Privacy-preserving Anomaly Detection of Encrypted Smart Contract for Blockchain-based Data Trading

Dajiang Chen, *Member, IEEE*, Zeyu Liao, Ruidong Chen, Hao Wang, Chong Yu, Kuan Zhang, *Member, IEEE*, and Ning Zhang, *Senior Member, IEEE*, Xuemin (Sherman) Shen, *Fellow, IEEE*

***Abstract*—In a blockchain-based data trading platform, data users can purchase data sets and computing power through encrypted smart contracts. The security of smart contracts is important as it relates to that of the data platform. However, due to the inability to apply to detection rules with complex structures and the inefficiency of detection, existing malicious code detection methods are not suitable for the encrypted smart contracts in blockchain-based data trading platforms with high transaction rate requirements. In this paper, a practical and privacy-preserving malicious code detection method is proposed for encrypted smart contract in blockchain-based data trading platform. Specifically, we design two kinds of miners to act as the malicious rule processor and the detector respectively for inspecting the encrypted smart contract. The rule processor generates an obfuscated map with the original open-source malicious rule set. The detector performs a malicious inspection algorithm by inputting the obfuscated map and the randomized tokens, where the latter is generated from smart contract. Then, we theoretically analyze the security syntax of the proposed method. The analysis results demonstrate the proposed scheme can achieve $\mathcal{L}$-secure against adaptive attacks. Extensive experiments are carried out through the open-source real rule sets, which show that the proposed scheme can reduce communication time and communication overhead.**

***Index Terms*—Data Trading Platform, Encrypted Smart Contract, Malicious Code Detection, Privacy-preserving**

I. Introduction

Massive amounts of data is constantly being produced with the development of emerging technologies, e.g., Blockchain (BC) [1]–[3], Cloud/Fog/Edge Computing [4], Artificial Intelligence (AI), and 5G Communication [5], [6], and their applications in Industrial Internet of Things (IIoT) [7], Internet of Medical Things (IoMT) [8], Smart City [9], *etc.* These data can be used for decision-making by leveraging big data analysis tools (e.g., AI algorithms and could computing) to benefit our economy and society [10]–[12]. As an emerging business model in the sharing economy, data trading can effectively collect enough target data for users and bring value to the data collector/owner [13]–[15].

Existing big data trading systems face two major problems as follows. (1) How to efficiently protect the copyright of data; and (2) how to transfer copyright (or right to use) of data between two parties without trusted third party. As an alternative solution, blockchain technology is widely used in data copyright protection, and data trading between untrusted parties [16]–[18]. Actually, as a key enabling technology of data sharing and trading, blockchain has been used for data ownership confirmation, smart transaction contracts, and digital currency payments [19]–[21]. A blockchain-based programming platform allows developers to build distributed applications based on smart contracts [22], so that the programs can be executed automatically in full accordance with the contracts on the blockchain.

In this paper, a practical application scenario of blockchain-based data trading platform is considered. The platform consists of multiple data centers, a large number of data users, and smart contracts. Data centers maintain massive valuable data that can be traded and have a large amount of computing resources to perform AI algorithms for data mining [23], [24]. Data users purchase data sets and computing power via smart contracts, and analyze the purchased data sets through the corresponding data analysis algorithms (which are one part of the smart contract and executed by the data center), e.g., AI algorithms [25]. The corresponding computing results are return to data users based on the code logic of the smart contract, and data centers receive corresponding digital currency rewards. According to the consensus, the blockchain network guarantees that the status of the contract and the returned results are not tampered with.

On the one hand, for the security and privacy of the contact content, the smart contract should be encrypted to ensure only the authorized parties (e.g., the corresponding data center) can obtain the plaintext of smart contract [26], [27]. On the other hand, due to the imperfect design of the blockchain-based platform itself, security vulnerabilities are not ruled out. Actually, regardless of environmental factors, inadvertent negligence or

Corresponding author: Ruidong Chen. This work is jointly supported by NSFC (No. 62002047 and 61872059), the International Scientific and Technological Innovation Cooperation Project in Sichuan Province (No. 2020YFH0062), and in part by the Demonstration of Scientific and Technology Achievements Transform in Sichuan Province under Grant 2022ZHCG0036.

Dajiang Chen, Zeyu Liao, and Hao Wang are with the Network and Data Security Key Laboratory of Sichuan Province, School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu, 611731, China (Email: djchen@uestc.edu.cn; {202122090439; 201922090416}@std.uestc.edu.cn).

Ruidong Chen is with School of Computer Science & Engineering, University of Electronic Science and Technology of China, Chengdu, 611731, China (Email: crdchen@163.com).

Chong Yu and Kuan Zhang is with the Department of Electrical and Computer Engineering, University of Nebraska-Lincoln, USA (e-mail: cyu6@huskers.unl.edu; kzhang22@unl.edu).

Ning Zhang is with the Department of Electrical and Computer Engineering, University of Windsor, ON, N9B 3P4, Canada (e-mail: ning.zhang@uwindsor.ca).

Xuemin (Sherman) Shen is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, N2L 3G1, Canada (e-mail: sshen@uwaterloo.ca).

intentionally malicious behaviors of smart contract developers may raise serious security risks in the smart contract, which will cause irreparable losses to data security and user property [28]. As a result, it is necessary to ensure that the content of encrypted smart contract is legal before being published to the blockchain to prevent malicious activities. Accordingly, a practical and privacy-preserving encrypted smart contract inspection is required to support blockchain-based data trading platform.

In the existing literature, privacy-preserving encrypted traffic detection methods have emerged recently, and most of them introduce a third-party middlebox which performs detection during the transmission [32]–[35]. In [32], a scheme named BlindBox is proposed, which ensures that the rules and plaintext data are visible only to the rule generator and endpoints respectively. It completes the detection of encrypted data without disclosing plaintext information. However, it incurs high computing/communication costs to implement garbled circuit [36] and oblivious transfer [37]. To improve the efficiency of the scheme above, an inspection scheme for encrypted traffic, namely PrivDPI, is presented [33]. In PrivDPI, the set of obfuscated rules can be generated from the rule set for malicious detection with an obfuscated rule generation algorithm, in which, the obfuscated rules obtained in preceding section can be reused to generate obfuscated rules in subsequent session. However, it requires high computing resource due to the usage of bilinear mapping. An encryption rule filter is designed for anomaly detection to store the encrypted action operations [34]. When the processed "content" option matches the randomized token successfully, the encrypted "action" of the rule can be restored and the corresponding operation should be executed. Later, a privacy-preserving inspection method of encrypted traffic is presented by using symmetric cryptographic techniques for IoT [35].

This paper aims to design a practical malicious code detection scheme of encrypted smart contract for blockchain-based trading platform. However, when privacy-preserving encrypted traffic detection meets blockchain-based trading platform, there are some technical challenges to be solved as follows. (1) *How to design a malicious code detection scheme that supports all forms of open-source Snort rules.* Note that, the rule used in existing schemes for malicious detection only consists single "content" option, while, in practice, a rule for malicious detection usually consists of multiple "content" options and an "action", such as, Snort rules (please refer to Sec. III-B). (2) *How to design a malicious code detection scheme with high efficiency.* Existing schemes only perform rough sliding window processing on each rule for obfuscated rules generation, which destroys the semantic integrity of the rules and is inefficient. Moreover, most of existing works require high computing/communication costs [32]–[35]. As a result, these schemes are not suitable for the high transaction rate requirements of blockchain-based data trading platforms. (3) *How to design a malicious code detection scheme in a distributed way.* In blockchain-based data trading platforms, a centralized encryption smart contract anomaly detection scheme will face the problem of single point failure, so it is necessary to utilize the node resources of the peer-to-peer network for a fee to achieve distributed encryption smart contract anomaly detection. Accordingly, it is necessary to design an efficient and distributed malicious code detection scheme that supports malicious rules with multiple "content" options for encrypted smart contract in a blockchain-based trading platform.

In this paper, a practical malicious code detection scheme of encrypted smart contract is proposed by leveraging lightweight cryptography techniques. In the proposed scheme, a new obfuscated rule generation algorithm is designed to support all forms of open-source Snort rules for malicious detection. To further improve efficiency and scalability of inspection in a distributed way, in the proposed scheme, two kinds of miners are selected to perform encrypted smart contract inspection as the malicious rule processor and the detector, respectively. Specifically, one kind of miner is responsible for rule processing (i.e., rule processor) to get an obfuscated map with the original open-source malicious rule set. Another kind of miner who acts as the detector uses the obfuscated map to perform the malicious code inspection of the encrypted smart contract from the developer. The main objective of the proposed scheme includes (1) Privacy-preserving of Smart Contract: before the encrypted smart contract reaches the data center, other participants except the developer cannot obtain the code content; (2) Confidentiality of Rules: the original rule set is chosen by data center of the trading platform, and is visible only to the rule processor and data center; and (3) Reliable Detection: when the rule processor and the detector are honest but curious, the false negative and false positive errors of detection can be negligible. A new security syntax (e.g., $\mathcal{L}$-secure) for encrypted smart contract inspection is defined formally. The formal proof of security proves that the proposed scheme is $\mathcal{L}$-secure against adaptive attacks.

The open-source Emerging Threats Snort Rules are utilized in proposed scheme. In Snort rules, a Snort rule can specify multiple "content" options, and the "content" options can be used in conjunction with "modifier" options to detect specific content in a traffic to trigger an appropriate action. The proposed scheme can be applicable to all forms of rules (the rule with single "content" option and the rule with multiple "content" options) in the existing open-source rule set. The proposed scheme analyzes and refines the rule structure, hides the "content" options, "modifier" options and "action" of each rule, and generates a small-capacity obfuscated map. When all the "content" options of a rule match and the corresponding "modifier" options match, the action operation of the rule can be restored. This greatly improves the accuracy of detection. We evaluate the performance of the proposed scheme with extensive experiments, and the experimental results demonstrate that the scheme has high detection rate, low time consumption/communication overhead. For example, the developer takes 285 ms to obtain randomized tokens with 5030 KB; the rule processor uses around 1987 ms to generate the obfuscated map generated by 5000 rules; and the detector takes approximately 723 ms to detect smart contract (with size 25 KB) by using the obfuscated map from 4000 rules.

We summarize the contributions of this paper as follows.

- We propose a privacy-preserving and practical malicious

code detection scheme of encrypted smart contract in blockchain-based data trading platform by leveraging lightweight cryptography techniques. As far as we know, it is a first privacy-preserving anomaly detection of encrypted data scheme that is suitable for all forms of open-source Snort rules in a distributed way.
- We define the security syntax of encrypted smart contract inspection scheme, i.e., the completeness and the $\mathcal{L}$-secure of a scheme. Given a leakage function $\mathcal{L}$, we prove that the simplified version of the proposed scheme is completeness and $\mathcal{L}$-secure against adaptive attacks.
- We conduct extensive simulation experiments to evaluate the performance of the proposed scheme on the open-source Emerging Threats Snort Rules. The results demonstrate that the proposed scheme can achieve high detection accuracy, small time cost, and low communication consumption.

II. Related Work

A. Smart Contract

A smart contract is a computer program that is verified by means of information dissemination and can run autonomously on the blockchain. In [22], Nick introduced the concept of smart contract. When a smart contract is deployed in a block, trigger conditions and corresponding response rules are needed to be preset, and the relevant status will also be recorded in the smart contract. If the condition is triggered, the relevant action from the calling node will be executed in response. Only the behavior of modifying the contract state or value eventually is recorded in the blockchain, other behaviors are not be disclosed.

As a programming platform with open source code, Ethereum enables the developers to design diverse distributed applications with smart contracts [29]. Ethereum can be used to guarantee behavior, design protocols, and process transactions such as enterprise management, voting applications, and e-commerce transactions [30]. Based on the deployed smart contract, Ethereum nodes can execute the contract in a decentralized manner. Generally, the smart contract is run by the local Ethereum virtual machine EVM of the node. EVM first interprets the contract program to obtain the bytecode file, and then executes it. When the transaction information related to the smart contract is stored in the blockchain, all nodes will run the contract according to the transaction information and change the state definitively.

B. Encrypted Traffic Detection

In recent years, many novel encrypted traffic detection technologies were proposed. To solve the dilemma between privacy of encrypted traffic and content security, a privacy-preserving deep packet filtering protocol was designed to perform filtering function over encrypted traffic while guaranteeing the data and rules privacy in software defined networks [31]. In [32], BlindBox was proposed to realize deep packet inspection (DPI) over encrypted traffic. However, the usage of complex cryptography such as garbled circuit and oblivious transfer leads to the lower efficiency in obfuscated rule preparation phase and hinders the practical application.

On the basis of BlindBox, a scheme named Embark was proposed in [48]. In Embark, a novel encryption scheme, namely PrefixMatch, was introduced to realize DPI over encrypted traffic in a cloud-based middlebox, which enabled the cloud-based middlebox to detect whether an encrypted IP address is in a valid encrypted range. Later, an novel scheme BlindIDS was presented to further improve the performance of DPI over encrypted traffic by using pairing-based public key techniques [49]. However, its versatility is greatly reduced due to its incompatibility with TLS protocol. Subsequently, a more efficient scheme PrivDPI was proposed by utilizing bilinear mapping and a novelty obfuscated rule generation algorithm [33]. Yuan *et al*. designed an encryption rule filter to realize DPI over encrypted traffic, in which, an "action" will be restored and the corresponding operation will be executed, when the rule content matches the token successfully [38]. However, an inspection rule contains multiple fragments with different lengths generally. In this case, it is infeasible to realize DPI over encrypted traffic with the scheme in [38] by generating tokens of different sizes, as it will result in significant communication overhead for token set transmission. Other related works also include [39]–[43]. In [39], an efficient privacy-preserving deep packet inspection system was proposed in secure outsourced middleboxes. In [40], a privacy-preserving anomaly detection system was proposed for industrial control systems. Zhang *et al*. designed a blockchain-based privacy-preserving quality-aware incentive scheme [41]. In [42], a deep learning-based vulnerability detection framework was developed for smart contracts. Ivanov *et al*. proposed a real-time smart contract security testing approach, namely transaction encapsulation, to increase the vulnerability coverage [43].

C. Searchable Symmetric Encryption

Searchable symmetric encryption supports the implementation of secure matching schemes that allow one party to outsource its data storage to another party in a private manner, while the data owner retains the ability to selectively search the data. Song *et al*. proposed a symmetric searchable encryption protocol in [44], which divides the plaintext file into "words" and encrypts the word collection. By scanning the entire ciphertext file and comparing the ciphertext words, the existence of the keyword in the ciphertext can be confirmed. In [45], Curtmola *et al*. standardized symmetric searchable encryption, and designed two schemes, namely SSE-1 and SSE-2, which can achieve indistinguishable security in non-adaptive/adaptive attack models. Later, Cash *et al*. presented a dynamic symmetric searchable encryption scheme to effectively search the encrypted database [46]. It is proved that the proposed scheme can achieve $\mathcal{L}$-secure against adaptive attacks.

The main objective of this paper is to design a practical encryption smart contract anomaly detection scheme on the blockchain-based data trading platform. However, existing methods face some challenges when applied to encrypted smart contract anomaly detection on blockchain-based data

TABLE I
A SUMMARY OF IMPORTANT NOTATIONS.

Notation	Definition
$G(K_l)$	The pseudorandom function with key K_l;
$F(K_s)$	The pseudorandom function with key K_s;
$f(K_h)$	The pseudorandom function with key K_h;
K_{AES}	The AES symmetric key;
M	The plaintext of smart contract at the Client;
$\mathcal{W}$	The set of tokens $\{W_1,\cdots,W_\ell\}$ for plaintext M;
$\mathcal{T}$	The set of tokens $\{T_1,\cdots,T_\ell\}$ for plaintext M;
$\mathcal{R}$	The set of Snort rules $\{R_1,R_2,\cdots,R_1\}$;
$seg_{i_j,k}$	The k-th segmentation of j-th content of i-th rule;
$\mathcal{P}$	The collection of secret shares $\{p_1,\cdots,p_n\}$ for processed content option *con*;
$\mathcal{Q}$	The collection of secret shares $\{q_1,\cdots,q_n\}$ for action.

TABLE II
A SUMMARY OF COMMON MODIFIERS IN SNORT RULES.

Notation	Definition
offset	Set the location to start searching;
depth	Set the maximum depth of search;
distance	After a "content" is successfully matched, the next "content" is matched after at least *distance* bytes;
within	After a "content" is successfully matched, the next "content" is matched after at most *within* bytes;
nocase	It specifies that the "content" string is not case sensitive;
http_header	It limits the search in the header of extracted HTTP client request and server response, and the extracted header can be unformatted;
http_client_body	The modifier restricts the search in the body of the HTTP client request message;
http_cookie	It restricts the search in the cookie header fields of HTTP requests and responses.

trading platforms. (1) They are not suitable for the detection rules of complex structures, e.g., Snort rules; (2) their detection efficiency cannot meet the demand of data trading platform for efficient transaction speed; and (3) they cannot be directly applied to a peer-to-peer network environment. Different from existing methods, we propose a privacy-preserving and practical malicious code detection scheme of encrypted smart contract in blockchain-based data trading platform by leveraging lightweight cryptography techniques, which can be applied to all forms of open-source Snort rules in a peer-to-peer network environment.

III. NOTIONS AND PRELIMINARIES

Notions. Let x and y be two bitstreams, $x\oplus y$ be the bitwise XOR of x and y, and $x\|y$ be the splicing of x and y. Random variables (RVs) are denoted by $X,Y,K,\cdots$, and the realization of them are denoted by $x,y,k,\cdots$. The main notations used in this paper are listed in Tab. I.

A. Preliminaries

By taking λ as a security parameter, several basic definitions are introduced as follows.

Definition 1: Let $F:\{0,1\}^{\alpha_1}\times\{0,1\}^{\beta_1}\to\{0,1\}^{\gamma_1}$ be an efficient and keyed function. F is a variable-input-length pseudo-random function, if the function

$$Adv^{pdf}_{F,\mathcal{A}}(\lambda)=\Pr[PRFReal^{\mathcal{A}}_F(\lambda)]-\Pr[PRFRand^{\mathcal{A}}_F(\lambda)] \quad (1)$$

is negligible for all probabilistic polynomial time (p.p.t.) adversary $\mathcal{A}$, in which, *PRFReal* and *PRFRand* are two games as follows. In *PRFReal*, the adversary $\mathcal{A}$ first obtains key K from a dictionary with index j, and then, queries function F with outputting y and inputting an index K and x. In *PRFRand*, the adversary $\mathcal{A}$ queries a random oracle R with outputting y and inputting an index j and x.

Definition 2: $H:\{0,1\}^{\alpha_2}\times\{0,1\}^{\beta_2}\to\{0,1\}^{\gamma_2}$ is called a *collision-resistant hash function* if the following experiment is negligible for any p.p.t. adversary $\mathcal{A}$.

- $\mathcal{A}$ succeeds if it outputs distinct x and x' with $H_K(x)=H_K(x')$, in which, $H_K(\cdot)=H(K,\cdot)$ and key K is selected from $\{0,1\}^{\alpha_2}$ uniformly at random.

B. Snort Rules

Snort is an intrusion detection system, which analyzes and summarizes the known intrusion behaviors, summarizes the intrusion features, and forms rules. Its rule set *Snort Emerging Threats* is used in this paper as the initial rules to realize inspection. A rule includes two parties: header and options. *Header*: the rule header of Snort rules contains the response action and data flow direction, which is the content before the first "(". *Options*: the rule options of Snort rules is the content between "(" and ")" , which are separated by ";". A Snort rule can have multiple "content" option, and the "content" option can have multiple modifiers. Only when the packet matches one or several rule options successfully, can it be regarded as an insecure packet with intrusion behavior by Snort DPI system. We present a simple Snort rule as follows.

$$\begin{aligned} &alert\ tcp\ \$EXTERNAL\_NET\ any \\ &\quad\to \$HOME\_NET\ 443\ (\\ &msg\ :\ \text{“}openssl\ Heartbleed\ attack\text{”}\ ; \\ &content\ :\ \text{“}ftp.log'',fast\_pattern,nocase) \end{aligned}$$

The modifiers can be roughly divided into two categories. The first one is the modifiers such as $offset$, $depth$, $distance$ and *within*, which explicitly limit the position of "content" with numbers. The second category is about content modifiers, such as "no case" which ignores case and a series of restrictive modifiers related to HTTP. When processing the packet payload, the location relationship can be clearly obtained from tokenization, but it is difficult to control the case format and HTTP related operations. Therefore, the proposed scheme focuses on the first type of modifiers. A list of important notations of *Snort* rule is shown in Table II.

IV. SYSTEM OVERVIEW

A. System Model

The system model is shown in Fig. 1, which includes four entities as follows. The data user, namely task requester (*TR*), sends the encrypted smart contract for data analysis.

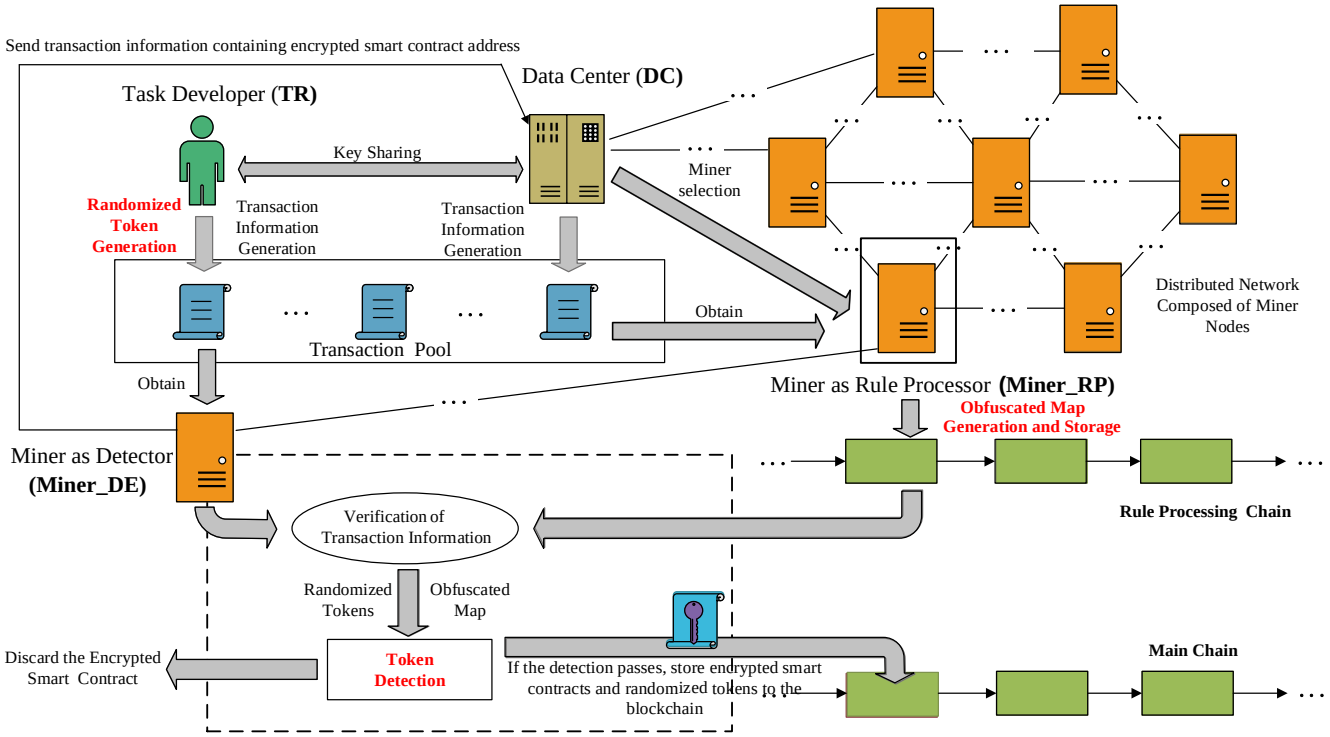


Fig. 1. The system architecture.

The data center (*DC*) sells its data and computing power to data user for data analysis The miner ($Miner_{RP}$) acts as a malicious rule processor, and the other miner ($Miner_{DE}$) performs anomaly detection and data forwarding. Moreover, there are two blockchains in our system: main chain and rule processing chain.

TR: When a task developer *TR* plans to send a smart contract on data trading platform to initiate a task request, who needs to generate a encrypted packet contained two data streams: an encrypted smart contract generated by symmetric key and a randomized token collection. Then the encrypted packet is sent to the trading pool by *TR*. If the randomized token collection passes the malicious detection, the encrypted packet will be packaged on blockchain. The encrypted packet can be obtained by *DC* from the blockchain.

$Miner_{RP}$: $Miner_{RP}$ is a miner who provides the processing of detection rules. $Miner_{RP}$ generates the obfuscated rules from detection rules (e.g., Snort rules), with which, $Miner_{DE}$ can detect malicious behavior if there is at least one "content" option in each rule to describe malicious code.

$Miner_{DE}$: $Miner_{DE}$ is a miner to monitor the encrypted packet from *TR* using the obfuscated rules issued by $Miner_{RP}$. If $Miner_{DE}$ detects malicious data in the encrypted packet, it will interrupt the flow path and send the encrypted packet to $Miner_{RP}$ for further analysis; otherwise, it will forensics the encrypted packet and packages it with other transactions on blockchain.

DC: After receiving the valid traffic from blockchain, *DC* first decrypts the encrypted packet, and then verifies the consistency of the content, i.e., whether the randomized token collection is generated correctly. If so, *DC* deploys the smart contract for the task of data mining; otherwise, *TR* is dishonest.

B. Threat Model

In this paper, the adversary model is considered as follows.

Trust Assumption on Endpoints TR and DC. It is assumed that *DC* is credible without considering the problems caused by imperfect programming language and virtual environment. It is assumed that the *TR* can be malicious. *TR* wants to design a smart contract with malicious code and escape malicious detection. For instance, *TR* can generate fraudulent tokens to avoid the malicious detection.

Semi-honest miner $Miner_{DE}$ and $Miner_{PR}$. It is considered that the miners $Miner_{DE}$ and $Miner_{PR}$ are semi-honest, e.g., honest-but-curious. Specifically, $Miner_{DE}$ must follow the steps of the proposed scheme to detect malicious code; but it wants to obtain the sensitive information from the smart contract, and attempts to infer the proprietary inspection rules from $Miner_{RP}$.

C. Design Goals

According to the above model, the design goals of the proposed system are as follows:

Privacy-Preserving of Smart Contract: This property requires that the privacy of smart contract of *TR* should be protected. Before the encrypted smart contract reaches the *DC*, other participants except the *TR* cannot obtain the code content.

Confidentiality of Rules: Only *DC* can choose the malicious rule set, and $Miner_{RP}$ can learn and process the rules. In other words, other participants cannot know whether the plaintext contains the content that matches the rules.

Efficiency and Accuracy: To facilitate practical application, our system should have negligible false negative and false positive errors with low computation/communication cost.

V. The Proposed Scheme

In this section, the proposed scheme is introduced, which consists of four phases: *Setup*, *Obfuscated Map Generation*, *Randomized Token Generation* and *Malicious Detection*.

A. Setup

The setup phase is run by the *TR* and *DC* to exchange keys between them. These keys are used in subsequent phases for the processing of rules and smart contracts, as well as the selecting of the $Miner_{RP}$.

Assumed that *TR* and *DC* have a pair of public/private keys (PK_{TR}, SK_{TR}) and (PK_{DC}, SK_{DC}), respectively, which can be generated by a Certificate Authority (CA).

Key Distribution: TR and *DC* negotiate a symmetric key K_{AES} for encrypting the smart contract with AES algorithm and a set of key $K_{SET} = \{K_s, K_h, K_l\}$ for generating obfuscated map and randomized tokens by using *Diffie-Hellman* key distribution scheme [47].

$Miner_{RP}$ Selection: After obtaining the keys K_{AES} and K_{SET}, a miner should be selected from a set of miners $\mathcal{A} = \{min_1, min_2, \cdots, min_\delta\}$ as rule processor $Miner_{RP}$ to process the original rule set. In order to provide the randomness of the miner choose, *DC* calculates the ID of miner from K_{AES}, K_{SET}, and a time stamp ts as follows.

$$id_s = H(K_{AES} \| K_{SET} \| ts) \mod \delta \tag{2}$$

According to Eq. 2, *DC* selects min_{id_s} as the rule processor. Moreover, for subsequent rule processing, the initial information of the transaction (i.e., Inf_{DC} and Sig_{DC}) needs to be put into the transaction pool of Rule Processing Chain (RP Chain) by *DC* in order to be shared with $Miner_{RP}$.

$$Inf_{DC} = Tran_{id} \| addr_{DC} \| addr_{min_{id_s}} \| Enc_{PK_{min_{id_s}}}(K_{SET}) \| ts \| id_s \tag{3}$$

$$Sign_{DC} = Sign_{SK_{DC}}(Inf_{DC}) \| Sign_{SK_{DC}}(Tran_{id} \| K_{SET} \| ts) \tag{4}$$

where $Tran_{id}$ is the transaction ID, $addr_{DC}$ and $addr_{minid_s}$ are the addresses of *DC* and miner respectively, $PK_{min_{id_s}}$ is the public key of miner min_{id_s}, $Sign(\cdot)$ is the signature algorithm designed in [50] and $Sign_{DC}$ denotes the signature of *DC*.

B. Obfuscated Rule Generation

Suppose that the open-source Snort rule set $\mathcal{R} = \{R_1, R_2, \cdots, R_\mathfrak{r}\}$ is selected as the detection rules, where the "content" option in each rule is used to match the smart contract to be inspected. An efficient obfuscated rule generation algorithm is proposed in this phase, which is based on the security framework of searchable symmetric encryption. Concretely, $Miner_{RP}$ generates obfuscated map through the following steps.

Key Acquisition: As the rule processor, miner min_{id_s} first obtains Inf_{DC} and $Sign_{DC}$ from the transaction pool of RP chain. Then min_{id_s} decrypts Inf_{DC} with its own private key $SK_{min_{id_s}}$ to obtain the set of keys $K_{SET} = (K_s, K_h, K_l)$. Next, min_{id_s} verifies the validity of the signature $Sign_{SK_{DC}}(Tran_{id} \| K_{SET} \| ts)$ from $Tran_{id}$, K_{SET}, and ts. Meanwhile, the obtained signature of *DC* can be used as part of generated obfuscated rule later.

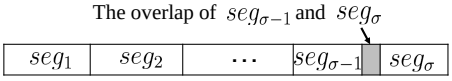


Fig. 2. Content segmentation.

Preliminary Treatment: For each Snort rule R_i, extracting the content options and its modifier options to get the $\{con_{i_j} : mod_{i_j}\}$ key value pair, where con_{i_j} is the j-th content of rule R_i, and the mod_{i_j} is the splicing of all modifiers corresponding to con_{i_j}. Note that, a Snort rule may contain multiple content options, and each content option may be modified by multiple modifier options. Moreover, each rule only has one header contained the security action (such as alert, log, pass, activate, and dynamic). Accordingly, rule R_i is initially processed as

$$R_i = \{con_{i_1} : mod_{i_1};\ con_{i_2} : mod_{i_2}; \cdots;\ con_{i_\tau} : mod_{i_{\tau_i}}; action_i\} \tag{5}$$

Content Segmentation: Taking the shortest length of all content options as len_{win}, all content options can be divided into several sub-contents with length len_{win}. As shown in Fig. 2, if the length of the last part is less than len_{win}, the last sub-content can be supplemented with the preceding bytes. Then, rule R_i is processed as

$$\begin{aligned} R_i = \{&seg_{i_1}1, seg_{i_1}2, \cdots, seg_{i_1\sigma_{i_1}} : mod_{i_1};\\ &seg_{i_2}1, seg_{i_2}2, \cdots, seg_{i_2\sigma_{i_2}} : mod_{i_2};\\ &\cdots\\ &seg_{i_{\tau_i}}1, seg_{i_{\tau_i}}2, \cdots, seg_{i_{\tau_i}\sigma_{i_{\tau_i}}} : mod_{i_{\tau_i}}; action_i\} \end{aligned} \tag{6}$$

Random ID Generation: An ID of each content of the rules is defined to uniquely identify the rule for facilitating the subsequent operations as follows. The Snort rules $\{R_1, R_2, \cdots, R_\mathfrak{r}\}$ are reordered randomly, and the new sequence number of R_i is denoted by rid_{R_i}. Then, the random ID of the j-th content of R_i can be denoted by id_{i_j} and expressed as $rid_{R_i} \| j \| b_{i_j}$, where $b_{i_j} \in \{0,1\}$, and $b_{i_j} = 1$ if $j = \sigma_{i_j}$; $b_{i_j} = 0$, otherwise. For example, a rule has 10 content options and its *rid* is 2. The fifth and tenth content option of the rule can be expressed as $id_{2_5} = 0002 \| 005 \| 0$ and $id_{2_{10}} = 0002 \| 010 \| 1$, respectively.

Action and Modifiers Hiding: In order to protect the privacy of rules, $Miner_{RP}$ needs to hide the action and content options. The proposed system aims to restore the content option through its segments, and then restore the final action that declared in rule header after the matching occurs through all the content options in the rule. To realize the efficient hide the action and modifiers, an efficient (n,n) secret sharing scheme is utilized as follows.

For action hide, we target on the security requirement that, only when all related content options of a rule are matched, the corresponding action can be recovered; otherwise, nothing can be revealed. Specifically, $\tau_i - 1$ random strings $q_{i_1}, \cdots, q_{i_{\tau_i - 1}}$ are generated with the same length as action, and let $q_{i_{\tau_i}} = q_{i_1} \oplus \cdots \oplus q_{i_{\tau_i - 1}} \oplus action_i$. The collection secret sharing is defined as $\mathcal{Q} = \{q_{i_1}, \cdots, q_{i_{\tau_i}}\}$, where each q_{i_j} is a share of

action. Then, the action can be restored as follows,

$$action_i = q_{i_1} \oplus \cdots \oplus q_{i_{\tau_i-1}} \oplus q_{i_{\tau_i}} \tag{7}$$

Next, the ID of each original content id_{i_j}, q_{i_j} and mod_{i_j} ($j \in \{1,2,\cdots,\tau_i\}$) are first processed by a hash function H. The corresponding hash value, secret share q_i and the modifiers mod_{i_j} are spliced together as C_{i_j}, i.e.,

$$C_{i_j} = H(id_{i_j}, q_{i_j}, mod_{i_j}) \| q_{i_j} \| mod_{i_j},\ \forall j \in \{1,2,\cdots,\tau_i\} \tag{8}$$

Similarly, given a content option con_{i_j} with σ_{i_j} segments, the corresponding C_{i_j} is also treated as a secret. Randomly choosing $\sigma_{i_j}-1$ strings $\{p_{i_j}(1),\cdots,p_{i_j}(\sigma_{i_j}-1)\}$, and the length of each bit string is the same bit length of C_{i_j}. Taking $p_{i_j}(\sigma_{i_j}) = p_{i_j}(1) \oplus \cdots \oplus p_{i_j}(\sigma_{i_j}-1) \oplus C_{i_j}$, we have

$$C_{i_j} = p_{i_j}(1) \oplus \cdots \oplus p_{i_j}(\sigma_i - 1) \oplus p_{i_j}(\sigma_{i_j}) \tag{9}$$

Finally, rule R_i can be processed as follows.

$$\begin{aligned} &\{seg_{i_1 1}, \cdots, seg_{i_1\sigma_{i_1}}{:}id_{i_1} \| p_{i_1}(1), \cdots, id_{i_1} \| p_{i_1}(\sigma_{i_1});\\ &seg_{i_2 1}, \cdots, seg_{i_2\sigma_{i_2}}{:}id_{i_2} \| p_{i_2}(1), \cdots, id_{i_2} \| 1 \| p_{i_2}(\sigma_{i_2});\\ &\cdots\\ &seg_{i_{\tau_i} 1}, \cdots, seg_{i_{\tau_i}\sigma_{i_{\tau_i}}}{:}id_{i_{\tau_i}} \| p_{i_{\tau_i}}(1), \cdots, id_{i_{\tau_i}} \| p_{i_{\tau_i}}(\sigma_{i_{\tau_i}})\} \end{aligned} \tag{10}$$

Obfuscated Map Generation: An obfuscated map Obm is generated by $Miner_{RP}$ as follows. For all $\theta \in \{1,2,\cdots,\sigma_{i_j}\}$, $(id_{i_j} \| p_{i_j}(\theta)) \oplus s$ is inserted into map $Obm[loc]$, where $s = f(K_h, seg_{i_j\theta})$, $loc = G(K_l, l)$ and $l = F(K_s, seg_{i_j\theta})$. To solve the hash collision problem, a variable-length hash bucket is built for each location. After obtaining the obfuscated map Obm, $Miner_{RP}$ puts the mapping table of Obm on RP chain as the body of a block. Specifically, the content of the block body contains

$$Inf_{RP}{=}Tran_{id} \| ts \| len_{win} \| Obm \| Sign_{SK_{DC}}(Tran_{id} \| K_{SET} \| ts) \tag{11}$$

and the signature

$$Sign_{RP} = Sign_{SK_{RP}}(Inf_{RP}) \tag{12}$$

where SK_{RP} is the secret key of $Miner_{RP}$.

For the details of *Obfuscated Rule Generation Algorithm*, please refer to Algorithm 1.

Algorithm 1: Obfuscated Rule Generation Algorithm

Require: The Snort rule set $\mathcal{R} = \{R_1, R_2, \cdots, R_\iota\}$, the shortest length len_{win} of all content options in rule set, and the initial information of transaction Inf_{DC} (i.e., Equ. (3)) from DC and the signature $Sign_{DC}$ (i.e., Equ. (4)) of DC, where R_i={con_{i_1}: mod_{i_1}; con_{i_2}: mod_{i_2};$\cdots$; con_{i_τ}: $mod_{i_{\tau_i}}$;$action_i$}.

Ensure: Inf_{RP} and $Sign_{RP}$.

(1) Key Acquisition.
- Obtaining $K_{SET} = (K_s, K_h, K_l)$ from $Enc_{PK_{min_{id_s}}}(K_{SET})$;
- Verifying the validity of the signature $Sign_{SK_{DC}}$ with transaction ID $Tran_{id}$, keys K_{SET}, and time stamp ts.

(2) Preliminary Treatment.
- For each $i \in \{1,\cdots,\iota\}$, $j \in \{1,\cdots,\tau_i\}$, con_{i_j} is divided into sub-contents with length len_{win}: $seg_{i_{j_1}1}, seg_{i_{j_1}2}, \cdots, seg_{i_{j_i}\sigma_{i_{j_i}}}$.

(3) Random ID Generation.
- The rules $\{R_1, R_2, \cdots, R_\iota\}$ are reordered randomly, and the new sequence number of R_i is denoted by rid_{R_i};
- The random ID of the j-th content of R_i is denoted by $id_{i_j} = rid_{R_i} \| j \| b_{i_j}$, where $b_{i_j} \in \{0,1\}$, and $b_{i_j} = 1$ if $j = \sigma_{i_j}$; $b_{i_j} = 0$, otherwise.

(4) Action and Modifiers Hiding.
- For all i, randomly choosing $\tau_i - 1$ bit strings $q_{i_1}, \cdots, q_{i_{\tau_i-1}}$ with the same length as $action_i$, and taking $q_{i_{\tau_i}} = q_{i_1} \oplus \cdots \oplus q_{i_{\tau_i-1}} \oplus action_i$;
- $\forall j \in \{1,2,\cdots,\tau_i\}$, let $C_{i_j} = H(id_{i_j}, q_{i_j}, mod_{i_j}) \| q_{i_j} \| mod_{i_j}$;
- For all C_{i_j}, randomly choosing $\sigma_{i_j} - 1$ strings $p_{i_j}(1), \cdots$, and $p_{i_j}(\sigma_{i_j} - 1)$ with the same length of C_{i_j}, and taking $p_{i_j}(\sigma_{i_j}) = p_{i_j}(1) \oplus \cdots \oplus p_{i_j}(\sigma_{i_j} - 1) \oplus C_{i_j}$, such that
$$C_{i_j} = p_{i_j}(1) \oplus \cdots \oplus p_{i_j}(\sigma_i - 1) \oplus p_{i_j}(\sigma_{i_j});$$
- Finally, rule R_i can be processed as Equ. (10).

(5) Obfuscated Map Generation.
- For all $\theta \in \{1,2,\cdots,\sigma_{i_j}\}$, $(id_{i_j} \| p_{i_j}(\theta)) \oplus s$ is inserted into map $Obm[loc]$, where $s = f(K_h, seg_{i_j\theta})$, $loc = G(K_l, l)$ and $l = F(K_s, seg_{i_j\theta})$.
- Taking SK_{RP} as the secret key of $Miner_{RP}$, and computing
$$Inf_{RP}{=}Tran_{id} \| ts \| len_{win} \| Obm \| Sign_{SK_{DC}}(Tran_{id} \| K_{SET} \| ts)$$
and the signature $Sign_{RP} = Sign_{SK_{RP}}(Inf_{RP})$.

C. Randomized Token Generation

To realize the detection on encrypted smart contract, TR needs to generate a randomized token collection $\mathcal{T}$, besides the traditional encrypted information $E_{K_{AES}}(M_{sc})$, where M_{sc} denotes the plaintext of smart contract. To obtain $\mathcal{T}$, TR first gets the Inf_{RP} and $Sign_{RP}$ from the RP chain, and checks the validity of the signatures $Sign_{DC}$ and $Sign_{RP}$. If so, TR divides the entire smart contract M into a token collection $\mathcal{W} = \{W_1, W_2, \cdots, W_\ell\}$, i.e.,

$$\mathcal{W} = \{W_1, W_2, \cdots, W_\ell\} = Token(M_{sc}) \tag{13}$$

by using a sliding-window-based tokenization algorithm $Token(\cdot)$, where each word W_i have the same length of sliding window len_{win}.

Then, TR utilizes pseudorandom functions and keys K_{SET} to generate randomized tokens $\mathcal{T}$ as follows. For each $\iota \in \{1,\cdots,\ell\}$, TR computes $T_i = F(K_s, W_\iota) \| f(K_h, W_\iota)$ by using two pseudorandom functions F and f with the keys K_s and K_h, respectively. Besides matching the rule's content options, it also needs to satisfy the corresponding requirements in the modifier options. Therefore, the starting position sp_ι of W_ι should be added as the following form.

$$T_\iota = F(K_s, W_\iota) \| f(K_h, W_\iota) \| sp_\iota \tag{14}$$

Next, TR can obtain the transaction information $TransInf_{TR} = Inf_{TR} \| Sign_{TR}$ by computing

$$Inf_{TR}{=}Tran_{id} \| ts \| len_{win} \| id_s \| addr_{TR} \| addr_{DC} \| \mathcal{T} \| E_{K_{AES}}(M_{sc}) \tag{15}$$

$$Sign_{TR} = Sign_{SK_{TR}}(Inf_{TR}) \tag{16}$$

where $addr_{TR}$ denotes the IP addresses of TR, $addr_{dest}$ indicates the IP address of DC, $\mathcal{T} = \{T_\iota\}_{\iota=1}^{\ell}$, and SK_{TR} denotes the secret key of TR.

Finally, TR puts $TransInf_{TR}$ into the transaction pool of main chain for subsequent token detection operations.

D. Malicious Detection

In token detection phase, a miner $Miner_{DE}$ is introduced as a detector to detect the malicious code in the smart contract. Before performing token detection, the following verification operations are required.

- $Miner_{DE}$ first obtains Inf_{TR} and $Sign_{TR}$ from the transaction pool of main chain, and Inf_{RP} and $Sign_{RP}$ from RP chain.
- And then, $Miner_{DE}$ compares whether the len_{win} in Inf_{TR} and Inf_{RP} are the same: if so, $Miner_{DE}$ continues with the following step; otherwise, the transaction is invalid.
- Next, $Miner_{DE}$ checks the validity of the two signatures $Sign_{TR}$ and $Sign_{RP}$: if so, $Miner_{DE}$ continues with the following step; otherwise, the transaction is invalid.
- Finally, $Miner_{DE}$ extracts the randomized token collection $\mathcal{T}$ and obfuscated mapping table Obm.

After accomplishing the verification, $Miner_{DE}$ performs the following steps.

- For $\iota \in \{1,2,\cdots,\ell\}$, $Miner_{DE}$ splits T_i into $t'_\iota = F(K_s, W_\iota)$, $t''_\iota = f(K_h, W_\iota)$ and $L_\iota = sp_\iota$, calculates the $loc = G(K_l, t'_\iota)$, and looks up the Obm mapping table through the location information loc to find the stored result $Obm[loc] = (id_{i_j} \| p_{i_j}(\theta)) \oplus s$.
- $Obm[loc]$ is XOR with t''_ι to obtain the splicing of ID id_{i_j} and the share $p_{i_j}(\theta)$, where $id_{i_j} = rid_{R_i} \| j \| b_{i_j}$.
- $Miner_{DE}$ maintains a table $\mathcal{T}_{con}$, and stores the divided id_{i_j}, $p_{i_j}(\theta)$, and sp_ι.
- For each id_{i_j}, let $\{id_{i_j}, p_{i_j}(\theta_\nu), sp_{\iota_\nu}\}_{\nu=1}^{\nu_0}$ be the set of triples with the same content ID id_{i_j}. If $|sp_{\iota_\nu} - sp_{\iota_{\nu-1}}| = len_{win}$ for $\nu < \nu_0$ and $|sp_{\iota_\nu} - sp_{\iota_{\nu-1}}| \le len_{win}$ for $\nu = \nu_0$, then, $Miner_{DE}$ computes

$$\hat{C}_{i_j} = \hat{H}_{i_j} \| \hat{q}_{i_j} \| \widehat{mod}_{i_j} = \bigoplus_{\nu=1}^{\nu_0} p_{i_j}(\theta_\nu) \tag{17}$$

- $Miner_{DE}$ maintains a table $\mathcal{T}_{rule}$ as follows. For each id_{i_j}, if (1) $\widehat{mod}_{i_j}$ is a valid modifier and the starting position sp_{ι_1} and the ending position $sp_{\iota_{\nu_0}} + len_{win}$ conform to this modifier, and (2) $\hat{H}_{i_j} = H(id_{i_j}, \hat{q}_{i_j}, \widehat{mod}_{i_j})$, then rid_{R_i}, $j \| b_{i_j}$, $\hat{H}_{i_j}$, $\hat{q}_{i_j}$, and $\widehat{mod}_{i_j}$ are stored in table $\mathcal{T}_{rule}$.
- $Miner_{DE}$ maintains a table $\mathcal{T}_{action}$ as follows. For each rule rid_{R_i}, if each element in $\{1\|0, 2\|0, \cdots, j_1 - 1\|0, j_1\|1\}$ is included in $\mathcal{T}_{rule}$, then $Miner_{DE}$ restores $action_i$ by computing

$$action_i = q_{i_1} \oplus q_{i_2} \oplus \cdots \oplus q_{i_{j_1}} \tag{18}$$

  Finally, rid_{R_i} and $action_i$ are stored in table $\mathcal{T}_{action}$ for $Miner_{DE}$ to record the matching rules and actions.

If $\mathcal{T}_{action}$ is empty after completing the detection, it means that the content of the smart contract is legal, and Inf_{TR} will be put on the blockchain for further data mining; otherwise, it is considered that the smart contract contains malicious information, and $Miner_{DE}$ directly discards Inf_{TR} .

After receiving Inf_{TR}, DC can obtain the smart contract M_{sc} from $E_{AES}(M_{sc})$ with key K_{AES}. Then DC checks the consistency of the smart contract M_{sc} and the randomized tokens $\mathcal{T}$. Finally, DC performs AI algorithm in smart contract M_{sc} over it's data set to get the final results, and return the results to DET.

VI. Security Analysis

In this section, a novel security model will be introduced, and then the security theorem and its proof will be presented.

A. Security Model

Definition 3: A privacy-preserving anomaly detection scheme $\Pi = (Setup, ObMapGen, RaTokenGen, Match)$ over message space $\mathcal{M}$ consists of a tuple of p.p.t. algorithms $Setup$, $ObMapGen$, $RaTokenGen$ and $Match$ as follows:

- $Setup(1^\lambda, KeyGen, KeyDist)$*:* Let λ be a security parameter. Algorithm $KeyGen$ outputs a key $k = (K_{CS}, K_{RP})$; a key distribution protocol $KeyDist$ can distribute K_{RP} to rule processor, and K_{CS} to sender and receiver, securely.
- $ObMapGen(K_{RP}, \mathcal{R})$*:* Algorithm $ObMapGen$ outputs an obfuscated map Obm by inputting the keys K_{RP} and a rule $\mathcal{R}$.
- $RaTokenGen(k, W_1, W_2, \cdots, W_\ell)$*:* Algorithm $RaTokenGen$ outputs a set of randomized tokens $\{T_1, T_2, \cdots, T_\ell\}$ by inputting the key k and a set of ℓ original tokens $\{W_1, W_2, \cdots, W_\ell\}$ for message $M \in \mathcal{M}$.
- $Match(Obm, T_1, T_2, \cdots, T_\ell)$*:* Algorithm $Match$ outputs a set of $Ture/Flase$ values $\{V_1, V_2, \cdots, V_\ell\}$ by inputting the obfuscated map Obm and the set of randomized tokens $T_1, T_2, \cdots, T_\ell$, where $Ture$ (i.e., 1) indicates that it matches successfully; $Flase$ (i.e., 0) means no match was found.

*Definition 4 (**Completeness**):* Scheme Π is completeness if for all efficient adversary $\mathcal{A}dv$, the following two equations

$$Adv^{fn-cor}_{\Pi,\mathcal{A}dv}(\lambda) = \Pr[FNCor^{\mathcal{A}dv}_{\Pi}(\lambda) = 1] \tag{19}$$

$$Adv^{fp-cor}_{\Pi,\mathcal{A}dv}(\lambda) = \Pr[FPCor^{\mathcal{A}}_{\Pi}(\lambda) = 1] \tag{20}$$

are negligible, where $FNCor^{\mathcal{A}dv}_{\Pi}(\lambda)$ and $FPCor^{\mathcal{A}dv}_{\Pi}(\lambda)$ are Experiment 1 and Experiment 2 respectively, as follows.

Experiment 1: $FNCor^{\mathcal{A}dv}_{\Pi}(\lambda)$

$k = (K_{CS}, K_{RP}) \leftarrow KeyGen(1^\lambda)$
$\mathcal{R} = \{R_1, \cdots, R_{|\mathcal{R}|}\} \leftarrow \mathcal{A}av(1^\lambda)$
$Obm \leftarrow ObMapGen(K_{RP}, \mathcal{R})$
$\mathcal{W} = \{W_1, W_2, \cdots, W_\ell\} \leftarrow \mathcal{A}dv(1^\lambda)$
$\mathcal{T} = \{T_1, T_2, \cdots, T_\ell\} \leftarrow RaTokenGen(K_{CS}, \mathcal{W})$
$Result \leftarrow Match(Obm, \mathcal{T})$
if $R_{i_0} = W_{j_0}$ (for some i_0 and j_0) and $Result = 0$, then $b = 1$
return b

*Definition 5 (**$\mathcal{L}$-secure Against Adaptive Attacks**):* An encrypted data malicious detection scheme scheme Π is $\mathcal{L}$-secure against adaptive attacks if, for all efficient adversary $\mathcal{A}$, there exists an efficient simulator $\mathcal{S}$, such that

$$Adv^{adap}_{\Pi,\mathcal{A},\mathcal{S}}(\lambda) = |\Pr[Real^{\mathcal{A}}_{\Pi}(\lambda) = 1] - \Pr[Ideal^{\mathcal{A}}_{\mathcal{L},\mathcal{S}}(\lambda) = 1]| \tag{21}$$

is negligible, where the games $Real$ and $Ideal$ are Game 1 and Game 2 respectively, as follows.

Experiment 2: $FPCor_{\Pi}^{\mathcal{A}dv}(\lambda)$

$k=(K_{CS},K_{RP}) \leftarrow KeyGen(1^{\lambda})$
$\mathcal{R}=\{R_1,\cdots,R_{|\mathcal{R}|}\} \leftarrow \mathcal{A}dv(1^{\lambda})$
$Obm \leftarrow ObMapGen(K_{RP},\mathcal{R})$
$\mathcal{W}=\{W_1,W_2,\cdots,W_{\ell}\} \leftarrow \mathcal{A}dv(1^{\lambda})$
$\mathcal{T}=\{T_1,T_2,\cdots,T_{\ell}\} \leftarrow RaTokenGen(K_{CS},\mathcal{W})$
$Result \leftarrow Match(Obm,\mathcal{T})$
if $R_{i_0} \neq W_{j_0}$ (for some i_0 and j_0) and $Result=1$, then $b=1$
return b

Game 1: $Real_{\Pi}^{\mathcal{A}}(\lambda)$

$k=(K_{CS},K_{RP}) \leftarrow KeyGen(1^{\lambda})$
$\mathcal{R}=\{R_1,\cdots,R_{|\mathcal{R}|}\} \leftarrow \mathcal{A}(1^{\lambda})$
$Obm \leftarrow ObMapGen(K_{RP},\mathcal{R})$
$\mathcal{W}=\{W_1,W_2,\cdots,W_{\ell}\} \leftarrow \mathcal{A}(1^{\lambda})$
$\mathcal{T}=\{T_1,T_2,\cdots,T_{\ell}\} \leftarrow RaTokenGen(K_{CS},\mathcal{W})$
$b \leftarrow \mathcal{A}(Obm,\mathcal{T})$

In $Real_{\Pi}^{\mathcal{A}}(\lambda)$, the challenger calls $KeyGen(1^{\lambda})$ to output $k=\{K_{CS},K_{RP}\}$. The adversary $\mathcal{A}$ selects a rule R for the challenger to create an obfuscated map Obm via $ObMapGen(K_{RP},R)$. Then $\mathcal{A}$ adaptively transmits a polynomial number of strings, which is extracted from the data packet, i.e., $\{W_1,W_2,\cdots,W_{\ell}\}$. After that, the challenger responds to $\mathcal{A}$ with the corresponding tokens $T_1,T_2,\cdots,T_{\ell}$. Finally, $\mathcal{A}$ outputs a decision bit b with inputting the tokens and Obm.

Game 2: $Ideal_{\mathcal{L},\mathcal{S}}^{\mathcal{A}}(\lambda)$

$\mathcal{R}=\{R_1,\cdots,R_{|\mathcal{R}|}\} \leftarrow \mathcal{A}(1^{\lambda})$
$\widetilde{Obm} \leftarrow \mathcal{S}(\mathcal{L}(\mathcal{R}))$
$\mathcal{W}=\{W_1,W_2,\cdots,W_{\ell}\} \leftarrow \mathcal{A}(1^{\lambda})$
$\mathcal{T}=\{T_1,T_2,\cdots,T_{\ell}\} \leftarrow RaTokenGen(K_{CS},\mathcal{W})$
$b \leftarrow \mathcal{A}(\widetilde{Obm},\mathcal{T})$

In $Ideal_{\mathcal{L},\mathcal{S}}^{\mathcal{A}}(\lambda)$, $\mathcal{A}$ selects a rule set $\mathcal{R}$, and a simulator $\mathcal{S}$ generates $\widetilde{Obm}$ based on leakage information $\mathcal{L}(\mathcal{R})$. Then $\mathcal{A}$ adaptively transmits a polynomial number of strings, which is extracted from the data packet, i.e., $\{W_1,W_2,\cdots,W_{\ell}\}$. After that, $\mathcal{A}$ obtains $T_1,T_2,\cdots,T_{\ell}$ from an oracle, which runs $RaTokenGen$ with inputting $\{W_1,W_2,\cdots,W_{\ell}\}$ and K_{CS}. Finally, $\mathcal{A}$ outputs a bit b. The $\mathcal{L}$ mentioned above is a state leakage function which describes what information is to be leaked from the inputs.

B. Security Theorem

In our security analysis, we assume that the symmetric encryption algorithm (E_K,D_K) and the public key encryption algorithm (Enc_{PK}, Dec_{SP}) are secure, which can be considered as psedudorandom permutations.

Here, we simplify our scheme Π_0 as follows. Assumed that $F:\mathcal{K}_s\times\{0,1\}^{l_1}\rightarrow\{0,1\}^{l_2}$, $G:\mathcal{K}_l\times\{0,1\}^{l_2}\rightarrow\{0,1\}^{l_3}$ and $f:\mathcal{K}_h\times\{0,1\}^{l_4}\rightarrow\{0,1\}^{l_5}$ are three pseudorandom functions; and $H:\{0,1\}^{l_6}\rightarrow\{0,1\}^{l_7}$ is a hash function;

- $Setup(1^{\lambda},KeyGen,KeyDist)$: it generates a secure key tuple $k=(K_s,K_h,K_l)$ between TR and DC, and sends key k to $Miner_{RP}$ securely.
- $ObMpGen(k,\mathcal{R})$: it calculates obfuscated map Obm with key k and the set of rules $\mathcal{R}$. The form of each map entry is $Obm[loc]=(id\|p)\oplus s$, where id is the random ID of each rule, p is the secret share used to restore the content options for a single rule con, $s=f(K_h,seg)$, $loc=G(K_l,l)$ is the input of obfuscated map, and $l=F(K_s,seg)$. Here, $con=H(id,q,Mod)\|q\|Mod$, where Mod is the modifier collection of a single content option and q is the secret share used to restore rule header $action$ for subsequent operation.
- $TokenGen(k,W_1,\cdots,W_{\ell})$: it generates a set of randomized tokens $\{T_1,\cdots,T_{\ell}\}$, where $T_i=F(K_s,W_i)\|f(K_h,W_i)\|sp_i$, where sp_i is the starting position of W_i.
- $Match(T_1,\cdots,T_{\ell},Obm)$: outputs a set of Ture/Flase values: $\{Ind_1,\cdots,Ind_{\ell}\}$, where $Ind_j=Ture$ if an $action$ is restored; otherwise, $Ind_j=Flase$.

Remark. In the subsequent of this paper, each rule in $\mathcal{R}$ contains only one content option and one response operation, and the length of the content option is the same as that of the plaintext token. Moreover, the length of response operation of all rules in $\mathcal{R}$ is equal.

Theorem 1: The proposed Scheme Π_0 is completeness if F, G and f are secure PRF and the hash function H utilized in the proposed scheme is *collision-resistant*.

Proof: $FNCor_{\Pi_0}^{\mathcal{A}dv}(\lambda)$ and $FPCor_{\Pi_0}^{\mathcal{A}dv}(\lambda)$ are shown in Experiment 3. It is clear that

$$Adv_{\Pi_0,\mathcal{A}}^{fn-cor}(\lambda)=\Pr[FNCor_{\Pi_0}^{\mathcal{A}}(\lambda)=1]=0 \tag{22}$$

Experiment 3: $FNCor_{\Pi_0}^{\mathcal{A}dv}(\lambda)$ (resp. $FPCor_{\Pi_0}^{\mathcal{A}dv}(\lambda)$)

$k=(K_{CS},K_{RP}) \leftarrow KeyGen(1^{\lambda})$
$\mathcal{R} \leftarrow \mathcal{A}dv(1^{\lambda})$
$Obm \leftarrow \emptyset$
for $i\in\{1,\cdots,|\mathcal{R}|\}$ do
  $id \leftarrow idGen(R_i)$
  $C_i=\{con_i:mod_i;\ action_i\} \leftarrow ruleSplit(R_i)$
  $\mathcal{S}_{eg}=\{seg_{i1},\cdots,seg_{i\sigma_i}:mod_i;action_i\} \leftarrow contentSplit(C_i)$
    for $\theta\in\{1,\cdots,\sigma_i\}$ do
      $s \leftarrow f(K_h,seg_{\theta})$
      $mapContent \leftarrow (id\|p)\oplus s$
      $l \leftarrow F(K_s,seg_{\theta})$
      $loc \leftarrow G(K_l,l)$
      $Obm \leftarrow Obm\cup(loc,mapContent)$
$\mathcal{W}=\{W_1,W_2,\cdots,W_{\ell}\} \leftarrow \mathcal{A}dv(1^{\lambda})$
for $\iota\in\{1,2,\cdots,\ell\}$ do $T_{\iota} \leftarrow F(K_s,W_{\iota})\|f(K_h,W_{\iota})\|sp_{\iota}$
$Result \leftarrow Match(Obm,\mathcal{T})$, where $\mathcal{T}=\{T_1,\cdots,T_{\iota}\}$
if $R_{i_0}=W_{j_0}$ (for some i_0 and j_0) and $Result=0$, then $b=1$
(if $R_{i_0}\neq W_{j_0}$ (for some i_0 and j_0) and $Result=1$, then $b=1$)
return b

We define (1) BAD_F be the event that there is a W_{ι_0} and seg_{θ_0} ($\theta_0\in\{1,\cdots,\sigma_{i_0}\}$ for some i_0 in $\{1,\cdots,|\mathcal{R}|\}$) with $W_{\iota_0}\neq seg_{\theta_0}$ such that $F(K_s,W_{i_0})=F(K_s,seg_{\theta_0})$; (2) BAD_f be the event that there is a W_{ι_0} and seg_{θ_0} ($\theta_0\in\{1,\cdots,\sigma_{i_0}\}$ for some i_0 in $\{1,\cdots,|\mathcal{R}|\}$) with $W_{\iota_0}\neq seg_{\theta_0}$ such that $f(K_h,W_{i_0})=$

$f(K_h, seg_{\theta_0})$; (3) BAD_G be the event that there is a W_{ι_0} and seg_{θ_0} ($\theta_0 \in \{1,\cdots,\sigma_{i_0}\}$ for some i_0 in $\{1,\cdots,|\mathcal{R}|\}$) with $W_{\iota_0} \neq seg_{\theta_0}$ such that $G(K_l, F(K_s, W_{i_0})) = G(K_l, F(K_s, seg_{\theta_0}))$; BAD_H be the event that $H(id_{i_0}, q', Mod_{i_0}) = H(id_{i_1}, q'', Mod_{i_1})$ with $(id_{i_0}, q', Mod_{i_0}) \neq (id_{i_1}, q'', Mod_{i_1})$ for some id_{i_0} id_{i_1}, q', q'', Mod_{i_0}, and Mod_{i_1} in obfuscated rule generation phase. Here, H can be regarded as a keying hash function, in where, the random ID id and random number q are used as a key.

$FPCor^{\mathcal{A}}_{\Pi}(\lambda) = 1$ happens only when one of the events BAD_F, BAD_f BAD_G, and BAD_H happens. Note that, adversary $\mathcal{A}$ does not know the keys (K_{CS}, K_{RP}) in Scheme Π_0. Therefore, from the fact that F, G and f are secure PRF and the hash function H used in the scheme is *collision-resistant*, we have the probability of BAD_F, BAD_f BAD_G, and BAD_H are negligible.

$$\begin{aligned} & Adv^{fp-cor}_{\Pi_0,\mathcal{A}dv}(\lambda) = \Pr[FPCor^{\mathcal{A}}_{\Pi_0}(\lambda) = 1] \\ \leq\ & \Pr[BAD_F] + \Pr[BAD_f] + \Pr[BAD_G] + \Pr[BAD_H] \\ \leq\ & negl(\lambda) \end{aligned} \tag{23}$$

■

Theorem 2: Scheme Π_0 is $\mathcal{L}$-secure against adaptive attacks if F, G and f are secure PRF.

Proof: We consider games $\mathbf{G}_0$, $\mathbf{G}_1$, and $\mathbf{G}_2$, in which, $\mathbf{G}_0$ is used to compute a distribution identical to $Real^{\mathcal{A}}_{\Pi_0}(\lambda)$, and $\mathbf{G}_2$ is used to compute a distribution identical to $Ideal^{\mathcal{A}}_{\mathcal{L},\mathcal{S}}(\lambda)$.

Game 4: $Input(\mathcal{R}, W_0, \cdots, W_\ell)//\mathbf{G}_0$

$k = (K_s, K_h, K_l) \leftarrow Setup(1^\lambda)$
$Obm \leftarrow \emptyset$
for $\iota \in \{1,2,\cdots,\ell\}$ do $T_\iota \leftarrow F(K_s, W_\iota) \| f(K_h, W_\iota) \| sp_\iota$
for $i \in \{1,\cdots,|\mathcal{R}|\}$ do
  $id \leftarrow idGen(R_i)$
  $\mathcal{C}_i = \{con_i : mod_i;\ action_i\} \leftarrow ruleSplit(R_i)$
  $\mathcal{S}_{eg} = \{seg_{i1}, \cdots, seg_{i\sigma_i} : mod_i; action_i\} \leftarrow contentSplit(C_i)$
    for $\theta \in \{1,\cdots,\sigma_i\}$ do
      $s \leftarrow f(K_h, seg_\theta)$
      $mapContent \leftarrow (id \| p) \oplus s$
      $l \leftarrow F(K_s, seg_\theta)$
      $loc \leftarrow G(K_l, l)$
      $Obm \leftarrow Obm \cup (loc, mapContent)$
return $b \leftarrow \mathcal{A}(Obm, T_1, \cdots, T_\ell)$

The first game $\mathbf{G}_0$ computes Obm and randomized tokens T_i as specified in the adaptive game. It selects a key set $k = (K_s, K_h, K_l)$ via $Setup(1^\lambda)$. For each W_i, $\mathbf{G}_0$ computes $F(K_s, W_i) \| f(K_h, W_i) \| sp_i$ as T_i. Moreover, it computes obfuscated map Obm using key set k for each original rule. Accordingly, for any p.p.t. adversary $\mathcal{A}$, we have

$$\Pr[G_0] = \Pr[Real^{\mathcal{A}}_{\Pi_0}(\lambda) = 1] \tag{24}$$

In Game $\mathbf{G}_1$, the outputs of tree random oracles $RF(\emptyset, \cdot)$, $RF'(\emptyset, \cdot)$ and $RF''(\emptyset, \cdot)$ are used to replace that of $F(K_s, \cdot)$, $f(K_h, \cdot)$ and $G(K_l, \cdot)$ in $\mathbf{G}_0$, respectively. This means that all of the T_i are uniform and independent strings.

Let Game G'_0 be the game which utilizes $RF(\emptyset, \cdot)$ to replace $F(K_s, \cdot)$ in Game G_0, and let Game G'_1 be the game which uses $RF'(\emptyset, \cdot)$ to replace $f(K_h, \cdot)$ in Game G'_0. Then, from that fact that F, f and G are pseudorandom functions, we can obtain

Game 5: $Input(\mathcal{R}, W_0, \cdots, W_\ell)//\mathbf{G}_1$

$k = (K_s, K_h, K_l) \leftarrow Setup(1^\lambda)$
$Obm \leftarrow \emptyset$
for $\iota \in \{1,2,\cdots,\ell\}$ do $T_\iota \leftarrow RF(\emptyset, W_\iota) \| RF'(\emptyset, W_\iota) \| sp_\iota$
for $i \in \{1,\cdots,|\mathcal{R}|\}$ do
  $id \leftarrow idGen(R_i)$
  $\mathcal{C}_i = \{con_i : mod_i;\ action_i\} \leftarrow ruleSplit(R_i)$
  $\mathcal{S}_{eg} = \{seg_{i1}, \cdots, seg_{i\sigma_i} : mod_i; action_i\} \leftarrow contentSplit(C_i)$
    for $\theta \in \{1,\cdots,\sigma_i\}$ do
      $s \leftarrow RF'(\emptyset, seg_\theta)$
      $mapContent \leftarrow (id \| p) \oplus s$
      $l \leftarrow RF(\emptyset, seg_\theta)$
      $loc \leftarrow RF''(\emptyset, l)$
      $Obm \leftarrow Obm \cup (loc, mapContent)$
return $b \leftarrow \mathcal{A}(Obm, T_1, \cdots, T_\ell)$

that

$$|\Pr[G_0] - \Pr[G'_0]| \leq negl(\lambda) \tag{25}$$

$$|\Pr[G'_0] - \Pr[G'_1]| \leq negl(\lambda) \tag{26}$$

$$|\Pr[G'_1] - \Pr[G_1]| \leq negl(\lambda) \tag{27}$$

for any efficient adversary $\mathcal{A}$. As a result, for any efficient adversary $\mathcal{A}$, we can obtain that

$$|\Pr[G_0] - \Pr[G_1]| \leq negl(\lambda) \tag{28}$$

Game 6: $Input(\mathcal{R}, W_0, \cdots, W_\ell)//\mathbf{G}_2$

$k = (K_s, K_h, K_l) \leftarrow Setup(1^\lambda)$
$Obm \leftarrow \emptyset$
for $\iota \in \{1,2,\cdots,\ell\}$ do $T_\iota \leftarrow F(K_s, W_\iota) \| f(K_h, W_\iota) \| sp_\iota$
for $i \in \{1,\cdots,|\mathcal{R}|\}$ do
  $id \leftarrow idGen(R_i)$
  $\mathcal{C}_i = \{con_i : mod_i;\ action_i\} \leftarrow ruleSplit(R_i)$
  $\mathcal{S}_{eg} = \{seg_{i1}, \cdots, seg_{i\sigma_i} : mod_i; action_i\} \leftarrow contentSplit(C_i)$
    for $\theta \in \{1,\cdots,\sigma_i\}$ do
      $s \leftarrow RF'(\emptyset, seg_\theta)$
      $mapContent \leftarrow (id \| p) \oplus s$
      $l \leftarrow RF(\emptyset, seg_\theta)$
      $loc \leftarrow RF''(\emptyset, l)$
      $Obm \leftarrow Obm \cup (loc, mapContent)$
return $b \leftarrow \mathcal{A}(Obm, T_1, \cdots, T_\ell)$

Note we consider Game G_2. In this game, random tokens $T_0, \cdots, T_\ell$ are generated by $F(K_s, \cdot)$ and $f(K_h, \cdot)$ with inputting $W_0, \cdots, W_\ell$.

Note that, if $F(K_s, \cdot)$ and $f(K_h, \cdot)$ are pseudo-random functions, then $\widehat{F}(K_s, K_h, \cdot)$ $(= (F(K_s, \cdot), f(K_h, \cdot)))$ is also a pseudo-random function. Then, we claim that,

$$|\Pr[G_1] - \Pr[G_2]| \leq negl(\lambda) \tag{29}$$

Actually, if there is an efficient adversary $\mathcal{A}_1$ such that

$$|\Pr[G_1] - \Pr[G_2]| > negl(\lambda) \tag{30}$$

then an efficient adversary $\mathcal{B}_1$ exists to satisfy

$$Adv^{pdf}_{\widehat{F},B_1}(\lambda) > negl(\lambda) \tag{31}$$

Here, the adversary $\mathcal{B}_1$ has access to an oracle $RF(\cdot, \cdot)$,

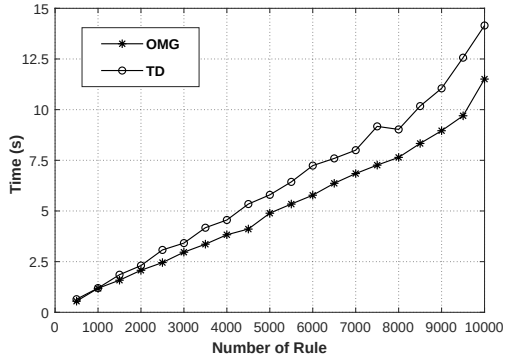


Fig. 3. Time consumption versus rule number.

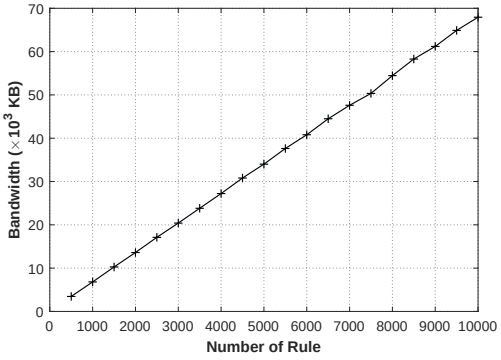


Fig. 4. Communication cost of obfuscated map versus rule number.

$RF'(\cdot,\cdot)$, and $RF''(\cdot,\cdot)$. $\mathcal{B}_1$ runs $\mathcal{A}_1$ to get $(\mathcal{R}, W_0, \cdots, W_\ell)$, and then uses output of $\mathcal{A}_1$ as its own.

An effective simulator $\mathcal{S}$ takes the information output by the leakage function $\mathcal{L}$ as the initial input. Specifically, $\mathcal{L}(\mathcal{R}) = \{|loc|, |mapContent|, |e|, \{|d|\}_{|e|}\}$, where $|loc|$ is the length of the input of the obfuscated map, $|mapContent|$ is the length of the value in the obfuscated map, $|e|$ is the number of entries in the obfuscated map, and $\{|d|\}_{|e|}$ is the size of the corresponding bucket for each entry. $\mathcal{S}$ simulates and constructs the obfuscated map $\widetilde{Obm}$ through the leakage function. Compared $\mathcal{S}$ with G_2, the keys and values in $\widetilde{Obm}$ are random independent strings so that

$$\Pr[G_2] = \Pr[Ideal^{\mathcal{A}}_{\mathcal{L},\mathcal{S}}(\lambda) = 1] \tag{32}$$

Accordingly, we have

$$\begin{aligned} Adv^{\text{adap}}_{\Pi_0,\mathcal{A},\mathcal{S}}(\lambda) &= |\Pr[Real^{\mathcal{A}}_{\Pi_0}(\lambda) = 1] - \Pr[Ideal^{\mathcal{A}}_{\mathcal{L},\mathcal{S}}(\lambda) = 1]| \\ &= |\Pr[G_0] - \Pr[G_2]| \\ &= |\Pr[G_0] - \Pr[G_1]| + |\Pr[G_1] - \Pr[G_2]| \\ &\leq negl(\lambda) \end{aligned} \tag{33}$$

Consequently, if F, f and G are secure PRFs, the scheme PESCI Π_0 is correct and $\mathcal{L}$-secure against adaptive attacks. ■

VII. Performance Evaluation

In this section, we conduct extensive experiments to discuss the performance of our scheme.

A. Experimental Settings

In our experiments, a desktop computer (AMD Ryzen 7 4800H) is used with 64-bit Linux operating system, Radeon Graphics 2.90 GHz and 16.0 GB of RAM. Moreover, all experiments are conducted with the libraries of Python. The pseudorandom functions F, G and f are implemented by SHA1, SHA224 and SHA256 respectively, while the hash function H is built on MD5. Both of them are built on the Python library *Crypto*. Moreover, the size of the function code of a smart contract is set to 5∼110 KB. The size of the sliding window is determined according to the length of the shortest "content" option in all selected rules. In our experiments, the size of the sliding window is set to 2 bytes, as the shortest length of "content" option of Snort rule set used in our experiments is 2 bytes.

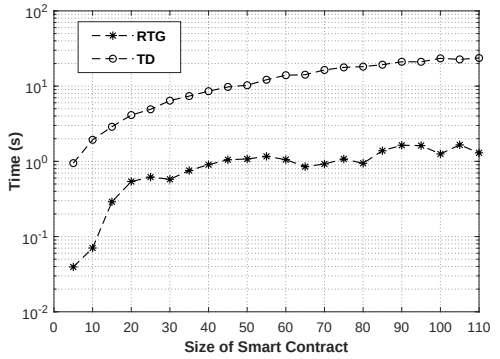


Fig. 5. Time consumption versus smart contract size.

B. Exploration of System Parameters

We now discuss the impact of system parameters on computation cost and communication cost of the proposed scheme. We first conduct an experiment to discuss the effect of the size of Snort rule set $|\mathcal{R}|$ and the size of smart contract file $|M|$ on the overhead of the proposed scheme. In our experiment, the size of rule set is chosen from 500 to 10000 with step 500, and the size of smart contract M is set as 15 KB. The time cost of the obfuscated map generation and token detection against the varied size of original rule set is first studied. As shown in Fig. 3, the time cost of both obfuscated map generation and token detection increases approximately linearly with the increase in the size of rule set. Then, the communication overhead of the proposed method versus the varied size of original rule set is discussed. From Fig. 4, it can be seen that, the communication overhead of the proposed scheme the bandwidth consumption of the generated obfuscated map increases linearly as the cardinality of rule set increases. Specifically, when the size of the smart contract is fixed at 15 KB, the time cost is around 112 ms and the bandwidth consumption is 5316 KB in the randomized token generation phase.

Then, the overhead of the randomized token generation and token detection under different size of smart contract (from 5 KB to 110 KB with step 5 KB) is discussed. Fig. 5 plots the time cost of two main phases under different size of smart contract. It can be seen that, the time cost is mainly concentrated on the token detection phase. The reason is that, with the increase of the size of smart contract file, the number of the tokens increases, which further leads to the

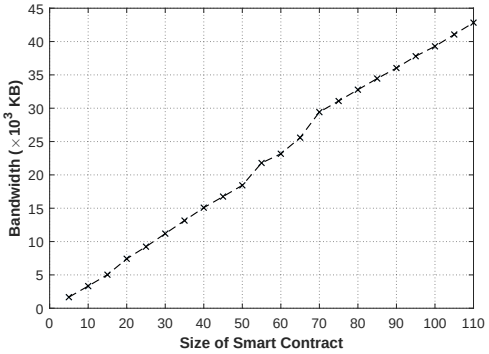


Fig. 6. Communication cost of token set versus smart contract size.

increase of time consumption of token detection. In Fig. 6, the communication overhead of two main phases under different size of smart contract is presented. From Fig. 6, we have that the communication overhead of randomized token generation increases with the increase of smart contract size linearly. Specifically, when the cardinality of the rule set is fixed at 3000, the time consumption of the obfuscated map generation phase is about 1927 ms and the bandwidth consumption is 14938 KB.

C. Performance Analysis

An experiment is carried out to discuss the overall performance of our scheme by taking the size of smart contracts from 5 KB to 110 KB with step 5 KB. In this experiment, four rule sets containing 4000 rules are selected. These sets have no intersection of rules and are all from suricata-5.0. It should be noted that the main difference between these rule sets is the length of the shortest "content" options, which are 2, 3, 4 and 5 bytes respectively. According to the shortest length from small to large, the four sets are defined as $ruleset_1$, $ruleset_2$, $ruleset_3$ and $ruleset_4$ respectively.

Fig. 7(a) shows the time cost of randomized token generation phase under different sizes of smart contract. From Fig. 7(a), we have the result that for four rule sets with different sliding windows (i.e. 2, 3, 4 and 5), the time consumption decreases with the increase of the size of sliding window. Meanwhile, for different rule sets, the time cost at this phase shows an upward trend as the size of the smart contract increases. Fig. 7(b) shows the time cost of obfuscated map generation phase. It can be seen that, the time cost of this phase raises with increasing the size of the sliding window or the size of smart contract. Fig. 7(c) shows the communication overhead of randomized token generation phase of the proposed scheme by changing the size of sliding window. As shown in this figure, with the increase of sliding window size, the communication overhead of randomized token generation phase decreases. The reason is that, for the larger sliding window, the fewer randomized tokens are generated, which reduces the bandwidth. Meanwhile, as the increase of smart contract size, the bandwidth corresponding to each experimental rule set in randomized token generation is increasing.

As shown in Table III, since the rules and sliding windows in the three rule sets are fixed, the time cost and communication overhead corresponding to each rule set in the obfuscated map generation phase is constant. Specifically, the communication cost corresponding to $ruleset_1$, $ruleset_2$, $ruleset_3$ and $ruleset_4$ is 8,488 KB, 14,623 KB, 16,042 KB and 20,125 KB, respectively.

TABLE III
THE TIME CONSUMPTION AND COMMUNICATION COST OF OBFUSCATED MAP GENERATION PHASE

Num of Rule Set	Time (ms)	Bandwidth (KB)
$ruleset_1$	1974	8,488
$ruleset_2$	2296	14,623
$ruleset_3$	2631	16,042
$ruleset_4$	3132	20,125

D. Comparison with Existing Schemes

We conduct an experiment to compare the time and communication overhead of our scheme with existing schemes (i.e., PrivDPI [33] and PETI [35]). In this experiment, the rule length *ruleLen* of the proposed scheme and PETI [35] are set to 8 bytes and 16 bytes, and the rule length *ruleLen* of PrivDPI [33] is set to 8 bytes, respectively; the number of rules is 6000 for $ruleLen = 8$ bytes, and 3000 for $ruleLen = 16$ bytes; and the size of the smart contract is set to 19.2KB. Since the ORG phase of PrivDPI in the subsequent session can reuse the obfuscated rules in the first session to reduce time and communication consumption [33], we consider two consecutive sessions in anomaly detection of encrypted smart contract to fairly compare the performance of the above schemes.

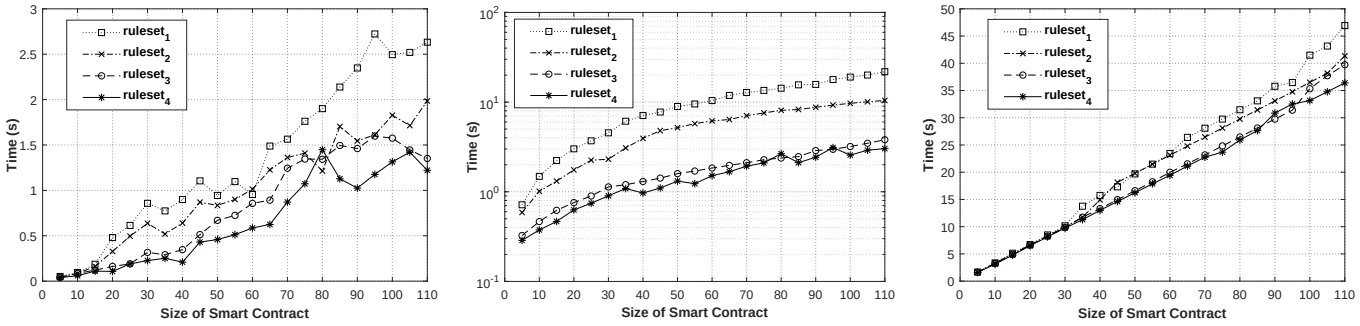
Fig. 8 shows the comparison of the time consumption of the proposed scheme with existing schemes in two consecutive sessions. It can be found that, (1) the computational overhead of our scheme is significantly smaller than that of PrivDPI and PETI in all three phases of two sessions; and (2)there is not much difference between the first and second execution of the proposed scheme, while the computational overhead of the ORG phase during the second session of PrivDPI is significantly reduced due to the fact that PrivDPI utilizes the results of the first session for simplicity of computation.

Fig. 9 plots the comparison of the communication cost of the proposed scheme with existing schemes in two consecutive sessions. It can be seen that, (1) in the ORG phase, the communication cost of the proposed scheme and PETI is almost the same, which are less than that of PrivDPI; (2) in the TD phase, there is no communication cost for all three schemes; and (3) the proposed scheme has a considerable advantage over privDPI and PETI in the RTG phase of the first session, and the communication overhead of PrivDPI in the RTG phase of the second session is extremely small due to the utilization of the results of the first session.

E. Discussion

In our system, it is assumed that the miners $Miner_{DE}$ and $Miner_{PR}$ are semi-honest. A feasible approach based on the admission and punishment mechanism to eliminate this assumption is as follows. Each miner participating in the

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3353827



(a) Time cost of randomized token generation

(b) Time cost of token detection

(c) Communication overhead of randomized token generation

Fig. 7. The time and communication cost of the proposed scheme versus varying size of smart contract in different rule set.

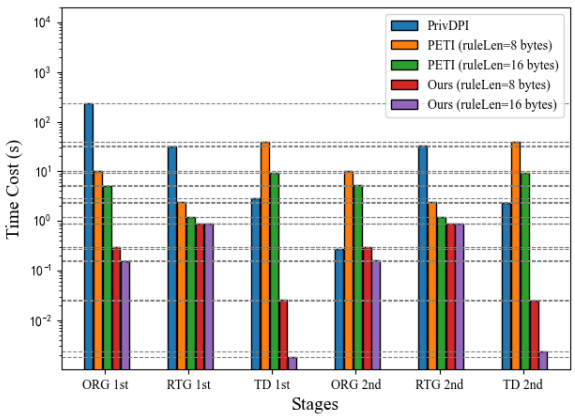


Fig. 8. The comparison of the time consumption of the proposed scheme with existing schemes.

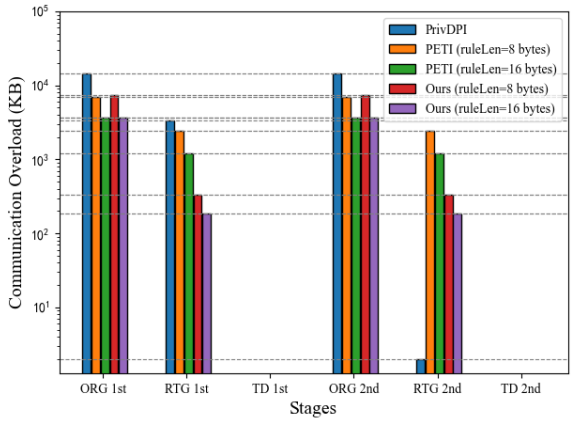


Fig. 9. The comparison of the communication cost of the proposed scheme with existing schemes.

task of *Obfuscated Rule Generation* and *Malicious Detection* needs to have a certain access mechanism and pay a certain amount of deposit for the penalty in case of dishonesty. When dishonest event occurs, data center first detects the honesty of $Miner_{PR}$. Note that, in the proposed protocol, $Miner_{PR}$ needs to process the matching rules into obfuscated map, and store it and the corresponding signature on the RP chain. If $Miner_{PR}$ is a dishonest node and generates a fake obfuscated map for a transaction, the data center can generate a real obfuscated map using the real rule set and key, and then compare it with the fake obfuscated map on the rule processing chain, so as to find out the evidence of dishonesty of $Miner_{PR}$. Then, data center verifies the honesty of $Miner_{DE}$. If node $Miner_{PR}$ is honest while $Miner_{DE}$ is dishonest in the detection process. The data center can extract the electronic contract and decrypt the transaction content; and then use the real rule set to detect the decrypted electronic contract. If the detection result is inconsistent with the system detection result, then $Miner_{DE}$ is dishonest. When a miner acts dishonestly, his qualification as a miner may be revoked or he may be fined depending on the severity of the behavior.

VIII. Conclusion

In this paper, a blockchain-based data trading platform has been considered, in which, data users can purchase data set and computing power through encrypted smart contracts. A privacy-preserving encrypted smart contract detection system has been proposed in the blockchain-based data trading platform with the help of two kinds of miners. One kind of miner acts as a rule processor to generate an obfuscated map with the original open-source malicious rule set; and another kind of miner acts as a detector to perform malicious inspection by inputting the obfuscated map and the randomized tokens of smart contract. We have is defined the security syntax of encrypted smart contract inspection, and proved that the proposed scheme is $\mathcal{L}$-secure against adaptive attacks. Experimental results demonstrate that our scheme can achieve malicious code detection with high detection accuracy, low time cost and low communication overhead. The assumption that all miners are semi-honest in the proposed scheme is relatively strong, which weakens the application in the practical peer-to-peer network environment. In the future, we would like to continue working on this topic and propose a privacy-preserving anomaly detection scheme for encrypted smart contracts under the dishonest miner model.

REFERENCES

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Business Review*, pp. 21260, 2008.

[2] Y. Liu, J. Liu, Q. Wu, H. Yu, Y. Hei and Z. Zhou, "SSHC: A Secure and Scalable Hybrid Consensus Protocol for Sharding Blockchains With a Formal Security Framework," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 3, pp. 2070-2088, 2022.

[3] M. Song, Z. Hua, Y. Zheng, H. Huang and X. Jia, "Blockchain-Based Deduplication and Integrity Auditing over Encrypted Cloud Storage," *IEEE Transactions on Dependable and Secure Computing*, to appear, 2023, doi: 10.1109/TDSC.2023.3237221.

[4] Y. Chen, J. Zhao, Y. Wu, *et al.*, "QoE-Aware Decentralized Task Offloading and Resource Allocation for End-Edge-Cloud Systems: A Game-Theoretical Approach," *IEEE Transactions on Mobile Computing*, to appear, 2022, doi: 10.1109/TMC.2022.3223119.

[5] N. Zhang, P. Yang, J. Ren, *et al.*, "Synergy of big data and 5g wireless networks: opportunities, approaches, and challenges," *IEEE Wireless Communications*, vol. 25, no. 1, pp. 12-18, 2018.

[6] W. Wu, C. Zhou, M. Li, *et al.*, "AI-native network slicing for 6G networks," *IEEE Wireless Communications*, vol. 29, no. 1, pp. 96-103, 2022.

[7] Y. Jiang, Y. Zhong and X. Ge, "IIoT Data Sharing Based on Blockchain: A Multileader Multifollower Stackelberg Game Approach," *IEEE Internet of Things Journal*, vol. 9, no. 6, pp. 4396-4410, 2022.

[8] M. Kumar, Kavita, S. Verma, *et al.*, "ANAF-IoMT: A Novel Architectural Framework for IoMT-Enabled Smart Healthcare System by Enhancing Security Based on RECC-VC," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 12, pp. 8936-8943, 2022.

[9] X. Li, H. Liu, W. Wang, *et al.*, "Big data analysis of the internet of things in the digital twins of smart city based on deep learning," *Future Generation Computer Systems*, vol. 128, pp. 167-177, 2022.

[10] W. Dai, C. Dai, K. K. R. Choo, *et al.*, "SDTE: A secure blockchain-based data trading ecosystem," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 725-737, 2019.

[11] D. Chen, S. Jiang, N. Zhang, *et al.*, "On Message Authentication Channel Capacity Over a Wiretap Channel," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 3107-3122, 2022.

[12] N. Zhang, N. Lu, N. Cheng, J. W. Mark, X. S. Shen, "Cooperative spectrum access towards secure information transfer for crns," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 11, pp. 2453-2464, 2013.

[13] S. Sedkaoui, M. Khelfaoui, "Sharing economy and big data analytics," *John Wiley & Son*s, 2020.

[14] X. Yang, R. Lu, J. Shao, X. Tang and A. A. Ghorbani, "Achieving Efficient Secure Deduplication With User-Defined Access Control in Cloud," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 1, pp. 591-606, 2022.

[15] Y. Li, L. Li, Y. Zhao, *et al.*, "Toward decentralized fair data trading based on blockchain," *IEEE Network*, vol. 35, no. 1, pp. 304-310, 2020.

[16] L. D. Nguyen, I. Leyva-Mayorga, A. N. Lewis and P. Popovski, "Modeling and Analysis of Data Trading on Blockchain-Based Market in IoT Networks," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6487-6497, 2021.

[17] D. Zhang, J. Le, X. Lei, T. Xiang and X. Liao, "Secure Redactable Blockchain With Dynamic Support," *IEEE Transactions on Dependable and Secure Computing*, to appear, 2023, doi: 10.1109/TDSC.2023.3261343.

[18] J. Zhang, Y. Ye, W. Wu and X. Luo, "Boros: Secure and Efficient Off-Blockchain Transactions via Payment Channel Hub," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 1, pp. 407-421, 2023.

[19] C. Li, S. Y. Liang, J. Zhang, *et al.*, "Blockchain-based data trading in edge-cloud computing environment," *Information Processing & Management*, vol. 59, no. 1, pp. 1-22, 2022.

[20] L. D. Nguyen, I. Leyva-Mayorga, A. N. Lewis, *et al.*, "Modeling and analysis of data trading on blockchain-based market in IoT networks," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6487-6497, 2021.

[21] B. An, M. Xiao, A. Liu, *et al.*, "Secure Crowdsensed Data Trading Based on Blockchain," *IEEE Transactions on Mobile Computing*, 2021.

[22] N. Szabo, "Formalizing and securing relationships on public networks," *First monday*, 1997.

[23] L. Ale, N. Zhang, H. Wu, *et al.*, Online Proactive Caching in Mobile Edge Computing Using Bidirectional Deep Recurrent Neural Network, *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5520-5530, 2019.

[24] D. Chen, Z. Zhao, X. Qin, *et al.*, "MAGLeak: A Learning-Based Side-Channel Attack for Password Recognition With Multiple Sensors in IIoT Environment," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 1, pp. 467-476, Jan. 2022.

[25] X. Liu, Y. Zheng, X. Yuan and X. Yi, "Securely Outsourcing Neural Network Inference to the Cloud With Lightweight Techniques," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 1, pp. 620-636, 2023.

[26] A. Kosba, A. Miller, E. Shi, *et al.*, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," *in Proc. of 2016 IEEE symposium on security and privacy (S&P)*, 2016, pp. 839-858.

[27] S. Steffen, B. Bichsel, M. Gersbach, N. Melchior, P. Tsankov, and M. Vechev, "zkay: Specifying and enforcing data privacy in smart contracts," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 1759-1776.

[28] P. Pal, and K. Sudharsana, "Wip: Criminal smart contract for private key theft in end to end encrypted applications," *in Proc. of International Conference on Information Systems Security*, Springer, 2019, pp. 21-32.

[29] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1-32, 2014.

[30] Y. Wu, S. Tang, B. Zhao, and Z. Peng, "Bptm: Blockchain-based privacy-preserving task matching in crowdsourcing," *IEEE ACCESS*, vol. 7, pp. 45605-45617, 2019.

[31] J. Shi, Y. Zhang, and S. Zhong, "Privacy-preserving network functionality outsourcing," *arXiv preprint*, arXiv:1502.00389, 2015.

[32] J. Sherry, C. Lan, R. A. Popa, and S. Ratnasamy, "BlindBox: Deep Packet Inspection over Encrypted Traffic," *in Proc. of SIGCOMM 2015*, London, United Kingdom, pp.213-226, 2015.

[33] J. Ning, G. S. Poh, J. Loh, J. Chia, and E. Chang, "PrivDPI: Privacy-Preserving Encrypted Traffic Inspection with Reusable Obfuscated Rules," *in Proc. of ACM CCS2019*, London, United Kingdom, 2019, pp. 1657-1670.

[34] X. Yuan, X. Wang, J. Lin, and C. Wang, "Privacy-preserving deep packet inspection in outsourced middleboxes," *in Proc. of IEEE INFOCOM 2016*, 2016, pp. 1-9, .

[35] D. Chen, H. Wang, N. Zhang, *et al.*, "Privacy-Preserving Encrypted Traffic Inspection with Symmetric Cryptographic Techniques in IoT," *IEEE Internet of Things Journal,* vol. 9, no. 18, pp. 17265-17279, 2022.

[36] A. C. Yao, "How to Generate and Exchange Secrets," *in Proc. of IEEE FOCS1986*, 1986.

[37] M. Naor and B. Pinkas, "Oblivious Transfer with Adaptive Queries," *in Proc. of IACR CRYPTO1999*, 1999.

[38] D. Yuan, Q. Li, G. Li, *et al.*, "PriRadar: A privacy-preserving framework for spatial crowdsourcing," *IEEE transactions on information forensics and security*, vol. 15, pp. 299-314, 2019.

[39] M. Deng, K. Zhang, P. Wu, M. Wen and J. Ning, "DCDPI: Dynamic and Continuous Deep Packet Inspection in Secure Outsourced Middleboxes," *IEEE Transactions on Cloud Computing*, to appear, doi: 10.1109/TC-C.2023.3293134.

[40] , S. Gao, J. Chen, B. Zhang, *et al.* "Privacy-Preserving Industrial Control System Anomaly Detection Platform," *Security and Communication Networks*, 2023.

[41] C. Zhang, M. Zhao, L. Zhu, W. Zhang, T. Wu and J. Ni, "FRUIT: A Blockchain-Based Efficient and Privacy-Preserving Quality-Aware Incentive Scheme," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 12, pp. 3343-3357, 2022.

[42] C. Sendner, H. Chen, H. Fereidooni, *et al.* "Smarter Contracts: Detecting Vulnerabilities in Smart Contracts with Deep Transfer Learning," *in Proc. of IEEE NDSS2023*, 2023.

[43] N. Ivanov, Q. Yan, A. Kompalli, "TxT: Real-time Transaction Encapsulation for Ethereum Smart Contracts," *IEEE Transactions on Information Forensics and Security*, vol.18, pp. 1141-1155, 2023.

[44] D. X. Song, D. Wagner, A. Perrig, "Practical techniques for searches on encrypted data," *in Proc. of IEEE S&P 2000*, Oakland, California, USA, 2000, pp. 44-55.

[45] R. Curtmola, J. Garay, S. Kamara, *et al.*, "Searchable symmetric encryption: improved definitions and efficient constructions," *Journal of Computer Security*, vol. 19, no. 5, pp. 895-934, 2011.

[46] D. Cash, J. Jaeger, S. Jarecki, *et al.*, "Dynamic searchable encryption in very-large databases: Data structures and implementation," *in Proc. of NDSS 2014*, San Diego, CA, USA, 2014, pp. 1-16.

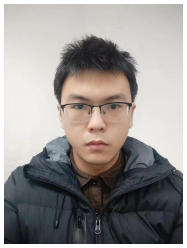
[47] A. Joux, "A one round protocol for tripartite Diffie-Hellman," *Journal of cryptology*, vol. 17, no.4, pp. 263-276, 2004.

[48] C. Lan, J. Sherry, R. A. Popa, *et al.*, "Embark: Securely outsourcing middleboxes to the cloud," *in proc. of 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, 2016, pp. 255-273.

[49] S. Canard, A. Diop, N. Kheir, *et al.*, "BlindIDS: Market-compliant and privacy-friendly intrusion detection system over encrypted traffic," *in proc. of the 2017 ACM on Asia Conference on Computer and Communications Security*, pp. 561-574, 2017.

[50] M. Bellare, and P. Rogaway, "The exact security of digital signatures-How to sign with RSA and Rabin," *in Proc. of International conference on the theory and applications of cryptographic techniques*, Springer, Berlin, Heidelberg, 1996, pp. 399-416.

Dajiang Chen (M'15) is currently an Associate Professor in the School of Information and Software Engineering at University of Electronic Science and Technology of China (UESTC). He was a postdoc research fellow with the BBCR group, Department of Electrical and Computer Engineering, University of Waterloo, Canada, from 2015 to 2017. He received the Ph.D. degree in information and communication engineering from UESTC in 2014. Dr. Chen served as the workshop chair for BDEC-SmartCity'19 (in conjunction with IEEE WiMob 2019). He also served as a Technical Program Committee Member for IEEE Globecom, IEEE ICC, and IEEE VTC. His current research interests include Physical Layer Security, Secure Channel Coding, and Machine Learning and its applications in Wireless Network Security and Wireless Communications.



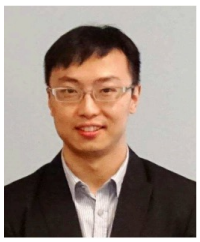
Zeyu Liao is currently a postgraduate student in the School of Information and Software Engineering, University of Electronic Science and Technology of China. He received the B.S. degree in Software Engineering in the School of information and software Engineering at University of Electronic Science and Technology of China, in 2021. His research interests include Security and Privacy Protection in Wireless Networks.

Ruidong Chen is currently an Associate Research Fellow in the School of Computer Science and Engineering, University of Electronic Science and Technology of China. He received the Ph.D. degree in information and communication engineering from University of Electronic Science and Technology of China in 2019. His research interests include Blockchain, and Security and Privacy Protection in Information Systems, Software, Networking, and Databases.

Hao Wang is currently a postgraduate student in the School of Information and Software Engineering, University of Electronic Science and Technology of China. He received the B.S. degree in Software Engineering in the School of information and software Engineering at University of Electronic Science and Technology of China, in 2019. His research interests include Security and Privacy Protection in different application scenarios of Wireless Networks (e.g., IoT, Edge Computing, and Internet of Vehicles).

Chong Yu (Graduate Student Member, IEEE) received the B.Sc. degree in communication engineering and the M.Sc. degree in communication and information system from Northeastern University, Shenyang, China, in 2015 and 2017, respectively. She is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Nebraska-Lincoln, Omaha, NE, USA. Her research interests include intelligent Internet of Things, cybersecurity, intelligent vehicle, cloud/edge computing, and machine learning.

Kuan Zhang (S'13-M'16) has been an assistant professor in the Department of ECE, University of Nebraska-Lincoln, since September 2017. He received his B.Sc. degree in communication engineering and his M.Sc. degree in computer applied technology from Northeastern University, China, in 2009 and 2011, respectively. He received his Ph.D. degree in ECE from the University of Waterloo in 2016. He was also a postdoctoral fellow with the Broadband Communications Research (BBCR) group, University of Waterloo from 2016 to 2017. His research interests include security and privacy for mobile social networks, cloud/edge computing, and cyber physical systems.



Ning Zhang (M'15-SM'18) is an Associate Professor in the Department of Electrical and Computer Engineering at University of Windsor, Canada. He received the Ph.D degree in Electrical and Computer Engineering from University of Waterloo, Canada, in 2015. After that, he was a postdoc research fellow at University of Waterloo and University of Toronto, Canada, respectively. His research interests include connected vehicles, mobile edge computing, wireless networking, and machine learning. He is a Highly Cited Researcher (Web of Science). He received an NSERC PDF award in 2015 and 6 Best Paper Awards from IEEE Globecom in 2014, IEEE WCSP in 2015, IEEE ICC in 2019, IEEE ICCC in 2019, IEEE Technical Committee on Transmission Access and Optical Systems in 2019, and Journal of Communications and Information Networks in 2018, respectively. He serves as an Associate Editor of IEEE Internet of Things Journal, IEEE Transactions on Cognitive Communications and Networking, and IEEE Systems Journal; and a Guest Editor of several international journals, such as IEEE Wireless Communications, IEEE Transactions on Industrial Informatics, and IEEE Transactions on Cognitive Communications and Networking.

Dr. Xuemin (Sherman) Shen (M'97-SM'02-F'09) received the B.Sc.(1982) degree from Dalian Maritime University (China) and the M.Sc. (1987) and Ph.D. degrees (1990) from Rutgers University, New Jersey (USA), all in electrical engineering. He is a University Professor and the Associate Chair for Graduate Studies, Department of Electrical and Computer Engineering, University of Waterloo, Canada. Dr. Shen's research focuses on wireless resource management, wireless network security, social networks, smart grid, and vehicular ad hoc and sensor networks. He is the elected IEEE ComSoc VP Publication, was a member of IEEE ComSoc Board of Governor, and the Chair of Distinguished Lecturers Selection Committee. Dr. Shen served as the Technical Program Committee Chair/Co-Chair for IEEE Globecom'16, Infocom'14, IEEE VTC'10 Fall, and Globecom'07, *etc*. He also serves/served as the Editor-in-Chief for IEEE IoT-J, IEEE Network, Peer-to-Peer Networking and Application, and IET Communications; a Founding Area Editor for IEEE Transactions on Wireless Communications; and an Associate Editor for IEEE Transactions on Vehicular Technology and IEEE Wireless Communications, etc. Dr. Shen received the IEEE ComSoc Education Award, the Joseph LoCicero Award for Exemplary Service to Publications, the Excellent Graduate Supervision Award in 2006, and the Premier's Research Excellence Award (PREA) in 2003 from the Province of Ontario, Canada. Dr. Shen is a registered Professional Engineer of Ontario, Canada, an IEEE Fellow, an Engineering Institute of Canada Fellow, a Canadian Academy of Engineering Fellow, a Royal Society of Canada Fellow, and a Distinguished Lecturer of IEEE Vehicular Technology Society and Communications Society.