Specy Network: Trusted Multichain Automation With Verifiable Specifications

***Abstract*—Smart contract have expanded the application scenarios of blockchain, allowing for the construction of complex business logic. To achieve the automatic execution of contract code under predetermined conditions, we proposed a automation protocol,Specy Network. It effectively addresses the problem of automated task execution in a multichain environment and features verifiability, trustworthiness, and high performance. We have fully designed and implemented Specy Network and validated its effectiveness through concrete use case.**

***Index Terms*—Automation, Blockchain, Smart Contract, Multichain**

I. Introduction

When relevant transactions occur, smart contracts will execute the corresponding logic code to complete access and modification of business state. However, due to the characteristic that smart contracts must be triggered by transactions, they cannot be executed automatically when the conditions are met, which restricts their applications [1]. For example, we hope to automatically exchange ETH tokens for USDT of equal value when the price of ETH rises to a certain range. In this paper, we propose Specy Network, a multi-chain automation protocol based on trusted execution environment(TEE) technology and domain-specific language technology. It simplifies the creation of automated actions for decentralized applications and enables real-time triggering of on-chain state transitions in response to real-world data changes. Specy Network will significantly expand the application scenarios of blockchain technology.

II. Automation protocol for blockchain

Specy Network is built on the Cosmos ecosystem and provides services in the form of a blockchain [2]. Fig.1 shows the complete implementation of its protocol. In addition to the basic blockchain service components, chain node also includes the following three parts: (1)Operator module: provides automation task, user, and fee management capabilities, allowing users to create automation tasks and pay for them. (2)Proof verifier module: provides task prover registration and result verification capabilities. (3)Cross-chain provider module: provides data interaction capabilities across multiple chains. In the following paragraphs, we will explain how to use Specy Network to automate tasks across multiple chains in a few steps.

- First Step “To become a task prover”: After the blockchain network is launched, node owners can run specific task check engine and register in the Specy Network proof verifier module to become a prover to provide condition checking for tasks(see Fig.2).

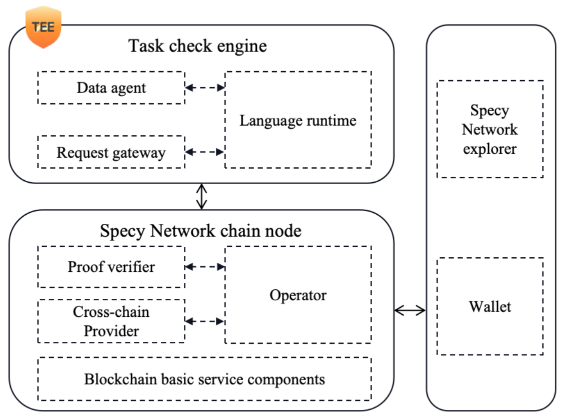


Fig. 1. Overall system architecture of Specy Network

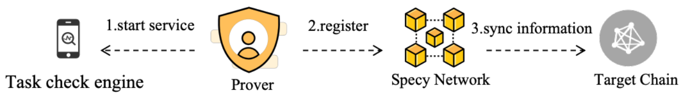


Fig. 2. Prover registration process

The task check engine program is implemented based on TEE(see Fig.1), which ensures the trustworthiness of task condition checking. It contains three key components: request gateway, data proxy, and language runtime. The request gateway is used to process requests, the data proxy is used to fetch data, and the language runtime is used to execute task condition code. The registration process is carried out in the form of a transaction, which is used to submit the public key and remote proof of the TEE to the blockchain for verification of the subsequent task check results. Relevant information will also be synchronized to the target chain through cross-chain provider [3].

- Second Step “Creating automation tasks”: Users define automated tasks based on their goals, create them through the Specy Network explorer, and submit them using wallet signatures(see Fig.1). The tasks consist of two parts:conditions and actions. To simplify the complexity of task definition, a domain-specific language was proposed, the syntax structure of which is shown in see Table I .

Conditions defined in this language can be interpreted and executed by the language runtime in the task execution engine. The behavior of a task describes the actions to be performed when the condition is met. Task need to identify the target chain where the action acts and the type of action. It will be submitted to the Specy Network operator module in the form of a transaction for processing and recording. In addition, in order to pay the inspection fee for the task prover, task owner need to deposit a certain number of tokens for the task.

TABLE I
SYNTAX OF SPECY NETWORK DOMAIN-SPECIFIC LANGUAGE

Elements	Syntax
$\langle SpecyTask \rangle$	$\longrightarrow \langle taskDecl \rangle \langle entities \rangle [inputdata][outputdata] \langle rules \rangle \langle execution \rangle$
$\langle taskDecl \rangle$	$\longrightarrow task\ \langle name \rangle$
$\langle entities \rangle$	$\longrightarrow$ entities $\{\ \{\langle entity \rangle\}^+\ \}$
$\langle entity \rangle$	$\longrightarrow\ entity \langle name \rangle \{\{\langle attibute \rangle\}^+\}$
$\langle attribute \rangle$	$\longrightarrow \langle name \rangle is \langle type \rangle$
$\langle inputdata \rangle$	$\longrightarrow inputdata\{\{\langle attribute \rangle\}^+\}$
$\langle outputdata \rangle$	$\longrightarrow outputdata\{\{\langle attribute \rangle\}^+\}$
$\langle rules \rangle$	$\longrightarrow rules\{\{\langle rule \rangle\}^+\}$
$\langle rule \rangle$	$\longrightarrow rule \langle name \rangle \{\{\langle ruleStmt \rangle\}^+\}$
$\langle ruleStmt \rangle$	$\longrightarrow\ \langle setStmt \rangle$
	$\mid \langle relationStmt \rangle$
	$\mid \langle logicalStmt \rangle$
	$\mid \langle queryStmt \rangle$
	$\mid \langle definitionStmt \rangle$
$\langle execution \rangle$	$\longrightarrow execute\{\{\langle executeStmt \rangle\}^+\}$
$\langle executeStmt \rangle$	$\longrightarrow checkrule(\langle name \rangle \mid \langle executeRuleDef \rangle)$
$\langle ruleDef \rangle$	$\longrightarrow\ \langle name \rangle [trueStmt][falseStmt]$
$\langle trueStmt \rangle$	$\longrightarrow true\ :(\langle name \rangle \mid \langle ruleDef \rangle)$
$\langle falseStmt \rangle$	$\longrightarrow false\ :(\langle name \rangle \mid \langle ruleDef \rangle)$

- Third Step "Task checking and execution": The task check engine will synchronize the automation tasks that have been created on the chain, obtain the necessary data required through the data proxy, and check the conditions of the task. When the requirements are met, the engine will create a transaction to trigger the task and send it to the target chain. Then, the engine will submit the execution result of the transaction to Specy Network. Specy uses the validity of the key verification result and records the execution information and payment fee. Throughout the process, each process needs to be signed based on TEE(see Fig.3).

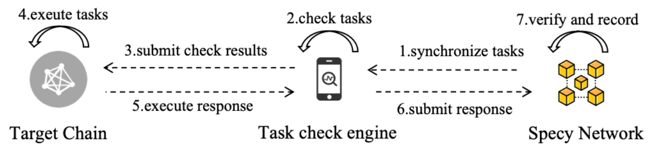


Fig. 3. Task checking and execution process

III. USE CASE

In this section, we take the decentralized social application scenario as an example. In this case, there is a certain scarcity of followers for social accounts, so users need to discover and follow popular users in a timely manner.Interestingly, there are some key opinion leaders (KOLs) who are usually able to timely discover hot users. Therefore, we only need to define an automatic follow-up task in Specy Network, the specific content is as follows:(1)Track the follow-up dynamics of KOL users.(2)Calculate the difference between the current user's follow-up list and the KOL user's follow-up list.(3)Follow the difference user list.Based on this analysis, the task is defined using domain-specific programming language (see Table II).

We deployed the example in the Specy Network implementation for testing. The results show that the system can automatically execute the defined behavior when the set conditions are met(see Fig.4).

IV. CONCLUSION

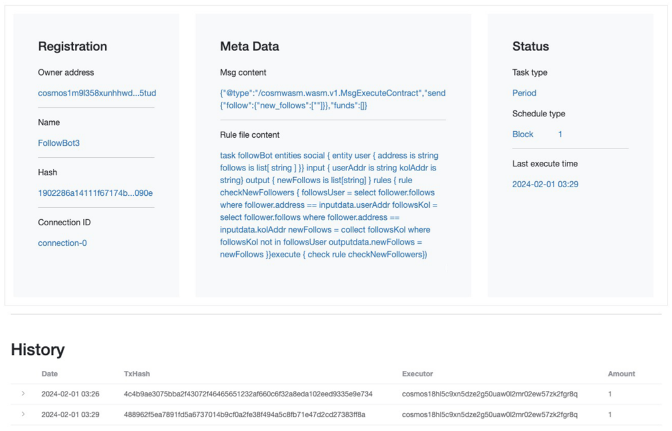
In this paper, we developed a blockchain automation protocol, Specy Network. It effectively reduces the complexity of automated task definition and can provide services for multiple chains. We have completed its development and implementation. The test case results show that Specy Network can well meet the automation needs of blockchain applications.

TABLE II
DEFINITION OF CONSTRAINTS FOR AUTOMATION TASK IN SOCIAL SCENARIOS

Name: followBot

Describe:

```
task followBot
entities social{
    entity user{
        address is string
        follows is list[string]
    }
}
input{
    userAddr is string
    kolAddr is string
}
output{
    newFollows is list[string]
}
rules{
    rule checkNewFollowers{
        followsUser=select follower.follows where follower.address==inputdata.userAddr
        followsKol=select follower.follows where follower.address==inputdata.kolAddr
        newFollows=collect followsKol where followsKol not in followsUser
        outputdata.newFollows= newFollows
    }
}
 execute{
    check rule checkNewFollowers
}
```



Registration

Owner address
cosmos1m9l358xunhhwd...5tud

Name
FollowBot3

Hash
1902286a14111f67174b...090e

Connection ID
connection-0

Meta Data

Msg content
{"@type":"/cosmwasm.wasm.v1.MsgExecuteContract","send
{"follow":{"new_follows":[""]}},"funds":[]}

Rule file content
task followBot entities social { entity user { address is string follows is list[string] }} input { userAddr is string kolAddr is string} output { newFollows is list[string] } rules { rule checkNewFollowers { followsUser = select follower.follows where follower.address == inputdata.userAddr followsKol = select follower.follows where follower.address == inputdata.kolAddr newFollows = collect followsKol where followsKol not in followsUser outputdata.newFollows = newFollows }}execute { check rule checkNewFollowers})

Status

Task type
Period

Schedule type
Block 1

Last execute time
2024-02-01 03:29

History

Date	TxHash	Executor	Amount
2024-02-01 03:26	4c4b9ae3075bba2f43072f46465651232af660c6f32a8eda102eed9335e9e734	cosmos18hl5c9xn5dze2g50uaw0l2mr02ew57zk2fgr8q	1
2024-02-01 03:29	488962f5ea7891fd5a6737014b9cf0a2fe38f494a5c8fb71e47d2cd27383ff8a	cosmos18hl5c9xn5dze2g50uaw0l2mr02ew57zk2fgr8q	1

Fig. 4. Task definition and execution records

REFERENCES

[1] Caldarelli, and Giulio. "Understanding the blockchain oracle problem: A call for action." Information ,vol. 11.11 (2020): 509,2020.

[2] Kwon, Jae, and Ethan Buchman."Cosmos whitepaper." A Netw. Distrib. Ledgers 27 ,2019.

[3] Kan, Luo, Yu Wei, Amjad Hafiz Muhammad, Wang Siyuan, Ling Chao Gao, and Hu Kai. "A multiple blockchains architecture on inter-blockchain communication." In 2018 IEEE international conference on software quality, reliability and security companion (QRS-C), pp. 139-145. IEEE, 2018.