# Formally Verifying Smart Contracts Using Theorem Proving

**M. Taimoor Zaeem, Dr. Adnan Rashid**
School of Electrical Engineering and Computer Science
National University of Sciences and Technology
Islamabad
mzaeem.bscs20seecs@seecs.edu.pk
adnan.rashid@seecs.edu.pk

## 1 Abstract

**Blockchain technology is recently being used in all sorts of applications. With the advent of Web3, blockchain technology is being used now in web applications. A major reason for this is smart contracts that enable us to implement decentralized applications in trust-less environments. Along with its adoption, attacks exploiting smart contract vulnerabilities are inevitably growing. To counter these attacks and avoid security breaches, several approaches have been explored such as documenting vulnerabilities, simulations or model checking using formal verification.**

**However, these approaches are inadequate to capture the blockchain and users' behavior properties. We propose to use higher-order-logic theorem proving as an alternative strategy to formally model and verify smart contract behavior in its execution environment. We will verify smart contract properties by proving theorems on the formal model.**

## 2 Introduction

**Blockchain technology can be defined as an ever-growing list of transactions, grouped in blocks, hence the name. Its key features include distributed consensus, cryptographic signatures, persistence, and execution in a trust-less environment. The advent of smart contracts has extended the functionalities of blockchain and is now being used in Web3, supply-chain management, digital wallet apps etc.**

**A smart contract is a piece of code that lives on the blockchain client. It is triggered and executed by specific transactions. The result of a smart contract execution changes global state, and the transaction is then stored in the blockchain. In this way, smart contracts can be used to implement decentralized applications. However, smart contracts – since they are a piece of code – are vulnerable to many security threats and attacks. As smart contracts operate in a distributed execution environment, taking adequate security measures is a challenge to this day.**

**Many approaches have been taken to analyze and test security measures for smart contracts. To guarantee correctness, many experts resort to using Formal Methods. Formal methods are a way to model a system mathematically and then use verification to prove properties related to the system. Traditional methods such as simulations aren't adequate to guarantee safety because these methods may skip some critical test cases and model checking has scalability issues. Theorem Proving is a technique where we model the system using mathematical axioms and propositions that guarantee completeness thus is powerful technique in analyzing a system.**

## 3 Literature Review

**In this section we provide the most relevant contributions and related work for the analysis of smart contracts. Different techniques such as model checking, program verification using static analysis, theorem proving, symbolic and concolic execution, runtime verification etc.**

**Program Verification techniques such as in [1] are used to verify the bytecode correctness of smart contracts, however in context of the blockchain, it fails to show vulnerabilities such as other entities in the blockchain.**

**Other techniques like symbolic and concolic execution [2] and runtime verification tools [3] also fail as they are good techniques to find vulnerabilities but cannot demonstrate correctness.**

**Model Checking [4] [5] [6] [7] on the other hand is a much better technique that demonstrates correctness and could also find vulnerabilities. They use automated methods such as Linear Temporal Logic. However, it runs into problems as the number of states and variables grow. This is known as the state-space explosion problem. Hence not a scalable technique.**

**Theorem Proving techniques have been [8] used for verifying low level formal semantics of the underlying programming language however theorem proving has not been used for modelling a blockchain system including smart contracts before.**

**The most relevant paper to our project is [9]. This paper uses model checking technique to demonstrate user and blockchain interaction. However, for reasons discussed above, we will improve the shortcoming of this paper by modelling the blockchain and users in an Interactive Theorem Prover and prove behavioral properties. Our modelling approach taken for this would be followed from [10].**

## 4 Formal Specification

**We have modelled the system as follows:**

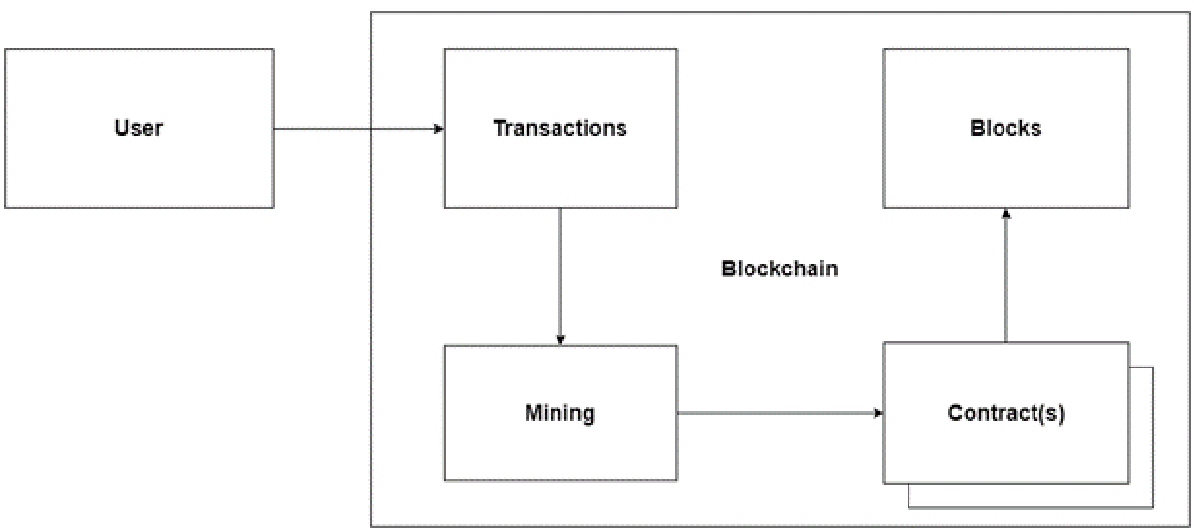


Figure 1: System Architecture

**Each block in the above diagram represent a process in the blockchain. Each process takes some time to perform its operation in the system. The time taken by each process is described using the following identifiers.**

- $t_r$: Transaction retrieval time
- $t_m$: Transaction mining time

- $t_c$: Transaction execution time in Smart Contract
- $t_b$: Transaction addition to blockchain time

**The transaction execution process in our system should be as follows. The `user` process initiates the transaction. The `transactions` process sends the transaction to `mining` process after `tr` time units. The `mining` process sends the transaction to `contract` process after `tm` time units. The `contract` process waits for `tc` time units before executing the transaction, then sends the transaction to the `blocks` process. The `blocks` process adds the transaction result to the blockchain after `tb` time units.**

### 4.1 Type Definitions

**The transaction consists of a user object, which is a tuple of two items, `user_alias` and `user_address`. The smart contract consists of register `reg` that is a list of user objects. These are defined as follows.**

```
user_alias: alias + no_alias
user_address: address + no_address
user: user_alias * user_address
reg: user list
```

## 5 Formal Model

### 5.1 Transactions

**Here goes detailed explanation of the definition. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis volutpat posuere velit nec dictum. Praesent at ex consectetur, porttitor mi at, molestie purus. Phasellus aliquam massa leo, dignissim bibendum felis lobortis nec. Suspendisse commodo pharetra lorem, eget commodo lacus congue ac. Nunc non ex ac odio vehicula sollicitudin ac facilisis urna. Cras varius nibh id magna dapibus, ac feugiat leo pellentesque. Nunc sed risus vel lectus mattis rhoncus. Proin lobortis at felis id ornare. Nullam ut augue justo. Ut et accumsan quam. Praesent euismod odio a nisi pellentesque auctor. Lorem ipsum dolor sit amet, consectetur adipiscing elit.**

**Etiam at condimentum urna. Proin est urna, euismod vel scelerisque a, tristique sed risus. Ut nec tortor tincidunt nibh convallis scelerisque eget et felis. Integer vestibulum rhoncus lorem in cursus. Cras dapibus non massa vitae ornare. Integer tempor nisi non sapien viverra, sit amet mollis ligula rhoncus. Mauris auctor at odio ut tincidunt. Integer pretium nulla nibh. Suspendisse potenti. Cras gravida pulvinar interdum. Aliquam at egestas libero. Mauris consectetur lorem erat. Aenean laoreet lorem semper velit suscipit, ac finibus lacus posuere. Etiam porttitor sapien quis purus volutpat scelerisque. Aenean ac diam sed purus efficitur laoreet eu id lectus. Nunc ultricies eros metus, sed mattis risus accumsan et.**

```
TRANSACTIONS tr dtr pending_tx mine =
  !t. (if pending_tx t <> [] then
        (if dtr t = 0 then
          (mine t = LAST_TX (pending_tx t)) /\
          (dtr (t+1) = tr)  /\
          (pending_tx (t+1) = RM_LAST_TX (pending_tx t))
        else
          (mine t = no_data)      /\
          (dtr (t+1) = dtr t - 1) /\
          (pending_tx (t+1) = pending_tx t))
      else
        (mine t = no_data) /\
        (dtr (t+1) = tr)   /\
        (pending_tx (t+1) = pending_tx t))
```

### 5.2 Mining

**Here goes detailed explanation of the definition. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis volutpat posuere velit nec dictum. Praesent at ex consectetur, porttitor mi at, molestie purus. Phasellus aliquam**

**massa leo, dignissim bibendum felis lobortis nec. Suspendisse commodo pharetra lorem, eget commodo lacus congue ac. Nunc non ex ac odio vehicula sollicitudin ac facilisis urna. Cras varius nibh id magna dapibus, ac feugiat leo pellentesque. Nunc sed risus vel lectus mattis rhoncus. Proin lobortis at felis id ornare. Nullam ut augue justo. Ut et accumsan quam. Praesent euismod odio a nisi pellentesque auctor. Lorem ipsum dolor sit amet, consectetur adipiscing elit.**

**Etiam at condimentum urna. Proin est urna, euismod vel scelerisque a, tristique sed risus. Ut nec tortor tincidunt nibh convallis scelerisque eget et felis. Integer vestibulum rhoncus lorem in cursus. Cras dapibus non massa vitae ornare. Integer tempor nisi non sapien viverra, sit amet mollis ligula rhoncus. Mauris auctor at odio ut tincidunt. Integer pretium nulla nibh. Suspendisse potenti. Cras gravida pulvinar interdum. Aliquam at egestas libero. Mauris consectetur lorem erat. Aenean laoreet lorem semper velit suscipit, ac finibus lacus posuere. Etiam porttitor sapien quis purus volutpat scelerisque. Aenean ac diam sed purus efficitur laoreet eu id lectus. Nunc ultricies eros metus, sed mattis risus accumsan et.**

```
MINING tm dtm mine sc_tx =
!t. (if  (mine t = user_data) then
      (if (dtm t = 0) then
        (sc_tx t = user_data) /\
        (dtm (t+1) = tm)
      else
        (sc_tx t = no_data) /\ (dtm (t+1) = dtm t - 1))
  else
    (sc_tx t = no_data) /\ (dtm (t+1) = tm))
```

### 5.3 Smart Contract

**Here goes detailed explanation of the definition. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis volutpat posuere velit nec dictum. Praesent at ex consectetur, porttitor mi at, molestie purus. Phasellus aliquam massa leo, dignissim bibendum felis lobortis nec. Suspendisse commodo pharetra lorem, eget commodo lacus congue ac. Nunc non ex ac odio vehicula sollicitudin ac facilisis urna. Cras varius nibh id magna dapibus, ac feugiat leo pellentesque. Nunc sed risus vel lectus mattis rhoncus. Proin lobortis at felis id ornare. Nullam ut augue justo. Ut et accumsan quam. Praesent euismod odio a nisi pellentesque auctor. Lorem ipsum dolor sit amet, consectetur adipiscing elit.**

**Etiam at condimentum urna. Proin est urna, euismod vel scelerisque a, tristique sed risus. Ut nec tortor tincidunt nibh convallis scelerisque eget et felis. Integer vestibulum rhoncus lorem in cursus. Cras dapibus non massa vitae ornare. Integer tempor nisi non sapien viverra, sit amet mollis ligula rhoncus. Mauris auctor at odio ut tincidunt. Integer pretium nulla nibh. Suspendisse potenti. Cras gravida pulvinar interdum. Aliquam at egestas libero. Mauris consectetur lorem erat. Aenean laoreet lorem semper velit suscipit, ac finibus lacus posuere. Etiam porttitor sapien quis purus volutpat scelerisque. Aenean ac diam sed purus efficitur laoreet eu id lectus. Nunc ultricies eros metus, sed mattis risus accumsan et.**

```
REGISTER_CONTRACT tc dtc sc_tx block_tx reg notify success =
!t. if (notify t = F) /\ (sc_tx t = user_data) then
        (if (dtc t = 0) then
            (if FIND_USER (user_data, reg t) then
              (block_tx t = (user_data,F)) /\
              (success t = F) /\ (notify t = T) /\
              (dtc (t+1) = tc)
            else
              (block_tx t = (user_data,T)) /\
              (reg (t+1) = user_data::(reg t)) /\
              (success t = T) /\ (notify t = T) /\
              (dtc (t+1) = tc))
        else
          (dtc (t+1) = dtc t - 1))
    else
      (dtc (t+1) = tc)
```

### 5.4 Blocks

**Here goes detailed explanation of the definition. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis volutpat posuere velit nec dictum. Praesent at ex consectetur, porttitor mi at, molestie purus. Phasellus aliquam massa leo, dignissim bibendum felis lobortis nec. Suspendisse commodo pharetra lorem, eget commodo lacus congue ac. Nunc non ex ac odio vehicula sollicitudin ac facilisis urna. Cras varius nibh id magna dapibus, ac feugiat leo pellentesque. Nunc sed risus vel lectus mattis rhoncus. Proin lobortis at felis id ornare. Nullam ut augue justo. Ut et accumsan quam. Praesent euismod odio a nisi pellentesque auctor. Lorem ipsum dolor sit amet, consectetur adipiscing elit.**

**Etiam at condimentum urna. Proin est urna, euismod vel scelerisque a, tristique sed risus. Ut nec tortor tincidunt nibh convallis scelerisque eget et felis. Integer vestibulum rhoncus lorem in cursus. Cras dapibus non massa vitae ornare. Integer tempor nisi non sapien viverra, sit amet mollis ligula rhoncus. Mauris auctor at odio ut tincidunt. Integer pretium nulla nibh. Suspendisse potenti. Cras gravida pulvinar interdum. Aliquam at egestas libero. Mauris consectetur lorem erat. Aenean laoreet lorem semper velit suscipit, ac finibus lacus posuere. Etiam porttitor sapien quis purus volutpat scelerisque. Aenean ac diam sed purus efficitur laoreet eu id lectus. Nunc ultricies eros metus, sed mattis risus accumsan et.**

```
BLOCKS tb dtb block_tx blocks =
!t. if (FST (block_tx t) = user_data) then
      (if (dtb t = 0) then
        (blocks t = (user_data,SND (block_tx (t-1)))::(blocks (t-1))) /\
        (dtb (t+1) = tb)
      else
        (dtb (t+1) = dtb t - 1))
    else
       (dtb (t+1) = tb)
```

### 5.5 Blockchain

**Here goes detailed explanation of the definition. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis volutpat posuere velit nec dictum. Praesent at ex consectetur, porttitor mi at, molestie purus. Phasellus aliquam massa leo, dignissim bibendum felis lobortis nec. Suspendisse commodo pharetra lorem, eget commodo lacus congue ac. Nunc non ex ac odio vehicula sollicitudin ac facilisis urna. Cras varius nibh id magna dapibus, ac feugiat leo pellentesque. Nunc sed risus vel lectus mattis rhoncus. Proin lobortis at felis id ornare. Nullam ut augue justo. Ut et accumsan quam. Praesent euismod odio a nisi pellentesque auctor. Lorem ipsum dolor sit amet, consectetur adipiscing elit.**

**Etiam at condimentum urna. Proin est urna, euismod vel scelerisque a, tristique sed risus. Ut nec tortor tincidunt nibh convallis scelerisque eget et felis. Integer vestibulum rhoncus lorem in cursus. Cras dapibus non massa vitae ornare. Integer tempor nisi non sapien viverra, sit amet mollis ligula rhoncus. Mauris auctor at odio ut tincidunt. Integer pretium nulla nibh. Suspendisse potenti. Cras gravida pulvinar interdum. Aliquam at egestas libero. Mauris consectetur lorem erat. Aenean laoreet lorem semper velit suscipit, ac finibus lacus posuere. Etiam porttitor sapien quis purus volutpat scelerisque. Aenean ac diam sed purus efficitur laoreet eu id lectus. Nunc ultricies eros metus, sed mattis risus accumsan et.**

```
BLOCKCHAIN tr dtr pending_tx notify tm dtm mine tc dtc sc_tx
reg  success tb dtb block_tx blocks =
  ((TRANSACTIONS tr dtr pending_tx mine) /\
   (MINING tm dtm mine sc_tx) /\
   (REGISTER_CONTRACT tc dtc sc_tx block_tx reg notify success) /\
   (BLOCKS tb dtb block_tx blocks))
```

## 6 Conclusion and Future Work

**The advent of smart contracts has extended the functionalities of blockchain and is now being used in Web3, supply-chain management, digital wallet apps etc. A smart contract is a piece of code that lives on the blockchain client. It is triggered and executed by specific transactions. However, smart contracts are vulnerable to many**

**security threats and attacks. As smart contracts operate in a distributed execution environment, taking adequate security measures is a complex challenge.**

**Many approaches have been taken to analyze security measures for smart contracts. To guarantee correctness, many experts tend to resort to using Formal Methods. Formal methods are a way to model a system mathematically and then use Formal Verification to prove properties of interest of the system. Theorem Proving is a technique where we model the system using mathematical axioms and propositions that guarantee completeness thus is powerful technique in analyzing a system. The formal model provided in this work could be used in the future by other researchers to prove other blockchain and smart contracts properties.**

**The source our work is available here[11].**

## Acknowledgements

## References

[1] Wolfgang Ahrendt, Richard Bubel, Joshua Ellul, Gordon J. Pace, Raúl Pardo, Vincent Rebiscoul, and Gerardo Schneider. Verification of Smart Contract Business Logic. In *8th International Conference on Fundamentals of Software Engineering (FSEN)*, volume LNCS-11761 of *Fundamentals of Software Engineering*, pages 228–243, Tehran, Iran, May 2019. Springer International Publishing. Part 6: Program Analysis.

[2] Christof Ferreira Torres, Julian Schütte, and Radu State. Osiris: Hunting for integer bugs in ethereum smart contracts. In *Proceedings of the 34th Annual Computer Security Applications Conference*, ACSAC '18, page 664–676, New York, NY, USA, 2018. Association for Computing Machinery.

[3] Joshua Ellul and Gordon J. Pace. Runtime verification of ethereum smart contracts. In *2018 14th European Dependable Computing Conference (EDCC)*, pages 158–163, 2018.

[4] Wonhong Nam and Hyunyoung Kil. Formal verification of blockchain smart contracts via atl model checking. *IEEE Access*, 10:8151–8162, 2022.

[5] Giancarlo Bigi, Andrea Bracciali, Giovanni Meacci, and Emilio Tuosto. *Validation of Decentralised Smart Contracts Through Game Theory and Formal Methods*, pages 142–161. Springer International Publishing, Cham, 2015.

[6] Xiaomin Bai, Zijing Cheng, Zhangbo Duan, and Kai Hu. Formal modeling and verification of smart contracts. In *Proceedings of the 2018 7th International Conference on Software and Computer Applications*, ICSCA '18, page 322–326, New York, NY, USA, 2018. Association for Computing Machinery.

[7] Ikram Garfatta, Kaïs Klai, Mohamed Graïet, and Walid Gaaloul. Model checking of vulnerabilities in smart contracts: a solidity-to-cpn approach. In *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, SAC '22, page 316–325, New York, NY, USA, 2022. Association for Computing Machinery.

[8] Jiao Jiao, Shuanglong Kan, Shang-Wei Lin, David Sanan, Yang Liu, and Jun Sun. Semantic understanding of smart contracts: Executable operational semantics of solidity. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1695–1712, 2020.

[9] Tesnim Abdellatif and Kei-Leo Brousmiche. Formal verification of smart contracts based on users and blockchain behaviors models. In *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–5, 2018.

[10] Osman Hasan and Sofiène Tahar. Performance analysis and functional verification of the stop-and-wait protocol in hol. *Journal of Automated Reasoning*, 42:1–33, 2008.

[11] Muhammad Taimoor Zaeen. Smart contract verification, 2024. Accessed: 09 May 2024.