2023 2nd International Conference on Emerging Trends in Electrical, Control, and Telecommunication Engineering (ETECTE) | 979-8-3503-0565-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/ETECTE59617.2023.10396687

Formal Modeling and Verification of Validator Voluntarily Exit in Ethereum 2.0 Beacon Chain

Muhammad Rashid
Dept of Computer Science
COMSATS University Islamabad
Sahiwal Campus, Pakistan
rashid674r@gmail.com

Imran Rasool
Dept of Computer Science
COMSATS University Islamabad
Sahiwal Campus, Pakistan
imranrasoolch@gmail.com

Nazir Ahmad Zafar
Dept of Computer Science
COMSATS University Islamabad
Sahiwal Campus, Pakistan
nazafar@gmail.com

Hamra Afzaal
Dept of Computer Science
Information Technology University
Lahore, Pakistan
hamraafzaal@hotmail.com

Abstract— **The Ethereum 2.0 Beacon chain is a big step toward improving the security, scalability, and decentralization. It is based on the Proof of Stake (PoS) consensus protocol that maintains the validator status, attestation details, and many more. The term validator is introduced in the PoS consensus protocol and its work is to propose blocks and vote for them by becoming a committee member to add those blocks to the blockchain. We address the problem related to validator's voluntary exit to vanish its active status and do its formal verification using the SPIN Model checker. There does not exist much research in this area as it was introduced recently in 2020. Further, this is the first work on formal verification of the beacon chain using the SPIN model checker. In this work, we formally specify the validator exit process of the Ethereum 2.0 Beacon Chain using the formal specification language, i.e., Process or Protocol Meta Language (PROMELA). We create a program graph for this process through the SPIN Model checker and describe safety and liveness properties using Linear temporal logic (LTL) in the form of a formula. The formal verification is performed to ensure correctness by giving the program graph and LTL formulas as input to the SPIN model checker whether the properties are satisfied with the program graph**

Keywords— Model checking, Formal Methods, SPIN, LTL, Blockchain, Beacon Chain, Ethereum 2.0, Program Graph

I. INTRODUCTION

Blockchain innovation and circulated records are drawing in huge consideration and setting off various activities in various enterprises. Be that as it may, the monetary business is viewed as an essential client of the blockchain idea. Blockchain has numerous applications like Bitcoin and Ethereum and some more. Our primary center is to figure out Ethereum [1]. Ethereum is a project that endeavors to construct the summed up innovation, on which all exchange-based state machine ideas might be fabricated [2]. Ethereum 2.0 is a significant move up to the Ethereum stage presenting another Proof of Stake (PoS) protocol, the Beacon chain protocol, with the essential objective of expanding the effectiveness, versatility, and security of Ethereum. Partaking hubs in the convention, called validators, secure a piece of their holdings of the underlying currency of the platform (Ether) so they might be picked as individuals from councils that propose and approve new blocks [3]. The Ethereum 2.0 Beacon Chain is introduced in 2020 with several features, e.g., recording the details of the active validators, exit validators, and details of votes in the blocks. Slashing and reward details are also recorded in the Beacon Chain. In 2022, the sharding concept is introduced in the Beacon Chain which is crosslinked to the Beacon Chain called as Phase 1 of Ethereum 2.0 but we are currently considering only the Beacon chain. Therefore, the Ethereum 2.0 with only a Beacon Chain is called phase 0 of the Ethereum 2.0.

An executable K-model of the Beacon Chain in Ethereum 2.0 is provided in [4]. The author gives a formal model of the Python code of the Beacon chain using the K tool. Currently, the formal model of the Beacon Chain in the SPIN model checker does not exist. Our major objectives to perform this research are 1) ensure safety 2) realize the importance of SPIN 3) perform high-level modeling. Therefore, in this work, we describe the formal model of the validator voluntarily exit process in the Ethereum 2.0 Beacon Chain. In the voluntarily exit process, validator does not want to propose and validate the blocks any more. After validator exit, he can withdraw his deposit ethers and also the rewards of validating the blocks. The complete process of validator exit process is explained in Ethereum 2.0 section. The SPIN model checker is used for verification of this process. The exit process is specified using the formal specification language, i.e., Process or Protocol Meta Language (PROMELA) to create a program graph. The syntax of PROMELA resembles the syntax of C language. A program graph is a model like an automaton but it is different from finite automaton. The end state is not compulsory in the program graph. Our first step of verification is completed after creating the program graph. After that, safety properties are formalized using Linear temporal logic (LTL). It is used to specify safety and liveness properties. Till now, we have the program graph, an automated model of the validator exit process, and the LTL formula for the properties we define on the voluntarily exit procedure. The model checker takes the program graph and LTL formulas as input and the model checker verifies these LTL formulas on the program graph. The SPIN model checker shows the results of the verification of the validator activation in the Ethereum 2.0 Beacon Chain.

The rest of the paper is organized as follows: the related work of Beacon chain and formal verification is presented in Section II. In Section III, introduction to model checking is given. In section IV, the Introduction to Ethereum 2.0 and Beacon chain is given and our validator exit procedure is also explained in this section. Formal specification and verification are done in section V, proof of the correctness of validator exit procedure is also given in same section. Concluded in Section VI.

II. Related Work

In this section, we describe the work done related to our research. In [3], the Gasper protocol of the Ethereum 2.0 is officially checked utilizing Coq verification tool. Gasper is a mix of Casper and Ghost. Casper is a conclusion of finalization gadget and Ghost means casting a vote on the Beacon Chain head. In [4], an executable K-model is developed for Phase 0, Ethereum 2.0 Beacon Chain. In [5], formal check of the Casper protocol is finished utilizing the Coq tool. Casper is the transformation of the Proof of Work (PoW) convention to the proof of Stake (PoS) convention in Ethereum 2.0. In [6], the formal verification of the Correct-by-Construction Casper protocol is done utilizing the Coq tool. Correct-by-Construction is likewise one more sort of Casper convention. It is not quite the same as that of Casper FFG (finalization-friendly gadget). In [7], the formal verification of the Beacon Chain is done using a deductive verification approach. Dafny language is used for this process of formal verification. In [8], Casper was introduced in this work and all its detail was described by the Ethereum 2.0 founder itself. In [9], in this paper Verilay, the first chain relay scheme is proposed that enables validating PoS protocols to produce finalized blocks, for example, Ethereum 2.0, Cosmos, and Polkadot. The concept does not require changes to the source blockchain protocols or validator operations. Signatures of block proposers are validated by a dedicated relay smart contract on the target blockchain. In [10], perform probabilistic and cost examinations of assailants in the Beacon Chain with 30 % of total stake. In [11], a complete review of update from Ethereum to Ethereum 2.0 is given. which is basically conversion from PoW to PoS protocol. In [12], the main features and differences of Proof-Stake-BFT proposals are presented with respect to Bitcoin, to appreciate their maturity and outline open issues and ongoing research challenges. In [13], the issues of smart contracts in Ethereum are addressed. A far-reaching concentration on comparison with these issues is given. In [14],-blockchain and its use cases are studied and an alternative way of authentication service has been proposed based on Ethereum Blockchain called DAuth. Furthermore, a prototype has been developed which enables user authentication on the site. In [15], the Ethereum blockchain is used to create a system which deals with home applications. In [16], a Web 3.0 based architecture is proposed which eliminates the centralized agencies and to promote a fully decentralized, secure, and transparent internet. Ethereum and smart contracts create a secure decentralized mechanism for initiating data-based payments. In [17], the authors deal with the smart contracts framework in blockchain and address the issues related to smart contract security vulnerabilities. In [18], the study has proposed a solution to completely decentralize the E-Commerce platform by using blockchain in conjunction with smart contracts and utilizes the decentralized storage like IPFS (Inter Planetary File System). In [19], the authors explore unification of execution traces for debug task in FBME - modular IDE for IEC 61499 applications. Further, they present the formal model of the execution trace representation and show the working on a simple example. In [20], the authors discuss about conversion of UML diagram into PROMELA language so it is easy to implement in the SPIN model checker. In [21], the modeling and verification of the timed automaton using SPIN model checker are given. In [22], SPIN model checker is used for the flight software. It deals with the spacecraft working formalism. In [23], the SPIN model checker for the formal verification of distance vector routing protocol is used. In [24], the authors address smart waste management system in which the author have integrated different techniques for providing modeling of smart waste management system using IoT and blockchain. The integrated technologies include blockchain, IoT, Unified Modeling Language (UML), and Temporal Logic of Action (TLA+) for modeling and automating waste management systems. There exists some other work of formal specification and verification is done using VDMSL in [25]–[27]. Formal Verification is done of consensus protocol in [28],[29].

In this work, we analyze, specify, and verify Ethereum 2.0 through the SPIN model checker. To the best of our knowledge, this is the first work on formal verification of the Beacon Chain in the SPIN model checker. In this work, we specifically focus on the formal verification of the validator voluntarily exit process in the SPIN model checker.

III. Model Checking

Model checking is a technique to check and increase the correctness of the systems. Correctness is ensured by checking different properties on the program graph. The program graph is the automated model we built for the System. Fig. 1 is demonstrated how model checker takes the LTL formula and program graph as input and checks whether program graph satisfies this LTL formula or not.

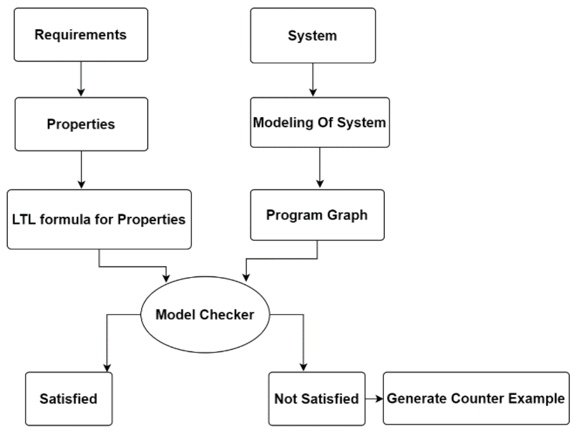


Fig. 1. Overview of model checking.

To verify the correctness of these program graphs, we use different safety and liveness properties on them. In the safety properties are in which we never want something bad to happen. Prefixes or finite words which violates the property somewhere are considered as the bad prefixes of the property. Safety properties can have further subcategories like invariants and Regular properties. LTL can be used to describe the property in the form of Boolean formula. In our research work we will formalize our property in the form of LTL.

IV. Ethereum 2.0

Ethereum was introduce by Vitalik Buterin in 2015. Ethereum at that time use Proof of Work protocol and had many other similarities to bitcoin. Ethereum founders aim to update it in four different phases which are Frontier, Homestead, Metropolis and Serenity. These four phases were decided to take place and now last update Serenity is going on. Each phase has its hard forks which we call the sub releases. It was decided when Ethereum move from Proof of Work consensus to PoS consensus, it will be named as Ethereum 2.0. Ethereum is now working on proof of stake consensus that is why we call this Ethereum 2.0. Now Serenity is going on and it has three hard forks like first is Beacon chain which is implemented in 2020 and second is sharding which take place in 2022 and third is Execution which will take place somewhere in future. Our work related to validator take place in Beacon chain so we will give its little introduction [8].

A. Beacon Chain

Beacon chain is a sub release in Serenity and take place in 2020. It keeps details of active validator, withdrawal, attestation, finalized slots and epochs. For our research, little introduction to epoch and slots is necessary.

Each slot is of 12 second and each epoch have 32 slots as shown Fig. 2. An epoch is of 6.4 minutes and in each slot one block is proposed and then attested by committee of 128 validators.

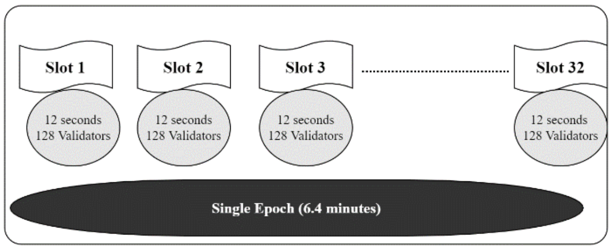

Slot 1
Slot 2
Slot 3
Slot 32
12 seconds 128 Validators
12 seconds 128 Validators
12 seconds 128 Validators
12 seconds 128 Validators
Single Epoch (6.4 minutes)

Fig. 2. Overview of an epoch

Beacon chain is heart of the Ethereum 2.0 PoS consensus. Beacon Chain also slashes and rewards the validator. If validator is dishonest at any point, it can be slashed by keeping their reward or by deducting complete amount of validator at stake.

B. Validator Voluntarily Exit procedure

Voluntarily exit is the complete procedure in which validator does not want to continue with its active status. Validator want to withdraw their deposit and rewards. Using flow diagram in Fig. 3, we will show the complete steps in voluntarily exit procedure. Initially, the status of validator will be check whether he is active or not. Validator must be active like he can propose and validates the blocks. In next step, it will be verified that the validator does not initiate its exit already.

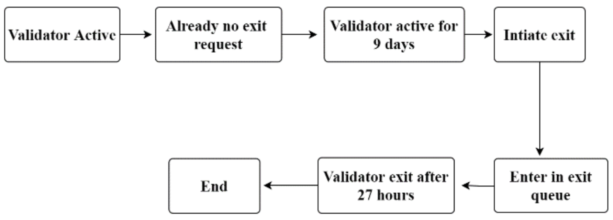


Fig. 3. Flow diagram of validator exit process.

Validator activation time period will be checked to determine how long validator remain active. It is necessary for validator to remain active for 9 days. Then in the next step validator can request for exit. After its request, validator will be considered as exit after 27 hours and can withdraw its ethers.

C. Algorithm

This algorithm shows that how procedure is taking place. All the conditions which are necessary shown in algorithm of Table I.

TABLE I. ALGORITHM OF VOLUNTARILY EXIT.

Algorithm: Voluntarily Exit process
Start **If** (validator is not active) Then return **Else**   **If** (validator initiated exit already)   Then return   **Else**   **If** (validator active time < 9days)   Then return   **Else**   Initiate Exit   Find exit queue epoch   Check queue churn limit   Set exit and withdrawable epoch End

V. FORMAL SPECIFICATION OF VALIDATOR EXIT

We will write formal specification of the system using PROMELA language. PROMELA is a basic language used in SPIN model checker. Its syntax resembles a lot to the C language. We will use this language to generate program graphs of the system. Each single part of code will be explained in this section.

A. Program Graph for Validator Exit Procedure

Program graph is an automaton which results from specification of the system written in PROMELA. Program is not like finite automata because in finite automaton final state is must but it is not the case in a program graph. Program graph can have or cannot have a final state. Therefore, program graph has five tuples:

- Set of locations: it is like state of automata
- Initial location: initial state or location in program graph.
- Set of conditions: conditions we define in program graph.
- Initial condition: initial condition define in program graph anywhere.
- Transition function: transition arrow in program graph labeled with conditions and messages.

Now we will write specification of the Python code of Beacon Chain for the validator voluntarily exit process. and will explain step by step in the form of blocks. Code is given in sequence, that is, initially data types then other blocks are system verification.

```
mtype={validator_active,check_active,check_exit,no_exit_request,still_active_status,check_active_time,validator_active_long_enough,signature_verified,start_exit,exit_initiated,compute_exit_queue_epoch,find_exit_queue_epoch,get_exit_queue_epoch,get_validator_inidces,check_withdrawable_epoch,exit_withdrawable_epoch_set,validator_voluntarily_leave,get_validator_indices}
int epoch=2548
int activation_epoch=10
int exit_epoch=2552
bit validator_exit_epoch
bit far_future_epoch
int computed_epoch=2548
int shard_committe_period=2048
int compulsory_activation_time
bit verify_signature
int compute_activation_exit
int active_validator_indices
int validator_churn
int validator_final_exit
int withdrawable_epoch
int max_seed_lookahead
int exit_queue_epoch
int  min_withdrawable_delay=256
int exit_queue_churn=4
```

All the variables are declared in this portion. The messages are kept mtype it is like enumeration type in C. Some variables are kept as bit either 1 or 0. Most of variables are int because of epochs we are considering.

1) Activation Check

```
Activation_check :
   Check_active
→activation_epoch<=epoch
→epoch<exit_epoch
→ validator_active
```

Initially it is checked whether validator is active or not. If validator is not active then it is not possible to initiate its voluntarily exit. If validator is active then we can move forward to next step of our voluntary exit process.

2) Check Exit Status

```
Exit_not_intiate :
→ check_exit
→validator_exit_epoch == far_future_epoch
- →validator_exit_epoch=validator_exit_epoch+1
→no_exit_request
→still_active_status
```

Now is the next step of checking whether a validator has initiate exit process already or not. If it is initiated already then, we do not need to initiate exit procedure and if it is not initiated already then, we will move to next step of our procedure.

3) Check Active time period

```
active_timeperiod :
→ check_active_time
→compulsory_activation_time=activation_epoch+shard_committe_period
→computed_epoch>=compulsory_activation_time
→validator_active_long_enough
→verify_signature==0
→verify_signature=verify_signature+1
→signature_verified
```

After checking active and exit status of validator status now we will check how long validator remain active because it is necessary for validator to remain active for 9 days at least which is total 2048 epochs which we called shard committee period

4) Initiate Exit

```
Intiate_Exit :
→ start_exit
→validator_exit_epoch != far_future_epoch
→exit_initiated
→compute_exit_queue_epoch
```

After all of those condition checked in pervious steps now voluntary exit is initiated. In this block code of we give initial steps in voluntarily exit and remaining processes will be explained in next blocks which are also sub part of exit initiation.

5) Exit Epoch checking

```
Exit_Queue_Epoch:
→find_exit_queue_epoch
→compute_activation_exit=epoch+1+Max_Seed_Lookahead
→exit_queue_epoch=exit_epoch+compute_activation_ext
→get_exit_queue_epoch
```

As exit process has been initiated therefore the first step of this process is to find the epoch in which validator will finally exit this process. In this block of code we verification code of finding that epoch in fist in first out queue(FIFO) because each validator which want to exit is must be in queue.

6) Validator Churn Limit

```
Validator_churn_limit:

→ get_validator_indices
→validator_churn=4
→exit_queue_churn>=validator_churn
→check_withdrawable_epoch
```

In this block code we consider the churn limit. Churn limit in Beacon chain is how many validators can exit in epoch when they get their turn in queue. We consider that in single epoch four validator can exit therefore four is churn limit of exit queue epoch.

7) Validator exit

```
Validator_exit_withdrawable_epoch:

→validator_final_exit=exit_queue_epoch

→withdrawable_epoch=validator_final_exit+
MIN_WITHDRAWABLE_DELAY
```

→exit_withdrawable_epoch_set
→validator_voluntarily_leave

This is the final block in which it is finalized in which epoch it will exit and it is also ensured that after 27 hours or we can say that after 256 epochs validator can withdraw his deposit amount and its rewards which validator got after validating blocks

B. *LTL Formulas for properties*

Linear temprol logic is used specify the properties. Properties are given in the form of description either it is safety or liveness and LTL formalize them into a formula. Here for validator exit procedure, we give one safety property and formalize that into LTL formula. Symbols we use in the formulas are

- *[]*: these square brackets consider complete path in program graph and all states of path are considered.
- <>: This symbol considers some states in complete path not all states in path.

1) Safety properties

Safety properties in which we do not want something bad to happen. For example, "two rains should not be on same track" So, in this way we never want in our property to something bad happen above example we gave for negative safety property. In positive safety property we want something good to happen for example "after yellow signal there should be green or red signal" So in such type of properties we want something happen in positive way. Because this is not a negated safety property.

Therefore, here we gave safety property according to the requirements. LTL formulas and their description are given below.

If a validator is not active or a validator initiate exit process already or a validator does not remain active for 9 days then the validator voluntary exit process will never be initiated.

LTL formula of this property is given below

---

ltlp1{[](<>(!(validator_active||no_exit_request||(computed_epoch>=compulsory_activation_time)))→ !(start_exit))}

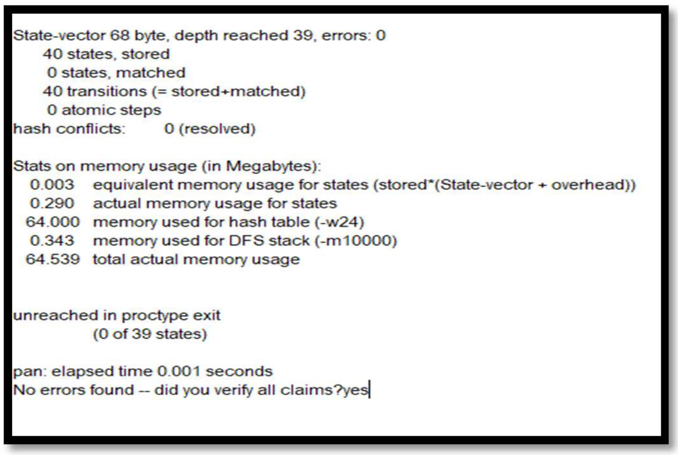
C. *Verification and results analysis through spin*

We perform verification through SPIN in following steps:

- Create program graph for the system of validator exit process.
- Define safety properties for the system to ensure its correctness.
- Formalize the safety properties into LTL formulas.
- Give LTL formula and program graph as input to the SPIN model checker.
- Model checker after verification will give results whether the properties are satisfied or not.

a. Verification results

Results or proof of correctness is given using SPIN model checker. All the properties are verified using model checker.



```
State-vector 68 byte, depth reached 39, errors: 0
     40 states, stored
      0 states, matched
     40 transitions (= stored+matched)
      0 atomic steps
hash conflicts:        0 (resolved)

Stats on memory usage (in Megabytes):
    0.003    equivalent memory usage for states (stored*(State-vector + overhead))
    0.290    actual memory usage for states
   64.000    memory used for hash table (-w24)
    0.343    memory used for DFS stack (-m10000)
   64.539    total actual memory usage


unreached in proctype exit
              (0 of 39 states)

pan: elapsed time 0.001 seconds
No errors found -- did you verify all claims?yes
```

Fig. 4. Proof of correctness.

As we can see from the previous figure, no error is found and all the properties are satisfied with our program graph for the validator voluntarily exit procedure.

VI. Conclusion

In this work, we have done formal verification of the validator voluntarily exit procedure in the Ethereum 2.0 Beacon Chain using the SPIN model checker. Initially, the validator exit process is briefly described and the algorithm is given for the process and then its formal specification is described in PROMELA language to generate the program graph through the SPIN model checker. The program graph depicts the working of the complete process of validator exit. After creating the program graph, we define safety properties using LTL formulas to ensure the correctness of the procedure. The program graph and LTL formulas are given as input to the model checker to perform formal verification using the model checking technique because it is very effective for verification purposes and it increases the correctness of the system. The results of our work show that our program graph satisfies the LTL formulas. Our research is useful for the Blockchain area, especially for Blockchain finance application safety purposes. In the future, we will extend this work by giving verification of the complete validator lifecycle which will also include validator activation.

References

[1] M. Nofer, P. Gomber, O. Hinz, and D. Schiereck, "Blockchain," *Bus. Inf. Syst. Eng.*, vol. 59, no. 3, pp. 183–187, 2017

[2] G. Wood, "Ethereum: a secure decentralised generalised transaction ledger," *Ethereum Proj. yellow Pap.*, vol. 151, pp. 1–32, 2014.

[3] M. A. Alturki, E. Li, D. Park, and B. Moore, "Verifying Gasper with Dynamic Validator Sets in Coq," 2020.

[4] M. A. Alturki, D. Bogdanas, C. Hathhorn, and D. Park, "An Executable K Model of Ethereum 2 . 0 Beacon Chain Phase 0 Specification".

[5] K. Palmskog, M. Gligoric, L. Pena, B. Moore, and G. Rosu, "Verification of Casper in the Coq Proof Assistant," 2018.

[6] E. Li, T. Serbanuta, D. Diaconescu, V. Zamfir, and G. Rosu, “Formalizing Correct-by-Construction Casper in Coq,” *IEEE Int. Conf. Blockchain Cryptocurrency, ICBC 2020*, pp. 26–28, 2020,

[7] F. Cassez, J. Fuller, and A. Asgaonkar, *Formal Verification of the Ethereum 2.0 Beacon Chain*, vol. 13243 LNCS. Springer International Publishing, 2022.

[8] V. Buterin and V. Griffith, “Casper the Friendly Finality Gadget,” pp. 1–10,2017

[9] M. Westerkamp and M. Diez, “Verilay: A Verifiable Proof of Stake Chain Relay,” *IEEE Int. Conf. Blockchain Cryptocurrency, ICBC 2022*, pp. 1–9, 2022.

[10] M. Neuder, D. J. Moroz, R. Rao, and D. C. Parkes, “Low-cost attacks on Ethereum 2.0 by sub-1/3 stakeholders,” pp. 1–11, 2021, [Online]. Available: http://arxiv.org/abs/2102.02247

[11] D. Ramos and G. Zanko, “Ethereum 2.0–A Review of the Causes and Consequences of the Upcoming Update to Ethereum’s Mainnet,” *Mobileyourlife.Com*, 2022, [Online]. Available: https://www.mobileyourlife.com/s/ETH_2_SciPaper_2.pdf

[12] S. Tucci-Piergiovanni, “Keynote: Blockchain consensus protocols, from Bitcoin to Ethereum 2.0,” *2022 IEEE Int. Conf. Pervasive Comput. Commun. Work. other Affil. Events (PerCom Work.*, pp. 1–1, 2022.

[13] M. Staderini, C. Palli, and A. Bondavalli, “Classification of Ethereum Vulnerabilities and their Propagations,” *2020 2nd Int. Conf. Blockchain Comput. Appl. BCCA 2020*, pp. 44–51, 2020,

[14] S. Patel, A. Sahoo, B. K. Mohanta, S. S. Panda, and D. Jena, “DAuth: A Decentralized Web Authentication System using Ethereum based Blockchain,” *Proc. - Int. Conf. Vis. Towar. Emerg. Trends Commun. Networking, ViTECoN 2019*, pp. 1–5, 2019.

[15] Q. Xu, Z. He, Z. Li, and M. Xiao, “Building an Ethereum-Based Decentralized Smart Home System,” *Proc. Int. Conf. Parallel Distrib. Syst. - ICPADS*, vol. 2018-Decem, pp. 1004–1009, 2019.

[16] K. B. Muthe, “DECENTRANET - AN ETHEREUM , PROXY RE-ENCRYPTION,” pp. 1–5, 2020.

[17] Z. Jiang, Z. Zheng, K. Chen, X. Luo, X. Tang, and Y. Li, “Exploring Smart Contract Recommendation: Towards Efficient Blockchain Development,” *IEEE Trans. Serv. Comput.*, vol. 16, no. 3, pp. 1822–1832, 2023.

[18] B. S. Liya, S. Pritam, S. Rohit Krishna, and K. Navin, “Decentralized E-Commerce Platform Implemented using Smart Contracts,” *Proc. - 2023 3rd Int. Conf. Smart Data Intell. ICSMDI 2023*, pp. 23–27, 2023.

[19] T. Liakh, R. Sorokin, D. Akifev, S. Patil, and V. Vyatkin, “Formal model of IEC 61499 execution trace in,” *2022 IEEE 20th Int. Conf. Ind. Informatics*, pp. 588–593, 2022.

[20] A. Amirat and A. Menasria, “Automatic Generation of PROMELA Code from Sequence Diagram with Imbricate Combined Fragments,” pp. 1–6, 2012.

[21] S. K. N, “Modeling and Verification of Timed Automaton Based Hybrid Systems Using Spin Model Checker,” 2016.

[22] P. R. Gliick, B. Labs, and M. Hill, “Using SPIN1 Model Checking for Flight Software,” pp. 105–113, 1978.

[23] K. Bhargavan, D. Obradovic, and C. A. Gunter, “Formal Verification of Standards for Distance Vector Routing Protocols,” vol. 49, no. 4, pp. 538–576, 2002.

[24] S. Latif, A. Rehman, and N. A. Zafar, "Blockchain and IoT Based Formal Model of Smart Waste Management System Using TLA+," *2019 International Conference on Frontiers of Information Technology (FIT)*, Islamabad, Pakistan, pp. 304-3045, 2019

[25] M. Iqbal, N. A. Zafar and E. H. Alkhammash, "Formally Identifying COVID-19 Patients for Providing Medical Services using Drones," *2021 International Conference of Women in Data Science at Taif University (WiDSTaif)*, Taif, Saudi Arabia, pp. 1-6, 2021.

[26] S. Latif, H. Afzaal and N. A. Zafar, "Intelligent traffic monitoring and guidance system for smart city," *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, Sukkur, Pakistan, pp. 1-6, 2018

[27] A. Rehman, S. Latif and N. A. Zafar, "Automata Based Railway Gate Control System at Level Crossing," *2019 International Conference on Communication Technologies (ComTech)*, Rawalpindi, Pakistan, pp. 30-35, 2019

[28] H. Afzaal, M. Imran, and M. Umar." Formal verification of fraud-resilience in a crowdsourcing consensus protocol." *Computers & Security* 131 (2023): 103290.

[29] H. Afzaal, M. Imran, and M. Umar “Formal verification persistence and liveness in the trust-based blockchain crowdsourcing consensus protocol." *Computer Communications* 192 (2022): 384- 401.