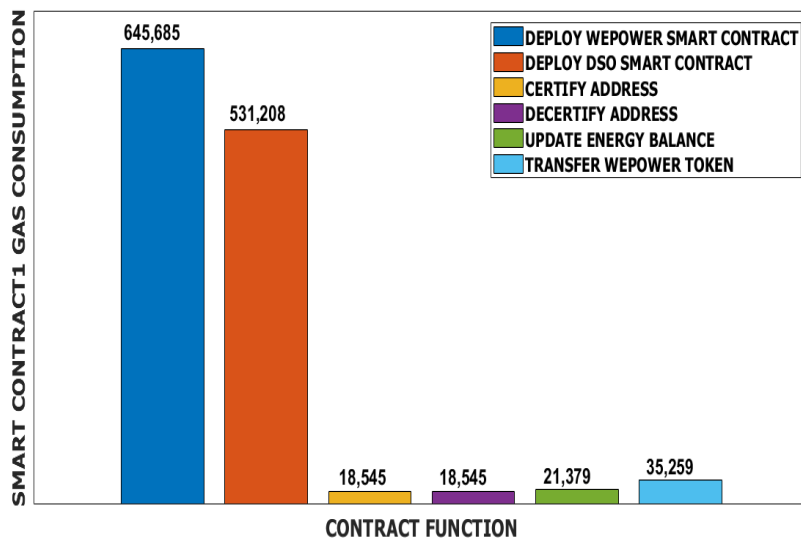


Figure 4.5: DSO Smart Contract Gas consumption.

4.2.4 Discussion of the results

The functioning of the proposed pay per use peer-to-peer energy trading platform has been confirmed in the development phase, so our first objective has been achieved. Experiments were conducted to evaluate the execution time and cost of the platform as the transaction and node load increased. As shown in Fig 4.1, there are two jumps in the performance of the model. The first occurs after nine trading rounds, which corresponds to 9000 days and 90,000 observations, and the MAPE drops from 3% to 2.42%. Then, after the 16th round, the MAPE drops from 2% to less than 1%. After the 20th round, the MAPE no longer decreases steadily, but fluctuates roughly in the range [0.39%, 0.61%]. The advantage of this type of prediction is that it uses the data from the chain to make the prediction. Since each node has a copy of the ledger that contains the data set used to train the model, it ensures the transparency of the prediction, which can be verified by any node in the network. As you can see from Tab 4.1, the value of the K-mean Smart Contract deployment is slightly higher and the gas consumption is much higher when the main K-mean Smart Contract function is called than when the purchase function is called (Niya et al., 2018). Calling K-mean smart contract is 160 times more expensive. However, the K-mean clustering is called only once before each trading round, which makes the gas cost reasonable and the implementation feasible and practical due to the low calling frequency. As for the performance of the modified PoW adapted to the platform in detecting fraudulent nodes, it can be seen in Fig 4.2 that fraudulent nodes are

always ahead of the others in the number of successfully mined blocks, even when the number of fraudulent nodes exceeds half of the entire network. Therefore, a fraudulent node can be easily detected. Since the nodes in the proposed architecture are not anonymous, they can be held accountable for any proven fraudulent behavior. As shown in Fig 4.6, the transaction latency increases as the transmission rate increases. This is due to the time it takes to mine the blocks and the fact that the block size in the Ethereum blockchain is limited to a maximum of 20 million gas and thus a limited number of transactions in a single block. The less gas a transaction consumes, the more transactions can be accommodated in a single block and consequently the lower the transaction latency. The increase in latency depends on the duration of block mining. Fig 4.3 and Fig 4.4 show that deploying the Wepower smart contract is more costly than deploying DSO and prosumer smart contracts together, and that updating the prosumer energy balance in the proposed system is less costly than transferring a Wepower energy token. The remaining smart contract functions show moderate gas consumption for each of the functions in both smart contracts.

4.3 Proposed decentralized PoL scheme for blockchain energy trading platforms

4.3.1 Determination of PoL execution cost and response time

Fig 4.6 shows the transaction latency with increasing sending rates for different number of target nodes. Fig 4.7 shows the gas consumption by invoking the Location verifier

smart functions.

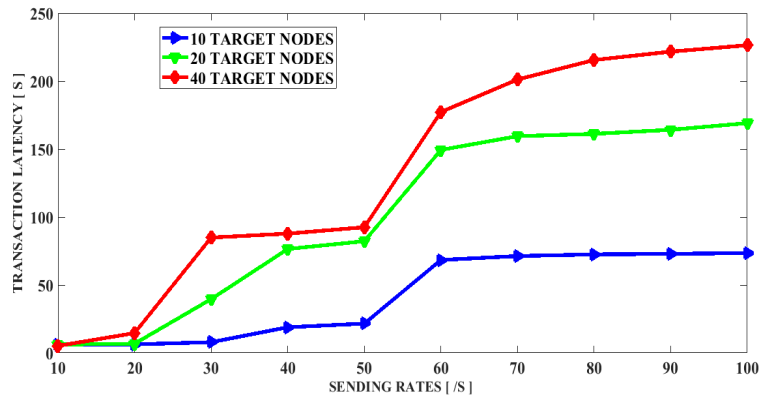


Figure 4.6: Transaction's latency with increasing send rates for different numbers of target node.

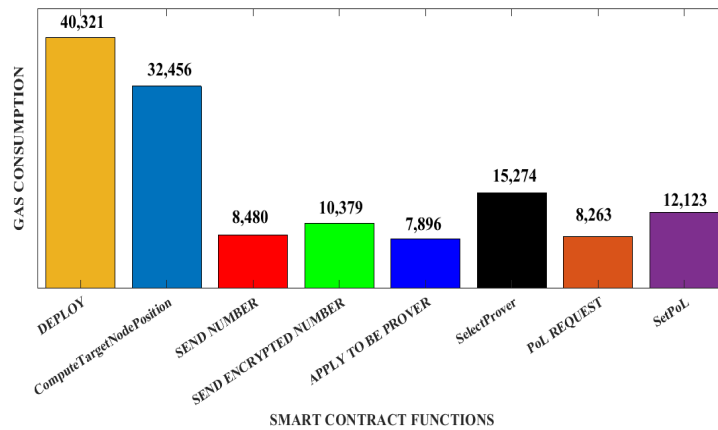


Figure 4.7: Location verifier smart contract gas consumption.

4.3.2 Discussion of the results

The functioning of the proposed decentralized PoL based authentication has been confirmed in the development phase, so our second objective has been achieved. Experiments were conducted to evaluate the execution time and cost of the platform with

increasing transaction and node load. As seen in Fig 4.6, the transaction latency increases with increasing sending rate. This is due to the block mining time and the fact that the block size on the Ethereum blockchain is limited to a maximum of 20 million gas, resulting in a limited number of transactions in a single block. The average transaction execution time is also quite high, which is due to the fact that some transactions in the sequence cannot be in the same blocks. Since some transactions in the sequence cannot be in the same blocks, the average transaction execution time is also quite high. In addition, the latency stabilizes after a certain transmission rate threshold, which is due to the fact that the execution of the sequence is asynchronous and cannot exceed a certain speed. Fig 4.7 shows the gas consumption of each smart contract function. As described in Fig 4.7, the execution of the smart contract functions is reasonable, but the gas consumption of Compute Target Node Position is significantly higher than the others because this function is very computationally intensive.

4.4 Decentralized smart contract-based OPF

4.4.1 Decentralized smart contract-based OPF execution costs

On-chain 3-bus DC-OPF solution, execution cost in U.S. dollars, corresponding to gas price fluctuation on 15/05/2022 is shown in Fig 4.8, while execution cost in dollars for gas price fluctuation on the same day, of the 14-bus AC-OPF problem solved with the proposed extended scheme for different numbers of participating provers is shown in

Fig 4.9.

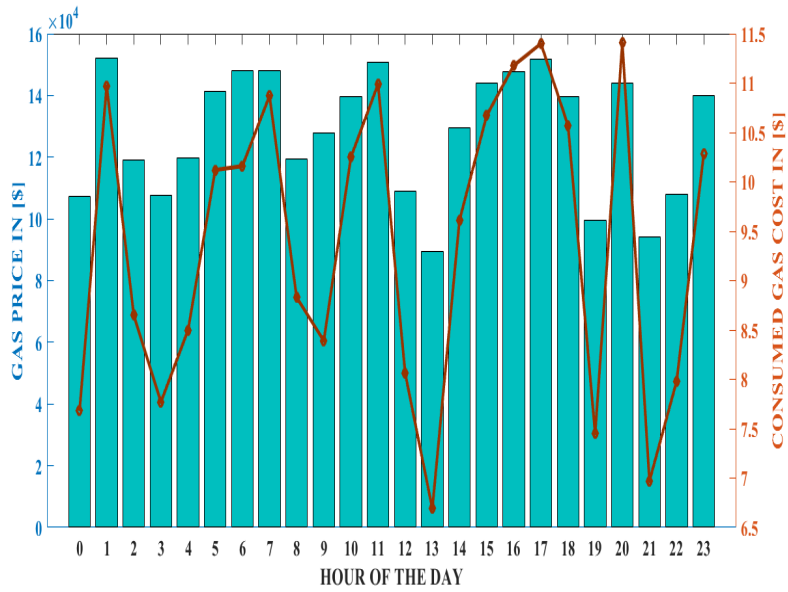


Figure 4.8: On-chain 3-bus DC-OPF solution, execution cost in US Dollars, according to gas price fluctuation in 15/05/2022.

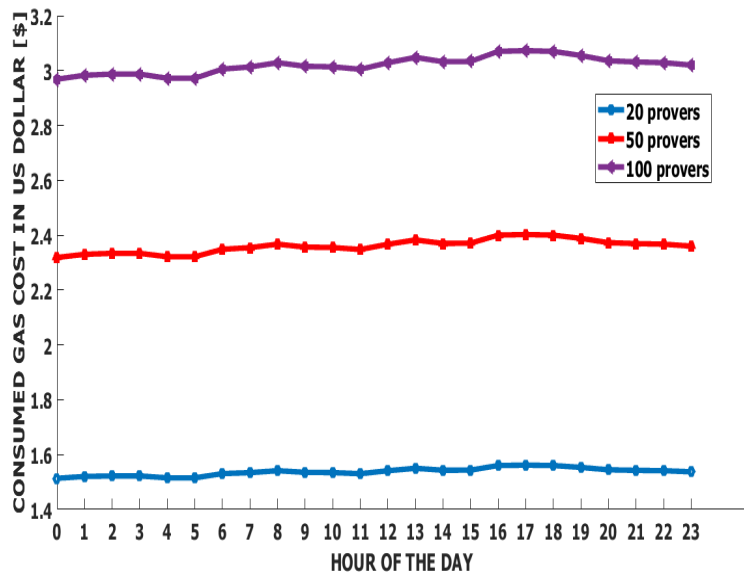


Figure 4.9: Execution cost of 14-bus AC-OPF solution, cost in US dollars, depending on gas price fluctuation on 15/05/2022, using the proposed scheme.

4.4.2 Discussion of the Results

The functioning of the proposed decentralized OPF-based demand response control system has been confirmed in the development phase, so our third objective has been achieved. Experiments were conducted to evaluate the execution time and cost of the platform with increasing transaction and node load. From the execution cost results for the two schemes presented in Sections 3 and 4, shown in Fig 4.10, it can be seen that the extended generalized scheme presented in Section 4 provides a much cheaper execution for a much more complex problem compared to the scheme presented in Section 3, without compromising the decentralization of the system. It can also be seen that the execution cost is much more stable over the daytime hours when using the enhanced model, while the execution cost shows a much more fluctuating pattern when solving the OPF problem on-chain. This is due to the fact that OPF is a NP problem and the number of iterations required to solve a given OPF problem is not deterministic. However, it can be seen that increasing the number of provers in the second scheme increases the execution cost, since the smart contract must loop through all provers and verify and compare all proposed results to determine the provers with the optimal solution. The number of provers allowed to participate in an OPF solution loop must be carefully selected before it is defined and fixed in the smart contract. On the other hand, this solution can still be considered expensive, as the operation can reach a few dollars per hour for very complex problems. However, we believe that this is still a better solution

than a centralized solution that requires the use of hardware. Such an approach has two drawbacks. First, it does not go in the direction of keeping P2P energy trading platforms as decentralized as possible, and second, it is not necessarily a cheaper solution because it requires the use of technical staff for ongoing maintenance and monitoring in addition to the high deployment costs, which are one-time fixed investment costs. Fig 4.11 shows the minimum hourly wage for various industrialized countries (world-population review, 2020)

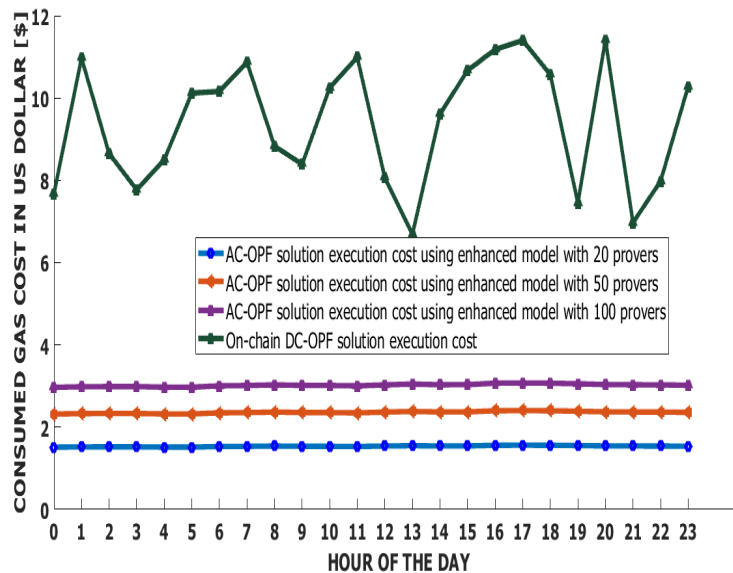


Figure 4.10: Comparison of execution costs between on-chain OPF solution for a 3-bus DC-OPF problem and the solution for a 14-bus AC-OPF problem using the enhanced proposed model.

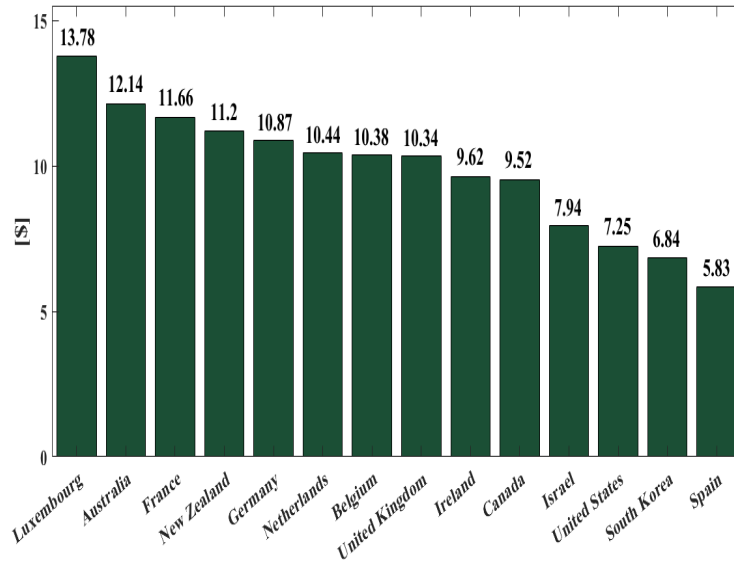


Figure 4.11: Hourly minimum wage for different industrialized countries.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATION

5.1 CONCLUSION

This thesis presents an Ethereum-based control and authentication of a fully decentralized pay-per-use energy trading platform where prosumers collaborate to form a single provider that supplies customers with a prepaid blockchain-based billing system. The potential and feasibility of such a platform are presented in this research, as well as its merits over the traditional energy tokenization energy trading system found in the literature. A detailed description of the architecture and operation of this platform, as well as its implementation, is provided in this thesis. In addition, a ToU pricing model based on machine learning and autonomous clustering based on a smart contract is presented. The aim was to demonstrate the relevance of the proposed platform and evaluate it using relevant metrics. We then presented a PoL for blockchain-based energy platforms, where no centralized entities are in charge of positioning target nodes. Rather, target locations are verified in a decentralized manner, enabled by random number generation using smart contracts and game theory suitable for blockchain energy trading applications. The usefulness and practicality of such a platform was discussed, as well as how location in such a system can be verified in a decentralized yet reliable manner to be

immune to potential fraud or collusion relevant to blockchain energy trading systems. It has been shown how, in the case of two counterparties, as in the case of blockchain-based energy applications, a game-theoretic scenario arises in which equilibrium is only possible if all participating verifiers specify the true origin of the token request, and how, in the case of two counterparties, it is possible to implement a decentralized and unpredictable random selection of verifiers to reduce the likelihood of collusion between them. As part of this work, we provided a detailed description of the architecture and operation of this platform and its implementation. Finally, we presented a blockchain-based demand-side management for smart grids and P2P energy trading platforms using an OPF control system based on smart contracts. In the proposed system, no centralized entities are entrusted with generation control of the various energy sources in the grid, and no control hardware deployed by the DSO is required. Rather, the various generators and owners of energy sources are forced by a decentralized consortium to comply with the calculated OPF solution, and those who do not comply would see their energy contribution wasted. The first approach, where the OPF is fully calculated by a smart contract, showed very high execution costs for a simplified linear approach to the OPF problem, leading us to conclude that while this is a fully decentralized solution, it is not realistic due to the complexity of the OPF problem. Therefore, a different approach was taken in which the OPF problem is not computed on a chain. Instead, different solutions proposed by different provers are compared and verified through a smart contract,

considering only the valid and cost-optimal proposals and rewarding those who proposed them. The proposed system was designed as a Nash game. The fewer reviewers proposed an optimal solution, the more a particular reviewer with an optimal solution proposal wins. This prevents both collusion between provers and laziness in computing the OPF solution, as they risk losing their wagered money if their proposal is not optimal compared to the others. This system was evaluated for solving a non-convex, non-linear AC-OPF problem for a 3-bus network and showed reasonable execution costs that increased with the number of participating examiners. Apart from being a decentralized solution with no central authority controlling the flow of electricity, the proposed system can even be considered a more cost-effective solution considering the cost of providing the hardware and the wages of the staff for continuous maintenance and monitoring in most industrialized countries.

5.2 NOVELTY

The novelty of our work is manifold. First, we proposed the first blockchain energy trading platform that bills based on consumption. Moreover, we proposed a new authentication system for blockchain P2P energy trading, fully decentralized and trustless and uses PoL, while all energy trading platforms presented in the literature rely to some extent on a trusted central authority to secure and authenticate transactions for buying and selling energy. Finally, we have presented the first fully decentralized demand response control system in P2P energy trading platforms, based on an OPF solution in a power grid.

5.3 CONTRIBUTION

In this thesis, firstly we have presented a P2P energy trading platform where energy can be traded in a decentralized manner on an Ethereum private blockchain. The key features which the architecture of the Proposed P2P energy trading system introduces are:

- **Dynamic Time of Use pricing:** This would maximize profit and enable consumers to get an appropriate and accurate demand response. The tariffs are proposed ahead of each trading period, based on energy consumption predictions using machine learning techniques. The tariffs are then implemented in the smart contract to enable autonomous billing. A new smart contract is created ahead of each upcoming trading round, with updated tariffs based on the trading round electricity consumption predictions. Moreover, the tariffs are made known immediately after each trading round and their validity can be verified by all participants, since each one would be having his copy of the shared ledger containing the training set data. Indeed, each participant is aware of the state history representing the energy consumption data that is used in the prediction process. Based on the predictions, ToU ranges are defined using the K-mean clustering algorithm, implemented on a smart contract, and executed on the blockchain. This ensures that the ToU ranges are reliable and bias-free, since they are defined by an independent autonomous smart third party.
- **Prevention against fraudulent behavior:** Considering energy is traded online, it is critical to ensure that the same energy is not sold more than once. Clients must as well upload valid data of their energy consumption. To achieve such a purpose, we propose in the present paper, that the DSO plays the role of a privileged participant, which is allowed certain credentials, making our proposed solution autonomic in nature instead of fully autonomous. First, only approved nodes can join the trading platform as

consumers. These nodes need to be certified by the DSO before being able to subscribe to any trading round. Any uncertified accounts have their subscription requests automatically denied. The DSO performs random checks on consumer nodes to detect any fraudulent or technically faulty ones. The DSO is the only agent given the credential to decertify accounts accordingly. Moreover, a customized consensus algorithm is proposed to spot suspicious nodes. Furthermore, a PoL protocol is implemented to confirm the aggregated data source.

- Automatic and autonomic running: Ethereum blockchain is used to implement the trading procedure through smart contracts, ensuring the forcible payment when demand is supplied, the forcible balance return in case of premature un-subscription, as well as the independent smart contract-based billing.

Secondly, the above proposed system is further enhanced by the use of Proof of location. Hence, a decentralized Proof-of-Location (PoL) system tailored to blockchain applications for energy trading was developed. It ensures that automated transactions are issued by the right nodes. Consumer nodes are located using triangulation and the PoL is carried out using smart contract-based random selection and a game-theoretic scenario suitable for blockchain energy trading.

Thirdly, a detailed approach was developed for implementing the decentralized optimal power flow (OPF) on a private blockchain smart contracts platform that enables an immutable and access-controlled transaction system for tokenized power assets. The model solves the OPF problem of a given grid where all constraints and fixed parameters are set within an immutable and autonomous smart contract. The only variable parameter is the load demand, which can be updated in the smart contract by a load monitoring unit, either periodically or in real time or even using short-term load forecasting, which would allow the smart contract to operate without any required

outside interaction. The smart contract solves the OPF problem for a local network. The OPF solution would be computed by a decentralized, unbiased smart entity and would thus be unchallengeable. The solution would be stored in the blockchain public ledger, making it safe from tampering. Prosumers would only receive the energy tokens they are entitled to if they comply with the OPF solution of the smart contract.

Finally, the proposed P2P energy trading platform performance is analysed, evaluated, tested and benchmarked for its prove it's superiority.

5.4 RECOMMENDATION

Finally, we believe it is useful to conclude this work with another challenge that has not yet been addressed and that is relevant to blockchain and its applications, as well as to the technologies that rely on current cryptographic algorithms in general, in order to outline future trends. Indeed, it is constructive to mention quantum computation as one of the promising directions for future developments. With the expected advent of quantum computing, current blockchain networks would become vulnerable and would need to shift to methods that use quantum-resistant cryptography and quantum networks (Ikeda, 2018; Kiktenko et al., 2018).

The current blockchain relies on cryptographic algorithms to generate key pairs such as Rivest-Shamir-Adleman (RSA) and elliptic curve cryptography. The key pair consists of a private key that is used by its owner to sign the transactions it issues, and the public one is used to verify the digital signature by the rest of the network. Cryptographic algorithms are one-way functions, which means that current computers cannot be used to derive the private key from the associated public key. However, with a quantum computer, any current cryptographic algorithm can be cracked. The generation of RSA cryptographic key pairs can be broken using Shor's quantum computer algorithm (Z.

Cai et al., 2019). Quantum computing is still in the early stages of research. The technology is expected to be ready for deployment within the next decade (Mavroeidis et al., 2018). In the meantime, researchers are working on developing quantum cryptography, as current cryptography would no longer be relevant in the presence of quantum computers. This cryptography, which is immune to quantum computing, is called post-quantum cryptography PQC. In the context of the blockchain, PQC involves the development of a quantum-resistant cryptographic signature. PQC is intended to be secure from quantum computers while being able to be implemented on a conventional computer. Many proposals have been submitted to the National Institute of Standards and Technology NIST. However, after 3 rounds of shortlisting, only three post-quantum cryptographic digital signature algorithms are left in the final selection, only one of which will be selected as the new standard (Alexeev et al., 2021). The three finalists are CRYSTALS -Dilithium (An official website of the United States government, 2022), Falcon (Kinningham et al., 2019), and Rainbow (Yasuda & Sakurai, 2016). This research is at the forefront of cryptographic theory applied to the energy industry (as an example). The extent of the work may become challenging in the future with the use of quantum computers. The work has also paved the way for a highly technical level of programming skills involving cryptographic algorithms.

REFERENCES

- Afzal, M., Huang, Q., Amin, W., Umer, K., Raza, A., & Naeem, M. (2020). Blockchain enabled distributed demand side management in community energy system with smart homes. *IEEE Access*, 8, 37428–37439.
- Ahl, A., Yarime, M., Tanaka, K., & Sagawa, D. (2019). Review of blockchain-based distributed energy: Implications for institutional development. *Renewable and Sustainable Energy Reviews*, 107, 200–211.
- Ahmad, A., Javaid, N., Mateen, A., Awais, M., & Khan, Z. A. (2019). Short-term load forecasting in smart grids: An intelligent modular approach. *Energies*, 12(1), 164.
- Aiman, S., Hassan, S., Habbal, A., Rosli, A., & Shabli, A. H. M. (2018). Smart electricity billing system using blockchain technology. *Journal of Telecommunication, Electronic and Computer Engineering (JTREC)*, 10(2-4), 91–94.
- Alexeev, Y., Bacon, D., Brown, K. R., Calderbank, R., Carr, L. D., & Chong. (2021, Feb). Quantum computer systems for scientific discovery. *PRX Quantum*, 2, 017001. Retrieved from <https://link.aps.org/doi/10.1103/PRXQuantum.2.017001> doi: 10.1103/PRXQuantum.2.017001
- Alves, B. (2019). Statista, installed electricity capacity in belgium 2000- 2019. Retrieved from <https://www.statista.com/statistics/1186751/belgium-installed-electricity-capacity>

- Amanbek, Y., Tabarak, Y., Nunna, H. K., & Doolla, S. (2018). Decentralized transactive energy management system for distribution systems with prosumer microgrids. In 2018 19th international carpathian control conference (iccc) (pp. 553–558).
- An official website of the United States government. (2022). Pqc standardization process: Third round candidate announcement. Retrieved from <https://csrc.nist.gov/News/2020/pqc-third-round-candidate-announcement>
- Antminer s9i. (2021). Retrieved from https://shop.bitmain.com/promote/antminer_s9i_asic_bitcoin_miner/specification
- Architecture overview. what is the sawtooth lake distributed ledger? (2015). Retrieved from <https://sawtooth.hyperledger.org/docs/core/releases/0.7/contents.html>
- Aste, T., Tasca, P., & Di Matteo, T. (2017). Blockchain technologies: The foreseeable impact on society and industry. *computer*, 50(9), 18–28.
- Anagnostopoulos, G. G., & Kalousis, A. (2019). A reproducible analysis of rssi fingerprinting for outdoor localization using sigfox: Preprocessing and hyperparameter tuning. In 2019 international conference on indoor positioning and indoor navigation (ipin) (pp. 1–8).
- Aste, T., Tasca, P., & Di Matteo, T. (2017). Blockchain technologies: The foreseeable impact on society and industry. *computer*, 50(9), 18–28.
- Asuquo, P., Cruickshank, H., Morley, J., Ogah, C. P. A., Lei, A., & Hathal. (2018). Security and privacy in location-based services for vehicular and mobile communications: an overview, challenges, and countermeasures. *IEEE Internet of Things Journal*, 5(6), 4778–4802.
- Barai, S., Biswas, D., & Sau, B. (2020). Sensors positioning for reliable rssi-based outdoor localization using cft. In 2020 ieee international symposium on sustainable energy, signal processing and cyber security (issc) (pp. 1–5).

- Bauwens, T., Gotchev, B., & Holstenkamp, L. (2016). What drives the development of community energy in europe? the case of wind power cooperatives. *Energy Research & Social Science*, 13, 136–147.
- Beikverdi, A. (2015). Nem launches, targets old economy with proof-of- importance. Retrieved from <https://cointelegraph.com/news/nem-launches-targets-old-economy-with-proof-of-importance>
- Biryukov, A., Khovratovich, D., & Pustogarov, I. (2014). Deanonimisation of clients in bitcoin p2p network. In *Proceedings of the 2014 acm sigsac conference on computer and communications security* (pp. 15–29).
- B'IS, K'IN, O. T., & ÇI'FCI, A. (2019). Forecasting of turkey's electrical energy consumption using lstm and gru networks. *Bilecik Şeyh Edebali Üniversitesi Fen Bilimleri Dergisi*, 8(2), 656–667.
- Bitcoin average transactions per block. (2021). Retrieved from https://ycharts.com/indicators/bitcoin_average_transactions_per_block.
- Bitcoin block time chart. (2021). Retrieved from <https://markets.bitcoin.com/crypto/BCH/chart/transaction-size>.
- The bitcoin forum index. (2021). Retrieved from <https://forum.bitcoin.com/technical-support/unconfirmed-bitcoin-transaction-for-more-than-5-days-t57444.html>
- bitinfocharts/blackcoin. (2021). Retrieved from <https://bitinfocharts.com/blackcoin/>
- Bitquery. (2022a). Retrieved from <https://explorer.bitquery.io/ethereum/token/0x178c820f862b14f316509ec36b13123da19a6054>
- Bitquery. (2022b). Retrieved from <https://explorer.bitquery.io/ethereum/token/0x4cf488387f035ff08c371515562cba712f9015d4>
- Bitquery. (2022c). Retrieved from <https://explorer.bitquery.io/ethereum/token/0xf4134146af2d511dd5ea8cdb1c4ac88c57d60404>

- Blockchain explorers. (2021). Retrieved from <https://chainz.cryptoid.info/strat/>
- Blockchain for a smart energy market: Best. (2022). Retrieved from <https://reiner-lemoine-institut.de/en/blockchain-energy-market-best/>
- Blundo, C., Iovino, V., & Persiano, G. (2009). Private-key hidden vector encryption with key confidentiality. In International conference on cryptology and network security (pp. 259–277).
- Boireau, O. (2018). Securing the blockchain against hackers. *Network Security*, 2018(1), 8–11.
- Bouraga, S. (2021). A taxonomy of blockchain consensus protocols: A survey and classification framework. *Expert Systems with Applications*, 168, 114384.
- Boureanu, I., Mitrokotsa, A., & Vaudenay, S. (2015). Practical and provably secure distance-bounding. *Journal of Computer Security*, 23(2), 229–257.
- Boureanu, I., & Vaudenay, S. (2015). Challenges in distance bounding. *IEEE Security & Privacy*, 13(1), 41–48.
- Britz, D. (2015). Recurrent neural network tutorial, part 4 implementing a gru/lstm rnn with python and theano. <http://www.wildml.com/2015/10/recurrent-neural-network-tutorial-part-4-implementing-a-grulstm-rnn-with-python-and-theano>.
- Buccafurri, F., Lax, G., Musarella, L., & Russo, A. (2021). An ethereum-based solution for energy trading in smart grids. *Digital Communications and Networks*.
- Butenko, A. (2016). Sharing energy: dealing with regulatory disconnection in dutch energy law. *European journal of risk regulation*, 7(4), 701–716.
- Buterin, V., & Griffith, V. (2017). Casper the friendly finality gadget. arXiv preprint [arXiv:1710.09437](https://arxiv.org/abs/1710.09437).

- Cai, X., Ren, Y., & Zhang, X. (2019). Privacy-protected deletable blockchain. *IEEE Access*, 8, 6060–6070.
- Cai, Z., Qu, J., Liu, P., & Yu, J. (2019). A blockchain smart contract based on light-weighted quantum blind signature. *IEEE Access*, 7, 138657-138668. doi: 10.1109/ACCESS.2019.2941153
- Cali, U., & Fifield, A. (2019). Towards the decentralized revolution in energy systems using blockchain technology. *Int. J. Smart Grid Clean Energy*, 8(3), 245–256.
- Cap, today's cryptocurrency prices by market. (2021).
- Carson, B., Romanelli, G., Walsh, P., & Zhumaev, A. (2018). Blockchain beyond the hype: What is the strategic business value. McKinsey & Company, 1.
- Castellanos, J. A. F., Coll-Mayor, D., & Notholt, J. A. (2017). Cryptocurrency as guarantees of origin: Simulating a green certificate market with the ethereum blockchain. In *2017 IEEE International Conference on Smart Energy Grid Engineering (SEGE)* (pp. 367–372).
- Castro, M., Liskov, B., et al. (1999). Practical byzantine fault tolerance. In *Osdi* (Vol. 99, pp. 173–186).
- Chan, K.-H., Ke, W., & Im, S.-K. (2020). Caru: A content-adaptive recurrent unit for the transition of hidden state in nlp. In *International conference on neural information processing* (pp. 693–703).
- Cho, K., Van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.
- Christidis, K., & Devetsikiotis, M. (2016). Blockchains and smart contracts for the internet of things. *Ieee Access*, 4, 2292–2303.

- Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.
- Coinmap. (2022). Retrieved from <https://coinmap.org>
- Community energy strategy: Full report. (Dec. 2014). United Kingdom.
- Corp, F.(2021). Foam –the consensus driven map of the world. Retrieved from https://foam.space/publicAssets/FOAM_Whitepaper.pdf
- Criddle, C. (2021). Bitcoin consumes “more electricity than argentina”. BBC News.
- cryptoID. (2021).Navcoin blockchain explorer. Retrieved from <https://chainz.cryptoid.info/nav/>
- cryptoID. (2021). Stratis blockchain explorer. Retrieved from <https://chainz.cryptoid.info/strat/>
- Dalhues, S., Zhou, Y., Pohl, O., Rewald, F., Erlemeyer, F., & Schmid, D. (2019). Research and practice of flexibility in distribution systems: A review. CSEE Journal of Power and Energy Systems, 5(3), 285–294.
- dash/hashrate chart. (2021). Retrieved from <https://bitinfocharts.com/comparison/dash-hashrate.html>
- Devine, M. T., Russo, M., & Cuffe, P. (2019). Blockchain electricity trading using tokenised power delivery contracts (Tech. Rep.). ESRI Working Paper.
- Dey, R., & Salem, F. M. (2017). Gate-variants of gated recurrent unit (gru) neural networks. In 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS) (pp. 1597–1600).
- Dib, O., Brousmiche, K.-L., Durand, A., Thea, E., & Hamida, E. B. (2018). Consortium blockchains: Overview, applications and challenges. International Journal On Advances in Telecommunications, 11(1&2), 51–64.

- Digibyte hashrate chart. (2021). Retrieved from <https://www.coinwarz.com/mining/digibyte/hashrate-chart#:~:text=DigiByte%20hashrate%20is%20a%20calculated,per%20Second%20or%20H%2Fs>
- Di Silvestre, M. L., Gallo, P., Guerrero, J. M., Musca, R., Sanseverino, E. R., & Sciumè, G. (2020). Blockchain for power systems: Current trends and future applications. *Renewable and Sustainable Energy Reviews*, 119, 109585.
- De Kwaasteniet, A. (2021). Miners, block time and orphans, a trinity. Retrieved from <https://medium.com/coinmonks/miners-block-time-and-orphans-a-trinity-680f45f8dd42>
- Dos Santos Silva, M. (2017). Study on “residential prosumers in the european energy union”.
- Du, M., Chen, Q., Liu, L., & Ma, X. (2019). A blockchain-based random number generation algorithm and the application in blockchain games. In *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)* (pp. 3498–3503).
- Du, Y., Wang, Z., & Leung, V. (2021). Blockchain-enabled edge intelligence for IoT: Background, emerging trends and open issues. *Future Internet*, 13(2), 48.
- Durvasulu, V., & Hansen, T. M. (2018). Market-based generator cost functions for power system test cases. *IET Cyber-Physical Systems: Theory & Applications*, 3(4), 194–205.
- Elsden, C., Nissen, B., Jabbar, K., Talhouk, R., Lustig, C., Dunphy, P., . . . Vines, J. (2018). Hci for blockchain: Studying, designing, critiquing and envisioning distributed ledger technologies. In *Extended abstracts of the 2018 CHI conference on human factors in computing systems* (pp. 1–8).
- Emmanuel, M., & Nimmy Chacko, A. T. (2020). Bscdl: A blockchain based smart contract digitized lottery scheme (Tech. Rep.).

- Energy in finland. (2020). Retrieved from [Online].Available: Toimitusvarmuus. Energiavirasto. Entrnce. (2019). Retrieved from <https://www.entrnce.com/>
- Ergun, H., Dave, J., Van Hertem, D., & Geth, F. (2019). Optimal power flow for ac–dc grids: Formulation, convex relaxation, linear approximation, and implementation. *IEEE transactions on power systems*, 34(4), 2980–2990.
- Ethereum hashrate. (2021).Retrieved from <https://2miners.com/eth-network-hashrate>
- EUR-Lex. (2019). Directive (eu) 2018/2001 of the european parliament and of the council of 11 december 2018. Retrieved from https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.L_.2018.328.01.0082.01.ENG
- Fernández, L. (2021). Cumulative installed power capacity in spain in 2021, by technology (in megawatts). Retrieved from <https://www.statista.com/statistics/1002759/installed-power-capacity-in-spain>
- Firo (zcoin) hashrate. (2021). Retrieved from <https://2miners.com/xzc-network-hashrate>
- Foti, M., & Vavalis, M. (2021). What blockchain can do for power grids? *Blockchain: Research and Applications*, 2(1), 100008.
- Gai, K., Wu, Y., Zhu, L., Qiu, M., & Shen, M. (2019). Privacy-preserving energy trading using consortium blockchain in smart grid. *IEEE Transactions on Industrial Informatics*, 15(6), 3548–3558.
- Gajic´, D. B., Petrovic´, V. B., Horvat, N., Dragan, D., Stanisavljevic´, A., Katic´, V., & Popovic´, J. (2022). A distributed ledger-based automated marketplace for the decentralized trading of renewable energy in smart grids. *Energies*, 15(6), 2121.

- Gambis, S., Killijian, M.-O., Roy, M., & Traoré, M. (2014). Props: A privacy-preserving location proof system. In 2014 IEEE 33rd International Symposium on Reliable Distributed Systems (pp. 1–10).
- Georgiadis, E. (2019). How many transactions per second can bitcoin really handle? theoretically. Cryptology ePrint Archive.
- Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. *Neural computation*, 12(10), 2451–2471.
- Gervais, A., Karame, G. O., Wüst, K., Glykantzis, V., Ritzdorf, H., & Capkun, S. (2016). On the security and performance of proof of work blockchains. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security* (pp. 3–16).
- Goldoni, E., Prando, L., Vizziello, A., Savazzi, P., & Gamba, P. (2019). Experimental data set analysis of RSSI-based indoor and outdoor localization in LoRa networks. *Internet Technology Letters*, 2(1), e75.
- Gordon, G., & Tibshirani, R. (2012). Karush-kuhn-tucker conditions. *Optimization*, 10(725/36), 725.
- Greece Europe RE SP. (2022). Retrieved from https://www.irena.org/IRENADocuments/Statistical_Profiles/Europe/Greece_Europe_RE_SP.pdf
- Grid+ whitepaper. (2021). Retrieved from <https://www.allcryptowhitepapers.com/grid-whitepaper/>
- Gruber, N., & Jockisch, A. (2020). Are GRU cells more specific and LSTM cells more sensitive in motive classification of text? *Frontiers in artificial intelligence*, 3, 40.
- Gupta, R., & Rao, U. P. (2017). An exploration to location based service and its privacy preserving techniques: a survey. *Wireless Personal Communications*, 96(2), 1973–2007.

- Gür, A. Ö., Öksüzer, S., & Karaarslan, E. (2019). Blockchain based metering and billing system proposal with privacy protection for the electric network. In 2019 7th international istanbul smart grids and cities congress and fair (icsg) (pp. 204–208).
- Haber, S., & Stornetta, W. S. (1990). How to time-stamp a digital document. In Conference on the theory and application of cryptography (pp. 437–455).
- Han, D., Zhang, C., Ping, J., & Yan, Z. (2020). Smart contract architecture for decentralized energy trading and management based on blockchains. *Energy*, 199, 117417.
- Hanke, T., Movahedi, M., & Williams, D. (2018). DFINITY technology overview series, consensus system. CoRR, abs/1805.04548. Retrieved from <http://arxiv.org/abs/1805.04548>
- Hashrate distribution. (2021). Retrieved from <https://www.blockchain.com/charts/pools>
- Hassan, M. H., Kamel, S., Selim, A., Khurshaid, T., & Domínguez-García, J. L. (2021). A modified rao-2 algorithm for optimal power flow incorporating renewable energy sources. *Mathematics*, 9(13), 1532.
- Hasse, F., Perfall, A., Hillebrand, T., Smole, E., & Lay, L. (2016). Blockchain—an opportunity for energy producers and consumers. pwc global power utilities. Retrieved from <https://www.pwc.com/gx/en/industries/assets/pwc-blockchain-opportunity-for-energy-producers-and-consumers.pdf>
- Heck, J. C., & Salem, F. M. (2017). Simplified minimal gated unit variations for recurrent neural networks. In 2017 IEEE 60th international midwest symposium on circuits and systems (mWSCAS) (pp. 1593–1596).
- Heineman, B. W. (2014). Who's responsible for the walmart mexico scandal. *Harvard Business Review*, 15.

- Hlaing, K. M., & Nyaung, D. E. (2019). Electricity billing system using ethereum and firebase. In 2019 international conference on advanced information technologies (icaait) (pp. 217–221).
- Hoy, M. B. (2017). An introduction to the blockchain and its implications for libraries and medicine. *Medical reference services quarterly*, 36(3), 273–279.
- Ikeda, K. (2018). Chapter seven - security and privacy of blockchain and quantum computation. In P. Raj & G. C. Deka (Eds.), *Blockchain technology: Platforms, tools and use cases* (Vol. 111, p. 199-228). Elsevier. Retrieved from <https://www.sciencedirect.com/science/article/pii/S0065245818300160> doi: <https://doi.org/10.1016/bs.adcom.2018.03.003>
- Ingabire, W., Larijani, H., & Gibson, R. M. (2021). Lora rssi based outdoor localization in an urban area using random neural networks. In *Intelligent computing* (pp. 1032– 1043). Springer.
- Installed capacity for electricity plants in france in 2020, by energy source (in megawatts). (2020). Retrieved from <https://www.statista.com/statistics/1263346/electrical-production-capacity-france/>
- Institut Fraunhofer pour les Systèmes Energétiques Solaires ISE à Fribourg . (2021). Energy-charts. Retrieved from https://www.energy-charts.de/power_inst.html
- ISGAN. (2022). Smarter stronger power transmission: Review of feasible technologies for enhanced capacity and flexibility. Retrieved from <https://www.iea-iskan.org/smarter-stronger-power-transmission-review-offeasible-technologies-for-enhanced-capacity-and-fl>
- Jain, A., Arora, S., Shukla, Y., Patil, T., & Sawant-Patil, S. (2018). Proof of stake with casper the friendly finality gadget protocol for fair validation consensus in ethereum. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 3(3), 291–298.

- Jain, R., Chiu, D., & Hawe, W. (1998). A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. *CoRR*, cs.NI/9809099.
- Jamil, M., & Mittal, S. (2020). Hourly load shifting approach for demand side management in smart grid using grasshopper optimisation algorithm. *IET Gener. Transm. Distrib*, 14(5), 808-815.
- Javali, C., Revadigar, G., Rasmussen, K. B., Hu, W., & Jha, S. (2016). I am alice, i was in wonderland: secure location proof generation and verification protocol. In *2016 IEEE 41st conference on local computer networks (lcn)* (pp. 477–485).
- Jayachandran, M., Rao, K. P., Gatla, R. K., Kalaivani, C., Kalaiarasy, C., & Logasabari- rajan, C. (2022). Operational concerns and solutions in smart electricity distribution systems. *Utilities Policy*, 74, 101329.
- Jenkins, N., Strbac, G., & Ekanayake, J. (2009). *Distributed generation*. Institution of Engineering and Technology.
- Kiktenko, E. O., Pozhar, N. O., Anufriev, M. N., Trushechkin, A. S., Yunusov, R. R., Kurochkin, Y. V., . . . Fedorov, A. K. (2018, may). Quantum-secured blockchain. *Quantum Science and Technology*, 3(3), 035004. Retrieved from <https://doi.org/10.1088/2058-9565/aabc6b> doi: 10.1088/2058-9565/aabc6b
- King, S., & Nadal, S. (2012). Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. self-published paper, August, 19(1).
- Kinningham, K., Levis, P., Anderson, M., Boneh, D., Horowitz, M., & Shih, M. (2019). Falcon — a flexible architecture for accelerating cryptography. In *2019 IEEE 16th international conference on mobile ad hoc and sensor systems (mass)* (p. 136-144). doi: 10.1109/MASS.2019.00025
- Kirli, D., Couraud, B., Robu, V., Salgado-Bravo, M., Norbu, S., Andoni, M., . . . Kiprakis, A. (2022). Smart contracts in energy systems: A systematic review of

fundamental approaches and implementations. *Renewable and Sustainable Energy Reviews*, 158, 112013.

Klomp, R., & Bracciali, A. (2018). On symbolic verification of bitcoin's script language. In *Data privacy management, cryptocurrencies and blockchain technology* (pp. 38–56). Springer.

Košťál, K., Krupa, T., Gembec, M., Vereš, I., Ries, M., & Kotuliak, I. (2018). On transition between pow and pos. In *2018 international symposium elmar* (pp. 207–210).

Kouhizadeh, M., & Sarkis, J. (2018). Blockchain practices, potentials, and perspectives in greening supply chains. *Sustainability*, 10(10), 3652.

Kounadi, O., Resch, B., & Petutschnig, A. (2018). Privacy threats and protection recommendations for the use of geosocial network data in research. *Social Sciences*, 7(10), 191.

Labazova, O., Dehling, T., & Sunyaev, A. (2019). From hype to reality: A taxonomy of blockchain applications. In *Proceedings of the 52nd hawaii international conference on system sciences (hicss 2019)*.

Lamport, L. (2019). The part-time parliament. In *Concurrency: the works of leslie lamport* (pp. 277–317).

Lamport, L., Shostak, R., & Pease, M. (2019). The byzantine generals problem. In *Concurrency: the works of leslie lamport* (pp. 203–226).

Larimer, D. (2014). Delegated proof-of-stake (dpos). *Bitshare whitepaper*, 81, 85.

- Legislation.gov.uk. (2022). Directive 2011/83/eu of the european parliament and of the council. Retrieved from <https://www.legislation.gov.uk/eudr/2011/83/introduction#>
- Li, W., Andreina, S., Bohli, J.-M., & Karame, G. (2017). Securing proof-of-stake blockchain protocols. In *Data privacy management, cryptocurrencies and blockchain technology* (pp. 297–315). Springer.
- Li, Z., Bahramirad, S., Paaso, A., Yan, M., & Shahidehpour, M. (2019). Blockchain for decentralized transactive energy management system in networked microgrids. *The Electricity Journal*, 32(4), 58-72. (Special Issue on Strategies for a sustainable, reliable and resilient grid)
- litecoin/hashrate chart. (2021). Retrieved from <https://bitinfocharts.com/comparison/litecoin-hashrate.html>
- Lora alliance tm strategy committee, lorawan geolocation whitepaper. (2021). Retrieved from https://lora-alliance.org/sites/default/files/2018-04/geolocation_whitepaper.pdf
- Lu, X., Guan, Z., Zhou, X., Wu, L., Du, X., & Guizani, M. (2019). An efficient and privacy-preserving energy trading scheme based on blockchain. In *2019 IEEE Global Communications Conference (GLOBECOM)* (pp. 1–6).
- Luu, L., Chu, D.-H., Olickel, H., Saxena, P., & Hobor, A. (2016). Making smart contracts smarter. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security* (pp. 254–269).
- Mavroeidis, V., Vishi, K., Zych, M. D., & Jøsang, A. (2018). The impact of quantum computing on present cryptography. CoRR, abs/1804.00200. Retrieved from <http://arxiv.org/abs/1804.00200>
- Mazieres, D. (2015). The stellar consensus protocol: A federated model for internet-level consensus. *Stellar Development Foundation*, 32, 1–45.

- Merrad, Y., Habaebi, M. H., Islam, M. R., Gunawan, T. S., Elsheikh, E. A., Suliman, F., & Mesri, M. (2022). Machine learning-blockchain based autonomic peer-to-peer energy trading system. *Applied Sciences*, 12(7), 3507.
- Merz, M. (2020). Enerchain - decentrally traded decentral energy. Retrieved from <https://enerchain.ponton.de/>
- Microgrid knowledge. (2021). Retrieved from <https://microgridknowledge.com/tag/lo3-energy/>
- Miller, D. (2018). Blockchain and the internet of things in the industrial sector. *IT professional*, 20(3), 15–18.
- Mohamed, N., & Al-Jaroodi, J. (2019). Applying blockchain in industry 4.0 applications. In 2019 IEEE 9th annual computing and communication workshop and conference (ccwc) (pp. 0852–0858).
- Monrat, A. A., Schelén, O., & Andersson, K. (2019). A survey of blockchain from the perspectives of applications, challenges, and opportunities. *IEEE Access*, 7, 117134–117151.
- Motepalli, S., & Jacobsen, H.-A. (2022). Decentralizing permissioned blockchain with delay towers. arXiv preprint arXiv:2203.09714.
- Munoz, M. F., Zhang, K., & Amara, F. (2022). Zipzap: A blockchain solution for local energy trading. arXiv preprint arXiv:2202.13450.
- Muzumdar, A., Modi, C., Madhu, G., & Vyjayanthi, C. (2021). A trustworthy and incentivized smart grid energy trading framework using distributed ledger and smart contracts. *Journal of Network and Computer Applications*, 183, 103074.
- Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, 21260.

- National statistics, digest of uk energy statistics (dukes): electricity. (2021). Retrieved from <https://www.gov.uk/government/statistics/electricity-chapter-5-digest-of-united-kingdom-energy-statistics-dukes>
- Neudecker, T., & Hartenstein, H. (2018). Network layer aspects of permissionless blockchains. *IEEE Communications Surveys & Tutorials*, 21(1), 838–857.
- Nguyen, G.-T., & Kim, K. (2018). A survey about consensus algorithms used in blockchain. *Journal of Information processing systems*, 14(1), 101–128.
- Niya, S. R., Schüpfer, F., Bocek, T., & Stiller, B. (2018). A peer-to-peer purchase and rental smart contract-based application (pursca). *it-Information Technology*, 60(5-6), 307–320.
- Nofer, M., Gomber, P., Hinz, O., & Schiereck, D. (2017). Blockchain. *Business & Information Systems Engineering*, 59(3), 183–187.
- Noor, S., Yang, W., Guo, M., van Dam, K. H., & Wang, X. (2018). Energy demand side management within micro-grid networks enhanced by blockchain. *Applied energy*, 228, 1385–1398.
- Nosouhi, M. R., Yu, S., Zhou, W., Grobler, M., & Keshtiar, H. (2020). Blockchain for secure location verification. *Journal of Parallel and Distributed Computing*, 136, 40–51.
- Nusair, K., & Alasali, F. (2020). Optimal power flow management system for a power network with stochastic renewable energy resources using golden ratio optimization method. *Energies*, 13(14), 3671.
- O’Dwyer, K. J., & Malone, D. (2014). Bitcoin mining and its energy footprint.
- Over 10 years of hourly energy consumption data from pjm in megawatts. (2022). Retrieved from https://www.kaggle.com/robikscube/hourly-energy-consumption?select=AEP_hourly.csv

- Pahlajani, S., Kshirsagar, A., & Pachghare, V. (2019). Survey on private blockchain consensus algorithms. In 2019 1st international conference on innovations in information and communication technology (iciict) (pp. 1–6).
- Park, C., & Yong, T. (2017). Comparative review and discussion on p2p electricity trading. *Energy Procedia*, 128, 3–9.
- Pham, A., Huguenin, K., Bilogrevic, I., Dacosta, I., & Hubaux, J.-P. (2015). Securerun: Cheat-proof and private summaries for location-based activities. *IEEE Transactions on Mobile Computing*, 15(8), 2109–2123.
- Phruksahiran, N., & Michanan, J. (2021). Iteration improvement of taylor-series estimation using hyperbolic systems for fm-radio source localization in bangkok. *Signal, Image and Video Processing*, 15(2), 247–254.
- Pinson, P., Madsen, H., et al. (2014). Benefits and challenges of electrical demand response: A critical review. *Renewable and Sustainable Energy Reviews*, 39, 686– 699.
- Poland - country commercial guide. (2019). Retrieved from <https://www.trade.gov/country-commercial-guides/poland-energy-sector>
- Polge, J., Robert, J., & Le Traon, Y. (2021). Permissioned blockchain frameworks in the industry: A comparison. *Ict Express*, 7(2), 229–233.
- Pop, C., Cioara, T., Antal, M., Anghel, I., Salomie, I., & Bertoncini, M. (2018). Blockchain based decentralized management of demand response programs in smart energy grids. *Sensors*, 18(1), 162.
- Portugal europe re sp. (2022). Retrieved from https://www.irena.org/IRENADocuments/Statistical_Profiles/Europe/Portugal_Europe_RE_SP.pdf
- Power ledger whitepaper. (2021). Retrieved from <https://www.allcryptowhitepapers.com/powerledger-whitepaper/>

- Rinaldi, G., Menon, P. P., Edwards, C., & Ferrara, A. (2019). Higher order sliding mode observers in power grids with traditional and renewable sources. *IEEE Control Systems Letters*, 4(1), 223–228.
- Rizal Batubara, F., Ubacht, J., & Janssen, M. (2019). Unraveling transparency and accountability in blockchain. In *Proceedings of the 20th annual international conference on digital government research* (pp. 204–213).
- Rosic, A. (2020). What is ethereum casper protocol? crash course. Retrieved from <https://blockgeeks.com/guides/ethereum-casper>
- Salimitari, M., & Chatterjee, M. (2018). A survey on consensus protocols in blockchain for iot networks. arXiv preprint arXiv:1809.05613.
- Salimitari, M., Chatterjee, M., & Fallah, Y. P. (2020). A survey on consensus methods in blockchain for resource-constrained iot networks. *Internet of Things*, 11, 100212.
- Sameeh, T. (2016). Two new models double spending attacks in bitcoins blockchain. *CoinDesk Inc.*
- Saroiu, S., & Wolman, A. (2009). Enabling new mobile applications with location proofs. In *Proceedings of the 10th workshop on mobile computing systems and applications* (pp. 1–6).
- Raper, J., Gartner, G., Karimi, H., & Rizos, C. (2007). Applications of location-based services: a selected review. *Journal of Location Based Services*, 1(2), 89–111.
- Raval, S. (2016). *Decentralized applications: harnessing bitcoin's blockchain technology*. " O'Reilly Media, Inc."
- Ravanelli, M., Brakel, P., Omologo, M., & Bengio, Y. (2018). Light gated recurrent units for speech recognition. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(2), 92–102.

- Riaz, M., Hanif, A., Masood, H., Khan, M. A., Afaq, K., Kang, B.-G., & Nam, Y. (2021). An optimal power flow solution of a system integrated with renewable sources using a hybrid optimizer. *Sustainability*, 13(23), 13382.
- Saxena, e. a., Shivam. (2019). Design and field implementation of blockchain based renewable energy trading in residential communities. *International conference on smart grid and renewable energy (SGRE)*, 2.
- Schwarz-Schilling, C., Neu, J., Monnot, B., Asgaonkar, A., Tas, E. N., & Tse, D. (2021). Three attacks on proof-of-stake ethereum. *arXiv preprint arXiv:2110.10086*.
- SEC, S. (2000). 2: Recommended elliptic curve domain parameters. *Standards for Efficient Cryptography Group, Certicom Corp*, 5.
- Sediq, A. B., Gohary, R. H., Schoenen, R., & Yanikomeroglu, H. (2013). Optimal tradeoff between sum-rate efficiency and jain's fairness index in resource allocation. *IEEE Transactions on Wireless Communications*, 12(7), 3496–3509.
- Shabani, M. (2019). Blockchain-based platforms for genomic data sharing: a decentralized approach in response to the governance problems? *Journal of the American Medical Informatics Association*, 26(1), 76–80.
- Shahidehpour, M., Yamin, H., & Li, Z. (2002). Market overview in electric power systems.
- Shibata, N. (2019). Proof-of-search: combining blockchain consensus formation with solving optimization problems. *IEEE Access*, 7, 172994–173006.
- Shu, F., & Lei, K. (2021). Vger: a vrf based cross-chain mechanism for blockchains. In *Journal of physics: Conference series* (Vol. 1780, p. 012038).
- Siano, P. (2014). Demand response and smart grids—a survey. *Renewable and sustainable energy reviews*, 30, 461–478.

- Song, J. G., Kang, E. S., Shin, H. W., & Jang, J. W. (2021). A smart contract-based p2p energy trading system with dynamic pricing on ethereum blockchain. *Sensors*, 21(6), 1985.
- Sotkiewicz, P. M., & Vignolo, J. M. (2007). Towards a cost causation-based tariff for distribution networks with dg. *IEEE Transactions on Power Systems*, 22(3), 1051– 1060.
- Sousa, J., Bessani, A., & Vukolic, M. (2018). A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform. In 2018 48th annual ieee/ifip international conference on dependable systems and networks (dsn) (pp. 51–58).
- Spectrecoin. (2020). Retrieved from <https://www.coinlore.com/coin/spectrecoin>
- Su, Y., & Kuo, C.-C. J. (2019). On extended long short-term memory and dependent bidirectional recurrent neural network. *Neurocomputing*, 356, 151–161.
- Sukhwani, H., Martínez, J. M., Chang, X., Trivedi, K. S., & Rindos, A. (2017). Performance modeling of pbft consensus process for permissioned blockchain network (hyperledger fabric). In 2017 ieee 36th symposium on reliable distributed systems (srds) (pp. 253–255).
- Suresh, A., Nair, A. R., Lal, A., Sarath, G., et al. (2020). A hybrid proof based consensus algorithm for permission less blockchain. In 2020 second international conference on inventive research in computing applications (icirca) (pp. 707–713).
- Swan, M. (2015). *Blockchain: Blueprint for a new economy*. " O'Reilly Media, Inc."
- Syscoin hashrate-chart. (2021). Retrieved from <https://www.coinwarz.com/mining/syscoin/hashrate-chart>

- Team, T. (2022). Etherscan. Retrieved from <https://etherscan.io/tx/0x3421fd298722a02e2eae65f78904f6acb92d7a4c3d4f256fe7ccd782ca717786>
- Thukral, M. K. (2021). Emergence of blockchain-technology application in peer-to-peer electrical-energy trading: a review. *Clean Energy*, 5(1), 104–123.
- Todorean, L., Antal, C., Antal, M., Mitrea, D., Cioara, T., Anghel, I., & Salomie, I. (2021). A lockable erc20 token for peer to peer energy trading. arXiv preprint arXiv:2111.04467.
- Tokgöz, A., & Ünal, G. (2018). A rnn based time series approach for forecasting turkish electricity load. In 2018 26th signal processing and communications applications conference (siu) (pp. 1–4).
- Total hash rate (th/s). (2021). Retrieved from <https://www.blockchain.com/fr/charts/hash-rate>
- Tovanich, N., Soulié, N., & Isenberg, P. (2021). Visual analytics of bitcoin mining pool evolution: on the road toward stability? In 2021 11th ifip international conference on new technologies, mobility and security (ntms) (pp. 1–5).
- Vale, Z., Faria, P., Abrishambaf, O., Gomes, L., & Pinto, T. (2021). Martine—a platform for real-time energy management in smart grids. *Energies*, 14(7), 1820.
- Vasin, P. (2014). Blackcoin’s proof-of-stake protocol v2. URL: <https://blackcoin.co/blackcoin-pos-protocol-v2-whitepaper.pdf> , 71.
- Verma, A. K., & Garg, A. (2017). Blockchain: An analysis on next-generation internet. *International Journal of Advanced Research in Computer Science*, 8(8).

- Wang, B., Zhao, S., Li, Y., Wu, C., Tan, J., Li, H., & Yukita, K. (2021). Design of a privacy-preserving decentralized energy trading scheme in blockchain network environment. *International Journal of Electrical Power & Energy Systems*, 125, 106465.
- Wang, G., Wang, S., Bagaria, V., Tse, D., & Viswanath, P. (2020). Prism removes consensus bottleneck for smart contracts. In *2020 crypto valley conference on blockchain technology (cvcbt)* (pp. 68–77).
- Wang, S., Ouyang, L., Yuan, Y., Ni, X., Han, X., & Wang, F.-Y. (2019). Blockchain-enabled smart contracts: architecture, applications, and future trends. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(11), 2266–2277.
- Wang, S., Taha, A. F., Wang, J., Kvaternik, K., & Hahn, A. (2019). Energy crowdsourcing and peer-to-peer energy trading in blockchain-enabled smart grids. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(8), 1612–1623.
- Wang, Z., & Anderson, C. L. (2021). A progressive period optimal power flow for systems with high penetration of variable renewable energy sources. *Energies*, 14(10), 2815.
- Wang, Z., Liu, F., Ma, Z., Chen, Y., Jia, M., Wei, W., & Wu, Q. (2021). Distributed generalized nash equilibrium seeking for energy sharing games in prosumers. *IEEE Transactions on Power Systems*, 36(5), 3973–3986.
- Wepower white paper. (2016). Retrieved 2022-09-30, from https://wepower.com/media/WhitePaper-WePower_v_2.pdf
- Whitepaper brikcoin. (2017). Retrieved from <https://icobench.com/ico/brikcoin/whitepaper>
- Wood, G., et al. (2014). Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014), 1–32.

- World population review. (2020). minimum-wage-by-country. Retrieved from <https://worldpopulationreview.com/country-rankings/minimum-wage-by-country>
- Wu, X., & Conejo, A. J. (2022). Distribution market including prosumers: An equilibrium analysis. *IEEE Transactions on Smart Grid*.
- Xing, C., Chen, Z., Chen, L., Guo, X., Zheng, Z., & Li, J. (2020). A new scheme of vulnerability analysis in smart contract with machine learning. *Wireless Networks*, 1–10.
- Xu, X., Pautasso, C., Zhu, L., Gramoli, V., Ponomarev, A., Tran, A. B., & Chen, S. (2016). The blockchain as a software connector. In 2016 13th working ieee/ifip conference on software architecture (wicsa) (pp. 182–191).
- Yasuda, T., & Sakurai, K. (2016). A multivariate encryption scheme with rainbow. In *Information and communications security Cham: Springer International Publishing*. (pp. 236–251).
- Ycharts. (2020). ethereum average gas price. Retrieved from https://ycharts.com/indicators/ethereum_average_gas_price
- Ycharts. (2020). ethereum price. Retrieved from https://ycharts.com/indicators/ethereum_price
- Ye, C., Li, G., Cai, H., Gu, Y., & Fukuda, A. (2018). Analysis of security in blockchain: Case study in 51%-attack detecting. In 2018 5th international conference on depend- able systems and their applications (dsa) (pp. 15–24).
- Young, J. (2020). Why the actual cost of mining bitcoin can leave it vulnerable to a deep correction. *Financial Times*.
- Zhang, J., Zhong, S., Wang, T., Chao, H.-C., & Wang, J. (2020). Blockchain-based systems and applications: a survey. *Journal of Internet Technology*, 21(1), 1–14.

- Zhang, S., & Lee, J.-H. (2019). Double-spending with a sybil attack in the bitcoin decentralized network. *IEEE transactions on Industrial Informatics*, 15(10), 5715–5722.
- Zhang, X., Qin, R., Yuan, Y., & Wang, F.-Y. (2018). An analysis of blockchain-based bitcoin mining difficulty: Techniques and principles. In *2018 chinese automation congress (cac)* (pp. 1184–1189).
- Zhang, Z., Zhou, L., Zhao, X., Wang, G., Su, Y., Metzger, M., . . . Zhao, B. Y. (2013). On the validity of geosocial mobility traces. In *Proceedings of the twelfth acm work- shop on hot topics in networks* (pp. 1–7).
- Zhou, Y., Kumar, D., Bakshi, S., Mason, J., Miller, A., & Bailey, M. (2018). Erays: reverse engineering ethereum’s opaque smart contracts. In *27th usenix security sym- posium (usenix security 18)* (pp. 1371–1385).
- Zhu, Z., & Cao, G. (2011). Applaus: A privacy-preserving location proof updating system for location-based services. In *2011 proceedings ieee infocom* (pp. 1889–1897).

LIST OF PUBLICATIONS

Merrad, Y., Habaebi, M. H., Elsheikh, E. A., Suliman, F. E. M., Islam, M. R., Gunawan, T. S., Mesri, M. (2022). Blockchain: Consensus Algorithm Key Performance Indicators, Trade-Offs, Current Trends, Common Drawbacks, and Novel Solution Proposals. *Mathematics*, 10(15), 2754. <https://doi.org/10.3390/math10152754>

Merrad Y, Habaebi MH, Toha SF, Islam M, Gunawan TS, Mesri M. (2022). Fully Decentralized, Cost-Effective Energy Demand Response Management System with a Smart Contracts-Based Optimal Power Flow Solution for Smart Grids. *Energies*.15(12):4461. <https://doi.org/10.3390/en15124461>

Merrad, Y., Habaebi, M. H., Islam, M., Gunawan, T. S., amp; Mesri, M. (2022). Robust Decentralized Proof of Location for Blockchain Energy Applications Using Game Theory and Random Selection. *Sustainability*, 14(10), 6123. <https://doi.org/10.3390/su14106123>

Merrad, Y., Habaebi, M. H., Islam, M. R., Gunawan, T. S., Elsheikh, E. A., Suliman,F., M., Mesri, M. (2022). Machine Learning-Blockchain Based Autonomic Peer-to- Peer Energy Trading System. *Applied Sciences*, 12(7), 3507. <https://doi.org/10.3390/app12073507>

Habaebi, M. H., Merrad, Y., Islam, M. R., Elsheikh, E. A., Sliman, F. M., Mesri, M. (2023). Extending CloudSim to simulate sensor networks. *Simulation*, 99(1), 3-22.00375497221105530.

Merrad, Y., Habaebi, M. H., Islam, M. R., & Gunawan, T. S. (2020). A Real-time Mobile Notification System for Inventory Stock out Detection using SIFT and RANSAC. *International Journal of Interactive Mobile Technologies*, 14(5). <https://doi.org/10.3991/ijim.v14i05.13315>

Merah, H., Merrad, Y., Habaebi, M. H., & Mesri, M. (2021). A novel PTS-based PAPR reduction scheme for FBMC-OQAM system without extra bit transmission of SI. *International Journal of Electronics*, 108(6), 928-944. <https://doi.org/10.1080/00207217.2020.1818847>