2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom) | 978-1-6654-1658-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/TRUSTCOM53373.2021.00075

A privacy-preserving framework for smart contracts based on stochastic model checking

Tingting Bao
Nanjing University of Finance and Economics
School of Information Engineering
Nanjing, China
1120190824@stu.nufe.edu.cn

Yang Liu*
Nanjing University of Finance and Economics
School of Information Engineering
Nanjing, China
yliu@nufe.edu.cn

***Abstract*—In the process of using smart contracts, users need to provide privacy data such as personal account information to the contract for transactions. However, as the number of users continues to increase, privacy data leakage has become more and more serious. Also, the current model checking methods do not involve verifying the privacy data in smart contracts. For these reasons, we introduce a privacy-preserving framework based on stochastic model checking called *VeriPrivData*. It formally expresses privacy data by data sensitivity. The smart contract is defined as the *Commitments* tuple with data sensitivity. Then the *Commitments* tuple is used to model as DTMC (Discrete Time Markov Chains). We extend PCTL (Probabilistic Computation Tree Logic) to ds-PCTL (Probabilistic Computation Tree Logic with data sensitivity) for describing the privacy requirements. Finally, we verify whether DTMC satisfies the ds-PCTL formula. In this paper, the *VeriPrivData* framework uses the stochastic model checking tool PRISM, and experiments are carried out. Experimental results show that this method can effectively avoid illegal disclosure of privacy data and enhance the protection of privacy data in the contract.**

Keywords—smart contracts, privacy data, data sensitivity, DTMC, stochastic model checking

I. INTRODUCTION

With the development of blockchain technology, smart contracts have preliminary applications in specific fields [1]. In the process of using the smart contract, privacy data such as name, address, phone number, and date of birth are inadvertently collected by the smart contract [2]. If there is no protection of that information, it is easy to be leaked and brings risks to personal safety [3]. For example, in ERC223 smart contract, attackers use the custom callback function to turn the contract owner into an address under their control, resulting in a large amount of user information leakage [4]. Therefore, the privacy-preserving of smart contracts is particularly important.

Model checking is a method for formal verification that is suitable for systems that can be expressed by a finite-state model [5]. The verification is performed on a computer, using a model checking software. The user provides the model checker with a finite-state model of the system, and a formally specified property it should possess. The model checker then checks if each state of the model satisfies the specification. Therefore, the verification of smart contracts based on the formal method can effectively validate whether a smart contract satisfies the privacy data property.

A. Related Work

Currently, there are many studies that protect the sensitive information of privacy data in smart contarcts in different ways. For example, [6] makes a privacy-protected data sharing agreement based on blockchain. This agreement allows users to exchange data and sharing with a standard index. PRVB is a novel system model based on blockchain oracle that guarantees the reliability of sensed data before it is published to the blockchain [7]. [8] proposes a PoBT algorithm that enables the security of block at both trade validation and block creation phases. These methods add a privacy-preserving mechanism on the blockchain, which does not allow unauthorized smart contracts to use data to achieve data protection research [9].

Because of the mathematical characteristics of formal verification, formal methods have received more and more attention to the security verification of smart contracts. Some literature proposes model checking methods for formal verification of a smart contract [10]. For example, Abdellatif proposes a PROMELA model for establishing smart contracts [11]. Using the SPIN model verification tool to verify the consistency of shopping smart contracts, it has been verified that the contract code and the contract text are consistent. In [12], Mavridou proposes a VeriSolid framework, which uses transition models and operational semantics to model and verify the action on the contract. The VeriSolid framework can generate Solidity code from the verified model to realize the correctness of smart contracts. Hirai performs a model checking of a smart contract used by the Ethereum Name Service [13].

The current model checking in smart contracts faces some shortcomings: (1) The lack of clear semantic description before and after using the smart contract, which led to the disclosure of privacy data in the contract. Therefore, how to describe smart contracts as a suitable model has become a vital issue [14]. (2) Many privacy-related laws and industry regulations contain requirements for the use of privacy data. For example, the US Medical Electronic Exchange Act stipulates that any child-oriented website shall not collect any other private information before collecting the age of the visitor [15]. The act is a typical restriction on the use of privacy data. Therefore, how to realize the restriction on the action of smart contracts using privacy data by the contract properties. (3) Most model checking methods focus on formal verification of some properties such as consistency, correctness, integer overflow [16]. The property of privacy data cannot be involved during model checking.

B. Our Contributions

Above the related work, we are using a stochastic model checking framework called *VeriPrivData* for the privacy data protection of smart contracts for Ethereum Solidity, which is shown in Fig. 1. Contrary to other methods that solve the privacy-preserving for smart contract, the *VeriPrivData* supports the description of privacy data and verification property of the privacy data. In particular, *VeriPrivData*

*Corresponding author

2324-9013/21/$31.00 ©2021 IEEE
DOI 10.1109/TrustCom53373.2021.00075

allows developers to graphically design a smart contract as a DTMC model, purpose the privacy requirements, perform stochastic model checking and verify whether DTMC satisfies the privacy requirements for using privacy data.

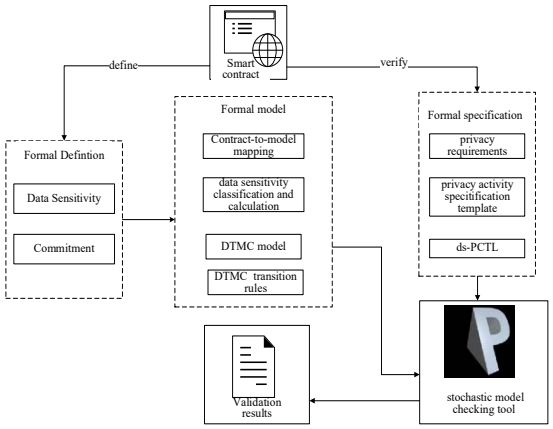

Fig. 1. *VeriPrivData* framework for smart contract.

In detail, the contributions to this paper are as follows:

- We extend the tuple of *commitments* with data sensitivity to describe the semantics of smart contracts.
- We use the privacy requirements to describe the restriction on the action of smart contracts using privacy data.
- We expand the PCTL to the ds-PCTL formula. And we can verify the sensitivity property of privacy data in smart contracts by the ds-PCTL formula.

For the rest of the paper, a formal definition of *commitment* with data sensitivity is introduced in section II. Section III describes a formalized modeling of smart contracts. Section IV presents a formal specification to the privacy requirements of users. Section V does a reference experiment on the e-commerce smart contract given in the blockchain platform. In section VI, the paper is summarized and the future work has prospected.

II. Formal Definition of Smart Contract

Smart contracts gain rapid exposure since the inception of blockchain technology. Yet, there is no unified definition for smart contracts. We use *commitments* with data sensitivity definition to describe the semantics of smart contracts in this section.

A. Data Sensitivity

The property of data can be expressed in terms of sensitivity. Weible proposes that data sensitivity is the level of privacy concern an individual feels for a type of data in a specified situation [17]. Affecting the data sensitivity can be divided into two factors: data types and data usage situation [22]. Data types refer to different kinds of data that will have a different preference on the data. The data usage situation refers to the data usage period, data usage type, and correlation between data [23]. These affecting data sensitivity factors are called sensitivity attributes.

Definition 1. (Sensitive attributes of data) Sensitive attributes of data is a tuple $ds= (v, u, \mu, £, \sigma)$, where v is the type of data in action, u is the preference sensitivity of data, μ is the time sensitivity of data, £ is the sensitivity of data usage, σ is the sensitivity of the correlation between data, among them:

Definition 1.1 (Data preference sensitivity) A data preference sensitivity is $u=fPreference(v)$, where $fPreference(v)$ is a function to quantify the value of sensitivity in the range of [0,1], v is the type of data. The preference sensitivity of data is measured according to the data type[24]. Different types of data will have a different impact on the data. The higher preference sensitivity of the same data, the more concern by users, the greater damage is after its disclosure.

Definition 1.2 (Data time sensitivity) A data time sensitivity is $\mu=fRetention(t)$, where $fRetention(t)$ is a function to quantify the value of sensitivity in the range of [0,1], t is a finite set of data retention periods in the contract. Due to the timeliness of data, it is necessary to consider the impact of the retention time of the data in the execution of smart contract. The more data-related function calls in the contract, the longer the data retention time, the higher the time sensitivity of data[25].

Definition 1.3 (Data usage sensitivity) A data usage sensitivity is $£=fPurpose(a)$, where $fPurpose(a)$ is a function to quantify the value of sensitivity in the range of [0,1], a is a finite set of data usage types in the contract. We define as a= {create, assign, transfer, collect, function, ,}, etc., in smart contract. The settings of sensitivity sat different usage types respectively.

Definition 1.4 (Data correlation sensitivity) A data correlation sensitivity is $\sigma=fRelation(v)$, where $fRelation(v)$ is a function to quantify the value of sensitivity in the range of [0,1], v is the type of data. The data sensitivity needs to consider the combined data. The leakage of a single data item will not cause, but the leakage of several data items at the same time may cause user losses. Therefore, it is necessary to consider whether the data item is a single data type [26]. There are four cases of sensitivity to the correlation between data. $fRelation(v)$=0 if the data type is composed of a single element and non-privacy data. $fRelation\ (v)$=0 if the data type is composed of multiple non-privacy data elements. If the data type consists of a single element that is privacy data, $fRelation(v)=u$. If the data type consists of n privacy data, $fRelation(v)=\frac{1}{n}\sum_{i=1}^{n} u_i$.

Through the above definition, all data in the contract is formally defined according to the sensitivity.

B. Commitment

A smart contract is viewed as a business exchange consisting of a set of reciprocal commitment [18]. We define *commitment* tuples as the basic unit of smart contract to express functions.

Definition 2. (*Commitment*) A *commitment* is a tuple $C= (x, y, p, A, r, \theta)$, where it means that the promisor(x) makes the action(A) under the premise(p) to the promises(y), and result(r) is produced, the comprehensive data sensitivity involved in the *commitment* is θ. In e-commerce smart contracts, the promisor and the promises are buyers, sellers, and smart contracts. p represents the preconditions for triggering the *commitment*.

Since the action of *commitment* includes the call relationship between data, we use a tuple to define it.

Definition 3. (Action) Action is represented as *A = (act(v), v, sender, receiver)*. In this tuple, *act(v)* is the name of the action, *v* is a set of data in *act(v)*, sender is the executor of the action, and receiver is the receiver of the action.

III. Formal Modeling of Smart Contract

According to the formal definition of a smart contract, the contract function expresses as a *commitment* [19]. This section first maps the smart contract code to *commitments* and further establishes a DTMC model with *commitments* for smart contracts.

A. Contract-to-Commitment Mapping

We should realize the transfer of functions, states, variables, etc., in the contract to the *commitment C (x, y, p, A, r, θ)*. Apply the Contract-to-Commitment mapping algorithm to input all functions and analyze function structure to map the elements of *commitment*. The Contract-to-Commitment mapping pseudo code is as follows:

```
Algorithm 1   Contract-to-Commitment
input：smart contract solidity code
output：x, y, act(v), r, p, θ
1 function MAPPINGSORT （solidity code）
2     let set f(j) is functions in solidity code
3     for f(j) do
4        variable v1,v2,v3,v4
5        enum status[ ]
6        modifier M1,M2
7        constructor C1, C2
8        r <- status[i ]
9        if verifyOpr(M1.address)==true then
10          x <- M1
11          if verifyOpr(M2.address)==true then
12             y<- smart contract
13          else
14             y <- M2
15          end if
16          act(v)<- C1,act(v)<- C2
17          if transaction is true && status[i]is met
18             r <- status[i ]+1
19             p <- sat
20          act(v)<- transaction , θ=ADD(act(v))
21       else
22          act(v)<- transaction,
23          r <- status[i].inactive.
24          p <- vio,θ=ADD(act(v))
25       endif
26    endif
27   j++
28   endfor
28 endfunction
29 function ADD(act()) return unit x
30       for act(k) from act(v)
31          act(k) <- getMsgFixSens(v)
32          k++
33          return x<- act(k).value
34       endfor
35 endfuntion
```

The Contract-to-Commitment mapping algorithm implements two functions: (1) It can map from contract function to *commitment*, such as map the function in the contract to the *commitment C*, map the contract initiator to the promisor *x*, map the contract receiver to promises *y*, the function trigger condition in the current contract is mapped to the premise *p* of *commitment*, the result after the function is executed is mapped to the result *r* of *commitment*, the modifier, function, and constructor called in the function are mapped to the action *A* in *commitment*; the variable in the function is mapped to the data *v*. (2) The GetMsgFixSens () of ADD() calls the method for data sensitivity classification and calculation to get the value *θ*.

B. Data Sensitivity Classification and Calculation

As data mapped by Contract-to-Commitment mapping grows, the type of data becoming more and more comprehensive. We should classify and identify the privacy data in the data set. A method for data sensitivity classification and calculation is proposed. The procedure is shown in Fig. 2.

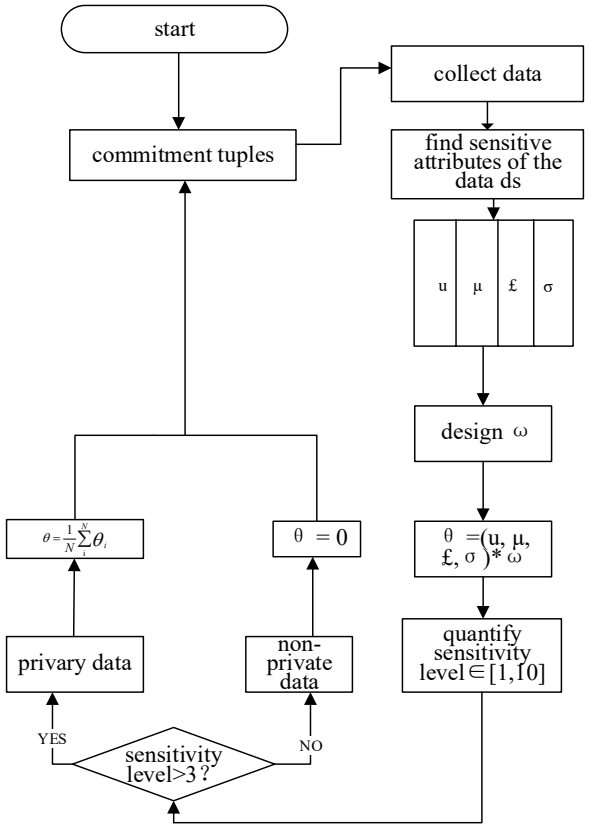

Fig. 2. Data sensitivity classification and calculation.

Firstly, actions and data are collected in the *commitment* tuples. According to the sensitivity attribute of data in Definition 1 *ds*= (*v, u, μ, £,σ*), we can obtain the sensitivity values of different attributes of data. Secondly, designing the weight of sensitive attributes, the comprehensive sensitivity value *θ* of this data is calculated. Thirdly, the data comprehensive sensitivity *θ* is quantified into sensitivity levels [1,10]. Fourly, the sensitivity level is compared with the specified threshold. If the sensitivity level is greater than the threshold, the data type is determined to be sensitive and called privacy data. If the sensitivity level is less than or equal to the threshold, the data type is called non-private data.

Meanwhile, the comprehensive sensitivity of non-private is zero. Finally, the sensitivity of all the privacy data of action is averaged, and the comprehensive sensitivity of the data of this action is θ.

Therefore, the smart contract is expressed as a series of *commitments* according to the formal definition and the mapping algorithm. The next step will introduce defining a *commitment* as a state in a smart contract model.

C. Smart Contract Model

Pee emphasizes that a smart contract is a state machine, and each execution of a smart contract can be seen as a transition from one state to another [28]. The execution of a smart contract is the process of transferring between different *commitment* states due to the uncertainty of the function call [29]. Therefore, the smart contract can be established as a stochastic process system model with Discrete Time Markov Chains [30].

Definition 4. (Smart contract model) Smart contract is showed as a DTMC (Discrete Time Markov Chains) model *(S, s_0, P, L)*, where *S= {s_0, s_1, s_2, …, s_m}* is a set of all *commitment*s states in a limited contract, s_0 is the initial state of the model, *P*:*S* x *S*->[0,1] is expressed as a transition probability matrix according to the transition rule between *commitments*, and the sum of all $s \in S$ is 1. *L*:*S*->2^{AP} is a label function, *AP* defines a set of *commitments* $C= \{C_1, C_2, \ldots, C_n\}$, C_i represents the *i*th *commitment* tuple *(x, y, p, A, r, θ)*.

D. Commitment Transition Rules

In our framework, we model smart contract as DTMC, which describes *commitmen*t tuple that can be in one of their possible states and can evolve to new states. Any smart contract operation may be considered as a transition where some *commitments* take place. Each transition may have an applicability condition that defines whether the operation may be executed.

In the smart contract model, *P*:*S* x *S*->[0,1] is expressed as a transition probability matrix according to the *commitments* transition rule [31]. Each transition rule is determined by the *commitment* condition. In the rules, C^p is the promise of *commitment C*, C^r is the result of the *commitment C*. Let *M* be an array with the comprehensive sensitivity and data *v*. There are three transition rules when *commitments* are made to different states in the contract.

If under the condition of C^p=true, the action *act(v)* in the state is satisfied, then *M* and the *commitment* C are entered into the subsequent *commitments* C', and *M'*, and *s* is marked as L(*s*)=sat and changes to *s'*, C^p changes to C^r.

$$s \in SC,\ \ p = C^p = true, r = C^r, \theta, act(v), M$$
$$M = Params(act(v), \theta_1, \theta_2, ...), s = (C, M)$$
$$Eval(s, C^p) \xrightarrow{p1} < (s', C^r), L(s) = sat >,$$
$$C^p = C^r = true, s' = (C', M')$$

If under the condition of C^p=true, the C^r in the state is not true, then *M* and the *commitment C* are entered into the subsequent default *commitments* $C^r$and *M'*, and the state *s* is marked as L(*s*)=vio and changes to *s'*, C^p Change to C^r.

$$s \in SC,\ \ p = C^p = \text{true}, r = C^r, \theta, act(v), M$$
$$M = Params(act(v), \theta_1, \theta_2, ...), s = (C, M)$$
$$Eval(s, C^p) \xrightarrow{p2} < (s', C^r), L(s) = vio >,$$
$$C^r = \text{false}, s' = (C', M')$$

If under the condition of C^p= true, the action *act(v)* in the state is satisfied. However, the action *act(v)* in the state is not satisfied. Then *M* and *commitment C* are entered into the subsequent *commitments C'* and *M'*, and the state *s* is marked as L(*s*)=bas and change to *s'*, C^p changes to C^r.

$$s \in SC,\ \ p = C^p = \text{true}, r = C^r, \theta, act(v), M$$
$$M = Params(act(v), \theta_1, \theta_2, ...), s = (C, M)$$
$$Eval(s, C^p) \xrightarrow{p3} < (s', C^r), L(s) = bas >,$$
$$act(v) = \text{false}, s' = (C', M')$$

IV. Formal Specification for Privacy Requirement

In smart contracts, different users have different privacy requirements for privacy data [32]. The restriction of the use of privacy data is called privacy requirement [33]. The smart contract model supports the verification of the property for user's privacy requirements. We need an approach to precisely specify the privacy requirement in Fig. 3 below:

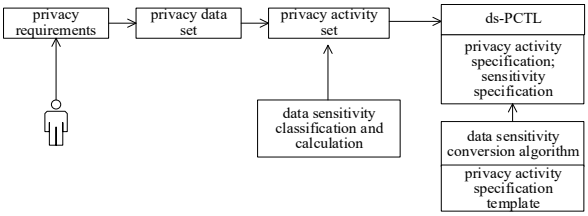

Fig. 3. Formal specification of privacy requirements.

We describe using constraints of privacy data by the privacy requirement. We systematically analyze the privacy data sets such as name, phone, etc., from the privacy requirements. Then we get the comprehensive sensitivity and status of the corresponding privacy data according to the sensitivity classification and calculation. Afterward, the privacy data sets are formally expressed as multiple privacy activities set according to the definition of privacy activities. Finally, the ds-PCTL formula is specified by using the privacy activity specification template and privacy sensitivity conversion algorithm.

A. Privacy Activity

To make a formal description of privacy requirements, it is necessary to perform semantic analysis on the privacy activity in privacy requirements.

Privacy activities are defined as the tuple a=*(S, D, OP, θ, sender, receiver)*. Which *S = {S_1, S_2, S_3...}* represents the status of privacy activities. *D* represents the privacy data involved in the activities. *OP* represents the actions described in the privacy requirements such as collect, use, expose and maintenance, etc.. *sender, receiver* indicate the data sender and data receiver. For example, in the e-commerce smart contract, the participants are mainly sellers, buyers, and contract parties. θ indicates the comprehensive sensitivity of privacy data.

By defining privacy requirements as privacy activities, the verification of privacy requirements is more simplified and versatile.

B. Formal Specification: ds-PCTL

The logic PCTL, which extends computation tree logic (CTL) with probabilities, does not test data sensitivity. Therefore, to address this limitation of PCTL, we consider an extension of PCTL that can express sensitivity property. We call the logic ds-PCTL. The syntax and semantics of this logic are described as follows:

1) Syntax

$\varphi ::= true \mid a \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid P_{\sim q}[\psi] \mid \theta_{\sim p}[\psi]$

$\psi ::= X\ \varphi \mid \varphi_1 \cup \varphi_2 \mid \varphi_1 \cup^{\leq n} \varphi_2$

Here, α is an atomic proposition, $\sim \in \{<,>,\leqslant,\geqslant\}$, $p \in [0,1]$, $q \in [0,1]$, φ denotes a state formula, ψ denotes a path formula. The temporal operators X and U are required to be immediately preceded by P, θ is the sensitivity operator [35].

2) Semantics

SC:=(S,s_0,P,L) for any state $s \in S$, φ and ψ be ds-PCTL state formulae[36]. The satisfaction relation $\vDash$ is defined for state formulae by:

- $s \vDash$ true, iff $\forall s \in S$;
- $s \vDash \alpha$, iff $\alpha \in L(s)$; $\alpha \in L(s)$
- $s \vDash \neg\varphi$, iff $s \vDash \varphi$ is false;
- $s \vDash \varphi_1 \wedge \varphi_2$, iff $s \vDash \varphi_1$ and $s \vDash \varphi_2$;
- $s \vDash P_{\sim p}[\psi]$, iff $Pr(s \vDash \varphi) \in p$ and $Pr(s \vDash \varphi) = Prs\{\omega \in Paths(s) \mid \omega \vDash \psi\}$;
- $s \vDash \theta_{\sim q}[\psi]$,iff $(s.\theta \vDash \varphi) \in p$ and $(s.\theta \vDash \varphi) = \sum\{\omega \in Paths(s) \mid \omega \vDash \psi\}$;

Given a path ω in DTMC ,the satisfaction relation is defined as follows:

- $\omega \vDash X\varphi$, iff $\omega[1] \vDash \varphi$;
- $\omega \vDash \varphi_1 \cup \varphi_2$, iff $\exists j \geqslant 0.(\pi[j] \vDash \varphi_2 \wedge (\exists 0 \leqslant k \leqslant j, \omega[k] \vDash \varphi_1$ and $\omega \vDash \sim p)$;
- $\omega \vDash \varphi_1 \cup^{\leq n} \varphi_2$, iff $\exists 0 \leqslant j \leqslant n.(\pi[j] \vDash \varphi_2 \wedge (\exists 0 \leqslant k \leqslant j), \omega[k] \vDash \varphi_1)$;

The formula $\theta_{\sim q}[\psi]=?$ represents the data sensitivity formula, where θ is the comprehensive sensitivity operator, q represents the trust value of the user, and ψ represents any path formula. The $\theta_{\sim q}[\psi]$ is true over a path ψ if comprehensive sensitivity θ in any state S in DTMC is specified, and it is required that θ in the path ψ satisfies the trust value q, and the model is said to satisfy the sensitivity specification of privacy data, so there is no illegal leakage of private data in path ψ.

3) Operator conversion algorithm

The data sensitivity can be expressed as $\theta_{\sim q}[\psi]$ formula to verify whether there is illegal leakage of private data. But it is impossible to use for automated verification because the θ operator cannot be employed in stochastic model checking [37]. For this reason, the conversion algorithm can convert $\theta_{\sim q}[\psi]$ into PCTL formulas.

In PRISM property specification language, the R operator expresses the expected value of a random variable associated with the reward value [38]. In $R_{\sim q}[\psi_{.\theta}]$ formula, we label the operator to calculate data sensitivity. The $R_{\sim q}[\psi_{.\theta}]$ means the average reward value of data sensitivity in path ψ satisfies q. Through this Algorithm 2, we can convert $\theta_{\sim q}[\psi]$ formula into $R_{\sim q}[\psi_{.\theta}]$ formula.

Algorithm 2 Operator conversion algorithm
Input: $\theta_{\sim q}[\psi]$
Output: $R_{\sim q}[\psi_{.\theta}]$
1 function conversion(ψ)
2 for any path ψ do
3 $\psi = s_0 s_1 \ldots . s_n$
4 for i=0 to n
5 θ = label $\frac{1}{n}\sum R$
6 $\theta[\psi]=\frac{1}{n}([s_{0.}\theta] + [s_{1.}\theta] + \cdots + [s_{n.}\theta])$
7 $= \sum_{i=0}^{n} R[\psi.\theta_I]\frac{1}{n}$
8 $=R[\psi.\theta]$
9 endfor
10 endfor
11 while(~q is ≤q) do
12 if $R^{max}[\psi.\theta] \leq q$ then
13 return $R_{\leq q}[\psi]$=true
14 else
15 return $R_{\leq q}[\psi]$=false
16 endif
17 endwhile
18 while(~q is ≥q) do
19 if $R^{min}[\psi.\theta] \geq q$ then
20 return $R_{\geqslant q}[\psi]$=true
21 else
22 return $R_{\geqslant q}[\psi]$=false
23 endif
24 endwhile
25 endfunction

C. Privacy activity specification template

We use the ds-PCTL formula to specify the privacy activities. The privacy activity description can map to the corresponding ds-PCTL formula by the privacy activity specification template. There are two types of templates to descript privacy activities. Table 1 is a privacy activity existence specification template that assume a single privacy activity. The second one is a specification template for multiple privacy activities. For simplicity, 'a' and 'b' are used to represent privacy activities.

TABLE I. PRIVACY ACTIVITY EXISTENCE SPECIFICATION TEMPLATE

privacy activity description	*ds-PCTL formula*
'a' in the contract will not occur	¬ (a)
'a' always occurs in the contract	G(a)
'a' in the contract will eventually occur	F(a)
Is the comprehensive sensitivity θ of the contract in the path φ lower than (higher than) q	$\theta_{\sim q}[\varphi]$
Whether the probability of the contract occurring in the path φ is lower (higher) than p	$P_{\sim p}[\varphi]$

TABLE II. MULTIPLE PRIVACY ACTIVITY RELATIONSHIP SPECIFICATION TEMPLATE

privacy activity description	*ds-PCTL formula*

'a' happens in the contract and 'b' occurs	(a)v(b)
If 'a' happens in the contract, 'b' must occur before or after 'a'	(a)U(b)
If 'a' happens in the contract, then 'a' is bound to occur in k periods	$a \cup^{\leq k} b$
Action 'a' happens after action 'b'	a->b

V. CASE STUDY

Here, the stochastic model checking tool PRISM is used to experiment with the proposed framework. PRISM is the most widely used graphical stochastic model checking tool implemented by Professor Kwiatkowska in the Java/C (++) language [35]. Various types of stochastic system models such as DTMC, CTMC, MDP, PA, PTA can be verified by PRISM [34].

We model an e-commerce smart contract derived from literature [39] as a DTMC. The model has 46 groups of states, a total of 57 state transitions. The e-commerce smart contract as a DTMC model is shown in Fig. 4.

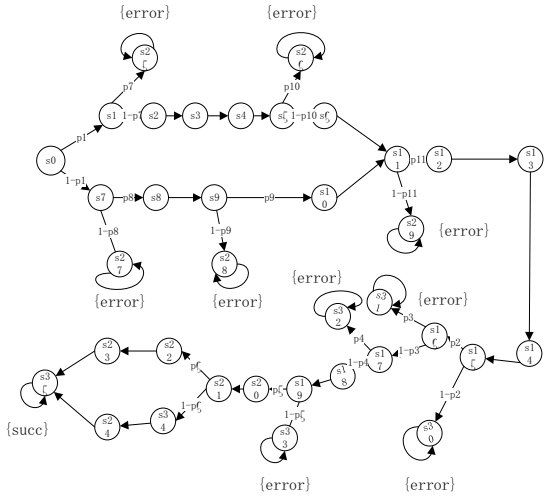

Fig. 4. DTMC model for e-commerce smart contract.

In the smart contract, we assume users accept the maximum trust value is 0.1. If the sensitivity of the data is lower than the trust value, the $\theta_{\sim q}[\psi]$ is true. The smart contract meets the privacy requirement of the user, so there is no leakage of illegal privacy data. Otherwise, the $\theta_{\sim q}[\psi]$ is false and the system has the problem of privacy data leakage. There are four privacy requirements of users in this case to verify whether the smart contract exists the illegal leakage.

Privacy requirement 1: The user address must be collected before being exposed to the contract party. If the address is exposed to the contract party, the order of users must be accepted before exposure. The comprehensive sensitivity of this path meets the requirements of user trust value.

For privacy requirement 1, the privacy activities are expressed as: a_1 =(S_7,*buyerAddr,collect,*θ_1*,B,C*),a_2=(S_8,*buyerAddr,transfer,*θ_2*, C,S*),a_3=(S_{10},*order,collect,*θ_3*,C,S*), where *C* represents the contract party, *B* represents the buyer, and *S* represents the seller. The privacy activity can be specified as ds-PCTL formula: $(a_1 \rightarrow a_2) \cap (a_2 \text{ U } a_3)$and the data sensitivity formula in the privacy activity is expressed as $\theta_{<=0.1}$ $[(a_1 \rightarrow a_2) \cap (a_2 \text{ U } a_3)]$. The contents of the data involved are shown in table below:

TABLE III. PRIVACY DATA OF PRIVACY REQUIREMENT 1

State	*Action*	*privacy data item*	*Comprehensive sensitivity*
S_7	create	buyerAddr	0.013
S_8	transfer	buyerAddr	0.034
S_{10}	collect	order	0.018

Verification of privacy activities: According to the path $(a_1 \rightarrow a_2) \cap (a_2 \text{ U } a_3)$, the probability of this path in the model is $P[(a_1 \rightarrow a_2) \cap (a_2 \text{ U } a_3)]= 0.49$. Verify the sensitivity of privacy data: According to the DTMC model, it is known that the states included in this path are S_7, S_8, S_9, S_{10}, and the comprehensive sensitivity of the path $\theta[(a_1 \rightarrow a_2) \cap (a_2 \text{ U } a_3)]=0.0214$, $\theta_{<0.1}$ $[(a_1 \rightarrow a_2) \cap (a_2 \text{ U } a_3)]$=true. Although the comprehensive sensitivity of the path satisfies the trust path, we find that the probabilities of path is low. This means the user address will be exposed to the contract party without user accepted. We should add the access mechanism of user address before exposure.

Privacy requirement 2: To prevent malicious buyers do not confirm, or buyers forget to confirm the receipt of goods, the order of buyers can be automatically confirmed within the specified time. The data comprehensive sensitivity of this path must not be higher than the trust value.

According to privacy requirement 2, the privacy activities are expressed as a_4=($S_{20,}$ signTime, create, θ_1, *B, C*), a_5=(S_{34},{trackingNumber,price,sellerAddr},transfer,θ_2,*C,S*),a_6=(S_{22},{now,signTime},collect,θ_3,*C,S)*, where *C* represents the contract party, *B* represents the buyer, and S represents the seller. The privacy activity specification can be expressed as a_4-> (a_5U a_6) by using the ds-PCTL specification template, which means that the contract order in the contract will never occur before the buyer confirms the order or the buyer confirms the order. At the same time, the privacy data sensitivity is expressed as $\theta_{<0.1}[\ a_4\text{->}(\ a_5 \text{ U } a_6)]$. The data involved is shown in the table below.

TABLE IV. PRIVACY DATA OF PRIVACY REQUIREMENT 2

State	*Action*	*privacy data item*	*Comprehensive sensitivity*
S_{20}	create	signTime	0.018
S_{34}	transfer	trackingNumber, price	0.082
S_{22}	collect	now, signTime	0.097

Verification of privacy requirement 2, the probability of this path in the contract is $P[a_4\text{->}(\ a_5 \text{ U } a_6)] = 0.9320$, and the sensitivity of the privacy data is verified. According to the smart contract model, the states included in this path include $S_{20,}$ $S_{21,}$ $S_{22,}$and S_{34}, the comprehensive sensitivity of the path $\theta[\ a_4\text{->}(\ a_5 \text{ U } a_6)]=0.0697$, $\theta_{<0.1}[\ a_4\text{->}(\ a_5 \text{ U } a_6)]$=true.

Then a comparative experiment was performed on two different types of e-commerce systems, among which the web e-commerce model comes from paper [33]. The traditional e-commerce web model accumulates 20 sets of states and a total of 22 state transitions.

Privacy requirement 3: After the buyer completes an order, how much the data sensitivity is? Is there any data leakage problem under the specified trust value?

Comparing with two e-commerce systems, the buyer completes an order specify as P[F "succ"], and the data sensitivity specifies as $\theta_{\sim q}$[F "succ"]. If the $\theta_{\sim q}$[F "succ"] formula is true, there is no data leakage problem in this system. Otherwise, it exists data leakage problem in the system. The verification results are shown in the table below:

TABLE V. VERIFICATION RESULT OF PRIVACY REQUIREMENT 3

system	*P[F "succ"]*	*Data sensitivity*	*$\theta_{\sim q}$[F "succ"]*	*conclusion*
web e–commerce	0.7741355	0.043	true	no data leakage problem
e-commerce smart contract	0.9320653	0.041	true	no data leakage problem

Privacy requirement 4: In the traditional e-commerce web platform and e-commerce smart contract platform, we assume there may also be a failure situation that buyers cannot finish an order. How does data sensitivity change in the failure situation? Is there any data leakage problem?

In Fig. 4, there are nine fail situations in the DTMC models. We will simulate these nine fail situations as nine fail paths. Comparing with two e-commerce systems, the data sensitivity specify as θ[F "error"]. If the θ[F "error"] formula less than 0.1, there is no data leakage problem in this system. Otherwise, it exists a data leakage problem in the system.

The data sensitivity in different fail paths is shown in Fig. 5. In this figure, X-axis represents the fail paths in the traditional e-commerce web platform and e-commerce smart contract. The Y-axis means the data sensitivity changes.

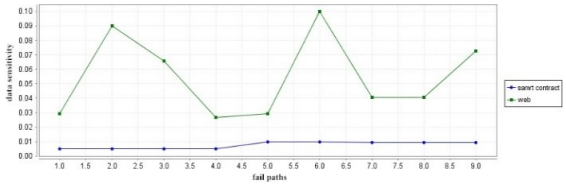

Fig. 5. Comparison of two e-commerce systems.

The above experimental results show that the privacy data sensitivity of two e-commerce systems is relatively low. In comparison, the sensitivity stability of smart contracts is better than that of the traditional web. The data sensitivity of the 6th path of traditional web system is higher than the trust value. Therefore, the e-commerce web system has private data leaking problems. According to the e-commerce web systems, we further find this path is very fragile because this path involves the payment information.

The sensitivity stability of smart contracts is better than the traditional web because the smart contract implements distributed payment system via combing blockchain. It can decrease the use of privacy data through peer-to-peer data transmission. However, the traditional e-commerce web system needs to transfer the payment data to the 3rd payment, such as Alipay. It causes privacy data disclosure risk to be higher. So, the 6th path involving the payment data is very fragile that if this path has run, it will cause illegal privacy data leakage. It is necessary to strengthen the protection of the privacy data of payment functions in this path.

VI. CONCLUSION

The *VeriPrivData* framework is proposed for the protection of privacy data in smart contracts. In this framework, the stochastic model checking technique has been exercised on the DTMC model access some properties in ds-PCTL. Comparing with the current model checking of smart contracts, this framework improves privacy-preserving of smart contracts by verifying the property of privacy data. Accrording to the four privacy requirments, we find the e-commerce smart contract leaks the proctection of user address data. Comparing with the traditional e-commerce web, the e-commerce smart contract data leakage problem is better than it. In the future, we will expand the property of privacy data and deal with more types of smart contracts.

ACKNOWLEDGMENT

This work was supported by the Six Talent Peaks Project of Jiangsu (No. RJFW-014), National Natural Science Foundation of China (No. 61303022), Natural Science Research Key Project of Jiangsu Higher Education Institutions of China (No.17KJA520002), Nanjing Scientific and Technological Innovation Project for Outstanding Overseas Returnees, Postgraduate Research and Practice Innovation Program of Jiangsu Province (No.KYCX20_1347).

REFERENCES

[1] S. Wang, L. Ouyang, Y. Yuan, X. Ni, X. Han and F. Y. Wang, "Blockchain-Enabled Smart Contracts: Architecture, Applications, and Future Trends," in IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 49, no. 11, pp. 2266-2277, Nov. 2019, doi: 10.1109/TSMC.2019.2895123.

[2] R. Gupta, S. Tanwar, F. Al-Turjman, P. Italiya, A. Nauman and S. W. Kim, "Smart Contract Privacy Protection Using AI in Cyber-Physical Systems: Tools, Techniques and Challenges," in IEEE Access, vol. 8, pp. 24746-24772, 2020, doi: 10.1109/ACCESS.2020.2970576.

[3] A. Kosba, A. Miller, E. Shi, Z. Wen and C. Papamanthou, "Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts," 2016 IEEE Symposium on Security and Privacy (SP), 2016, pp. 839-858, doi: 10.1109/SP.2016.55.

[4] http://solidity.readthedocs.io/en/v0.4.21/common-patterns.html#state-machine

[5] Z. Liu and J. Liu, "Formal Verification of Blockchain Smart Contract Based on Colored Petri Net Models," 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), 2019, pp. 555-560, doi: 10.1109/COMPSAC.2019.10265.

[6] Y. Murray and D. A. Anisi, "Survey of Formal Verification Methods for Smart Contracts on Blockchain," 2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS), 2019, pp. 1-6, doi: 10.1109/NTMS.2019.8763832.

[7] C. Zhang, L. Zhu, C. Xu and K. Sharif, "PRVB: Achieving Privacy-Preserving and Reliable Vehicular Crowdsensing via Blockchain Oracle," in IEEE Transactions on Vehicular Technology, vol. 70, no. 1, pp. 831-843, Jan. 2021, doi: 10.1109/TVT.2020.3046027.

[8] S. Biswas, K. Sharif, F. Li, S. Maharjan, S. P. Mohanty and Y. Wang, "PoBT: A Lightweight Consensus Algorithm for Scalable IoT Business Blockchain," in IEEE Internet of Things Journal, vol. 7, no. 3, pp. 2343-2355, March 2020, doi: 10.1109/JIOT.2019.2958077.

[9] V. Avdiienko, "Mining Apps for Abnormal Usage of Sensitive Data," 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, 2015, pp. 426-436, doi: 10.1109/ICSE.2015.61.

[10] Y. Murray and D. A. Anisi, "Survey of Formal Verification Methods for Smart Contracts on Blockchain," 2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS), 2019, pp. 1-6, doi: 10.1109/NTMS.2019.8763832.

[11] T. Abdellatif and K. Brousmiche, “Formal Verification of Smart Contracts Based on Users and Blockchain Behaviors Models,” 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS), 2018, pp. 1-5, doi: 10.1109/NTMS.2018.8328737.

[12] K. Nelaturu, A. Mavridoul, A. Veneris and A. Laszka, “Verified Development and Deployment of Multiple Interacting Smart Contracts with VeriSolid,” 2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), 2020, pp. 1-9, doi: 10.1109/ICBC48266.2020.9169428.

[13] Hirai Y, “Defining the Ethereum Virtual Machine for Interactive Theorem Provers,” Financial Cryptography and Data Security, 2017, vol. 10323, Springer, doi: 10.1007/978-3-319-70278-0_33.

[14] X. Shu, D. Yao and E. Bertino, “Privacy-Preserving Detection of Sensitive Data Exposure,” in IEEE Transactions on Information Forensics and Security, vol. 10, no. 5, pp. 1092-1103, May 2015, doi: 10.1109/TIFS.2015.2398363.

[15] Y. Murray and D. A. Anisi, “Survey of Formal Verification Methods for Smart Contracts on Blockchain,” 2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS), 2019, pp. 1-6, doi: 10.1109/NTMS.2019.8763832.

[16] S. Lee, S. Park and Y. B. Park, “Formal Specification Technique in Smart Contract Verification,” 2019 International Conference on Platform Technology and Service (PlatCon), 2019, pp. 1-4, doi: 10.1109/PlatCon.2019.8669419.

[17] R. J. Weible, “Privacy and data: An empirical study of the influence of types of data and situational context upon privacy perceptions,” 1994.

[18] K. Bhargavan, A. Delignat-Lavaud, and C. Fournet, “Formal verification of smart contracts,” 2016 ACM workshop on programming languages and analysis for security, 2016, pp. 91-96, doi: 10.1145.2993600.2993611.

[19] J. Bernal Bernabe, J. L. Canovas, J. L. Hernandez-Ramos, R. Torres Moreno and A. Skarmeta, “Privacy-Preserving Solutions for Blockchain: Review and Challenges,” in IEEE Access, vol. 7, pp. 164908-164940, 2019, doi: 10.1109/ACCESS.2019.2950872.

[20] M. Zholbaryssov, C. N. Hadjicostis and A. D. Dominguez-Garcia, “Privacy-Preserving Distributed Coordination of Distributed Energy Resources,” 2020 59th IEEE Conference on Decision and Control (CDC), 2020, pp. 4689-4696, doi: 10.1109/CDC42340.2020.9303977.

[21] N. B. Truong, K. Sun, G. M. Lee and Y. Guo, “GDPR-Compliant Personal Data Management: A Blockchain-Based Solution,” in IEEE Transactions on Information Forensics and Security, vol. 15, pp. 1746-1761, 2020, doi: 10.1109/TIFS.2019.2948287.

[22] W. Xie, C. Zheng, X. Luo and X. Wang, “A Sensitivity Assessment Method for Marine Big Data,” 2020 IEEE 5th International Conference on Signal and Image Processing (ICSIP), 2020, pp. 1075-1080, doi: 10.1109/ICSIP49896.2020.9339376.

[23] Y. C. Yau, P. Khethavath and J. A. Figueroa, “Secure Pattern-Based Data Sensitivity Framework for Big Data in Healthcare,” 2019 IEEE International Conference on Big Data, Cloud Computing, Data Science & Engineering (BCD), 2019, pp. 65-70, doi: 10.1109/BCD.2019.8885114.

[24] P. Mindek, G. Mistelbauer, and E. Groller, “Data-sensitive visual navigation,” the 33rd Spring Conference on Computer Graphics, 2017, no. 9, pp. 1-10, doi:10.1145.3154.3154361.

[25] J. Domingo-Ferrer, O. Farràs, and J. Ribes-González, “Privacy-preserving cloud computing on sensitive data: A survey of methods, products and challenges,” Computer Communications, 2019, vol. 140-141, pp. 38-60, doi:10.1016.2019.04.011.

[26] V. U. Rani and M. S. Rao, "PrivGuard: Sensitivity Guided Anonymization based PPDM with Automatic Selection of Sensitive Attributes," 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS), 2021, pp. 1832-1837, doi: 10.1109/ICACCS51430.2021.9441991.

[27] C. Lian and Z. Chen, “Anonymous Privacy Protection Algorithm Based on Sensitive Attribute Classification,” 2020 2nd International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI), 2020, pp. 222-226, doi: 10.1109/MLBDBI51377.2020.00048.

[28] S. J. Pee, E. S. Kang, J. G. Song and J. W. Jang, “Blockchain based smart energy trading platform using smart contract,” 2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), 2019, pp. 322-325, doi: 10.1109/ICAIIC.2019.8668978.

[29] De Kruijff J and H. Weigand, “Ontologies for commitment-based smart contracts,” OTM Confederated International Conferences, Springer, Cham, vol. 10574, 2017, pp. 383-398, doi:10.1007.978-3-319-69459-7_76.

[30] Y. Liu, “PCTL∗ stochastic model checking label-extended probabilistic Petri net system model,” 2014 IEEE 5th International Conference on Software Engineering and Service Science, 2014, pp. 287-290, doi: 10.1109/ICSESS.2014.6933565.

[31] A. Morozov, K. Ding, M. Steurer and K. Janschek, “OpenErrorPro: A New Tool for Stochastic Model-Based Reliability and Resilience Analysis,” 2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE), 2019, pp. 303-312, doi: 10.1109/ISSRE.2019.00038.

[32] B. Bünz, S. Agrawal, and M. Zamani, “Zether: Towards privacy in a smart contract world,” International Conference on Financial Cryptography and Data Security. Springer, Cham, vol. 12059, 2020, pp. 423-443, doi:10.1007.978-3-030-51280-4_23.

[33] Jin W, “Privacy Requirement Modeling and Verification in Cloud Computing,” the 18th ACM/IEEE International Conference of Model-Driven Engineering Languages and systems 2015, doi: 10.7544/issn1000-1239.2015.20150513.

[34] Baier, Christel, “Model checking probabilistic systems,” Handbook of Model Checking, Springer, Cham, 2018, pp. 963-999. doi: 10.1007.978-3-319-10575-8_28.

[35] Y. Wang, N. Roohi, M. West, M. Viswanathan and G. E. Dullerud, “Statistically Model Checking PCTL Specifications on Markov Decision Processes via Reinforcement Learning,” 2020 59th IEEE Conference on Decision and Control (CDC), 2020, pp. 1392-1397, doi: 10.1109/CDC42340.2020.9303982.

[36] M. Kwiatkowska, G. Norman, and D. Parker, “PRISM: Probabilistic symbolic model checker,” International Conference on Modelling Techniques and Tools for Computer Performance Evaluation, vol. 2324, Springer, Berlin, Heidelberg, 2002, pp. 200-204, doi:10.1007.3-540-46029-2_13.

[37] X. Sun, H. Khedr, and Y. Shoukry, “Formal verification of neural network controlled autonomous systems,” Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, 2019, pp. 147-156, doi:10.1145.3302504.3311802.

[38] M. Lahijanian, S. B. Andersson and C. Belta, “Control of Markov decision processes from PCTL specifications,” Proceedings of the 2011 American Control Conference, 2011, pp. 311-316, doi: 10.1109/ACC.2011.5990952.

[39] https://docs.soliditylang.org/en/v0.8.4/solidity-by-example.html.