Quantitative cooperation analysis among cross-chain smart contracts

Hong Su1,2 · Bing Guo1 · Junyu Lu1 · Xinhua Suo1

Received: 20 June 2021 / Accepted: 14 January 2022 / Published online: 18 February 2022
© The Author(s), under exclusive licence to Springer-Verlag London Ltd., part of Springer Nature 2022

Abstract
In cross-chain scenarios, there are different blockchains that need cooperation. The cooperation between different blockchains is completed through smart contracts, which jointly complete cross-chain tasks. When numerous cooperative smart contracts are involved, smart contracts form a complex interactive network, which makes it difficult to evaluate the cooperation. A general model is needed to quantitatively analyze the cross-chain cooperation of associated smart contracts. In this paper, we model the cooperation among smart contracts as conditions and their corresponding actions, the quantitative condition-trigger model. Then, a method of calculating trigger probability by using graph weight is proposed. As the edge weight lacks the information of interaction probability, we introduce the dimension of the edge weight to calculate the interaction probability. The results show that the proposed method can effectively analyze the cross-chain cooperation between smart contracts.

Keywords Cross-chain cooperation · Quantitative condition-trigger model · All-trigger · Dynamic-trigger

1 Introduction

In cross-chain scenarios, a system may consist of different blockchains due to various purposes. In order to cooperate with different blockchains, cross-chain interoperability is introduced [1, 2] to ensure that the operation or state changes in one blockchain lead to operation or state changes in other blockchains. Operations and the state changes can be abstracted to actions. Then, the cross-chain interoperability is that the source action triggers the target action. There are different relationships between the source actions and the target actions. One (source) action triggers one (target) action (one-to-one); one action triggers several actions (one-to-many); several actions trigger one action (many-to-one); or several actions trigger several other actions (many-to-many).

Taking Fig. 1 as an example, it is a system for processing foods (fruit) data. The system consists of four blockchains: a sensor blockchain, a delivery blockchain, a monitoring blockchain, and a customer blockchain. Multiple blockchains are used mainly for two reasons: (1) different blockchains have different functions. The monitoring blockchain is usually a permissioned blockchain composed of nodes from the government, which is intended for auditing and is not suitable to consist of nodes from the sensor blockchain. (2) Multiple blockchains increase the processing capabilities, as there may be many blockchains in a city. If one blockchain is used for all, the processing capability is limited. Now, we analyze the trigger relationship. The sensor blockchain processes the planting data obtained from crop sensors. Then, the fruits enter the delivery system, and the corresponding information enters the delivery system and the food monitoring system at the same time. This is a one-to-many relationship. Finally, the information of these two blockchains goes to the customer blockchain, which provides information for customers to judge and select fruits. This is a many-to-one format.

In the above format, the triggering relationship is predefined. In other cases, the source action only triggers some candidate actions in other blockchains dynamically (this is called dynamic-trigger and the predefined trigger is called the all-trigger; they are further described in Sects. 3.2.1 and 3.2.2). Li et al. [3], Gačić et al. [4], Dang et al. [5] propose to use multiple blockchains (or sidechains) to
increase the processing capability. Then, one problem is how to select a candidate blockchain when a task arrives, that is, the load balancing issue of blockchains. There are two requirements for selecting a candidate blockchain. (1) A candidate blockchain should have the capacity to further process, because the load of a blockchain changes over time, which depends on their unpredictable dynamic processing status. For example, the total number of unconfirmed transactions (TNUT) can be regarded as the current load of a blockchain. Even for the bitcoin, its TNUT changes from about 1,000 transactions to more than 135,000 transactions between April 7, 2021, and September 30, 2021. (2) Each candidate blockchain should have an equal opportunity to run the target action. Then, the load of each candidate can be balanced. Otherwise, if all requests are sent to a blockchain that blockchain may be busy processing these transactions. With these two requirements, a candidate blockchain should be dynamically selected according to its current load, and we abstract this kind of trigger as a dynamic-trigger.

Actual systems may contain a combination of the above triggers. Predefined triggers are combined with dynamic-triggers, and even triggers are cascaded. In Fig. 1, action $a_1$ triggers action $a_2$ and $a_3$ as predefined; while it is dynamic to select of one of actions $a_{41}$ and $a_{42}$ to analyze the data from sensors. Due to the complexity of triggering, it is important to analyze the relationship between related smart contracts. (1) The analysis helps us to understand whether related blockchains are cooperative or not. For example, it helps us to know whether $a_1$ triggers $a_3$ or not. (2) It provides the ability to analyze systems that dynamically add certain actions. In this case, actions are dynamically added to the running systems, which requires understanding whether the new action cooperates with others or not. (3) It helps to understand where the bottleneck of a system is, as quantitative analysis of the system helps to identify the bottleneck.

Blockchain networks become more and more complex when using dynamic and pre-defined blockchain systems. However, at present, the relationship among these blockchains is predefined and case-specific, lacking a quantitative method for analyzing cooperations between these systems. In graph theory, network flow is used to qualitatively analyze the node relationship using the edge weight. However, it is difficult to distinguish between predefined trigger and dynamic-trigger, because all pre-defined triggers are due to occur and the dynamic-trigger is selected to occur. Therefore, we add the dimensions to distinguish the two kinds of triggers as described in Sect. 3 and use it as a quantitative method to reflect the relationship among actions (trigger probability). Besides the introduction of weights with dimension, the concept of calculation precedence is also proposed for selecting triggers to calculate their probability when among several trigger pairs are involved.

In this paper, we focus on cooperation between different blockchains, including different trigger models and the quantitative analysis of the cooperation between different blockchains. The main contributions are as follows.

(1) We propose a method for analyzing the probabilities of cooperation between blockchains, which is calculated by graph weight with dimensions. As the trigger relationship can form a complex network, if we use traditional methods to calculate the cooperation probabilities between each paired action, the calculation process is complex and even difficult to analyze the whole system. Therefore, we adopt the graph theory to analyze cooperation. An action can be viewed as a graph node and its interaction with other actions as edges. These nodes and edges form a graph. However, the network flows in the graph theory cannot reflect the probabilities of blockchain cooperation (referring to Sect. 3.2.2 for more details). Therefore, we introduce the weight dimension and the derived dimension to calculate the cooperation probabilities.

(2) We propose the quantitative condition-trigger model as a basic interaction model among cross-chain actions, which helps us to analyze the relationship between cross-chain actions through network weights with dimension rather than the details of each action (for example, which blockchain the action belongs to). If one action triggers

---

1. [https://www.blockchain.com/charts/mempool-count](https://www.blockchain.com/charts/mempool-count).
2.2 Related work

Cross-chain interoperability ensures correct execution of actions on different blockchains. Hash-locking [13, 14] aims to ensure atomic exchange between blockchains, which requires a specific sequence of steps. In the initial step, the corresponding assets are locked on different blockchains; in the second step, the first user attempts to acquire the assets he exchanges by exposing its own secret; and later, the second user can retrieve the assets it exchanges through the secret. To exchange data between blockchains [15, 16], it requires to ensure the correct execution sequence of actions. To ensure the correctness of cross-chain payment protocols, a new specification formalism formalizes its protocols and ensures that these protocols work correctly even when clock skews exist [17]. These works focus more on the correct synchronization of actions between different blockchains to achieve a specific goal; while they lack a common model to analyze the relationship between actions used in the cross-action interoperability.

In the early stage of cross-chain interoperability, the relationships between cross-chain actions are determined, in which actions have no competition. It can be seen in the bidding process of an e-commerce exchange [18, 19] or in the energy trading process [20, 21]. Participants and their actions are determined, and it can be abstracted into an all-trigger model. Later, a multiple-blockchain solution has been used to solve the scalability issue. To meet industrial standards, work [3] proposes a novel blockchain architecture, which allows multiple blockchains with different consensus algorithms to run in parallel. Gaži et al. [4] propose sidechains to increase the transaction throughput by offloading processing from the mainchain to its sidechains. Dang et al. [5] propose the use of sharding to increase the transaction throughput by loading tasks to one of several candidate blockchains. They are the detailed format of the dynamic-trigger, while they lack corresponding quantitative analysis model.

There are also works for information propagation between blockchains. Different blockchains exchange information via a blockchain router or a special network. A blockchain router behaves like Internet router [1, 2], with one central ‘hub’ blockchain connecting other blockchains. Blockchains communicate through the ‘hub’ blockchain. Blockchains can also exchange information through a special network. Cross-chain exchange methods [2, 22] are by special roles of validator, connector, surveillant, and nominator, with the aim to keep the transactions atomic or ensure data validity. These methods are the basis for cross-chain communication and facilitate the transfer cross-chain information.
3 Trigger model

The cooperation between smart contracts is done through the interaction of actions, that is, whether actions trigger other actions. In this section, we describe the condition-trigger model and different types of (action) triggers. The triggers can be direct or indirect.

(1) Direct-trigger actions are actions that interact without the help of other actions. In an all-trigger, if the source action is triggered, the target action will be triggered. In a dynamic-trigger, an action only has the probabilities of triggering other actions, and therefore, we need to evaluate the trigger probabilities.

(2) Indirect-trigger actions are actions that are not invoked directly, and their interaction is through other actions. For such trigger, we need to know whether an action leads to the expected action in an indirectly-triggered action. Direct-triggers can be cascaded to form an indirect-trigger.

3.1 Condition-trigger model

Since the action of a smart contract is done under certain conditions (requests from other smart contracts), we propose the model of condition and trigger, called the condition-trigger model. In this model, an action in a blockchain is triggered by its condition. If another blockchain wants to trigger an action, it should do something to match the condition of the target action. If the action always occurs, the condition can be set as true.

In Fig. 2, action b has the condition cb. Action a attempts to trigger action b with condition cb. After passing the condition to blockchain n, action b is triggered. As there may be complex condition-trigger relationships between blockchains, it is important to analyze their relationships.

Now, we formally describe conditions and triggers. A condition (suppose c) is generated by its action (suppose a), and the condition (c) triggers another action (suppose d'), which is shown in (1).

\[
\begin{align*}
\text{generate} & \quad a \rightarrow c \quad \text{(1)} \\
\text{trigger} & \quad c \rightarrow d' 
\end{align*}
\]

where c means a condition, and a, d' mean two actions.

In (1), there are keywords of “generate” and “trigger”. For simplification, these two keywords are omitted. An arrow with the default behavior indicates that a condition triggers an action or an action generates a condition.

To describe the relationship among conditions and actions, we introduce set of conditions and actions. The notation of C is as the set of conditions in (2), and A is as the set of actions in (3).

\[
\begin{align*}
C &= \{c_1, \ldots, c_i, \ldots, c_n\} \quad \text{(2)} \\
A &= \{a_1, \ldots, a_i, \ldots, a_n\} \quad \text{(3)}
\end{align*}
\]

where \( c_i \) is a condition in a cross-chain cooperation.

\[
A \rightarrow C \quad \text{(4)}
\]

Now, we define the condition set \( C_a \) of an action \( a_i \) as (5), and the action set \( A_c \) of condition \( c_i \) as (6).

\[
\begin{align*}
C_a &= \{c_i | c_i \in C \text{ and } c_i \text{ is a condition generated by } a_i\} \quad \text{(5)} \\
A_c &= \{a_i | a_i \in A \text{ and } a_i \text{ is an action triggered by } c_i\} \quad \text{(6)}
\end{align*}
\]

If we put the above two expressions together, we get (7), which specifies that two actions can be associated with a condition. In (7), \( a' \) is the source action that triggers other actions, and a is the target action that is triggered by other actions.

\[
a' \rightarrow a \quad \text{(7)}
\]

Since a condition is triggered by an action, we can omit the condition and describe the relationship between different actions directly. Similarly, we define the (trigger) target set \( T_a \) of an action \( a_i \) as (8), and the (trigger) source set \( S_a \) of action \( a_i \) as (9).

\[
\begin{align*}
T_a &= \{a_j | a_j \text{ is triggered by conditions generated by } a_i\} \quad \text{(8)} \\
S_a &= \{a_j | a_j \text{ generates the conditions that triggers } a_i\} \quad \text{(9)}
\end{align*}
\]

The relationship among trigger pairs can be represented graphically. Its nodes are actions, and its edges are direct triggers from the source action to the target action. All conditions and actions form a directed acyclic graph called the trigger graph. In a trigger graph, an action may trigger multiple actions, or an action is triggered by multiple conditions. (Each condition can trigger this action.) A trigger graph is shown in Fig. 3.

Cooperation relationships are visible in a trigger graph. Such as whether two actions are directly triggered (a direct trigger pair) or indirectly triggered (an indirect trigger pair). In Fig. 3, \( a_1 \) and \( a_2 \) are an indirect trigger pair because \( a_3 \) is between them. \( a_3 \) and \( a_7 \) are a direct trigger pair. If there are both direct and indirect triggers between...
two actions, we call them the hybrid trigger pair, as \( a_1 \) and \( a_5 \).

The above types of actions are related because one action can trigger another. Actions are associated when there is a directed path from one action to another, and this path is called a trigger path. Through the trigger path, two associated actions can be defined as in formula (10), where \( a_i \) triggers \( a_j \).

\[
\exists p, a_i \in tp, a_j \in tp
\]

where \( tp \) is a trigger path.

Similarly, we define non-associated actions. If there is no trigger path among action \( a_i \) and action \( a_j \), these two actions are not associated.

The above qualitatively describes the cooperation, which specifies that one action will trigger another one. However, in some cases, one action only has the probability to trigger another one, and there is no guarantee that one action must trigger another action. We need to quantitatively evaluate the probability that one action will trigger another action, which is called quantitative cooperation analysis.

To quantitatively analyze the cross-chain actions, we further adjust the trigger graph. Edge weights represent the probability that one action triggers another action. We first analyze the basic models between direct trigger pairs and then extend it to analyze indirect trigger pairs.

It is difficult to adopt the joint probability method [23] to analyze cooperation between actions. The reason is that there may be multiple paths from the source action (\( a_i \)) to the target action (\( a_j \)); in each path, \( a_i \) trigger \( a_j \), which may result in the sum of these trigger probabilities being bigger than 1 (referring to part 3.2.2 for more details). Meanwhile, it is not a single joint probability instead it is the combination of several joint probabilities. As these actions may form a complex interactive network, it is difficult to iterate and analyze all combinations. Then, in this paper, we utilize trigger graph to analyze.

### 3.2 Trigger types

In this section, we describe the basic trigger model between direct actions, the dynamic-trigger, and the all-trigger.

#### 3.2.1 Dynamic-trigger

In some cases, multiple actions are candidates to be triggered by a condition, while the number of actually triggered actions is limited. This can be used for load balancing between multiple blockchains as described in Sect. 1. In other words, not all direct trigger actions allow simultaneous triggering.

This kind of trigger is called the dynamic-trigger model, where one action sends a trigger condition to other actions, and only one of these actions is allowed to run. When there are \( n \) candidate actions, the probability of each action to run is \( 1/n \), as in (11). In this paper, we limit only one action that can be triggered for simplification. If several actions (assuming \( k \) actions) can be triggered, we should assign a probability \( k/n \) to each action.

\[
\forall a_j \in T_{a_i}, w_e = 1/n
\]

where \( e \) is the edge from \( a_i \) to \( a_j \), and \( w_e \) is its weight.

Figure 4 is a dynamic-trigger, in which action \( s \) triggers several actions from \( t_1 \) to \( t_n \). For each edge, we give a weight of \( 1/n \) to indicate that this action triggers one of its target actions.

#### 3.2.2 All-trigger

In this model, an action triggers all its target actions. All target actions run when the trigger condition matches. Figure 5 shows an example, in which action \( s \) trigger actions from \( t_1 \) to \( t_n \).

We still analyze this case by the edge weight. Suppose state \( s \) is the source action. Intuitively, a weight of 1 should be assigned to each edge. However, each can be triggered separately, we give each edge a weight of 1 with a different
dimension. This kind of dimension is called the (edge) weight dimension. The name of a weight dimension is formatted as \( (12) \). For example, \( ds_2 \) indicates that it is the dimension of the second trigger of action \( s \), and we just use \( ds \) if the sequence is not necessary to mention.

\[
\text{d}_{\text{<actionname> }<\text{sequence}>} \quad \text{or} \quad \text{d}_{\text{<actionname>}} \quad \quad (12)
\]

Different weight dimensions are required because the weights from an all-trigger cannot be summed directly. Figure 6 is one example. Action \( s \) (all-trigger) triggers actions from \( m_1 \) to \( m_i \). Action \( m_1 \) (dynamic-trigger) triggers \( l_i \) with 1/2 probability as it only selects one of two actions (\( l_i \) and \( l_j \)) to run. Similarly, action \( l_k \) also has 1/2 probability to run. If we directly add them, \( l_o \) is triggered with \((1/2+1/2)\), which is incorrect as there are probabilities to run action \( l_j \) or \( l_i \).

3.3 Trigger combinations

Dynamic-trigger and all-trigger can be combined. Here, we give four possible combinations, all–all, all–dynamic, dynamic–all, and dynamic–dynamic combinations.

3.3.1 All–all combination

In this form, there are two cascaded all-trigger types as shown in Fig. 7. Two related weight dimensions are generated, and we introduce binary operation ‘#’ that means a new weight dimension (\( d_{nd} \)) is generated from the old weight dimension (\( d_{nd} \)), which forms (\( d_{nd}#d_{nd} \)). \( d_{nd} \) is the child weight dimension of \( d_{nd} \), and \( d_{nd} \) is the parent weight dimension of \( d_{nd} \). In Fig. 7, action \( r \) and \( v \) are target actions of the second all-trigger of action \( s \). Weight dimension \( d_{ni} \) is generated on the first trigger. Weight dimensions \( d_{ni} \) and \( d_{ni} \) are generated at the second trigger \( m \), which results weights of \( d_{ni}#d_{ni} \) and \( d_{ni}#d_{ni} \). Assuming the weight of the first action \( s \) is \( d \) (following cases are also based on this assumption), the weight of action \( r \) (a target action of \( m \) is
shown in (13). Notice the weight of an action is the weight of its incoming edge in the trigger graph, indicating the probability of the action to be triggered.

\[
d\xrightarrow{\text{all-trigger}} d\#d_{s_1} \quad d\xrightarrow{\text{all-trigger}} d\#d_{s_1} \#d_{m_1}
\]  

(13)

where \(d\) is the weight of action \(s\), and the arrow means that the former action triggers the latter action.

### 3.3.2 All–dynamic combination

In this type, an all-trigger is followed by a dynamic-trigger. Different dimensions are formed in the all-trigger, and weights are divided in the dynamic-trigger. An example is shown in Fig. 8.

In Fig. 8, it shows that the source action is \(s\) and its target actions of the all-trigger are \(m_1, m_2, \text{and so on}\). \(m_1\) has a dynamic-trigger. Since the number of \(m_1\)’s target actions in the dynamic-trigger is \(n\), the weight of each action is \(1/n\) of \(d_{s_1}\), and we notate it as \(d_{s_1} (1/n)\). Here, we place \(1/n\) after the dimension symbol instead of as its coefficient. The reason is that this format keeps the branch information of this dynamic-trigger, which will be used in future calculation. The weight of action \(l_i\) (a target action of \(m_1\)) is shown in (14).

\[
d\xrightarrow{\text{all-trigger}} d\#d_{s_1} \quad \text{dynamic-trigger} \quad d\#d_{s_1} (1/n)
\]

(14)

where \(d\) is the weight of action \(s\).

### 3.3.3 Dynamic–all combination

In this type, the first trigger is a dynamic-trigger, and the second trigger is an all-trigger. The dynamic-trigger divides the trigger probabilities of the former. The edge weight change process is shown in (15).

\[
d\xrightarrow{\text{dynamic-trigger}} d(1/n) \quad \text{all-trigger} \quad d(1/n)\#d_{s_1}
\]

(15)

where \(d\) is the weight of action \(s\).

![Fig. 8 A combination of all-trigger and dynamic-trigger type. Source action \(s\) first generates new weight dimensions \(d_{s_1}, d_{s_2}\) and \(d_{s}\) in an all-trigger. One of its target actions \(m_1\) generates a dynamic-trigger to produce \(d_{s_1} (1/n)\)](image)

In expression (15), the weight is changed to \(1/n\) in the dynamic-trigger, and a new dimension is generated in the subsequent all-trigger. The edge weight changes from \(d\) to \(d(1/n)\#d_{s_1}\).

### 3.3.4 Dynamic–dynamic combination

In this case, there are two successive dynamic-triggers, each of which divides the original value. Assume the first dynamic-trigger is between \(n\) paired actions and the second is between \(m\) paired action. Then, the weight change process is shown in (16).

\[
d\xrightarrow{\text{dynamic-trigger}} d(1/n) \quad \text{dynamic-trigger} \quad d(1/mn)
\]

(16)

where \(d\) is the weight of action \(s\).

### 4 Trigger probabilities

In Sect. 3, we describe the trigger probabilities between basic trigger pairs and their combinations, where there is only one path from the source action to the target action. While in a complex network, there may be multiple paths from the source action to the target action, which results in different combinations of the weight dimensions. In this section, we describe how to calculate trigger probabilities between two actions when multiple trigger paths exist.

#### 4.1 Assumptions

To make our analysis more focused, there are two assumptions. (1) First one is that each condition \(c_j\) of \(C_{a_i}\) can trigger the target action \(a_i\), which is called the independent trigger assumption. There is no requirement for the combination of two or more conditions to trigger a target action. This means we only discuss conditions where the relationship is or. If an action is triggered by a combination of multiple conditions, the relationship between them is and; the analysis is similar to the or relationship, and we discuss it in Sect. 4.5. (2) The second one is that there is no loop between actions, the loop assumption. If one action (\(a_i\)) triggers another action (\(a_j\)), \(a_j\) will not trigger back to \(a_i\). If there is a finite loop, we can replace the loop with several repeating condition-trigger pairs.
4.2 Probabilities calculation

4.2.1 Relationship among different combinations of weight dimension

After traversing the trigger graph, we may get multiple weights from different paths. We put them in parentheses and call it the weight array of the result. Here is an example.

\[(D_1(x), D_2(y), \ldots)\]

where \(D_1, D_2\) are dimensions, and \(x, y\) are probabilities.

The trigger probability can be calculated from the weight array of the result. However, their weight dimensions may be different. Before calculating, we need to discuss the relationship between different weight dimensions. We start with the definition of the dimension tuple, as in Definition 1.

**Definition 1** Let \(w\) be a weight of an action. Its **dimension tuple** \((D)\) is a tuple composed of all the sub-dimensions of \(w\). Its **dimension length** of \(D\) is the number of dimensions in \(D\). Its **sub-dimension tuple** \((S)\) is a subset of \(D\), where \(S\) is either one or more adjacent dimensions of \(D\). Dimensions \(D_1\) and \(D_2\) are adjacent dimensions of \(w\), which means that there is no other dimension(s) between \(D_1\) and \(D_2\) in \(w\).

For example, a weight is \(d_a(1/2) \# d_b(1/4) \# d_c\). Its dimension tuple is \(\{d_a, d_b, d_c\}\), and its sub-dimension tuples are \(\emptyset, \{d_a\}, \{d_b\}, \{d_c\}, \{d_a, d_b\}, \{d_a, d_c\}, \{d_b, d_c\}\), and \(\{d_a, d_b, d_c\}\). Or we can also represent these sub-dimension tuples as \(\emptyset, d_a, d_b, d_c, d_a \# d_b, d_a \# d_c, d_b \# d_c, \) and \(d_a \# d_b \# d_c\). The latter format is more convenient in this paper.

Based on the sub-dimension tuple, we introduce the concept of the common starting sub-dimension tuple or the common sub-dimension tuple.

**Definition 2** Let \(D_1, D_2, D_3\), and so on are several dimension tuples. Their **common sub-dimension tuple** is a sub-dimension tuple \(D\), which matches the following criteria.

\[D_1 = D \# D'\]
\[D_2 = D \# D''\]
\[D_3 = D \# D'''\]

... where \(D, D', D'',\) and \(D'''\) are different sub-dimension tuples.

For example, given weights of \(d_a \# d_b \# d_a\) and \(d_a \# d_b \# d_c\), we can choose a common sub-dimension tuple \(D\) as \(d_a \# d_b \# d_c\). These expressions can be expressed as \(D \# d_a\) and \(D \# d_c\).

Meanwhile, for the convenience of analysis, we introduce the weight dimension of weight \(w\) at position \(i\) (the position is 1-based), notated as \(d(w, i)\). For example, assume weight \(w\) is \(d_a(1/2) \# d_b(1/4) \# d_c\); then \(d(w, 1) = d_a\), and \(d(w, 3) = d_c\).

4.2.2 Calculation among the same dimension

In this case, all weights are of the same dimension. Assume the dimension is \(D\), and weights are \(D(x), D(y),\) and \(D(z)\). They are results of one or more dynamic-triggers after \(D\). Otherwise, if they encounter any all-trigger after \(D\), there is at least a dimension difference. The calculation is to directly add them, as shown in (17).

\[(D(x), D(y), D(z)) \rightarrow D(x + y + z)\]  

This format can be extended. Suppose there are two dimensions \(w'\) and \(w''\) as in (18).

\[w' = D_1(x_1) \# D_2(x_2) \# D_3(x_3) \# \ldots \# D_n(x_n)\]
\[w'' = D_1(y_1) \# D_2(y_2) \# D_3(y_3) \# \ldots \# D_n(y_n)\]

As they have a common sub-dimension tuple \(S\) as in (19), they be calculated as in (20).

\[S = D_1 \# D_2 \# D_3 \# \ldots \# D_n\]
\[w = w' + w''\]
\[= D_1(x_1 + y_1) \# D_2(x_2 + y_2) \# D_3(x_3 + y_3) \# \ldots\]

We explain this process with Fig. 9. The original trigger relationship is in the upper part. In the first step (the middle part), we directly sum the trigger probabilities \(D_1(x_1)\) and \(y_1)\), because they are from branches of the same dynamic-trigger(s). The lower part is step 2, where \(x_2\) and \(y_2\) are calculated similarly. Then, all independent possibilities are calculated in this way.

The proof for this calculation is in two aspects. (1) For the same trigger pair, the trigger probabilities should be...
summed. For a trigger pair, different trigger probabilities are different trigger paths from the source action to the target action. For example, in Fig. 9, action $D_1$ triggers $D_2$ in two paths with probabilities of $y_1$ and $x_1$, respectively, which means that action $D_2$ can be triggered $y_1$ and $x_1$ times in one trigger. These two paths have no dependence, and then, the final trigger probability is their sum. (2) The trigger relationship remains unchanged. After calculation, the trigger relationship should not change because we only merge several trigger paths to a single path instead of changing the trigger relationship. The final trigger path is still the source action triggering the target action. For example, in Fig. 9, action $D_2$ triggers action $D_3$; after the calculation, the trigger relationship is the same (action $D_2$ to trigger action $D_3$). Then, we get the conclusion that both trigger relationships and trigger probabilities remain the same after the calculation of the same dimension.

With this calculation, the same weight arrays have been calculated.

### 4.2.3 Calculation among dimension and its derived dimension

In this case, the weight calculation is performed between a dimension and its derived dimension. Assume two inputs are $D(x_1)$ and its derived dimension $D(x_2)$, and its derived dimension tuple is $D$ and its derivation $D#d_k$.

For this case, (1) the two weights are from two branches of a dynamic-trigger, and (2) one of its branches passes through an all-trigger, which leads to $D(x_2)$, $D#d_k$. The corresponding dimension tuple is $D$ and its derivation $D#d_k$.

As these two actions come from two dynamic branches of an action, either of them can trigger the target action. The probability of the target action to be triggered is the sum of the probabilities of the two inputs, as shown in (21).

\[(D(x_1), D(x_2), D#d_k) \rightarrow D(x_1 + x_2 + y) \quad (21)\]

Brief proof: These trigger probabilities are (1) from the different branches of a random path to generate two trigger paths and (2) an action encounters an all-trigger to generate the derived dimension. Because they are not two paths of the same all trigger, they are still independent. Then, the final trigger probability is their sum. After calculation, the derived dimension is eliminated, and other dimensions and their values are not changed, meaning that this calculation does not affect other trigger relationships. Therefore, the trigger probabilities are merged in this calculation while maintaining other trigger relationships.

### 4.2.4 Calculation among different dimensions of one action

Input weights are branches of an all-trigger, such as $D#d_k(x), D#d_k(y), D#d_k(z)$. This case is characterized by: (1) inputs are branches of an all-trigger to generate different dimensions ($d_k$, $d_k_2$, $d_k_3$); (2) inputs may later encounter a dynamic-trigger to change the coefficient ($x$, $y$ and $z$ may not be equal). Thus, the probability is the probability of complements that no action is triggered, as in (22).

\[(D#d_k(x), D#d_k(y), D#d_k(z)) \rightarrow D(1 - (1 - x) + (1 - y)(1 - z)) \quad (22)\]

Brief proof: when D is triggered, its branches $d_k_1$, $d_k_2$, $d_k_3$, etc., are triggered. These branches are dependent as their existence may overlap, which prevents these trigger probabilities from directly adding up. Conversely, the trigger probability is that the opposite when none of them happen. Meanwhile, after calculation, only one action’s dimension is merged, and other trigger relationships are not changed, which means this calculation does not impact other trigger relationships.

### 4.2.5 Calculation among different dimensions

In this case, inputs are from different paths. The characteristics of this case are as follows. For any inputs $i_1$ and $i_2$, $d(i_1, 1)$ and $d(i_2, 1)$, they are not from an all-trigger but from a dynamic-trigger. An example is shown in Fig. 10. Different branches of a dynamic-trigger can be triggered independently. Then, the trigger probability is the sum of all probabilities, as shown in (23).

\[(D_1(x), D_2(y), D_3(z)) \rightarrow (x + y + z) \quad (23)\]

Brief proof: These trigger probabilities are from different trigger paths and have no correlation, which results in that

![Fig. 10](image-url)

Fig. 10 Paths with different dimensions. Input 1 and input 2 are from two different paths that have different dimensions
these triggers are independent. Then, the final trigger probability is their sum. However, in this calculation, the dimension information (trigger relationship) may be lost, and then, we do this calculation after other kinds of calculation as in Sect. 4.3.

4.3 Calculations precedence of probabilities

Trigger probabilities can be calculated using the above methods. However, different calculation order may produce different results. For example, in (24), the calculation of \( d_j(x) \# d_{k_1}(y_1) \) and \( d_j(x) \# d_{k_2}(y_2) \) is between different dimensions of one action; the calculation of \( d_j(x) \) and \( d_j(x) \# d_{k_1}(y_1) \) is between dimension and its derived dimension. These two different calculations have different results.

\[
(d_j(s), d_j(x), d_j(x) \# d_{k_1}(y_1), d_j(x) \# d_{k_2}(y_2), \ldots) \tag{24}
\]

The calculation precedence is as follows. (1) The same dimension should be calculated first. It does not change the trigger relationship because all the dimensions are preserved, which indicates that trigger relationships are preserved.

(2) Different dimensions are calculated finally. The calculation as in Sect. 4.2.5 will reduce all the dimension information (trigger relationship). Therefore, it needs to be calculated at last.

(3) The remaining calculations are different dimensions of one action (3a) or the calculation of dimension and its derived dimensions (3b), which should be between (1) and (2).

Since 3a and 3b may contain each other mutually, we calculate them in the dimension length order to select the pair that has the longest dimension length, which is called the longest dimension rule. In 3a and 3b, the dimension length is reduced by only 1 after calculation. When we select the longest ones, other trigger relationships are still kept.

Now, we prove why we choose the longest dimension rule. Trigger relationships form a tree after loop unrolling as in the assumption. The trigger probability is calculated from the longest, similar to cutting the leaves of a tree, which retains the structure of other unrelated leaves and branches. Otherwise, if we cut off the root or branch node of a tree, it will make the structure of subtree changes, that is, irrelevant trigger relationship has been changed.

4.4 Probabilities and association judgment

After the calculation, we get the final probability that an action is triggered by other actions, which we call the final trigger probability or the trigger probability. In this subsection, we use the trigger probability to analyze several aspects of cooperation.

**Association Judgment.** Assume the trigger probabilities between action \( a_a \) and action \( a_b \) is \( p_t \). If \( p_t \) is greater than 0, then \( a_a \) and \( a_b \) are associated. The trigger probability reflects how frequently one action triggers another action. The greater \( p_t \) is, more frequently \( a_b \) will be triggered after action \( a_a \). If \( p_t \) is 1, \( a_b \) will be triggered by action \( a_a \).

**Bottleneck analysis.** In some cases, there are bottlenecks that prevent the further processing of an action (assume action as). We can use the trigger probability to find the corresponding blockchains. A bottleneck blockchain has the feature that its actions have high trigger probabilities to be triggered by action as. Assuming two blockchains, BCa and BCb, they have similar processing capabilities. If the trigger probabilities of BCa are 0.95 and BCb is 0.1, we should choose to add new blockchains to reduce the load of BCa. Meanwhile, when new blockchains of BCa increase, successive blockchains may be the bottleneck blockchains, and the trigger probabilities can be used for further analysis of the successive blockchains. Therefore, we can use the trigger probabilities to discover bottlenecks and increase corresponding blockchains.

4.5 Discussion for assumption

There are two assumptions in the previous section to simplify the discussion, the looping assumption, and the independent trigger assumption, which we discuss further here.

4.5.1 Loop assumption

In the previous assumption, we assume that triggers are not in a loop, or they form a finite loop which can be replaced by duplicate condition-trigger pairs. For an infinite loop, the special point is that each trigger repeats infinitely, which makes subsequent actions bound to be triggered, even if the trigger probability is small, because a small probability multiplied by a big number (number of loops) will reach 1.

We discuss infinite loop triggers by an example in Fig. 11, where \( a_4 \) is the target trigger. Regardless of how small \( p_o \) is, if it is a deterministic value greater than 0, \( a_4 \) is triggered after \( a_1 \) is triggered. The reason is that when the number of loops (\( l_i \)) is large enough, \( l_i \times p_o \) will be equal to or greater than 1, and \( a_4 \) is triggered. Furthermore, \( l_i \times p_o \) may be bigger than 1, which means that subsequent actions are triggered more than once when \( a_1 \) is triggered.

The infinite loop then generates a trigger probability of 1, which means that when the loop is triggered (such as \( a_1 \) in Fig. 11 is triggered), subsequent actions of the loop (such as \( a_4 \)) will be triggered. Even a random trigger is
encountered, the trigger possibility remains 1, because the number of triggers is unlimited. Then, we use 1 as the trigger probability for subsequent actions. Unit 1 is used to indicate the target action is bound to be triggered.

4.5.2 Independent trigger assumption

The independent trigger assumption is that any single condition can trigger an action, where these conditions have an or relationship to trigger the action. However, in some cases, an action is triggered when several conditions match, which is an and relationship between these conditions. Conditions to trigger an action are called a trigger group.

For any trigger group, the trigger probability is their joint probability. For example, in (25), the trigger group has a trigger probability of $x_1 \times x_2 \times \ldots \times x_n$.

\[
(D_1(x_1), D_2(x_2), \ldots, D_n(x_n))
\] (25)

5 Evaluation

In this section, we try to evaluate the proposed approach. We first introduce the verification environment and then evaluate the all-trigger and dynamic-trigger, respectively.

5.1 Environment

This subsection includes two main parts, the blockchains and the communication protocol for cross-chain triggers. The communication protocol should (1) be secure and (2) be confirmed in the blockchain to maintain a consistent state. Then, we put the condition of a trigger as data in the blockchain transaction, which is confirmed by the consensus algorithm. Blockchains synchronize these data between them.

5.1.1 Blockchains

Depending on the test case, 5 or 7 blockchains are used in the evaluation. The consensus algorithms of blockchains are PoW. These blockchains are publicly available blockchains, which means that (1) the nodes of one blockchain can connect to the nodes of another blockchain through P2P methods, and (2) the nodes of a blockchain can obtain chain of blocks from other blockchains and fetch corresponding transactions for condition-triggers.

5.1.2 Connection topology

The topology among blockchains is a ring topology. There are two subtypes of topology, one with 5 blockchains, and another one with 7 blockchains, as shown in Fig. 12.

5.1.3 Data format

In the proposed condition-trigger model, the condition are generated by the corresponding action, which further triggers other actions. The condition is identified by a special starting string (“Cross-chain:trigger:condition”) and placed in the data structure of a transaction. This transaction has a special type to identify it as a cross-chain transaction. Blockchains synchronize cross-chain...
transactions rather than all transactions, aiming to reduce the amount of the data to be synchronized.

5.1.4 Data propagation

There are different ways to propagate the message between blockchains. It includes the non-blockchain intermediate method and the blockchain-based propagation method. In the non-blockchain intermediate method, the data are obtained from the source blockchain to the target blockchain by a third party (or user). This method relies on some methods to ensure the validation of the data, such as the method used in the hash-locking [13].

The blockchain-based propagation method can be based on a central blockchain, such as the blockchain router method [2], which helps to propagate transactions between blockchains. Blockchains connect to the blockchain router, publish data to the blockchain router and obtain data from the blockchain router.

Cross-chain data can also be propagated directly among blockchains instead of introducing a central blockchain. If each one connects to all other blockchains, the number of connections increases with the square of the number of blockchains. We adopt a ring topology of blockchains, where each blockchain only connects to another blockchain. Directly connected blockchains convert and seal the corresponding transaction of its adjacent blockchain. The process iterates to transfer data between indirectly connected blockchains.

In the method used in the evaluation, blockchains seal relevant data into their own blockchain after obtaining the chain of blocks from other blockchains. Miners are encouraged to mine data from other blockchains, and the incentive mechanism is adjusted accordingly. The reward for miners consists of three parts. (1) A relatively large reward is given to the miner who mines a block, as it uses heavy CPU resources to mine a block, 1 internal coin in the verification blockchain. (2) A relatively low reward is given if a new block contains transaction(s) for the condition-trigger model, 1/8 internal coin in the verification blockchain. Because it uses the network and corresponding CPU resources to process external transactions. However, this process does not require as many resources as the process of mining a block. (3) A little reward is given for each condition-trigger transaction, 0.001 coins in the verification blockchain. Because there is not much work after the synchronization and corresponding process. The reward for mining extra data \( \text{rewardExtra} \) is shown in (26).

\[
\text{rewardExtra} = 5/4 \text{coins} + 0.001 \text{coins} \times n \quad (26)
\]

where \( n \) means the number of transactions which contain data from other blockchains of condition-trigger model.

5.1.5 Limit the number actions to be triggered

Concurrency problems may occur when actions attempt to run after receiving a dynamic trigger. Assume a dynamic-trigger triggers one of multiple actions \( a_1, a_2, \) to \( a_3 \). If one of them (suppose \( a_3 \)) finds that there is no other action to run, it attempts to run; at the same time, another one (suppose \( a_3 \)) may also want to run, and no other actions are found to be running. Then, \( a_1 \) and \( a_3 \) begin to run at the same time. This causes multiple actions to run.

To avoid this, one kind of transaction (its data field begins with “Cross-chain:trigger:data”) is adopted to indicate that an action is ready to run, which is called trigger token or TT. If there is only one TT in a certain time, the corresponding action gets the token to run. If there is more than one TT sent out, we select the TT with the smallest blockchain identifier to run.

When there is only one token, its sending action should wait certain time (the waiting time) to avoid concurrency issues. The waiting time spent on propagation includes the time to transfer a TT over the network and mine it. Then, the propagation time \( t_{p-i} \) in blockchain \( i \) contains the time used in the network \( t_{n-i} \) and the confirmation time (the mining time) \( t_{c-i} \), which is shown in (27). As the ring topology is used, the waiting time \( t_w \) should be the sum of all propagation time among them, as shown in (28). As the time may change, we also add additional time \( t_a \) to ensure that the propagation and confirmation are completed.

\[
t_{p-i} = t_{n-i} + t_{c-i} \quad (27)
\]

\[
t_w = \sum_{i=1}^{n} t_{p-i} + t_a \quad (28)
\]

\( t_a \) is relatively long to ensure that corresponding information has been propagated to other blockchains. In fact, \( t_a \) can be optimized. If a smart contract does not respond to a dynamic-trigger, it can send a token denying token (TDT) to notify others. Other smart contracts run without waiting for additional time \( t_a \). Then, (28) is reformed to (29), where \( t_w \) is the maximum time of \( t_{p-i} \).

\[
t_w = \max \left( \sum_{i=1}^{n} t_{p-i} \right) \quad (29)
\]

where \( i \) stands for blockchain \( i \).

5.2 All-trigger

In this subsection, we verify the all-trigger model. The topology is a 5-blockchain topology. Three kinds of verifications are carried out, A1, A1 and A3, whose logic connections are shown in Fig. 13. Note that a logic connection shows how actions are triggered, which is different.
from the topology diagram. For example, blockchain 3 is logically triggered by blockchain 1, although blockchain 3 is not directly connected to blockchain 1 (referring to Fig. 12).

In the A1 case, a user triggers the initial action through a transaction in blockchain 1, which can be regarded as probability 1. (The user is not shown in Fig. 13.) Then, the corresponding smart contract on blockchain 2 obtains the condition and performs corresponding actions in blockchain 2. Later, blockchain 3 obtains the corresponding state from blockchain 2 and triggers the corresponding action.

A2 case is similar to A1. The difference is that the condition of an all-trigger from blockchain 1 has been sent to two blockchains, blockchains 2 and 3. The target blockchain 4 obtains two all-triggers from blockchains 2 and 3. When the first one comes, the action of blockchain 4 is triggered. In A3 case, three blockchains receive and send out the all-trigger conditions from blockchain 1.

The weight calculation is as follows, which indicates that the trigger probability of the target action is 1.

Formula (30) is for A1.

$$1 \times 1 = 1$$  

Formula (31) is for A2, in which there are two inputs ($d_{i1}, d_{i2}$) for the target smart contract on blockchain 4.

$$1 - (1 - 1) \times (1 - 1) = 1$$  

Formula (32) is for A3, in which there are three inputs ($d_{i1}, d_{i2}, d_{i3}$) for the target smart contract on blockchain 5.

$$1 - (1 - 1) \times (1 - 1)(1 - 1) = 1$$  

We carried out 27 round verifications aiming to compare their completion time. The result is shown in Fig. 14.

Although the completion time is different among A1, A2, and A3, target actions are triggered in all 27 test rounds. It indicates that a target action can be triggered along an all-trigger path. We also randomly shut down one or two blockchain nodes. The action is still triggered, which indicates that the failure of some nodes does not affect the whole blockchain. The reason is that the blockchain consensus algorithm can reach an agreement and keep the final status unified.

### 5.3 Dynamic-trigger

There are also three types of verifications for the dynamic-trigger type, named D1, D2, and D3. The logic connections are shown in Fig. 15. In the D1 case, a user triggers the initial action in blockchain 1, and blockchain 1 sends a dynamic-trigger to blockchain 2 and blockchain 3. Then, blockchain 2 or 3 wins the trigger, which sends an all-trigger to blockchain 4.

D2 case is similar to D1 in the first steps: blockchain 1 sends a dynamic-trigger to blockchain 2 and blockchain 3. Then, blockchain 2 or 3 wins the trigger, which sends an all-trigger to blockchain 4. The difference is that blockchain 2 sends an all-trigger to blockchain 5 and blockchain 3 sends an all-trigger to blockchain 4. Since only blockchain 2 or 3 gets the dynamic-trigger, the target blockchain (blockchain 5) is dynamically triggered with a probability of 1/2.
In the case of D$_3$, the first step is still similar. However, blockchain 1 sends a dynamic-trigger to blockchains 2, 3, and 4, and only blockchain 2 sends the all-trigger to blockchain 5. Since there are three blockchains to race for a dynamic-trigger, blockchain 5 is triggered dynamically with a probability of $1/3$.

The trigger results are shown in Figs. 16, 17 and 18. The y-axis is the number of test rounds, and if an action in the blockchain is triggered in a test round, we draw a small circle for the blockchain at that test round.

Figure 16 shows the results of D$_1$, from which we see that the probability of winning the dynamic-trigger from blockchain 1 is almost equal for blockchain 2 and 3. This also validates the dynamic-trigger function described in Sect. 3.2.1.

For blockchain 4, it is triggered with the probability of 1. Its weight array is $(1/2, 1/2d_{i_1})$, which can be regarded as the calculation among dimension (1) and its derived dimension (i), and then, the final weight is $1/2 + 1/2 = 1$.

For blockchain 7, it triggers only with the probability of $1/2$. Its weight array is $(1/2 * d_{i_1}, 1/2 * d_{i_2})$, and the final weight is $1/2 * (1 - (1 - 1) * (1 - 1)) = 1/2$. This indicates that dimensions from the same action cannot be
added. It also indicates that the dimensional information should be retained during the calculation.

From Figs. 17 and 18, we see that the probability of a blockchain action to run is smaller with the increase of the number of competitive actions. The trigger probability is the reciprocal of blockchain numbers, because each action has the same probability to run.

### 5.3.1 Optimization of waiting time

In this part, we try to verify the optimization of waiting time by TDT (time denying transaction). The logic connection is shown in Fig. 19. There is a dynamic-trigger from blockchain 1 to blockchain 2 and 3. We measure the effect of the optimization by the completion time of a smart contract (named SOp), which runs on blockchain 4 and can be triggered by an action from blockchain 2 or blockchain 3. In order to make the process reflect the propagation and confirmation time, the logic of SOp is simple; it only records the start time and completion time of the trigger.

Two comparative experiments, O1 and O2, are carried out. In these two experiments, we dynamically select blockchain 2 or 3 not to respond (the dynamic selection is determined by the dynamic trigger string hash), which is called non-responsive blockchain (NRB). In O1, NRB does not send TDT to notify other blockchains, and then, other blockchains wait until timeout. (The timeout value is set to 1200 seconds.) In O2, NRB sends a TDT. We have conducted 35 rounds of tests on O1 and O2, respectively. The results are shown in Fig. 20.

From Fig. 20, we see that the waiting time in O2 changes from the minimum value (212 seconds) to the maximum value (821 seconds). In O1, dynamic-triggers are done until timeout. Regardless of the performance of the blockchain network, the waiting time of O1 is still 1200 seconds. Another point is that the timeout value should be large enough to ensure that dynamic-triggers can be propagated and confirmed in other blockchains. In this verification, its value (1200 seconds) is much larger than the maximum values (821 seconds) in the O1 case.

### 6 Conclusion

In this paper, we address the issues of the triggering relationship between cross-chain smart contracts deployed on different blockchains. We first propose the condition-trigger model as the basic model of analysis and its different trigger types. Trigger types include the all-trigger and the dynamic-trigger, concerning whether it limits the number of successive smart contracts to be triggered. Second, we propose to use the weights of the graph (formed by different actions) to analyze the probability of cooperation. Since the edge weight cannot be directly used as the cooperation probabilities, we introduce the dimension in the edge weight. Finally, we evaluate the condition-trigger model and its different trigger types, the all-trigger and dynamic-trigger. The evaluation results show that our proposed methods can analyze the (action) cooperation relationship between smart contracts in different blockchains.

**Funding** This work was supported in part by the National Natural Science Foundation of China under Grant No. 61772352; National Key Research and Development Project under Grant No. 2020YFB1711800 and 2020YFB1707900; the Science and Technology Project of Sichuan Province under Grant No. 2019YFG0400, 2020YFG0479, 2020YFG0322, and the R&D Project of Chengdu City under Grant No. 2019-YF05-01790-GX.

**Availability of data and material** Not applicable

**Code availability** Not applicable

**Declarations**

**Conflicts of interest/Competing interests** The authors declare that they have no conflict of interest.

**Ethics approval** Not applicable
Consent to participate  Not applicable
Consent for publication Not applicable

References

1. Kan L, Wei Y, Muhammad A H, et al (2018) A multiple blockchains architecture on inter-blockchain communication. In: 2018 IEEE international conference on software quality, reliability and security companion (QRS-C). IEEE, pp 139–145
2. Wang H, Cen Y, Li X (2017) Blockchain router: a cross-chain communication protocol[AC]. In: Proceedings of the 6th international conference on informatics, environment, energy and applications. ACM, pp 94–97
3. Li W, Sforzin A, Fedorov S, Karame G (2017) Towards scalable and private industrial blockchains. In: Proceedings of the ACM workshop on blockchain, cryptocurrencies and contracts, pp 9–14
4. Gazi P, Kiayias A, Zindros D (2019) Proof-of-stake sidechains. In: 2019 IEEE symposium on security and privacy (SP). IEEE, pp 139–156
5. Dang H, Dinh TTA, Loghin D, et al (2019) Towards scaling blockchain systems via sharding. Proceedings of the 2019 international conference on management of data, pp 123–140
6. Kannengiesser N, Pfister M, Greulich M, et al (2020) Bridges between islands: cross-chain technology for distributed ledger technology. Proceedings of the 53rd hawaii international conference on system sciences
7. Borkowski M (2019) Cross blockchain technologies: review, state of the art, and outlook. http://dsg.tuwien.ac.at/projects/tast/pub/tast-white-paper-4. pdf. White Paper, Technische Universita¨t Wien. Version, 1
8. Zie JY, Deneuville JC, Briffaut J et al (2019) Extending atomic cross-chain swaps. In: Data privacy management, Cryptocurrencies and Blockchain Technology. Springer, Cham, pp 219–229
9. Liu Z, Xiang Y, Shi J et al (2019) Hyperservice: Interoperability and programmability across heterogeneous blockchains. In: Proceedings of the 2019 ACM SIGSAC conference on computer and communications security, pp 549–566
10. Dolgui A, Ivanov D, Potryasaev S et al (2020) Blockchain-oriented dynamic modelling of smart contract design and execution in the supply chain. Int J Prod Res 58(7):2184–2199
11. Li T et al (2021) FAPS: a fair, autonomous and privacy-preserving scheme for big data exchange based on oblivious transfer, Ether cheque and smart contracts. Inf Sci 12(544):469–84
12. Dwivedi AD, et al. Optimized blockchain model for internet of things based healthcare applications. In: 2019 42nd international conference on telecommunications and signal processing (TSP) 2019 Jul 1 (pp 135–139). IEEE
13. Dai B, Jiang S, Zhu M, et al. (2020) Research and implementation of cross-chain transaction model based on improved hash-locking. In: International conference on blockchain and trust-worthy systems. Springer, Singapore, pp 218–230
14. Herlihy M (2018) Atomic cross-chain swaps. In: Proceedings of the 2018 ACM symposium on principles of distributed computing. ACM, pp 245–254
15. Wang Q, Wang S, Zhang P et al (2019) An achieving data exchange cross-chain alliance protocol. J Phys Conf Ser 1213:042037
16. Lin W, Yin X, Wang S, et al (2020) A Blockchain-enabled decentralized settlement model for IoT data exchange services. Wireless Netw. 1–15
17. Van Glabbeek R, Gramoli V, Tholoniat P (2019) Cross-chain payment protocols with success guarantees
18. Chen YH, Chen SH, Lin IC (2018) Blockchain based smart contract for bidding system. In: 2018 IEEE international conference on applied system invention (ICASI). IEEE, pp 208–211
19. Mengelkamp E, Notheisen B, Beer C et al (2018) A blockchain-based smart grid: towards sustainable local energy markets. Comput Sci Res Develop 33(1–2):207–214
20. Karumba S, Kanhere SS, Jurdak R, Sethuvenkatraman S (2020) HARB: a hypergraph-based adaptive consortium blockchain for decentralised energy trading. IEEE Internet Things J. https://doi. org/10.1109/JIOT.2020.3022045
21. Zhang B, Jiang C, Yu JL et al (2016) A contract game for direct energy trading in smart grid. IEEE Trans Smart Grid 9(4):2873–2884
22. Qasse Ilham A, Talib Manar Abu, Nasir Qassim (2019) Inter blockchain communication: a survey. In: Proceedings of the ArabWIC 6th annual international conference research track (ArabWIC 2019). Association for Computing Machinery, New York, NY, USA, Article 2, pp 1–6
23. Kakimoto M, Endoh Y, Shin H et al (2018) Probabilistic solar irradiance forecasting by conditioning joint probability method and its application to electric power trading. IEEE Trans Sustain Energy 10(2):983–993
24. Azaïs R, Bardet JB, Génadot A et al (2014) Piecewise deterministic Markov process-recent results. In: ESAIM: Proceedings. EDP Sciences 44:276–290

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.