Lecture Notes in Networks and Systems 320

Javier Prieto
Alberto Partida
Paulo Leitão
António Pinto *Editors*

Blockchain and Applications

3rd International Congress

Springer

Lecture Notes in Networks and Systems

Volume 320

The series "Lecture Notes in Networks and Systems" publishes the latest developments in Networks and Systems—quickly, informally and with high quality. Original research reported in proceedings and post-proceedings represents the core of LNNS.

Volumes published in LNNS embrace all aspects and subfields of, as well as new challenges in, Networks and Systems.

The series contains proceedings and edited volumes in systems and networks, spanning the areas of Cyber-Physical Systems, Autonomous Systems, Sensor Networks, Control Systems, Energy Systems, Automotive Systems, Biological Systems, Vehicular Networking and Connected Vehicles, Aerospace Systems, Automation, Manufacturing, Smart Grids, Nonlinear Systems, Power Systems, Robotics, Social Systems, Economic Systems and other. Of particular value to both the contributors and the readership are the short publication timeframe and the world-wide distribution and exposure which enable both a wide and rapid dissemination of research output.

The series covers the theory, applications, and perspectives on the state of the art and future developments relevant to systems and networks, decision making, control, complex processes and related areas, as embedded in the fields of interdisciplinary and applied sciences, engineering, computer science, physics, economics, social, and life sciences, as well as the paradigms and methodologies behind them.

Indexed by SCOPUS, INSPEC, WTI Frankfurt eG, zbMATH, SCImago.

All books published in the series are submitted for consideration in Web of Science.

More information about this series at http://www.springer.com/series/15179

Javier Prieto · Alberto Partida ·
Paulo Leitão · António Pinto
Editors

Blockchain and Applications

3rd International Congress

Springer

Editors
Javier Prieto
Building Edificio I+D+i, Room 24.1
University of Salamanca
Salamanca, Salamanca, Spain

Paulo Leitão
Research Centre in Digitalization
Instituto Politécnico de Bragança
Bragança, Portugal

Alberto Partida
Universidad Rey Juan Carlos
Móstoles, Madrid, Spain

António Pinto
ESTG, Instituto Politécnico do Porto
Felgueiras, Portugal

ISSN 2367-3370 ISSN 2367-3389 (electronic)
Lecture Notes in Networks and Systems
ISBN 978-3-030-86161-2 ISBN 978-3-030-86162-9 (eBook)
https://doi.org/10.1007/978-3-030-86162-9

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

The 3rd International Congress on Blockchain and Applications 2021 will be held in Salamanca from 6 to 8 of October. This annual congress will reunite blockchain and artificial intelligence (AI) researchers, who will share ideas, projects, lectures, and advances associated with those technologies and their application domains.

Among the scientific community, blockchain and AI are seen as a promising combination that will transform the production and manufacturing industry, media, finance, insurance, e-government, etc. Nevertheless, there is no consensus with schemes or best practices that would specify how blockchain and AI should be used together. Combining blockchain mechanisms and artificial intelligence is still a particularly challenging task.

The BLOCKCHAIN'21 congress is devoted to promoting the investigation of cutting-edge blockchain technology, to exploring the latest ideas, innovations, guidelines, theories, models, technologies, applications and tools of blockchain and AI for the industry, and to identifying critical issues and challenges those researchers and practitioner must deal with in the future research. We want to offer researchers and practitioners the opportunity to work on promising lines of research and to publish their developments in this area.

The technical program has been diverse and of high quality, and it focused on contributions to both, well-established and evolving areas of research. More than 44 papers have been submitted to 38 from over 20 different countries (Canada, France, Germany, India, Ireland, Italy, Jordan, Luxembourg, Malaysia, Malta, Morocco, Netherlands, Oman, Portugal, Slovenia, Spain, Sweden, United Arab Emirates, and USA).

We would like to thank all the contributing authors, the members of the Program Committee, the sponsors (IBM, Indra, EurAI, AEPIA, AFIA, APPIA, and AIR Institute), and the Organizing Committee for their hard and highly valuable work. We are especially grateful for the funding supporting by project “XAI - XAI - Sistemas Inteligentes Auto Explicativos creados con Módulos de Mezcla de Expertos,” ID SA082P20, financed by Junta Castilla y León, Consejería de Educación, and FEDER funds. Their work contributed to the success of the BLOCKCHAIN'21 event and, finally, the Local Organization Members and the

Program Committee Members for their hard work, which was essential for the success of BLOCKCHAIN'21.

Javier Prieto
Alberto Partida
Paulo Leitão
António Pinto

Organization

General chair

Javier Prieto Tejedor	University of Salamanca, Spain, and AIR Institute, Spain

Advisory Board

Ashok Kumar Das	IIIT Hyderabad, India
Abdelhakim Hafid	Université de Montréal, Canada
António Pinto	Instituto Politécnico do Porto, Portugal

Program Committee Chairs

Alberto Partida	Universidad Rey Juan Carlos, Spain
Paulo Leitao	Technical Institute of Bragança, Portugal

Organizing Committee

Juan M. Corchado Rodríguez	University of Salamanca, Spain, and AIR Institute, Spain
Javier Prieto Tejedor	University of Salamanca, Spain, and AIR Institute, Spain
Roberto Casado Vara	University of Salamanca, Spain
Fernando De la Prieta	University of Salamanca, Spain
Sara Rodríguez González	University of Salamanca, Spain
Pablo Chamoso Santos	University of Salamanca, Spain
Belén Pérez Lancho	University of Salamanca, Spain
Ana Belén Gil González	University of Salamanca, Spain
Ana De Luis Reboredo	University of Salamanca, Spain
Angélica González Arrieta	University of Salamanca, Spain

Emilio S. Corchado Rodríguez	University of Salamanca, Spain
Angel Luis Sánchez Lázaro	University of Salamanca, Spain
Alfonso González Briones	University Complutense of Madrid, Spain
Yeray Mezquita Martín	University of Salamanca, Spain
Javier J. Martín Limorti	University of Salamanca, Spain
Alberto Rivas Camacho	University of Salamanca, Spain
Ines Sitton Candanedo	University of Salamanca, Spain
Elena Hernández Nieves	University of Salamanca, Spain
Beatriz Bellido	University of Salamanca, Spain
María Alonso	University of Salamanca, Spain
Diego Valdeolmillos	AIR Institute, Spain
Sergio Marquez	University of Salamanca, Spain
Marta Plaza Hernández	University of Salamanca, Spain
Guillermo Hernández González	AIR Institute, Spain
Ricardo S. Alonso Rincón	University of Salamanca, Spain
Javier Parra	University of Salamanca, Spain

Program Committee

Regio A. Michelin	University of New South Wales, Australia
Mo Adda	University of Portsmouth, UK
Rishav Agarwal	University of Waterloo, Canada
Imtiaz Ahmad Akhtar	Higher Colleges of Technology, Sweden
Sami Albouq	Islamic University of Madinah, Saudi Arabia
Ricardo Alonso	AIR Institute, Spain
Alejandro Alfonso Fernández	DigitelTS, Spain
Diego Andina	Universidad Politécnica de Madrid, Spain
Artem Barger	IBM, Israel
Francisco Luis Benítez Martínez	University of Granada, Spain
Javier Bermejo Higuera	Universidad Internacional de La Rioja, Spain
Bill Buchanan	Napier University, UK
Roben C. Lunardi	IFRS, Brazil
Roberto Casado Vara	University of Salamanca, Spain
Arnaud Castelltort	Montpellier, France
Giovanni Ciatto	University of Bologna, Italy
Victor Cook	University of Central Florida, EE.UU.
Manuel E. Correia	CRACS/INESC TEC; DCC/FCUP, Portugal
Gaby Dagher	Boise State University, Idaho, USA
Ashok Kumar Das	International Institute of Information Technology, India
Pankaj Dayama	IBM, India

Andreea-Elena Drăgnoiu	University of Bucharest, Romania
Enrique De la Cal Marín	University of Oviedo, Spain
Josep Lluis De La Rosa	TECNIO Centre EASY Innovation, UdG, Spain
Monika di Angelo	TU Wien, Austria
Roberto Di Pietro	Hamad Bin Khalifa University, College of Science and Engineering, Qatar
Joshua Ellul	University of Malta, Malta
Enes Erdin	Florida International University, EE.UU.
Xinxin Fan	IoTeX, India
Manuel J. Fernandez	University of Seville, Spain
Iñaki Fernández	Université de Lorraine, France
Christof Ferreira Torres	Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg
Nikos Fotiou	Athens University of Economics and Business, Greece
Paula Fraga Lamas	University of A Coruña, Spain
Muriel Franco	University of Zurich, EE.UU.
Felix Freitag	Universitat Politècnica de Catalunya, Spain
Shahin Gheitanchi	Altera Corporation, EE.UU.
Raffaele Giaffreda	CREATE-NET, Italy
Mongetro Goint	Université Le Havre Normandie, France
Hélder Gomes	Escola Superior de Tecnologia e Gestão de Águeda, Universidade de Aveiro, Portugal
Dhrubajyoti Goswami	Concordia University, Canada
Ram Govind	Indian Computer Emergency Response Team, Ministry of Electronics and IT, India
Volker Gruhn	Universität Duisburg-Essen, Germany
Hao Guo	Northwestern Polytechnical University, China
Abdelatif Hafid	University of Montreal, Canada
Yahya Hassanzadeh-Nazarabadi	Ferdowsi University of Mashhad, Irán
Farzin Houshmand	University of California, EE.UU.
Qin Hu	George Washington University, EE.UU.
Maria Visitación Hurtado	University of Granada, Spain
Hans-Arno Jacobsen	University of Toronto, Canada
Marc Jansen	University of Applied Sciences Ruhr West, Germany
Eder John Scheid	University of Zurich, Switzerland
Raja Jurdak	QUT, Australia
Salil Kanhere	University of New South Wales, Australia
Christos Karapapas	Athens University of Economics and Business, Greece
Samuel Karumba	UNSW, Australia
Denisa Kera	Asia Research Institute (STS Cluster), Australia

Mohamed Laarabi	Mohammadia School of Engineering Rabat, Morocco
Oscar Lage	TECNALIA, Spain
Chhagan Lal	University of Padova, Italy
Xabier Larrucea	TECNALIA, Spain
Anne Laurent	LIRMM – UM, France
Xianfeng Li	Macau University of Science and Technology, China
Sotirios Liaskos	School of IT, York University, Canada
Alexander Lipton	Massachusetts Institute of Technology, EE.UU.
Fengji Luo	The University of Sydney, Australia
João Paulo Magalhaes	ESTGF, Porto Polytechnic Institute, Portugal
Yacov Manevich	IBM, Israel
Stefano Mariani	Università degli Studi di Modena e Reggio Emilia, Italy
Luis Martínez	University of Jaén, Spain
Imran Memon	Zhejiang University, China
Suat Mercan	Florida International University, EE.UU.
Yeray Mezquita	University of Salamanca, Spain
Juan Jose Morillas Guerrero	Universidad Politécnica de Madrid, Spain
Imtiaz Muhammad Anas	Boston University, EE.UU.
Daniel Jesus Munoz Guerra	University of Malaga, Spain
Ahmed Nadeem	King Saud University, Saudi Arabia
Malaw Ndiaye	Université Cheikh Anta Diop, Senegal
Mark Nejad	University of Delaware, EE.UU.
Mariusz Nowostawski	Norwegian University of Science and Technology, Norway
Andrea Omicini	Alma Mater Studiorum–Università di Bologna, Italy
Kazumasa Omote	University of Tsukuba, Japan
Arindam Pal	Data61, CSIRO, Australia
Rafael Pastor Vargas	UNED, Spain
Miguel Pincheira	Fondazione Bruno Kessler, Italy
António Pinto	ESTG, P.Porto, Portugal
Pedro Pinto	Polytechnic Institute of Viana do Castelo, Portugal
Steven Platt	Universitat Pompeu Fabra, Spain
Matthias Pohl	Otto-von-Guericke-Universität Magdeburg, Germany
Javier Prieto	University of Salamanca, Spain
Yuansong Qiao	Athlone Institute of Technology, Ireland
Bruno Rodrigues	University of Zurich, Switzerland
David Rosado	University of Castilla-La Mancha, Spain
Grigore Rosu	University of Illinois at Urbana-Champaign, EE. UU.

Gernot Salzer	Vienna University of Technology, Austria
Georgios Samakovitis	University of Greenwich, UK
Altino Sampaio	Instituto Politécnico do Porto, Escola Superior de Tecnologia e Gestão de Felgueiras, Portugal
Elio San Cristóbal Ruiz	UNED, Spain
Ricardo Santos	ESTG/IPP, Portugal
Nuria Serrano	AIR Institute, Spain
Wazen Shbair	University of Luxembourg, SnT, Luxembourg
Pradip Sharma	University of Aberdeen, UK
Chien-Chung Shen	University of Delaware, EE.UU.
Ajay Kumar Shrestha	University of Saskatchewan, Canada
Mark Staples	CSIRO, Australia
Radu State	University of Luxembourg, Luxembourg
Burkhard Stiller	University of Zurich, Switzerland
Wilhelm Stork	Karlsruhe Institute of Technology, Germany
Marko Suvajdzic	University of Florida, EE.UU.
Stefan Tai	TU Berlin, Germany
Chamseddine Talhi	École de Technologie Supérieure, Canada
Teik Guan Tan	Singapore University of Technology and Design, Malaysia
Ege Tekiner	Florida International University, EE.UU.
Subhasis Thakur	National University of Ireland, Galway, Ireland
Llanos Tobarra	UNED, Spain
Aitor Urbieta	IK4-Ikerlan Technology Research Centre, Spain
Julita Vassileva	University of Saskatchewan, Canada
Massimo Vecchio	Fondazione Bruno Kessler, Italy
Eduardo Vega-Fuentes	University of Glasgow, UK
Sebastián Ventura	University of Cordoba, Department of Computer Science and Numerical Analysis, Spain
Luigi Vigneri	IOTA Foundation, Germany
Roopa Vishwanathan	New Mexico State University, Mexico
Marco Vitale	Foodchain SpA, Italy
Chenggang Wang	University of Cincinnati, EE.UU.
Yawei Wang	George Washington University, EE.UU.
Komminist Weldemariam	IBM Research, Africa, and Queen's University, Canada
Amr Youssef	Concordia University, EE.UU.
Uwe Zdun	University of Vienna, Austria
Kaiwen Zhang	École de technologie supérieure de Montréal, Canada
Mirko Zichichi	Universidad Politécnica de Madrid, Spain
Avelino F. Zorzo	PUCRS, Brazil
André Zúquete	University of Aveiro, Portugal

BLOCKCHAIN’21 Sponsors

Sponsors

IBM

indra

Organizers

Instituto Politécnico do Porto

Universidad de Salamanca

International Institute of Information Technology Hyderabad

Université de Montréal

Alma Mater Studiorum A.D. 1088

Universidad Rey Juan Carlos

ipb Instituto Politécnico de Bragança

Support from National Associations

Contents

BLOCKCHAIN-Main Track

Formal Analysis of Smart Contracts: Model Impact Factor on Criminality

Malaw Ndiaye(✉) and Karim Konaté

UCAD, Dakar, Senegal
{malaw.ndiaye,karim.konate}@ucad.edu.sn

Abstract. Smart contracts certainly provide a powerful functional surplus for maintaining the consistency of transactions in applications governed by blockchain technology. However, the intended level of automation might cause cascading effects that have to be checked by formal methods of algorithmic proof. Our smart contract formal model analysis framework uses the Finite State Machine (FSM) theory which is a model of behavior composed of states, transitions and actions. In this model, a state stores information about the past, a transition indicates a state change and is described by a condition that would need to be fulfilled to enable the transition. An action is a description of an activity that is to be performed at a given moment. These conditions are properties that are checked during program execution. This formal analysis framework allows us to define a set of invariants on Finite State Machine behavior model and to propose an anomaly detection system based on the invariants of the smart contract.

1 Introduction

Ethereum, taken as a whole, can be viewed as a transaction-based state machine. The state can include such information as account balances, reputations, trust arrangements, data pertaining to information of the physical world; in short, anything that can currently be represented by a computer is admissible. Transactions thus represent a valid arc between two states; the 'valid' part is important there exist far more invalid state changes than valid state changes. Invalid state changes might, e.g., be things such as reducing an account balance without an equal and opposite increase elsewhere. A valid state transition is one that is produced by a transaction [36].

Smart contract is the program deployed in a distributed network that can acquire outside information via transations and update the internal state automatically. Majority of smart contract procedures are based on blockchain technology. Existing smart contracts control digital currencies principal. Whereas they were found having defects and deficiencies in the course of their operations, leading to more serious consequences, such as "The DAO" and Ethereum Parity wallet incident, which caused a large number of cryptocurrencies to be stolen, causing a large loss. If these program bug cannot be processed, smart contracts will be difficult to manipulate real assets [10].

J. Prieto et al. (Eds.): BLOCKCHAIN 2021, LNNS 320, pp. 3–13, 2022.
https://doi.org/10.1007/978-3-030-86162-9_1

1.1 Contributions

This paper makes the following contributions:

- We propose a framework for analyzing the smart contract model, we use an approach based on Finite State Machine theory to model the execution of smart contracts in the Ethereum environment.
- We will show the inadequacies of the model in relation to malicious smart contracts, i.e. show how the model facilitates attacks and steals capital.
- Proposal for an anomaly detection model based on smart contract invariants.

1.2 Organize

This paper is organized as follows:

- Section 2 presents background study and related work.
- Section 3 presents the study framework of the program as a whole and examines the flaws in the model of smart contract promoting crime.
- Section 4 describes our anomaly detection system based on the invariant calculation method in smart contract systems.
- Section 5 offers new research directions based on anomaly detection in smart contract systems.

2 Background Study and Related Work

2.1 Smart Contracts Operational Mechanism

Smart contracts generally have two attributes: value and state (Fig. 1). The triggering conditions and the corresponding response actions of the contract terms are preset using triggering condition statements such as "If-Then" statements. Smart contracts are agreed upon and signed by all parties and submitted in transactions to the blockchain network, then transactions are broadcasted via the P2P network, verified by the miners and stored in the specific block of the blockchain [22,34].

The creators of the contracts get the returned parameters (e.g., contract address), then users can invoke a contract by sending a transaction. Miners are motivated by the system's incentive mechanism and will contribute their computing resources to verify the transaction. More specially, after the miners receive the contract creation or invoking transaction, they create contract or execute contract code in their local Execution Environment. Based on the input of trusted data feeds and the system state, the contract determines whether the current scenario meets the triggering conditions. If the conditions are met, the response actions are strictly executed. After a transaction is validated, it is packaged into a new block. The new block is chained into the blockchain once the whole network reaches a consensus [10,34].

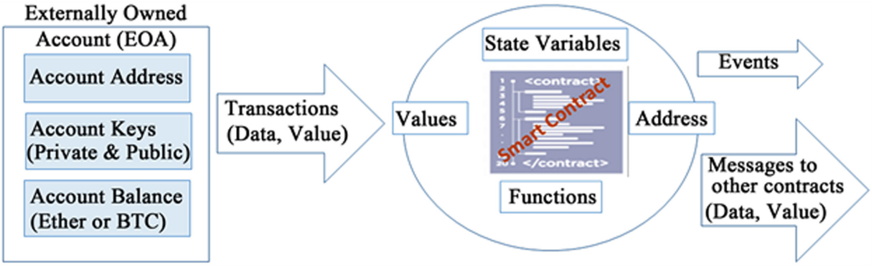

Fig. 1. Smart contracts operational mechanism

2.2 Smart Contracts Formal Model

A model M^* is a pair (Q, δ^*), where Q is a set of states and δ^* is a set of sequences of states satisfying property $\sum$ below. The set of states can be thought of as the set of all conceivable states of a program; i.e., all possible combinations of values of variables and "program counter" values. A sequence $q_1, q_2, \ldots \in \delta^*$ represents an execution that starts in state q_1, performs the first program step to reach state q_2, performs the next program step to reach state q_3 etc. The execution terminates if and only if the sequence is finite. The set δ^* represents all possible executions of the program, starting in any possible state [19].

Contract automata M^* is a quintuple:

$$\mathbf{M}^* = (\mathbf{Q}, \sum, \delta^*, s^*, F^*) \tag{1}$$

Among them:

- Q $= \{q_1^*, q_2^*, ..., q_m^*\}$. Q is the set about all states of contract execution automata, q_i^* is contained in the state set of contract party, $q_i^* \in q_i$, (i=1, ... , m);
- $\sum$ is the set of all input events;
- δ^* is the set of all the transition functions,
  $\delta^* : Q \times \sum \rightarrow Q$
- s^* is the initial state, $s^* \in Q$
- F^* is the set of termination states, $F^* \subset Q$.

2.3 Transaction Formal Model

A transaction (formally, T) is a single cryptographically signed instruction constructed by an external actor [3,28,36,38]. While it is assumed that the ultimate external actor will be human in nature, software tools will be used in its construction and dissemination. There are two types of transactions: those which result in message calls and those which result in the creation of new accounts with associated code (known informally as "contract creation"). Both types specify a number of common fields: **nonce** (T_n), **gasPrice** (T_p), **gasLimit** (T_g), **to** (T_t), **value** (T_v), **init** (T_i), **data** (T_d) [30] (Fig. 2).

$$T = \cup T_\alpha \equiv \cup T_\alpha \quad \alpha \in \{n, p, g, t, v, i, d, ...\} \tag{2}$$

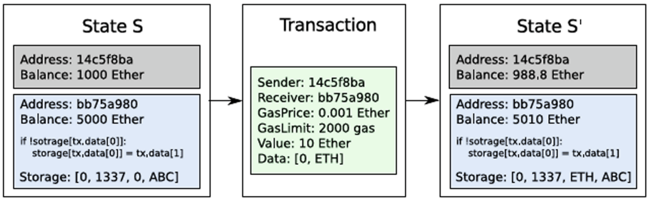

Fig. 2. Transaction model

3 Smart Contracts Sequential Execution Model

3.1 The Smart Contracts Sequential Programs

The specific model of deterministic sequential programs can be obtained by structuring the general state into [29]:

$$Q = (\pi, u) \tag{3}$$

π is the control component and assumes a finite number of values, taken to be labels or locations in the program. $L = \{l_0, l_1, ...l_n\}$

u is the data component and will usually range over an infinite domain. It be structured into state variables and data structures or functions (Fig. 1).

The transition relation δ can also be partitioned into a next-location function $N(\pi, u)$ and a data transformation function $D(\pi, u)$. $N(l, u)$ will actually depend on u only if the statement at l is a conditional.

We can thus express δ in terms of N and D:

$$\delta[(\pi, u), (\pi', u')] \iff \pi' = D(\pi, u) \& u' = N(\pi, u) \tag{4}$$

3.2 Smart Contracts Execution Model

An execution is a sequence of external transactions T each nesting one or more internal transactions (transitions T_α). Each transition starts with a message and proceeds in a sequence of commands. Commands may load or store data from and to the private storage, perform local computations (not affecting the storage), and initiate nested transitions [28].

a) Transition Invariant

A transition invariant T is a superset of the transitive closure of the transition relation δ restricted to the accessible states Q [30]. Formally,

$$\delta^* \cap Q \times Q \subseteq T \tag{5}$$

Transition is valid

$$\Leftrightarrow \forall q_i \exists q_{i+1} \ / \ \delta(q_i, T_\alpha) = q_{i+1} \tag{6}$$

b) State Invariant

A state invariant is a superset of Q. Given the transition invariant T and the set of starting states $I \in Q$, the set

$$I \cup \{q' | q \in I \; and \; (q', q) \in T\} \tag{7}$$

is a state invariant. Conversely, a transition invariant can be strengthened by restricting it to a given state invariant [30].

In other words, an execution is normal if there is a finite sequence of consecutive valid transitions which begins with an initial state q_1 and ends with a final state q_n, without blocking any state (Fig. 3).

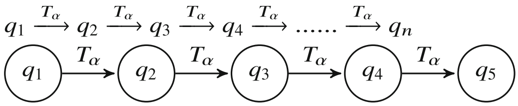

Fig. 3. Smart contracts execution model

3.3 Model and Vulnerable Smart Contracts Attack Vectors

In software systems, programming errors are usually the root cause leading to security breaches such as denial of service, buffer overflow, format string, code injection, etc. Coding errors may result either from defectively designed language features such as no built-in protection for accessing memory, or from invalid logic having high-level semantic error [12].

Smart contracts may contain vulnerabilities, which cause contracts to run on an unplanned scenario. However, these vulnerabilities are still harmless until an adversary takes advantage by exploiting them. Generally, he must send transactions, which are termed as attack vectors in security field, to exploit these vulnerabilities [23].

In formal verification (model verification, symbolic execution, theorem proof, translation and type verification) [1,2,4,5,14,18,21,22,28,31] as in the detection of vulnerabilities [6–9,11,13,15,20,24,25,27,32,33,37], all smart contracts which do not respect the properties of theorems 1 and 2, are carriers attack vectors. Table 1 represent the properties which are likely to be violated.

3.4 Model and Criminal Smart Contracts

We refer to smart contracts that facilitate crimes in distributed smart contract systems as criminal smart contracts (CSCs) [16]. In blockchain, the main activity of criminal smart contracts is based on Darkleaks, Generic public Leakage, Private Leakage, Key Theft, website defacement contract, Data Feed corruption,

Table 1. Ethereum application layer vulnerabilities attacks

Model impact factor		Properties Violation		
Attack name	Attack vectors	Theorem 1	Theorem 2	
		Termination	Fairness	Correctness
DAO ATTACK	Reentrancy	✗	✔	✗
Parity multisignature wallet	Delegate call injection	✔	✔	✗
	Erroneous visibility	✔	✔	✗
	Unprotected suicide	✔	✔	✗
	Frozen Ether	✔	✔	✗
BECToken attack	Integer overflow	✔	✗	✗
GovernMental attacks	DOS unbounded operations	✔	✔	✗
	Unchecked call return value	✗	✔	✗
	Call-stack depth limit	✗	✔	✗
	Transaction-ordering dependence	✔	✔	✗
	Timestamp dependence	✔	✔	✗
HYIP attack	DOS unexpected revert	✗	✔	✗
Fomo3D attacks	Generating randomness	✔	✔	✗
	DOS block stuffing	✗	✔	✔
ERC-20 signature replay	Insufficient signature information	✗	✔	✗
Rubixi attack	Erroneous constructor name	✔	✔	✗

Password theft. An example of a CSC is a smart contract for (private-)key theft. Such a CSC might pay a reward for delivery of a target key sk, such as a certificate authority's private digital signature key. The validity of criminal smart contracts is an indicator of criminals' success.

In their previous work, Juel and et al. [16,17] demonstrated that the execution of criminal smart contracts is always based on a time T_{end}. The time T_{end} marks the end of the execution whatever the state of the transaction i.e. the desirable terminal state is not always accessible therefore properties (4) and (7) are not always respected. This thesis is confirmed by the work of Yilei and et al. [35]. In their studies, they proposed a CSC based on PublicLeaks by formulating random factors such as the donation ratio. This contract is divided into five terminal states, one of which is unique in PublicLeaks because of its random nature (Table 2).

Table 2. Criminal smart contracts attacks

Model impact factor		Properties violation			
Attack name	Type of action	Correctness	Reachability	Real-time	Liveness
Leakage of secrets	Darkleaks	✔	✗	✗	✔
	Generic public leakage	✔	✗	✗	✔
	Private leakage	✔	✗	✗	✔
Key compromise	Key theft	✔	✗	✗	✔
Calling card crimes	Website defacement contract	✔	✗	✗	✔
	Data feed corruption	✔	✗	✗	✔
Password theft	Password theft	✔	✗	✗	✔

4 Proposition

It is important to understand that formal verification does not solve the problem of malicious smart contracts crime. To solve the problem of contracts while respecting the properties mentioned above, we propose an anomaly detection system on smart contracts. The idea is to create a consensus mechanism based on the behavior of smart contracts whose principle will be based on the prototype of normal behavior of the smart contract.

Anomaly detection overcomes the limitation of misuse detection by focusing on normal system behaviors, rather than attack behaviors. This approach is characterized by two phases: in the training phase, the behavior of the system is observed in the absence of attacks, and invariant calculation technique used to create a profile of such normal behavior. In the detection phase, this profile is compared against the current behavior of the system, and any deviations are flagged as potential attacks. Unfortunately, systems often exhibit legitimate but previously unseen behavior, which leads anomaly detection techniques to produce a high degree of false alarms. Moreover, the effectiveness of anomaly detection is affected greatly by what aspects of the system behavior are learnt.

Thus our model is composed of two modules: an invariant calculation algorithm to determine the normal profile of a smart contract, and an algorithm for monitoring the execution of the contract (Fig. 4).

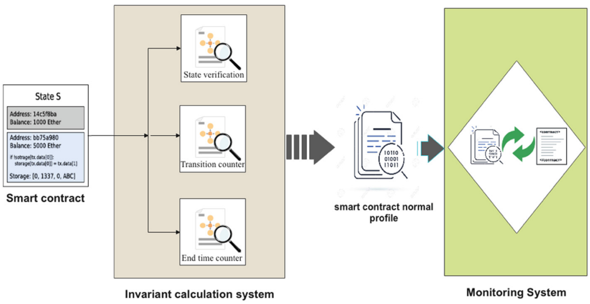

Fig. 4. Smart contracts anomalies detection system

5 Challenges

Given that all of the proposed solutions use techniques based on formal verification, it would be of great importance to orient the field of research towards the anomalies detection in smart contract systems. Anomaly detection is based on a program or host or network. Many distinct techniques are used based on type of processing related to behavioral model.

They are: Statistical based, Operational or threshold metric model, Markov Process or Marker Model, Statistical Moments or mean and standard deviation model, Invariant Model, Multivariate Model, Time series Model, Cognition based, Finite State Machine Model, Description script Model, Machine Learning based, Baysian Model, Genetic Algorithm model, Neural Network Model, Computer Immunology based.

The application of these detection techniques relating to the behavior model can solve the problem of attacks in smart contract systems.

Additional work could be carried out on smart contract anomaly detection techniques using artificial intelligence, ontology-based smart contracts for detecting malicious behavior, deep learning technique.

A future project could consist of working on a behavior detection model based on artificial intelligence because so far, cybersecurity systems using artificial intelligence have proven to be the most effective in protecting blockchain.

6 Conclusion

Termination, Fairness, Correctness, Reachability, Safety, Liveness and Real-time properties of a smart contract must be guaranteed in advance, before formal instantiation in a blockchain. This is certainly important for developers, as well as suppliers and consumers that rely on the soundness of a smart contract. Moreover, it furnishes a source of trust for users because trust is maintained by algorithmic concepts. For example, proving the correctness of smart contracts, a model of the actual correct behaviour of a contract is necessary in first place. Determining whether a contract reacts correctly is not always as trivial as it seems, and proving it (automatically) means that the behaviour must be defined as conditions in a formal notation, for instance (temporal) first order logics [26]. However, a good approach to smart contract model allows us to find a solution to the problems related to blockchain crime. Anomaly detection provides an answer to various problems such as vulnerable smart contracts and criminal smart contracts.

References

1. Abdellatif, T., Brousmiche, K.-L.: 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), pp. 1–5. IEEE (2018)

2. Amani, S., Bégel, M., Bortin, M., Staples, M.: Towards verifying ethereum smart contract bytecode in Isabelle/HOL. In: Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs, pp. 66–77. ACM (2018)
3. Bartoletti, M., Galletta, L., Murgia, M.: A true concurrent model of smart contracts executions. In: International Conference on Coordination Languages and Models, pp. 243–260. Springer (2020)
4. Bhargavan, K., et al.: Short paper: formal verification of smart contracts. In: Proceedings of the 11th ACM Workshop on Programming Languages and Analysis for Security (PLAS), in Conjunction with ACM CCS, pp. 91–96 (2016)
5. Bigi, G., Bracciali, A., Meacci, G., Tuosto, E.: Validation of decentralised smart contracts through game theory and formal methods. In: Programming Languages with Applications to Biology and Security, pp. 142–161. Springer (2015)
6. Brent, L., et al.: Vandal: a scalable security analysis framework for smart contracts. arXiv preprint arXiv:1809.03981 (2018)
7. Chen, W., Zheng, Z., Cui, J., Ngai, E., Zheng, P., Zhou, Y.: Detecting ponzi schemes on ethereum: towards healthier blockchain technology. In: Proceedings of the 2018 World Wide Web Conference, pp. 1409–1418 (2018)
8. Di Angelo, M., Salzer, G.: A survey of tools for analyzing ethereum smart contracts. In: 2019 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPCON). IEEE (2019)
9. Dika, A.: Ethereum smart contracts: Security vulnerabilities and security tools. Master's thesis, NTNU (2017)
10. Feng, T., Yu, X., Chai, Y., Liu, Y.: Smart contract model for complex reality transaction. Int. J. Crowd Sci. **3**(2), 184–197 (2019). https://doi.org/10.1108/IJCS-03-2019-0010
11. Fu, Y., et al.: EVMFuzzer: detect EVM vulnerabilities via fuzz testing. In Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp. 1110–1114. ACM (2019)
12. Hadjidj, R., Yang, X., Tlili, S., Debbabi, M.: Model-checking for software vulnerabilities detection with multi-language support. In: 2008 Sixth Annual Conference on Privacy, Security and Trust, pp. 133–142. IEEE (2008)
13. He, J., Balunović, M., Ambroladze, N., Tsankov, P., Vechev, M.: Learning to fuzz from symbolic execution with application to smart contracts. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pp. 531–548 (2019)
14. Hirai, Y.: Formal verification of deed contract in ethereum name service, November 2016 (2016). https://yoichihirai.com/deed.pdf
15. Jiang, B., Liu, Y., Chan, W.K.: ContractFuzzer: fuzzing smart contracts for vulnerability detection. In: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, pp. 259–269. ACM (2018)
16. Juels, A., Kosba, A., Shi, E.: The ring of gyges: using smart contracts for crime. Aries **40**, 54 (2015)
17. Juels, A., Kosba, A., Shi, E.: The ring of gyges: investigating the future of criminal smart contracts. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 283–295 (2016)
18. Kalra, S., Goel, S., Dhawan, M., Sharma, S.: Analyzing safety of smart contracts. In: NDSS, Zeus (2018)
19. Lamport, L.: "Sometime" is sometimes "not never" on the temporal logic of programs. In: Proceedings of the 7th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pp. 174–185 (1980)

20. Luu, L., Chu, D.-H., Olickel, H., Saxena, P., Hobor, A.: Making smart contracts smarter. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 254–269 (2016)
21. Murray, Y., Anisi, D.A.: Survey of formal verification methods for smart contracts on blockchain. In: 2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS), pp. 1–6. IEEE (2019)
22. Nehai, Z., Piriou, P.-Y., Daumas, F.: Model-checking of smart contracts. In: 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), pp. 980–987. IEEE (2018)
23. Nguyen, Q.-B. Nguyen, A.-Q., Nguyen, V.-H., Nguyen-Le, T., Nguyen-An, K.: Detect abnormal behaviours in ethereum smart contracts using attack vectors. In: International Conference on Future Data and Security Engineering, pp. 485–505. Springer (2019)
24. Nguyen, T.D., Pham, L.H., Sun, J., Lin, Y., Minh, Q.T.: sFuzz: an efficient adaptive fuzzer for solidity smart contracts. arXiv preprint arXiv:2004.08563 (2020)
25. Nikolic, I., Kolluri, A., Sergey, I., Saxena, P., Hobor, A.: Finding the greedy, prodigal, and suicidal contracts at scale. In: Proceedings of the 34th Annual Computer Security Applications Conference, pp. 653–663 (2018)
26. Osterland, T., Rose, T.: Correctness of smart contracts for consistency enforcement. ERCIM NEWS **110**, 18–19 (2017)
27. Parizi, R.M., Dehghantanha, A., Choo, K.-K.R., Singh, A.: Empirical vulnerability analysis of automated smart contracts security testing on blockchains. In: Proceedings of the 28th Annual International Conference on Computer Science and Software Engineering, pp. 103–113. IBM Corp. (2018)
28. Permenev, A., Dimitrov, D., Tsankov, P., Drachsler-Cohen, D., Vechev, M.: VerX: safety verification of smart contracts. In: 2020 IEEE Symposium on Security and Privacy, SP, pp. 18–20 (2020)
29. Pnueli, A.: The temporal logic of programs. In: 18th Annual Symposium on Foundations of Computer Science (SFCS 1977), pp. 46–57. IEEE (1977)
30. Podelski, A., Rybalchenko, A.: Transition invariants. In: 2004 Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science, pp. 32–41. IEEE (2004)
31. So, S., Lee, M., Park, J., Lee, H., Oh, H.: VeriSmart: a highly precise safety verifier for ethereum smart contracts. In: 2020 IEEE Symposium on Security and Privacy (SP), pp. 1678–1694. IEEE (2020)
32. Torres, C.F., Schütte, J., State, R.: Osiris: hunting for integer bugs in ethereum smart contracts. In: Proceedings of the 34th Annual Computer Security Applications Conference, pp. 664–676 (2018)
33. Wang, H., Li, Y., Lin, S.-W., Ma, L., Liu, Y.: VULTRON: catching vulnerable smart contracts once and for all. In: Proceedings of the 41st International Conference on Software Engineering: New Ideas and Emerging Results, pp. 1–4. IEEE Press (2019)
34. Wang, S., Ouyang, L., Yuan, Y., Ni, X., Han, X., Wang, F.-Y.: Blockchain-enabled smart contracts: architecture, applications, and future trends. IEEE Trans. Syst. Man Cybern.: Syst. **49**(11), 2266–2277 (2019)
35. Wang, Y., Bracciali, A., Li, T., Li, F., Cui, X., Zhao, M.: Randomness invalidates criminal smart contracts. Inf. Sci. **477**, 291–301 (2019)
36. Wood, G.: Ethereum: a secure decentralized generalized transaction ledger (EIP-150 revision) (2020). http://gavwood.com/paper.pdf

37. Wustholz, V., Christakis, M.: Harvey: a greybox fuzzer for smart contracts. arXiv preprint arXiv:1905.06944 (2019)
38. Xia, X., Ji, Q., Le, J.: Research on transaction dependency mechanism of self-healing database system. In: 2012 International Conference on Systems and Informatics (ICSAI2012), pp. 2357–2360. IEEE (2012)

A Blockchain-Enabled Fog Computing Model for Peer-To-Peer Energy Trading in Smart Grid

Saurabh Shukla(✉), Subhasis Thakur, Shahid Hussain, and John G. Breslin

National University of Ireland Galway, Galway, Ireland
{saurabh.shukla,shahid.hussain,subhasis.thakur,
john.breslin}@nuigalway.ie

Abstract. The advancement in renewable energy sources (RESs) technology have changed the role of traditional consumers to prosumers. In contrast to the traditional power grid, the Smart Grid (SG) network provides a platform for peer-to-peer (P2P) energy trading between prosumers to buy or sell energy according to their requirements. The potential benefits of P2P energy trading can be realized through an efficient service provider of the communication network infrastructure. However, the current communication network is a trustless environment and thereby is unable to fully support the P2P energy trading requirements. Existing techniques in P2P energy trading with blockchain suffers from large network delay due to large network size; this further affects the network performance for P2P trading. In this paper, we present a novel Blockchain-Based Smart Energy Trading (BSET) algorithm along with a Blockchain-Enabled Fog Computing Model (BFCM) for P2P energy trading in Smart Grid. The proposed BSET algorithm provides a fully trusted minimum latency communication network that enables the prosumers to trade energy within their local premises. The algorithm was implemented using iFogSim, Truffle, ATOM, Anaconda, and Geth and evaluated against state-of-the-art communication network models for P2P energy trading. The simulation results revealed the effectiveness in terms of secure trading and network latency.

Keywords: Smart grid · Smart meter · Cyber-physical system · Fog computing · Blockchain · Cryptography · Cloud computing · Microgrid · Internet-of-Things

1 Introduction

Nowadays, the increasing demand for timely electrical energy consumption and monitoring has given rise to the role of a smart grid network over the traditional grid. The traditional grid is a centralized and one-way transmission of energy. Whereas the SG network is distributed two-way transmission of energy and information. Where prosumers and consumers have a major role to play in buying and selling energy in a decentralized manner. The current SG network extends the controlling, computation, monitoring and sensing of the information and electrical energy flow in a bidirectional way when compares to a traditional network where different buyers and sellers participate in the auction

J. Prieto et al. (Eds.): BLOCKCHAIN 2021, LNNS 320, pp. 14–23, 2022.
https://doi.org/10.1007/978-3-030-86162-9_2