Formalizing Dynamic Behaviors of Smart Contract Workflow in Smart Healthcare Supply Chain

Mohammad Saidur Rahman[1(✉)], Ibrahim Khalil[1], and Abdelaziz Bouras[2]

[1] RMIT University, Melbourne, Australia
{mohammadsaidur.rahman,ibrahim.khalil}@rmit.edu.au
[2] Qatar University, Doha, Qatar
abdelaziz.bouras@qu.edu.qa

Abstract. We present a formal model for smart contract workflow using Colored Petri-Net in the context of a blockchain-based healthcare supply chain in this paper. Ensuring traceability of products is a crucial issue in a smart healthcare supply chain. Blockchain and smart contracts are two enabling technologies that ensure the traceability of products and prevent data tampering in the smart healthcare supply chain. In a blockchain-based supply chain, a workflow of smart contracts needs to created and executed based on the input data. The selection of smart contracts in the workflow is data-driven and dynamic. Hence, it is necessary to verify the correctness of the dynamic execution of smart contracts. In this paper, we develop a Colored Petri-Net based formalism to verify the correctness of dynamic behaviors of the smart contract workflow. We conduct experiments to evaluate the performance of our proposed model.

Keywords: Blockchain · Formal model · Smart contract · 12 healthcare supply chain · Colored petri-net

1 Introduction

With the help of Internet-of-Things (IoT) and smart devices, the smart supply chain can play an essential role during the pandemic situation, such as COVID-19. For example, keeping records of pharmaceutical products in the whole supply chain process in a contactless manner can be a perfect example of the smart supply chain in healthcare. However, supply chain participants can tamper data if they fail to comply with the product handling policy. Therefore, it is necessary to ensure the traceability of the supply chain data and prevention of data tampering throughout the process.

Blockchain provides data traceability and protection from tampering at the same time [12,13]. Hence, the blockchain technology has become a new norm in different applications such as supply chain [15], healthcare [10,17], smart grid [2],

N. Park et al. (Eds.): SecureComm 2020, LNICST 336, pp. 391–402, 2020.
https://doi.org/10.1007/978-3-030-63095-9_25

and smart transport [1,6,7,9]. Data are stored in the blockchain as a transaction [16]. At present, *smart contract* [3] is an essential part of blockchain-based systems. Smart contracts are a great advancement in blockchain technology [11]. A smart contract is a computer program that is executed automatically when a certain condition is satisfied. In a blockchain-enabled smart supply chain, business rules are abstracted as smart contracts and deployed in the blockchain networks. Smart contracts validate blockchain transactions before they make any changes to the distributed ledger.

In this paper, we address the issue of verifying the correctness of smart contract execution that is dynamic in nature and data-driven. The blockchain-based smart healthcare supply chain (SHSC) involves many stakeholders such as producers, distributors, logistics service providers, retailers, and consumers. All stakeholders are connected to a blockchain network to ensure traceability, transparency, integrity, and trust by preventing data modification. Business contracts among stakeholders are stored in multiple smart contracts. These smart contracts are executed whenever a particular condition is met. For example, a payment smart contract, between a supplier and customer, processes the payment to the supplier if x units of a pharmaceutical item are delivered to the customer. However, the execution of smart contracts may be dependent on data. For instance, a supplier may not deliver the required x units of the product. Assume that y units are delivered at first, and z units are delivered later, such that $x = y + z$. Hence, two different blockchain transactions are generated with the respective number of delivered items. Hence, the payment smart contract should process the payment only if the sum of units delivered is equal to x. Here, the execution of the payment smart contract is data-driven and behaves dynamically. As a supply chain task is a complex task by nature, we need a series of smart contracts, called *smart contract workflow*, from different participants to fulfill a particular supply chain task. Hence, it is necessary to verify if a generated smart contract workflow is complying with different dynamic conditions in SHSC.

The primary objective of this paper to develop a model that would verify the correctness of the execution of smart contracts with the dynamic behavior in the blockchain-based SHSC. In this paper, we model the dynamic behaviors of smart contracts in a workflow based on the business rules using Colored Petri-Net (CP-Net). Colored Petri-Net [5,18] is a version of Petri-Net. CP-Net is a popular formalization method that is used for modeling the dynamic behavior of entity [14]. This formalization is being used to check the soundness of smart contracts [4,8,19] in blockchain-based systems. CP-Net allows us to describe the smart contract behavior and logic through different tokens as colors [4]. Dynamic conditions of smart contracts can be expressed, and vulnerabilities in smart contracts can be detected easily in CP-Net. Overall, unnecessary loss is avoided in blockchain-based SHSC.

The rest of the paper is organized as follows. Section 2 discusses some of the related works. Section 3 describes the proposed Colored Petri-Net based formalism for smart contract workflow. Experimental results and performance of the proposed framework are analyzed in Sect. 4. We conclude the paper in Sect. 5.

2 Related Work

In this section, we discuss some of the Colored Petri-Net based formal modeling approaches for smart contracts in blockchain systems.

The research work in [8] proposes a Colored Petri-Net based formal verification method that identifies smart contracts' logical vulnerabilities in the blockchain system. Initially, the smart contract models with possible attackers based on hierarchical CP-Net. Next, the smart contract models are executed for validating the functional correctness. Authors in [8] demonstrate that the CP-Net-based formalism can detect the smart contract's logical vulnerabilities as well as the non-logical vulnerabilities in the contracts, such as the limitations of the smart contract development platform.

Authors in [4] present a multilevel modeling solution for smart contracts for analyzing the security of smart contracts. The model improves bytecode's program logic rules in the first place. Next, the Hoare logics are used for creating a CP-Net model. The wrong execution paths are shown by the model to analyze the security of a smart contract.

However, none of the aforementioned works model the data-driven dynamic behaviors of smart contract workflow, which is necessary for the safe execution of smart contracts in blockchain-based SHSC.

3 Proposed Model

In this section, we discussed our proposed Colored Petri-Net based modelling of smart contract workflow for the blockchain enabled SHSC.

3.1 Overview of Data-Driven Smart Contract Composition

In SHSC, smart contracts take data in terms of transactions and pass data to functions that implement business rules. Each function has at least one pre-condition and post-condition known as *guard conditions*. The blockchain network should verify an input value of the guard condition before smart contracts are deployed to the blockchain.

As a task in the supply chain is complex in general, a smart contract may not be enough to execute all of the business rules when a blockchain transaction is generated. Hence, multiple smart contracts need to be selected and composed to execute all business rules. Smart contract composition may be performed based on the data that is provided in the transaction. Therefore, data-driven compositions of smart contracts are required to fulfill a supply chain task in the SHSC.

In SHSC, we introduce four types of smart contract composition logics:(1) *sequential*, (2) *aggregation*, (3) *split*, and (4) *loop*. An overview of each of the composition logic is illustrated in Fig. 1. Assume that the composability between two smart contracts are represented using the operator "→". In the *sequential composition logic*, a smart contract SC sends a data x to another smart contract

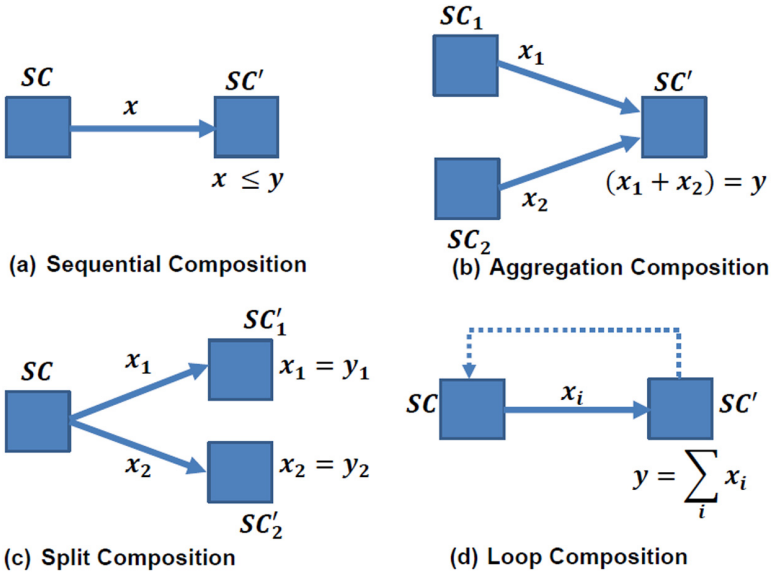

Fig. 1. overview of data-driven smart contract composition logic

$SC^{'}$. Smart contracts SC and SC is sequentially composable (i.e., $SC \rightarrow SC^{'}$) if $x = y$, where y is the expected value by $SC^{'}$ (see Fig. 1(a)). Assume that a smart contract $SC^{'}$ requires a data x_1 from the smart contract SC_1 and x_2 from another smart contract SC_2 such that $y = x_1 + x_2$, where y is the expected data in $SC^{'}$. Here, $(SC_1 \uplus SC_2) \rightarrow SC^{'}$ is called *aggregation composition* of smart contract (see Fig. 1(b)), and "$\uplus$" is the aggregation operator. A *split composition* is a composition logic where a smart contract SC sends data to multiple smart contracts. Assume that a smart contract SC sends a data x_1 to a smart contract $SC_1^{'}$ and x_2 to another smart contract $SC_2^{'}$. Now, $(SC_1^{'} \parallel SC_2^{'}) \rightarrow SC$ is a split composition (see Fig. 1(c)), and "$\parallel$" is the split operator. A *loop composition logic* can be defined as a series of sequential composition between two smart contracts SC and $SC^{'}$ which can be represented as $SC \xrightarrow{i} SC^{'}$. Here, "$\xrightarrow{i}$" indicates that the sequential composition $SC \rightarrow SC^{'}$ should be repeated i times (see Fig. 1(d)). The smart contract SC sends data x to the smart contract $SC^{'}$ unless $y = \sum_{i} x$, where y is the expected data at $SC^{'}$.

3.2 Colored Petri-Net Formalism of Composite Smart Contracts

In this section, we present Colored Petri-net formalism of different types of data-driven smart contract compositions.

Coloured Petri Net (CP-net) has several places. Every place in the net has a corresponding value type. The set of value types is called a color set. Each of the tokens in a place has a value that belongs to that type. Every single

arc has a variable and transition has a *precondition* and a *postcondition*. These preconditions and postconditions are called the guard. A precondition of an input arc of a transition is an expression with multiple variables. These variables are independent. A postcondition is an expression with variables of both input arcs and output arcs. A transition in a CP-net is enabled if and only if for each input place a token can are a well-established process modeling technique that has formal semantics. These semantics are used to model and analyze several processes, including protocols, manufacturing systems, and business processes.

We assume that a Colored Petri net (CP-Net) represents the behavior of a smart contract that works based on input data. The CP-Net for smart contracts consists of one input place and one output place. The input place is used for absorbing information, and the output place is used for emitting information. Combinedly, the CP-Net with input and output places facilitates the definition of the composition operators and the analysis as well as the verification of specific properties (e.g., reachability, deadlock, and liveness). Each event is activated when a particular token is obtained. In addition, the output place will have a certain number of tokens once the required events are fired. Tokens in CP-Net may have different colors representing the preconditions of smart contracts being executed. At any given time, a smart contract can be in one of the following states: initial, ready, processing, postponed, failed, or finished. When a smart contract is in the ready state, a token is in its corresponding input place, whereas the finished state means there is a token in the corresponding output.

Definition 3.1 (Smart Contract Net). A Smart Contract Net (SC-Net) is a CP-Net, i.e., a tuple $SCN = (P, T, F, \Sigma, C)$ where:

- P is a finite set of places such that $P = P_I \cup P_D$, where:
  - P_I is the set of internal places, and
  - P_D is the set of data places. Assume that ${}^{\triangleleft}p$ and $p^{\triangleright}$ are source and sink places, respectively. For $\forall {}^{\triangleleft}p, p^{\triangleright} \in P_D$.
- T is a finite set of transitions representing the operations of the service,
- $F \subseteq (P \times T) \cup (T \times P)$ is a set of directed arcs representing flow relation and (P_I, T, F) is a work flow net. (P_I, T, F) has two special places: i and o. Place i is a source place: ${}^{\triangleleft}i = \emptyset$ and Place o is a sink place: $o^{\triangleright} = \emptyset$. If we add a transition t to (P_I, T, F) which connects place o with i (i.e., ${}^{\triangleleft}t = i$ and $t^{\triangleright} = o$), then the resulting Petri net is strongly connected.
- Σ is the set of color sets.
- C is the color function. Assume that $E = \{e\} \in \Sigma$ is the color set that has only one possible value which stands for the control token in (P_I, T, F). C is defined from P to Σ such that $\forall p \in P_I, C(p) = e$.

Here, the smart contract net is a Colored Petri net. Internal places P_I represent the internal control logic. Data places P_D represent the data exchanged between services. The internal places are tagged with color e that stands for the control token in (P_I, T, F). The data places are tagged with the color sets that represent data types. The function F works as a guard function. The condition of the function F must be satisfied by a color value in E to trigger an event.

Based on the definition of the smart contract net stated above, a smart contract can be defined as follows:

Definition 3.2 (Smart Contract). A Smart Contract is a tuple $SC =< addr, owner, A_{pre}, G, tran, type, dom, desc, C, SCN >$ where:

- $addr$ - is the unique address of the smart contract,
- $owner$ - is the ID of owner of the smart contract,
- A_{pre} - the set of attributes and their values representing the initial state before the execution,
- $G = \{G_1, G_2, \ldots, G_f\}$, where G is the set of f number of functions G_i in the smart contract,
- $tran$ - is the transaction containing meta data that needs to be verified by the blockchain,
- $type$ - is a constant value representing the type of the smart contract,
- dom - is the domain of operation the smart contract. For example, if the smart contract contains the functions related to the manufacturing then the domain of the smart contract is manufacturer. Other possible domains can be distributor, retailer, and consumer in SHSC.
- $desc$ - is the textual description of the smart contract.
- CM - is a set of its component smart contracts such that $CM = \{CM_1, CM_2, \ldots, CM_m\}$, where CM_i is the i-th smart contract in CM and m is the number of component smart contracts involved in the smart contract. If $CM = \{\}$, then SC is a component smart contract. Otherwise, SC is a composite smart contract, and
- $SCN = (P, T, F, \Sigma, CM)$ is the smart contract net modelling the dynamic behavior of the smart contract using a CP-Net.

We assume that any two smart contracts (SC and $SC^{'}$) that are eligible for any of the aforementioned data-driven compositions have the same type and domain description as mentioned in the Colored Petri-net formalism of smart contract (see Definition 3.2).

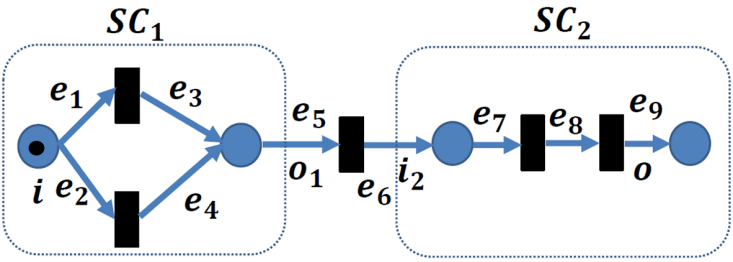

Fig. 2. Colored petri-net modelling of sequential composition of SC_1 and SC_2

Definition 3.3 (Sequential Composition). A sequential composition (S_{SC}) of smart contracts SC_1 and SC_2 is a tuple $S_{SC} =< A_{pre}, G, tran, type, CM, SCN >$ where:

- $CM = CM_1 \cup CM_2$
- $SCN = (P, T, F, \Sigma, C)$ such that
  - $P = P_1 \cup P_2$,
  - $T = T_1 \cup T_2 \cup \{t\}$,
  - $F = F_1 \cup F_2 \cup \{(o_1, t), (t, i_2)\}$,
  - $i = i_1$ and $o = o_2$,
  - $P_I = P_{I_1} \cup P_{I_2} \cup (o_1, t)$,
  - $P_D = P_{D_1} \cup P_{D_2} \cup (i_2, t)$,
  - $\Sigma = \Sigma_1 \cup \Sigma_2$,
  - $C = C_1 \cup C_2$.

Given two smart contracts SC_1 and SC_2, the invocation of SC_2 depends on the output data of SC_1. Therefore, the color function $C(o_1), C(i_2) \in C$ must satisfy $C(i_2) \subseteq C(o_1)$. Otherwise, the second smart contract SC_2 cannot be invoked by the first smart contract SC_1. Hence, the sequence composition of smart contracts would fail. Figure 2 presents the CP-Net of sequence composition of smart contracts.

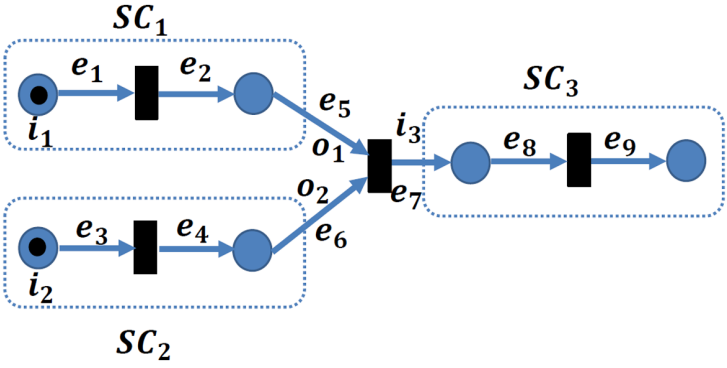

Fig. 3. Colored petri-net modelling of aggregation composition of SC_1, SC_2, and SC_3

Definition 3.4 (Aggregation Composition). An aggregation composition (A_{SC}) of smart contracts SC_1, SC_2, and SC_3 that is denoted as:

$$A_{SC} =< A_{pre}, G, tran, type, CM, SCN >, \text{ where:}$$

- $CM = CM_1 \cup CM_2 \cup CM_3$
- $SCN = (P, T, F, \Sigma, C)$ such that
  - $P = P_1 \cup P_2 \cup P_3 \cup i, o$,
  - $T = T_1 \cup T_2 \cup T_3 \cup \{(t, i_1), (t, i_1), (t, o)\} \cup \{t\}$,

- $F = F_1 \cup F_2 \cup F_3 \cup \{(t, i_1), (t, i_2), (o_1, t), (o_2, t), (t, i_3), (o_3, t)\}$,
- $i = \{i_1, i_2, o_1, o_2\}$ and $o = \{o_1, o_2, o_3\}$,
- $P_I = P_{I_1} \cup P_{I_2} \cup P_{I_3} \cup \{(o_1, t), (o_2, t)\}$,
- $P_D = P_{D_1} \cup P_{D_2} \cup P_{D_3} \cup (i_3, t)$,
- $\Sigma = \Sigma_1 \cup \Sigma_2 \cup \Sigma_3$,
- $C = C_1 \cup C_2 \cup C_3$.

Assume that three smart contracts (SC_1, SC_2, and SC_3) are given. The invocation of SC_3 depends on the output data of SC_1 and SC_2. Therefore, the color function $C(o_1), C(o_2), C(i_3) \in C$ must satisfy $C(i_3) \subseteq \{C(o_1) \cup C(o_2)\}$ to activate the execution of SC_3. Otherwise, the aggregation composition of smart contracts would fail. Figure 3 presents the CP-Net of aggregation composition of smart contracts.

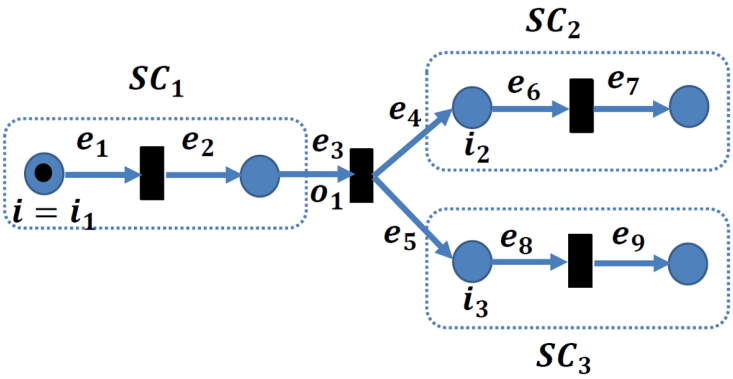

Fig. 4. Colored petri-net modelling of split composition of SC_1, SC_2, and SC_3

Definition 3.5 (Split Composition). A split composition (SP_{SC}) of smart contracts SC_1, SC_2, and SC_3 is a tuple $SP_{SC} =< A_{pre}, G, tran, type, CM, SCN >$ where:

- $CM = CM_1 \cup CM_2 \cup CM_3$
- $SCN = (P, T, F, \Sigma, C)$ such that
  - $P = P_1 \cup P_2 \cup P_3 \cup i, o_1, o_2$,
  - $T = T_1 \cup T_2 \cup T_3 \cup \{(t, i_1), (t, o_1), (t, o_2)\} \cup \{t\}$,
  - $F = F_1 \cup F_2 \cup F_3 \cup \{(t, i_1), (o_1, t), (t, i_2), (o_2, t), (t, i_3), (o_3, t)\}$,
  - $i = \{i_1, o_1\}$ and $o = \{o_1, o_2, o_3\}$,
  - $P_I = P_{I_1} \cup P_{I_2} \cup P_{I_3} \cup \{(o_1, t)\}$,
  - $P_D = P_{D_1} \cup P_{D_2} \cup P_{D_3} \cup \{(t, i_1), (o_1, t), (o_2, t), (o_3, t)\}$,
  - $\Sigma = \Sigma_1 \cup \Sigma_2 \cup \Sigma_3$,
  - $C = C_1 \cup C_2 \cup C_3$.

For any three given smart contracts (SC_1, SC_2, and SC_3), the invocation of SC_2 and SC_3 depends on the output data of SC_1. Therefore, color functions $C(o_1), C(i_2), C(i_3) \in C$ must satisfy $\{C(i_2) \cup C(i_3)\} \subseteq C(o_1)$ to activate the execution of SC_2 and SC_3. Otherwise, the split composition of smart contracts would fail. Figure 4 presents the CP-Net of split composition of smart contracts.

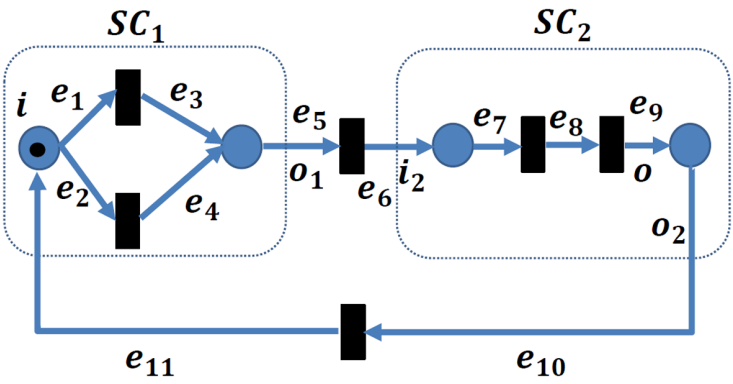

Fig. 5. Colored petri-net modelling of loop composition of SC_1 and SC_2.

Definition 3.6 (Loop Composition). A loop composition (L_{SC}) of smart contracts SC_1 and SC_2 is a tuple $L_{SC} =< A_{pre}, G, tran, type, CM, SCN >$ where:

- $CM = CM_1 \cup CM_2$
- $SCN = (P, T, F, \Sigma, C)$ such that
  - $P = P_1 \cup P_2$,
  - $T = T_1 \cup T_2 \cup \{(t, i_1), (o_2, t), t\}$,
  - $F = F_1 \cup F_2 \cup \{(t, i), (o_1, t), (t, i_2), (o_2, t), (t, i_1), (o, t)\}$,
  - $i = \{i_1, o_2, \}$ and $o = \{o_1, o_2\}$,
  - $P_I = P_{I_1} \cup P_{I_2} \cup \{(o_1, t), (o_2, t)\}$,
  - $P_D = P_{D_1} \cup P_{D_2} \cup (i_1, t), (i_2, t)$,
  - $\Sigma = \Sigma_1 \cup \Sigma_2$,
  - $C = C_1 \cup C_2$.

Given two smart contracts SC_1 and SC_2, the invocation of SC_2 depends on the output data of SC_1 and vice versa. Therefore, the color function $C(i_1), C(o_1), C(i_2), C(o_2) \in C$ must satisfy $C(i_2) \subseteq C(o_1)$ and $C(i_1) \subseteq \{C(i) \cup C(o_2)\}$. Figure 5 presents the CP-Net of loop composition of smart contracts.

4 Experimental Results and Performance Analysis

In this section, we discuss the performance of our proposed framework for modeling the blockchain-based supply chain management system. We conduct several experiments to generate different smart contract workflows and investigate the performance. We use synthetic supply chain data to generate several supply chain workflows. The performance is investigated in terms of the ability to identify workflows that are not sound. We also show the time required to check the soundness of smart contract workflows under different settings.

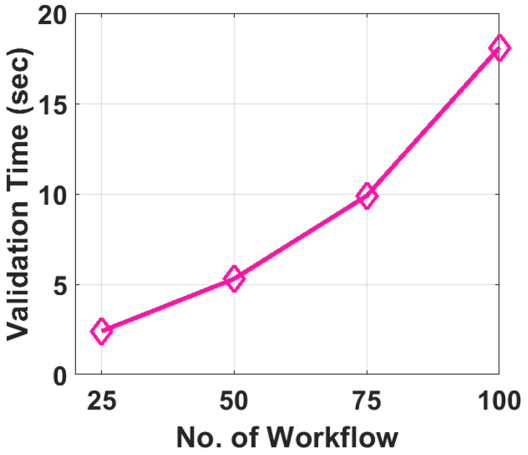

Fig. 6. Validation time (sec) for different number of smart contract workflows requested at a time.

All experiments were executed using a desktop PC with an Intel i5-6600 quad-core CPU without hyperthreading in Windows 10 operating system. We created a JAVA based server program using Apache Tomcat 8.0 to handle simultaneous requests and Petri-net based workflow validation tasks. We consider a collection of 500 smart contracts of different types from different stakeholders.

Table 1 shows the correctness of the validation process of different number of smart contract workflows. The smart contract workflows, conforming the Colored

Table 1. Correctness of soundness checking.

No. of workflow	Sound	Not sound	Correctness
25	24	1	100%
50	43	7	100%
75	69	6	100%
100	88	12	100%

Petri-Net based formalism, are referred as *sound*. Non-conforming workflows are referred as *not-sound*. Results show that our proposed framework identifies the conforming and non-conforming workflows with 100% accuracy.

Figure 6 shows the validation time for different number of simultaneous smart contract workflow validation requests. The validation time is measured in seconds. According to the result shown in Fig. 6, the validation time increases almost exponentially with the increment of validating workflows.

5 Conclusion

In this paper, we present a smart contract workflow validation model for blockchain-based supply chains for smart healthcare. At first, data-driven smart contract composition for the workflow is discussed. Next, a formal model is proposed to analyze the correctness of smart contract workflow before executing the actual blockchain network for different input data. The key benefit of the framework is that it minimizes the cost of smart contract execution. Experimental results demonstrate that the validation task can be performed efficiently for multiple simultaneous validation requests. Our formalism technique can be used to capture dynamic behaviors in any blockchain-based supply chain and other systems. However, we plan to model more complex dynamic behaviors in future work.

Acknowledgement. This work is part of the NPRP11S-1227-170135 project. The authors would like to express their gratitude to the QNRF (Qatar Foundation) for its support and funding for the project activities.

References

1. Ahmed, S., Rahman, M.S., Rahaman, M.S., et al.: A blockchain-based architecture for integrated smart parking systems. In: 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), pp. 177–182. IEEE (2019)
2. Bhattacharjee, A., Badsha, S., Shahid, A., Livani, H., Sengupta, S.: Block-phasor: a decentralized blockchain framework to enhance security of synchrophasor. In: IEEE Kansas Power and Energy Conference, Manhattan, Kansas, USA (2020)
3. Bistarellia, S., Mazzanteb, G., Michelettib, M., Mostardab, L., Sestilib, D., Tiezzib, F.: Ethereum smart contracts: analysis and statistics of their source code and opcodes. Internet Things **11**, 100198 (2020)
4. Duo, W., Xin, H., Xiaofeng, M.: Formal analysis of smart contract based on colored petri nets. IEEE Intell. Syst. **35**, 19–30 (2020)
5. Entezari-Maleki, R., Etesami, S.E., Ghorbani, N., Niaki, A.A., Sousa, L., Movaghar, A.: Modeling and evaluation of service composition in commercial multiclouds using timed colored petri nets. IEEE Trans. Syst. Man Cyber. Syst. **50**(3), 947–961 (2020)
6. Kudva, S., Badsha, S., Sengupta, S., Khalil, I., Zomaya, A.: Towards secure and practical consensus for blockchain based vanet. Inf. Sci. (2020). https://doi.org/10.1016/j.ins.2020.07.060

7. Kudva, S., Norderhaug, R., Badsha, S., Sengupta, S., Kayes, A.: Pebers: practical Ethereum blockchain based efficient ride hailing service. In: IEEE International Conference on Informatics, IoT and Enabling Technologies (2020)
8. Liu, Z., Liu, J.: Formal verification of blockchain smart contract based on colored petri net models. In: 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), vol. 2, pp. 555–560. IEEE (2019)
9. Maskey, S.R., Badsha, S., Sengupta, S., Khalil, I.: Bits: blockchain based intelligent transportation system with outlier detection for smart city (2020)
10. Rahman, M.S., Khalil, I., Arachchige, P.C.M., Bouras, A., Yi, X.: A novel architecture for tamper proof electronic health record management system using blockchain wrapper. In: Proceedings of the 2019 ACM International Symposium on Blockchain and Secure Critical Infrastructure, pp. 97–105 (2019)
11. Ream, J., Chu, Y., Schatsky, D.: Upgrading blockchains: smart contract use cases in industry. Retrieved December **12**, 2017 (2016)
12. Scholl, H.J., Pomeshchikov, R., Rodríguez Bolívar, M.P.: Early regulations of distributed ledger technology/blockchain providers: a comparative case study. In: Proceedings of the 53rd Hawaii International Conference on System Sciences (2020)
13. Swan, M.: Blockchain: Blueprint for a New Economy. O'Reilly Media Inc, Sebastopol (2015)
14. Tan, W., Fan, Y., Zhou, M., Tian, Z.: Data-driven service composition in enterprise SOA solutions: a petri net approach. IEEE Trans. Autom. Sci. Eng. **7**(3), 686–694 (2009)
15. Tian, F.: An agri-food supply chain traceability system for china based on RFID & blockchain technology. In: 2016 13th International Conference on Service Systems and Service Management (ICSSSM), pp. 1–6. IEEE (2016)
16. Vakilinia, I., Badsha, S., Arslan, E., Sengupta, S.: Pooling approach for task allocation in the blockchain based decentralized storage network. In: 15th International Conference on Network and Service Management, IEEE (2019)
17. Yue, X., Wang, H., Jin, D., Li, M., Jiang, W.: Healthcare data gateways: found healthcare intelligence on blockchain with novel privacy risk control. J. Med. Syst. **40**(10), 218 (2016). https://doi.org/10.1007/s10916-016-0574-6
18. Zhang, L., Yao, S.: Using the c-net for formalizing workflow patterns. In: 2010 Second International Conference on Information Technology and Computer Science, pp. 102–105 (2010)
19. Zupan, N., Kasinathan, P., Cuellar, J., Sauer, M.: Secure smart contract generation based on petri nets. In: Rosa Righi, R., Alberti, A.M., Singh, M. (eds.) Blockchain Technology for Industry 4.0. BT, pp. 73–98. Springer, Singapore (2020). https://doi.org/10.1007/978-981-15-1137-0_4