ADVANCES IN CYBER-PHYSICAL SYSTEMS

Vol. 4, No. 1, 2019

A REPRESENTATIVE FRAGMENT METHOD OF ANALYZING COMPLEX SYSTEMS OF SMART CONTRACTS

Michal Horodelski[1] *and Piotr Filipkowski*[1]

[1]*Institute of Mathematics and Computer Science,*
Faculty of Mathematics Informatics
and Landscape Architecture, The John Paul II Catholic University of Lublin, Poland
Author's e-mail: *michal.horodelski@kul.pl*
[2]*Institute of Information Systems and Digital Economy,*
The Collegium of Economic Analysis of Warsaw School of Economics, Poland
Author's e-mail: *pfilip@sgh.waw.pl*

Submitted on 24.06.2019

***Abstract*: The paper presents the use of states of explosion-proof method for analyzing the behavior of systems that provide smart contract technology. The selected example system is ShadowEth, whose main task is to ensure sufficient confidentiality of information stored in the Ethereum blockchain currency. The Petri network model for the ShadowEth system has been presented. The system properties according to the specifications have been defined. Properties described in a certain extension of the TCTL logic and verification have been carried out.**

***Index Terms*: petri net, time, smart-contract, shadoweth, Ethereum.**

I. INTRODUCTION

The work contains a description of the method allowing to analyze the systems that provide smart contract technology. This method is resistant to the explosion of the modeled system states. The basis of the presented method is the Petri network theory. The extension used in comparison to the normal network has reading places, as in contextual networks, and the time that has elapsed since the token appeared in the network places. This treatment allows reader to model the passage of time directly for system objects that are represented by a token, place, presence of a token in a place or relations between these objects. The authors assume that the reader knows at least a base of Petri network model.

The second chapter describes a system called ShadowEth [10], which provides an extension of smart-contract technology, and approximates selected problems in it. The third chapter presents the basics of the modeling method used. The formal description of the Petri network extension and the ShadowEth system model are described. In the paper, the ways to represent the semantics of the model and the language of the system model description are presented. The fourth chapter presents Petri nets models of protocol. The fifth chapter presents the analysis of the behavior of the described system, and the results of the system simulation. The final chapter presents conclusions and directions for further research.

II. SYSTEM SHADOWETH

ShadowEth system [10] is an extension technology of smart systems providing contract [2, 4] in such a way that it provides protection / confidentiality of the data in the smart contract in three areas: specification, performance and state of the contract.

Smart contracts are programs whose operating principle (data storage method) is based on blockchain technology. Smart contract allows for the autonomous transactions by two contracting interveners. In other words, the smart contract is a protocol allowing digital access to verify correctness and support the negotiation process within the contract concluded by two parties. The term “smart contract” was used for the first time by Szabo [9] in 1994 in the publication "Smart contract", in which he explained the principle of operation of such contracts on the example of taking out a loan for the purchase of a car.

Smart contract operates using blockchain technology, which not only implements the account book of the signed contacts and transactions, but also is the basis for the functioning of the Ethereum currency.

A. BLOCKCHAIN AND SMART CONTRACT TECHNOLOGY

The blockchain technology allows to perform an electronic ledger, also known as a blockchain, which is distributed and publicly available [6, 7]. Blockchain technology with the help of users called miners/workers allows for the authorization of transactions concluded in the system between clients/interveners/parts.

The method of authorizing a new block containing transactions and attaching a block to the chain is carried out by the so-called consensus mechanism. The literature contains a description of various mechanisms / algorithms authorizing data such as Proof of Work (PoW), Proof of Stake (PoS), Practical byzantine fault tolerance (PBFT) or Delegate Proof of Stake (DPoS).

Certainly the most known and fundamental mechanism is the PoW mechanism derived from the

Bitcoint cryptocurrency. PoW solves the problem of so-called sybil attack of attaching virtual nodes to the peer-to-peer network in order to obtain a numerical advantage over the approval of the next block. PoW's operation is carried out with the help of participants (miners/workers) of the peer-to-peer network.

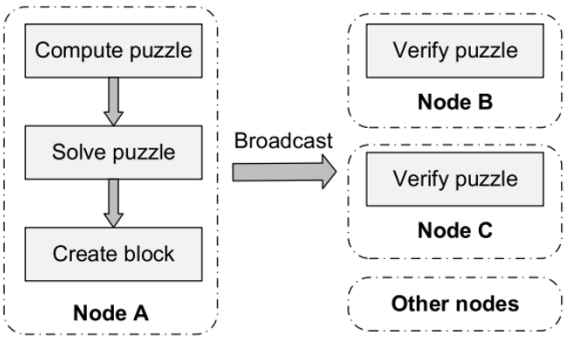

Fig. 1. The mechanism of consensus PoW

Source: [14], p. 2.

According to the diagram from Figure 1, miners generate and solve a computing problem called nonce. The solution of a nonce is a difficult task that requires time and computing power. However, the verification of the solution is much faster and much less expensive. Then the result is included in the set of transactions closed in a block and sent to other network participants in order to confirm verification. If 51% of the miners approve the solution, the block maintains its attachment to the chain and waits for perpetual approval/signature by generating next 6 blocks by the miners. One block is approximately every 10 minutes. Therefore, the perpetual approval of the next chain block takes about an hour for the network. It also happens that alternative blocks are attached to the chain next to each other. In this case, forks are formed. However, only the branch that will become the longest is maintained. Blocks from other branches are canceled.

In a peer-to-peer network, different ways of broadcasting information about a new block are used. These methods, called propagation mechanisms, can be divided into groups. The author refers to three groups in which the action is based on:

- Advertisement-based propagation.
- Sendheaders propagation.
- Unsolicited push propagation.

For example, in the first group, when node A receives information about the new block, it sends a message called *inv message* to other peer-to-peer nodes connected to the network. When node B receives a message from A, it will follow the established rules. If B already has information about the new block, he will not take any action. Otherwise, B will respond to the announcement by sending a reply message. When A receives a reply message, it will send B full information about the new block. The second group is the improvement of the previous one by sending information about block headers.

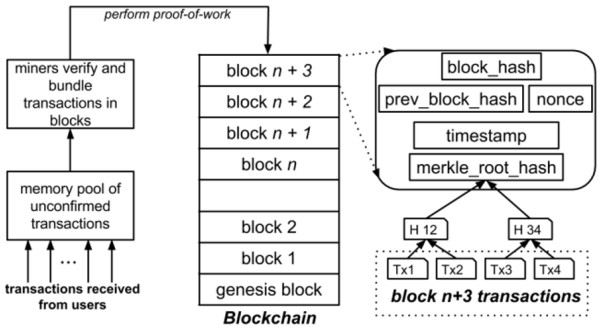

Fig. 2. Diagram of adding another block in PoW

Source: [15], pp. 3416–3452.

In a peer-to-peer network, different ways of broadcasting information about a new block are used. These methods, called propagation mechanisms, can be divided into groups. The author refers to three groups in which the action is based on:

- Advertisement-based propagation.
- Sendheaders propagation.
- Unsolicited push propagation.

For example, in the first group, when node A receives information about the new block, it sends a message called *inv message* to other peer-to-peer nodes connected to the network. When node B receives a message from A, it will follow the established rules. If B already has information about the new block, he will not take any action. Otherwise, B will respond to the announcement by sending a reply message. When A receives a reply message, it will send B full information about the new block. The second group is the improvement of the previous one by sending information about block headers.

B. SHADOWETH PROTOCOL

The main task of the ShadowEth system, described in [10, ShadowEth], is to provide a confidential platform for private smart-contract, which can be stored in the generally available accounting book of the Ethereum currency. The system offers confidentiality in three areas: specification, performance and smart-contract status.

One of the components of the system is the trusted enclave network, which stores data in the TEE-DS network acting as a trusted distributed data store.

The confidentiality of the code and information is maintained through a trusted communication channel (prepared before the contract is initiated) between the user and the TEE-DS network. Just before the conclusion of the contract, i.e. the transfer to the block chain (account book), it is encrypted. The contract can only be decrypted in the trusted enclave network.

In order to achieve the confidentiality of smart-contract performance, the only information that goes to the accounting ledger is information that allows him to be called (found) and allows its verification. The TEE-DS network generates a unique public and private key pair for each contract. The private key remains secret and the public key is published. When writing a contract to the accounting ledgers, the arguments are encrypted with the help of a public key. Only the participant in the enclave group who owns the private key can decrypt the data.

In order to ensure the confidentiality of the smart-contract status in the accounting book (Ethereum platform), only the hash / signature of the entire contract is stored, not the complete information as in the classic smart-contract. Access to information about contract details is possible only within the enclave network. In the case of insufficient memory, data is sent to external media, but in a securely encrypted way.

The structure of the system is analogous to the usual smart-contract system.

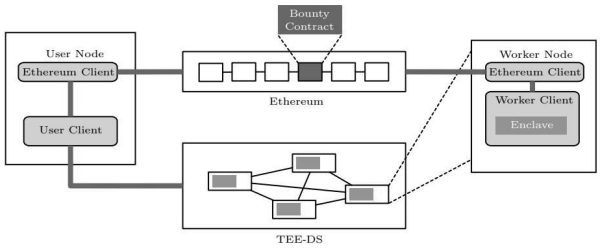

Fig. 3. ShadowEth system architecture

Source: [10], p. 546.

In the ShadowEth architecture scheme, there is a chain of blocks – Ethereum. Inside the chain, there may be a special case of the contract called Bounty-contract.

Bounty contract is the part of the contract that is publicly available and included in the Ethereum accounting book. Every Ethereum user can view bounty-contract, although there is no access to the details of the contract.

The User Node consists of the User Client and Ethereum Client components. User Client is a set of interfaces for the end user, thanks to which Etherum Client support is possible. The use of interfaces does not require the privileges of the participants of the enclave network.

The Worker Node, in the same way, consists of the Worker Client and Ethereum Client components. The Worker Client component is responsible for creating and managing confidential contracts. To use the Worker Client component, the user must be in the enclave network and have the appropriate permissions.

TEE-DS (being a distributed database) stores detailed information on confidential contracts.

Conclusion of contracts and their recalling works according to the defined protocol. The ShadowEth protocol can be divided into two scenarios:

contract deployment – the user defines the business logic of the confidential contract, compiles the code and places it in the TEE-DS. The information that allows him to be identified / searched is sent to the bounty contract. In this part, a pair of keys was generated, private (remaining only in the enclave) and a public key generally available.

contract invocation – it involves calling a private contract (deployed) in the blockchain, and it is possible for the users to do it just like an ordinary contact in the Ethereum network.

The Contract deployment section includes the following stages:

- the user sends the contract source codes to the TEE-DS through a secured communication channel (thanks to Keysession).
- TEE-DS generates a pair of keys Keyc_p and Keyc_s, unique for each contract (p – public, s – private/ secret).
- TEE-DS sends Keyc_p back to the user.
- The user sends identification information for a confidential contract to Bounty contract – which signals a new contract (private).
- Bounty contract creates a new record with the contract in its contract list in which it adds identification information, sets the version to 0, and the status to “deployed”.
- Once the deployed transaction has been recognized, the code of the private contract is sent to the remaining TEE-DS employees.

A private contract deployed in a block chain can be called by users just like a normal contact in the Ethereum network. The calling of a private contract is carried out according to the following steps:

- the user initiates a contract invocation by the user's client. The user must specify a contract Keyc_p, remuneration and appropriate parameters including the function name and its calling arguments.
- The user's client forwards the request:

1) adds a time stamp (date) to the content of the request;

2) generates a confidential Keyreq key which is used only for the given request;

3) when using Keyc_p, encrypts the request with the exception of pay and sends data to the Ethereum client.

- The Ethereum client generates an invoked transaction containing the contract Keyc_p, an encrypted request, and remuneration sent to the Bounty Contract.
- In the moment when the Bounty Contract receives the requested transaction:

1) verifies the identification data to make sure that the contract exists and that the request comes from one of the users included in the list of contract owners;

2) adds a new entry (new pass) with information attached to the referenced transaction to the list-to-do with the TODO designation and in the meantime, sends the remuneration to the account.

- At the moment when the requested transaction was recognized, the workers can see the tasks completed. They can choose any task and get related information, including Keyc_p and an encrypted request from Bounty Contract.
- using Keyc_p worker, one can ask TEE-DS for the contract code and load the contract into your enclave network together with the encrypted request;
- In the enclave, the contract-gate firstly decrypts the request to get a list of arguments of the called function and Keyreq and then uses a specific function;
- If the execution of the function ends without error, the contract-gate will distribute the modification of the contract status for the remaining employees in the TEE-DS and will generate a response containing:

1) version of the contract before execution;

2) contract status hash after execution;

3) the value returned by the function (when the function returns something), which is encrypted by Keyreq;

4) information about the agreement (when it is);

5) IVS.

- The worker's client sends a response to the Bounty Contract, which can verify the correctness of the IVS with Keyc_p, to ensure that the worker has completed this task correctly.
- After verification, the Bounty contract updates the list entry to be marked as FINISHED, fills in the return value (if present) and updates the contract information (version number, contract status hash).
- Then the user can provide the value return and decrypt it using Keyreq.
- If the result is marked as an agreement, the Bounty contract will generate an agreement transaction with the information about the agreement in a reply message. The first Worker who completes this process will receive remuneration.
- When the result is confirmed by BC, other TEE-DS workbenches will accept modifications.

III. DESCRIPTION OF METHODOLOGY OF MODELING

Construction of the development prefix for c-TdPN systems whose behavior is represented by the time-based transit system may be similar to the TPN systems in the algorithm described in [1] on page 9. The difference in the new proposed solution is the skeleton of this algorithm to another model Petri nets, another way to determine symbolic states $\widetilde{Post}(s)$ and $Post_t(s)$ and another way to verify the stop condition $s \notin Pass$.

In the c-TdPN model, the transition may be time-barred due to the passage of time in contrast to the TPN transitions.

The urgency of making a transition can be enforced by assuming the maximum time of performing the transition from the moment it is possible [11] or by adding restrictions to the presence of tokens in places.

For practical purposes, the c-TdPN will be added for places restrictions to the length of stay in one token, called invariants (inv).

The model of the c-TdPN system with invariants is the c-TdPN model extended by an additional inv relation allowing to determine the maximum length of the token stay in the selected place of the system model.

C. FORMAL DESCRIPTION OF C-TDPN AND TTS MODELS

Model c-TdPN is represented by structure $N = (P, T, F, C, I, m)$, where:

- P is a finite set of element called *places*;
- T is a finite and disjoined with P set of elements called *transitions*;
- $F \subseteq (P \times T) \cup (T \times P)$ is a set of pairs representing arcs connecting places with transitions and transitions with places;
- $C \subseteq P \times T$ is (disjoined with F) a set of pairs representing arcs called reading arcs, connecting places with transitions;
- I is a function that assigns for each arc $(p,t) \in F$ an interval of $[0, \infty]$ and for each arc $(p,t) \in C$ exact interval $[0, \infty]$;
- m is a function, called initial state, which assigns for each place a finite set of undistinguishable elements called tokens.

Functions such as m represent possible states of system wherein m represents *initial state*.

Transitions represent *actions* that the system can perform if at any place from where the arc leads to them, we can choose at least one token if the time has elapsed since the appearance of such a token falls within the arc of accessibility assigned to the arc.

Performing such an action causes the consumption of the selected tokens and creation of a token in each place where the arc originates from the transition, which represents this action.

Starting from the initial state, the execution of sequentially enabled transitions creates a graph of the executed transitions and achieved states.

The time that has elapsed since placing the token in the place is called the token's *age*.

Limitation of the structure N to (P,T,F) (respectively to (P,T,F,C)) with the interpretation

described is a well-known model of Petri net (resp. contextual Petri net) and limitation to (P,T,F,m) (resp. to (P,T,F,C,m)) network system (resp. contextual network system) with initial state m. Formally, it is a graph with two types of nodes (places and transitions) with arcs connecting nodes of different types.

Timed Transition System (TTS) of the system whose model is N (c-TdPN) is the structure $S=(Q,Q_0,\Sigma,\rightarrow)$ where:

- Q is a set of system states;
- $Q_0 \subseteq Q$ is a set of initial system states;
- Σ is a finite set of system actions;
- a relation $\rightarrow \subseteq Q\times(\Sigma\cup\mathbb{R}_+\cup\{0\}$ is a set of edges representing the change in the system state.

Changing the state of the system from q to q' can occur due to finalization of the action $a\in\Sigma$ or by the passage of $d\in R\cup\{0\}$ moments. The change is represented by $q\xrightarrow{x}q'$ which means $(q,x,q')\in\rightarrow$ where $x\in\{a,d\}$. For $x=d$ we can write $q'=q+d$ which means the state q after d of moments without the occurrence of any system action. Both the possibility of action and the passage of time depend on the type of system model.

Path (or process) in S we will call the maximum sequence of successive states due to the alternating time units $d_i\in R\cup\{0\}$ and the action instances $a_i\in\Sigma$. Formally, the path r in S, starting in the state $q_i\in Q_0$ is a graph:

$$r=q_i\xrightarrow{d_i}(q_i+d_i)\xrightarrow{a_i}q_{i+1}$$
$$\xrightarrow{d_{i+1}}(q_{i+1}+d_{i+1})\xrightarrow{a_{i+1}}q_{i+2}\xrightarrow{d_{i+2}}\dots$$

In a special case, the path can be started in any state $q\in Q$ by calling it the suffix of the process. The set of all paths (runs) beginning in q is represented by $p(q)$.

The model of contract deployed in ShadowEth protocol, shown in Fig. 4, is represented by structure $N=(P,T,F,C,I,m_0)$, where:

- $P=\{P_1,P_2,...,P_9\}$;
- T is a set of transition, represented by squares with captions;
- F is represented by arrows;
- C does not occur;
- m_0 is a function that assigns 1 to places P_1, P_2 and 0 for others.

D. THE SEMANTIC OF THE MODEL C-TDPN

Let $N=(P,T,F,C,I,inv,m_0)$ mean the c-TdPN model with invariants and $S_N=(Q,\{q_0\},\Sigma,\rightarrow)$ a timed transition system representing his behavior, where:

$Q=(P\rightarrow\mathbb{R}\cup\{0\})\times(T\rightarrow\mathbb{R}_+\cup\{0\}$,

$q_0=(m_0,v_0)$,

$\Sigma=T$,

$\rightarrow\subseteq Q\times(T\cup\mathbb{R}_+\cup\{0\})\times Q$,

v_0 is a function that assigns a value to each place at 0. The clocks are assigned to the places of the network model N. For each $p_i\in P$ there is exactly one clock $x_i\in X$. We also assume that there may be at most one token in the place. Hence, if x_p is the clock assigned to the place p, $inv(x_p)$ means the invariant assigned to this place.

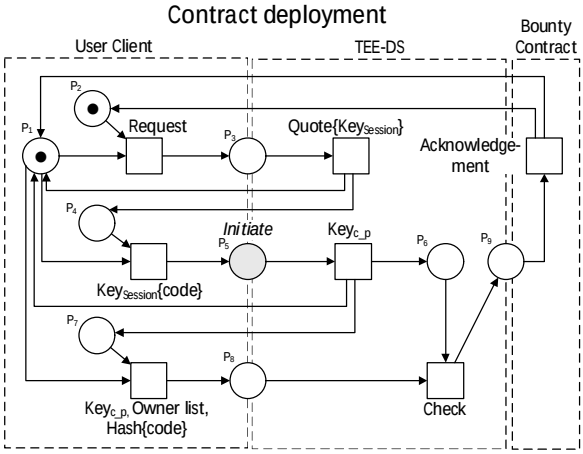

Fig. 4. Contract deployment model in ShadowEth protocol

Source: own study

Change of state $q=(m,v)$ to $q'=(m',v')$ was created as a result of the launch of the transition t, marked as $q\xrightarrow{t}q'$, is possible if:

$m'=(m-Ft)\cup tF$,

for each $p\in Ft\cup Ct$:

$m(p)\geq 1$,

$\min I(p,t)\leq v(p)\leq\max I(p,t)$,

$v'(p)=0$ when $p\in tF$,

when $v(p)=x_p$ is the indication of the clock assigned to the place p.

Change of state q to q' was created as a result of the passage \$d\$ units of time, marked $q\xrightarrow{d}q'$, is possible if:

$m'=m$,

$d\in R_+\cup\{0\}$,

for each $p\in Ft\cup Ct: v'(p)=v(p)+d$, and

for each place p where the token is:

$v'(p)=v(p)+d$,

$v'(p)\leq inv(p)$.

E. DISCRETIZATION OF THE TIMED TRANSITION SYSTEM

Transition system S_N after discretization is a discrete system

$$SA_N = (QA, QA_0, \Pi, \rightarrow) \tag{0.1}$$

where sets QA, QA_0, Π, $\rightarrow$ mean successively a set of symbolic states, initial symbolic states, actions, and system transitions. System SA_N can be represented by a graph with vertices in QA, the edges described by the relation $\rightarrow \subseteq QA \times \Pi \times QA$ and marked with actions from Π. The system actions represent running transitions in the set T.

The symbolic state aggregates some states of the system S_N with the same marking. Symbolic state s is represented by $s = (m, Z)$, where m is the marking of the network N, and Z is a zone limiting the indications of clocks assigned to places where there is at least one token in m and a zero clock.

Clock indications are limited by a range $[0, min)$, where min it's the smallest value $inv(p)$ for the place where the token is ∞ or when there are no limitations.

The zero clock constantly indicates the value 0. Symbolic initial state $s_0' = (m_0, Z_0')$ is the start marking m_0 and zone Z_0. This zone limits the indications of the clocks assigned to the places where there are tokens in the initial state in such a way that each clock has to indicate 0 (Z_0' is a zero matrix).

Change of the symbolic state s to s'' represented by $\rightarrow$ is about:

determination of symbolic state aggregating all possible states of the system S_N created by launching the transition t in s, represented by $s' = PostTd_t(s)$,

designation for s' all system states S_N resulting from the longest possible passage of time represented by $\underbrace{PostTd}(s')$.

In brief, $s'' = (\underbrace{PostTd} \circ PostTd_t)(s)$, where the zone w s' cannot be empty. In the case of a zone in s'' or s' is empty, that means t was not allowed in s.

Symbolic state $s' = PostTd_t(s) = (m', Z')$ is created by launching an enabled transition t in the symbolic state s, where:

$$m' = (m - Ft) \cup tF,$$

$$Z' = ((Z \cap \forall_{p \in Ft} \{\min I(p,t) \le x_p \le \max I(p,t)\})_{|Ol}) \cap \prod_{x_j \in Ne} \{x_j = 0\}$$

a clock x_p is associated with the place p.

State $s' = PostTd_t(s)$ in practice, it is calculated in stages:

designation of new marking $m' = (m - Ft) \cup tF$;

for Z' assigning a zone created by applying for Z limitations resulting from the possibility of a transition t:

$$Z' = Z \cap \forall_{p \in Ft} \{\min I(p,t) \le x_p \le \max I(p,t)\}$$

for Z' assigning it to a canonical character;

leaving in Z' limitations for the clock x_0 and those clocks assigned to places $m - Ft$ in which there is at least one token (set Ol);

adding successively to Z' limitations of new resetted clocks associated with places tF (set Ne);

clock differences from Ne and those who were already in the zone Z' because no limitations are set to ∞;

at the end for Z' assigning it to a canonical character.

Symbolic state

$$s'' = \underbrace{PostTd}_d(s') = (m'', Z'') \tag{0.2}$$

is a result of the longest possible passage of time d from the moment the state was established s', where:

$$m'' = m',$$

$$Z'' = \overrightarrow{Z'} \cap \forall_{x \in X_{Z'}} x \le d \wedge x \le inv(x)$$

and $\overrightarrow{Z'}$ is the future of the zone Z'.

The lack of a finite value is the future of the zone $d = \infty$ means no limit to the maximum delay: $\underbrace{PostTd}_d(s') = \underbrace{PostTd}(s')$, which reduces the last point to: $Z'' = \overrightarrow{Z'} \cap \forall_{x \in X_{Z'}} x \le inv(x)$.

The set QA_N consists only of the symbolic state $s_0 = \underbrace{PostTd}_d(s_0')$ and all symbolic states can be obtained by applying the assembly $\underbrace{PostTd} \circ PostTd_t$ for previously created states starting from s_0, where $t \in T$ is an enabled transition for these states.

Fig. 5 is presenting example of graph SA_N.

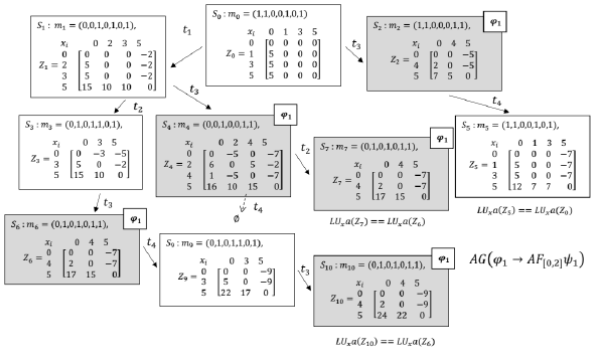

Fig. 5. Form of presentation of the behavior fragment – zone graph (overview data)

Source: own study

F. AN ALGORITHM THAT GENERATES A REPRESENTATIVE FRAGMENT OF THE STATE SPACE

Those definitions are used interchangeably in algorithm No. 1 to create a new algorithm for producing zone graphs for models of c-TdPN systems with invariants and an additional stop condition.

Data:

system – system c-TdPN with invariants with additional technical place g stored token and assigned clock point at 0 ($x_g = 0$),

$s'_0 = (m_0, Z'_0)$ – initial moment of system model, for all $i, j: Z'.z_{ij} = 0$,

d – limiting the maximum moment of generating space state or $d = \infty$ when there is no limit,

$s_0 = \overrightarrow{PostTd_d}(s'_0)$,

$wait := \{s_0\}$

$pass = \emptyset$

approx – chosen approximation of states: no-app, simple k, k_x or LU_x

Results: Generated discretized space of states c-TdPN, saved in *Pass*

Algorithm:

```
1: pass=GenerateZoneGraph(system,wait,pass,approx):
2: while wait ≠ ∅ do
3:    s = pop(wait);
4:    if ¬approx.Include(s,pass) then
5:       for t ∈ system.enabledTd(t,s) do
6:          s' = PostTd_d(PostTd_t(s));
7:          if Z_s' ≠ ∅ then
8:             wait = wait ∪ {s'};
9:          end if
10:      end for
11:      pass = pass ∪ {s};
12:   end if
13: end while
```

Algorithm No. 1. The algorithm that generates the zone graph of the c-TdPN system models

After calling the function *GenerateZoneGraph*() set *pass*, if the execution of the algorithm is completed, it stores all nodes of the system model space state until the moment d, or a representative fragment of it in the case $d = \infty$.

The main function loop is to check if the states are still untested (lines 2-13). In the case when there are no more undiscovered states, the execution of the algorithm ends. Otherwise, the undiscovered state is assigned to the symbol s and removed from the collection *wait* (line 3). Next, if in *pass* there is a condition that has the same approximation *approx* as s, this algorithm goes to the next run of the main loop, and s is skipped (line 4). Otherwise, for each active transition t in state s the new symbolic state is calculated s' and added to set *wait* if it has a non-empty zone (lines 5–10). Then information about the examined condition is saved s (line 11), and the algorithm execution goes back to the main loop.

G. APPROXIMATE SYMBOLIC STATES

Four well-known approximations were chosen for the study of c-TdPN with invariants, including the solution used in the TAPAAL tool. These are the following approximations k, k_x, LU_x [12], ext_{pl} [13]. Each of the approximations was adapted for the c-TdPN model with invariants.

Let $s' = (m', Z')$ mean the state created by the approximation of the state $s = (m, Z)$ oraz $x_i, x_j \in X$ set of zone clocks Z i Z' (including zero clock). The approximation does not change the marking of the symbolic state, but only the values of constraints written in the zone, so as not to increase the area of actuations of active transitions in s.

A simple approximation k consists in choosing one integer k, equal to the largest finite constraint in the network model. If the clock's indication exceeds the number k, its further value is unnecessary. New zone $Z' = k_a(Z)$ is calculated according to the formula:

$$z'_{ij} = \begin{cases} \infty & dla \quad z_{ij} > k \\ -k & dla \quad z_{ij} < -k \\ z_{ij} & dla \quad pozostałych \end{cases}$$

The approximation of k_x differs from the approximation of k, that for each x clock a separate value k_x corresponding to the largest finite constraint in the model for this clock is selected. The zone $Z' = k_a^x(Z)$ is calculated according to the formula:

$$z'_{ij} = \begin{cases} \infty & dla \quad z_{ij} > k_{x_i} \\ -k_{x_j} & dla \quad z_{ij} < -k_{x_j} \\ z_{ij} & dla \quad pozostałych \end{cases}$$

H. ADDITIONAL STOP CONDITION OF ALGORITHM

Algorithm No. 1 has an additional stop condition defined, thanks to which it is possible to generate the initial fragment of the state space of the tested system model up to a certain point of d without using approximation. This will generate a poorer piece of behavior (discretized space of the system model states), although it will allow verification of its properties on the basis of contained paths and statuses that can be achieved without exceeding the moment d.

This fragment can be used for:

faster obtaining a representative model of behavior limited to a given moment,

faster verification of ownership with early breaking time,

checking whether the tested system action will be carried out until d,

answer to the question about what may happen in the system to the highest specified d moment.

An additional stop condition also allows to test the algorithm's operation:

until d with the selected approximation,

until d without approximation.

I. TDPN-TCTL (TTS) LOGIC TO DESCRIBE THE SYSTEM MODEL BEHAVIOR.

The verified properties will be described in the logic called c-TdPN-TCTL, which is an extension of the TCTL logic.

Let N, S_N and SA_N be defined as in the previous chapter.

Syntax of logic c-TdPN-TCTL has the following form:

$$\varphi := true|\gamma|\varphi \wedge \varphi|\varphi \vee \varphi|\neg\varphi|E\varphi U_I\varphi|A\varphi U_I\varphi$$

where *true* is keyword and $g \in GMEC$.

The semantics of logic c-TdPN-TCTL was defined by the operator ‘ :

$$\begin{array}{rll} s \models true & & \\ s \models \gamma & wtw & m \text{ zapewnia spełnienie formuły } \gamma \\ s \models \neg\varphi & wtw & \neg(s \models \varphi) \\ s \models \varphi \wedge \psi & wtw & s \models \varphi \wedge s \models \psi \\ s \models \varphi \vee \psi & wtw & s \models \varphi \vee s \models \psi \\ s \models x_g \in I & wtw & x_g \in I \cap [-s.z_{0g}, s.z_{g0}] \\ s \models E\varphi U_I\psi & wtw & \exists\rho \in \Pi(s), \exists_{i\geqslant 0}: \\ & & \rho[i] \models x_g \in I \wedge \psi \text{ oraz} \\ & & \forall_{j:0\leqslant j<i}\rho[j] \models \varphi \\ s \models A\varphi U_I\psi & wtw & \forall\rho \in \Pi(s), \exists_{i\geqslant 0}: \\ & & \rho[i] \models x_g \in I \wedge \psi \text{ oraz} \\ & & \forall_{j:0\leqslant j<i}\rho[j] \models \varphi \end{array}$$

where $s = (m, Z)$ is a symbolic state, x_g is a clock assigned to a token located in the technical (artificial) place of the system, used to measure time from the moment of its initiation. Whereas $s.z_{ij}$ means limiting clocks i and j from the node zone s.

J. ALGORITHMS TO VERIFY THE PROPERTIES OF A SYSTEM MODEL.

Let N, S_N and SA_N mean successively the system model, its behavior model and behavior model after discretization.

In the further part, the symbolic state will be called a node with some selected features. Let j, y be GMEC formulas. Node w has the following features:

$w.m$ – marking,

$w.Z$ – zone,

$w.z_{ij}$ specific upper limit of the difference of clocks $x_i - x_j$ from zone Z,

$-w.z_0 g$ and $w.z_{g0}$ is the earliest and the latest possible indication x_g in node w,

$w.potomni$ – set of nodes to which the arrow goes $\Rightarrow$ from node w ($w.potomni = \emptyset$ when there are no arrows),

$w.ref$ is a node approximated from w (always if $w.ref \neq \emptyset$ then $w.potomni = \emptyset$),

$w.spel(j)$ – true when marking $w.m$ ensures truthfulness j,

$w.espel(j, I)$ – true when the formula j is met at the time of the interval $I \cap w.Z$ at least in one path passing through the node w, where I is the interval in which the clock will be displayed x_g.

Let I, J compartments be positive semi-axes, wherein $I = [I.a, I.b]$ and $J = [J.a, J.b]$. The operations are defined for compartments:

subtraction (reduction) of the scalar value c: $J - c = [J.a - c, J.b - c]$,

safe reduction:

$$I = I.redukuj(c) = [\max\{0, I.a - c\}, \max\{0, I.b - c\}] \quad (0.3)$$

In addition, let the zone graphs SA_N mean the system state space N after discretization and $s_0 = \overline{\overline{PostTd_d}}(s_0')$ starting node of the graph.

K. PROPERTIES OF VERIFICATION ALGORITHM $EF_I j$.

Property $EF_I j$ will be verified using the function $EF(j, I)$ and recursive auxiliary function $EFRef(wezel, j, I)$. The function of the auxiliary function is to check whether $wezel$ passes a path that has a beginning in the node s_0, in which ownership is fulfilled $F_I j$ (fulfilled j at the moment with $I \cap wezel.Z$). Function $EF(j, I)$ returns the truth when property $EF_I j$ is met in the graph being examined SA_N (in the symbolic state s_0).

Node $s_0 = \overline{\overline{PostTd_d}}(s_0')$, where s_0' is an initial symbolic state of graph SA_N.

```
EFRef(wezel, φ, I):
 1: if I.b < −wezel.z_0s then
 2:   return FALSE;
 3: else if wezel.espel(φ, I) then
 4:   return TRUE;
 5: else if wezel.potomni ≠ ∅ then
 6:   for w ∈ wezel.potomni do
 7:     if EFRef(w, φ, I) then
 8:       return TRUE;
 9:     end if
10:   end for
11:   return FALSE;
12: else if wezel.ref ≠ ∅ then
13:   for w ∈ wezel.ref.potomni do
14:     if EFRef(w, φ, I.redukuj(wezel.ref.z_0g − wezel.z_0g)) then
15:       return TRUE;
16:     end if
17:   end for
18:   return FALSE;
19: else
20:   return FALSE;
21: end if
```

Algorithm No. 2. Algorithm for the verification of EF system properties

The procedure performing function *EFRef* in *wezel* checks five cases.

First of all if *wezel* occurs later than the interval I procedure returns false because all fragments of paths coming from this node are irrelevant (line 1–2).

Second, if the formula j is met when marking *wezel.m* and clock limitations x_g they contain a moment from the interval I procedure returns the truth (lines 3–4).

Third, when *wezel* has child nodes, it should be checked whether at least one of these nodes passes a path that fulfills ownership $F_I j$. If so, the procedure returns the truth, otherwise it is false (lines 5-11).

Fourth, if the previous cases did not take place, the procedure checks whether the node *wezel* has an approximation in SA_N (whether $wezel.ref \neq \emptyset$). If so, the procedure returns true when one of the child nodes of the node *wezel.ref* passes the path that fulfills the property $F_J j$, where $J = I.redukuj(wezel.ref.z_{0g} -$ $- wezel.z_{0g})$ (lines 12–17). Otherwise, the procedure returns false (line 18).

IV. SYSTEM WITH SMART-CONTRACT TECHNOLOGY

A. SHADOWETH TIMED PETRI NET MODEL

Model of ShadowEth, which authors have prepared, includes not only model of contract deployment (Fig. 4) but also model of contract invocation (Fig. 6). Second model was also invented with the help of sequence diagram presented in [10, ShadowEth].

Contract invocation model represents the next part of the protocol and is responsible for calling the private contract deployed in the blockchain.

In both models (Figures 4 and 6), it was assumed that the ranges of token availability are in the form of $[0,1]$ and an invariant with every invoice limiting the token's age to the one unit moment at each place.

V. ANALYZING THE BEHAVIOR OF THE SYSTEM AND SIMULATION TESTS

With the algorithm No. 1 can generate a representative portion of the behavior model of the system in the form of discretized graph zones which were analyzed in model checking.

Then, using algorithms such as algorithm No. 2, it is possible to automatically verify the assumed property of the system model.

During the research, the main focus was on verifying the confidentiality of information stored in contracts. A formal verification of ownership was carried out [5]. The confidentiality of concluding contracts on three levels was examined: specification of the contract, performance of the contact and its status.

Selected verification results are presented in the following tables.

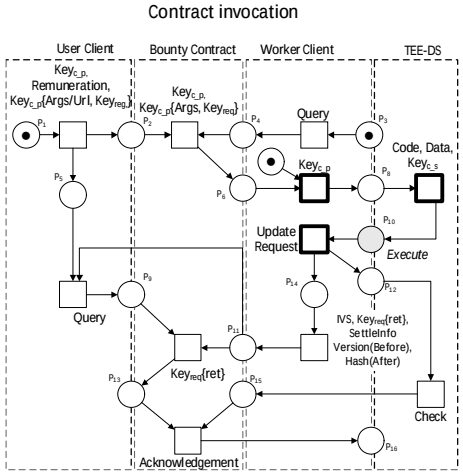

Fig. 6. Contract invocation model in ShadowEth protocol

Source: own study

Table 1

Contract Deployed Verification

Property	MEANING	Verification
$AF_I\varphi_1$	Sending a request for $Key_{session}$ for confidential communication: $\varphi_1 := \{P_3 = 1\}$	true
$AF_I\varphi_2$	The user has received $Key_{session}$: $\varphi_2 := \{P_1 = 1, P_4 = 1\}$	true
$AF_I\varphi_3$	TEE-DS. generates a pair of keys: Key_{c_p} i Key_{c_s} dedicated to the contract: $\varphi_3 := \{P_5 = 1\}$	true
$AF_I\varphi_4$	The user has received Key_{c_p}: $\varphi_4 := \{P_6 = 1, P_7 = 1\}$	true
$AF_I\varphi_5$	The user has sent information about a new confidential contract to Bounty contract: $\varphi_5 := \{P_9 = 1\}$	true

The next table summarizes the verified properties for the Contract invocation.

Table 2

Contract invocation Verification

Property	MEANING	Verification
$AF_I\psi_1$	The user has called the contract: $\psi_1 := \{P_2 = 1, P_5 = 1\}$	true
$AF_I\psi_2$	Bounty contract authorizes the call and adds a new entry to the list: $\psi_2 := \{P_6 = 1\}$	true
$AF_I\psi_3$	TEE-DS. at the request of Worker Client, he performs the contract using the private key: $\psi_3 := \{P_{10} = 1\}$	true
$AF_I\psi_4$	Bounty contract confirms the performance of the contract: $\varphi_5 := \{P_{16} = 1\}$	true

VI. SUMMARY

The paper presents the application of the developed method, explosion-proof state of the system model, modeling and analysis of the system property using smart-contract technology on the example of the ShadowEth system. The protocol for contract deployed and contract invocation was described, the method used was presented, models for both parts of the protocol were presented, the properties related to confidentiality were defined, and ownership verification was provided in both models. In all cases, properties were met. It was shown that it is possible to use the developed method for modeling and verification of smart-contract systems.

REFERENCES

[1] Boucheneb, H., Gardey, G., & Roux, O. H. (2009). TCTL model checking of time Petri nets. Journal of Logic and Computation, 19(6), 1509-1540. Christidis, K., & Devetsikiotis, M. (2016). Blockchains and smart contracts for the internet of things. Ieee Access, 4, 2292–2303.

[2] Luu, L., Chu, D. H., Olickel, H., Saxena, P., & Hobor, A. (2016, October). Making smart contracts smarter. In Proceedings of the 2016 ACM SIGSAC conference on computer and communications security (pp. 254–269). ACM.

[3] Kosba, A., Miller, A., Shi, E., Wen, Z., & Papamanthou, C. (2016, May). Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In 2016 IEEE symposium on security and privacy (SP) (pp. 839–858). IEEE.

[4] Atzei, N., Bartoletti, M., & Cimoli, T. (2017, April). A survey of attacks on ethereum smart contracts (sok). In International Conference on Principles of Security and Trust (pp. 164–186). Springer, Berlin, Heidelberg.

[5] Bhargavan, K., Delignat-Lavaud, A., Fournet, C., Gollamudi, A., Gonthier, G., Kobeissi, N., ... & Zanella-Béguelin, S. (2016, October). Formal verification of smart contracts: Short paper. In Proceedings of the 2016 ACM Workshop on Programming Languages and Analysis for Security (pp. 91–96). ACM.

[6] Cong, L. W., & He, Z. (2019). Blockchain disruption and smart contracts. The Review of Financial Studies, 32(5), 1754–1797.

[7] Cuccuru, P. (2017). Beyond bitcoin: an early overview on smart contracts. International Journal of Law and Information Technology, 25(3), 179–195.

[8] Zhang, F., Cecchetti, E., Croman, K., Juels, A., & Shi, E. (2016, October). Town crier: An authenticated data feed for smart contracts. In Proceedings of the 2016 aCM sIGSAC conference on computer and communications security (pp. 270–282). ACM.

[9] Szabo, N. (1994). Smart contracts, 1994. Virtual School.

[10] Yuan, R., Xia, Y. B., Chen, H. B., Zang, B. Y., & Xie, J. (2018). Shadoweth: Private smart contract on public blockchain. Journal of Computer Science and Technology, 33(3), 542-–556.

[11] Akshay, S., Genest, B., & Hélouët, L. (2014). Timed Petri Nets with (restricted) Urgency.

[12] Behrmann, G., Bouyer, P., Larsen, K. G., & Pelánek, R. (2006). Lower and upper bounds in zone-based abstractions of timed automata. International Journal on Software Tools for Technology Transfer, 8(3), 204–215.

[13] David, A., Jacobsen, L., Jacobsen, M., & Srba, J. (2012). A forward reachability algorithm for bounded timed-arc Petri nets. arXiv preprint arXiv:1211.6194.

[14] Li X., Jiang P., Chen T., Luo X., Wen Q., A survey on the security of blockchain systems, Future Generation Computer Systems, 2017, p. 2.

[15] Conti M., Kumar E. S., Lal C., Ruj S., A survey on security and privacy issues of bitcoin, IEEE Communications Surveys & Tutorials, 20(4), 2018, pp. 3416–3452.

Piotr Filipkowski was born in Bialystok in 1976. Doctor of economics (in Business Information Systems), assistant professor at the Institute of Information Systems and Digital Economy of the Warsaw School of Economics. He gained his research experience in projects carried out at the Military Institute of Aviation Medicine (assistant professor of Executive Board – coordination of strategic projects), Department of Information Society Technology, KUL JPII (assistant professor). He graduated from the Faculty of Electrical Engineering and the Faculty of Management at Bialystok University of Technology, Institute of Systemic Research of the Polish Academy of Sciences and the Faculty of Management at the University of Lodz. President of the European Center for Information Society Technologies. The Secretary of Postgraduate Studies "Cybersecurity management", The Secretary & Member of CEA Council of the Warsaw School of Economics. Author of numerous publications in the area of information society technologies and co-creator of the modeling and simulation system and the ® sign for the platform of proprietary intelligent systems solutions. He lectures "Management Information Systems", "Introduction to Business Information Systems". Author of lectures "Digital Technologies in Business Relations", "Mobile Digital Office" in English and Polish.

Michal Horodelski was born in Zamoyskiego, Zamość, in 1985. He obtained Master's degree in Mathematics, Bachelor of Computer Science, Lecturer at the Institute of Mathematics and Computer Science, The John Paul II Catholic University of Lublin, postgraduate at Institute of Computer Science Polish Academy of Science in Warshaw. He gained his research experience in project carried out at the Military Institute of Aviation Medicine (Distributed platform for modelling and simulation of information flows in the DIS/HLA standard), Vice President of the European Centre for Information Society Technologies. A graduade of The Faculty of Mathematics, Informatics and Landscape Architecture, The John Paul II Catholic University of Lublin.