Hindawi
Security and Communication Networks
Volume 2020, Article ID 8819958, 22 pages
https://doi.org/10.1155/2020/8819958

Research Article

Modelling the Mimic Defence Technology for Multimedia Cloud Servers

Feng Feng,[1] **Xiabing Zhou**,[2] **Bin Li**,[1] **and Qinglei Zhou**[1]

[1]*School of Information Engineering, Zhengzhou University, Zhengzhou 450001, China*
[2]*School of Computer Science and Technology, Soochow University, Suzhou 215006, China*

Correspondence should be addressed to Xiabing Zhou; zhouxiabing@suda.edu.cn and Bin Li; iebinli@zzu.edu.cn

Received 15 August 2020; Revised 27 September 2020; Accepted 26 October 2020; Published 28 November 2020

Academic Editor: Zhihua Xia

A current research trend is to combine multimedia data with artificial intelligence and process them on cloud servers. In this context, ensuring the security of multimedia cloud servers is critical, and the cyber mimic defence (CMD) technology is a promising approach to this end. CMD, which is an innovative active defence technology developed in China, can be applied in many scenarios. However, although the mathematical model is a key component of CMD, a universally acceptable mathematical model for theoretical CMD has not been established yet. In this work, the attack problems and modelling difficulties were extensively examined, and a comprehensive modelling theory and concepts were clarified. By decoupling the model from the input and output of the specific system scene, the modelling difficulties were effectively avoided, and the mathematical expression of the CMD mechanism was enhanced. Furthermore, the process characteristics of the attack behaviour were identified by using a specific mathematical mapping method. Finally, based on the decomposition problem of large prime factors and convolution operations, an intuitive and exclusive CMD mathematical model was proposed. The proposed model could clearly express the CMD mechanism and transform the problems of attack and defence in the CMD domain into corresponding mathematical problems. These aspects were considered to qualitatively assess the CMD security, and it was noted that a high level of security can be realized. Furthermore, the overhead of CMD was analyzed. Moreover, the proposed model can be directly programmed.

1. Introduction

With the development of information technology, multimedia has been widely applied in the human society. Moreover, the emergence of artificial intelligence (AI) technologies has considerably enhanced the ability to analyse, process, and utilise multimedia. Because multimedia processing and AI technologies are computation- and storage-intensive, they are realized through the relatively mature distributed computing [1] and cloud computing [2] techniques. Cloud computing is based on server clusters. To enable the efficient processing of multimedia data, ensuring the security of multimedia cloud servers is essential. Multimedia cloud servers face nearly the same issues as those of cyberspace, such as network attacks and multimedia data protection. The multiple reports of network security incidents indicate that attackers mostly exploit the software and hardware vulnerabilities, and backdoor attacks constitute the majority of network security incidents. Nevertheless, at the current level of science and technology, the unknown vulnerabilities and backdoors and the associated security threats cannot be eliminated [3]. Traditional defence technology tends to rely on the attack technology as a priori knowledge, which involves hysteresis and passivity, and thus, the traditional approaches cannot block unknown vulnerabilities and backdoors. Therefore, the development of innovative network security theories and techniques has become a key research direction in the field of network security. Moving target defence (MTD) is a new type of active defence technology developed in the United States to achieve a considerable advantage in the field of network attack and defence [4]. At present, the research and application of this technology are mainly concentrated in developed countries, with the United States as the main hub.

Cyber mimic defence (CMD) is an innovative active defence technology independently developed in China, whose emergence can offset the unbalanced situation of the network attack and defence techniques [5]. The CMD mechanism facilitates its application in the multimedia cloud server cluster environment. The redundancy characteristics in the cluster environment conform with the CMD aspects. In particular, the computing power of the cluster environment can satisfy the performance overhead of the CMD to realize computationally intensive operations such as redundant encryption or decryption. Furthermore, the already mature virtual technology [6, 7] can provide support for the CMD and has been widely used in server cloud environments. In summary, the CMD technology can be applied to multimedia cloud servers. In this study, two cases were considered to enable the protection of multimedia cloud servers through CMD. One approach involves constructing a mimic multimedia cloud server architecture, and the other approach involves the use of the mimic encryption to protect the multimedia data. For the first approach, heterogeneous redundancy is achieved with the granularity at software or hardware on the multimedia cloud servers. It can be heterogeneous at the hardware level, operating system level, database level, server software level, background application level, and can also be heterogeneous at several levels at the same time, which will form a rich heterogeneous redundant server pool. Scheduling these heterogeneous servers through the CMD mechanism can greatly protect software and hardware on them. For the second approach, the main purpose is to protect multimedia data. The use of heterogeneous redundant encryption to protect multimedia data and the use of hash fingerprint comparison to detect and shield threats are in line with the principle of CMD and can greatly improve the security of multimedia data. China has actively promoted the research on the theory and technology of CMD, and a theoretical system of CMD has been established [8, 9] and is considerably different from the static and fixed traditional systems. In this context, it is necessary to clarify the technical aspects of the mechanism of the mimic defence at the theoretical level [10] and to establish an intuitive and exclusive mathematical model according to the CMD mechanism to formulate a mapping relationship with the mathematical aspects. In this manner, the mechanism and protection capabilities of CMD can be clarified, and the research on CMD can be further promoted. Nevertheless, a universally accepted mathematical model for CMD has not been established yet.

In this study, by extensively analysing the attack problems and modelling difficulties, a clear modelling concept was established. By decoupling the model from the specific system input and output scenes, a clear mathematical expression was formulated, while avoiding the modelling difficulties. Furthermore, the process characteristics of the attack were highlighted by establishing a specific mathematical mapping method. The excellent security capacities of CMD were demonstrated using the proposed model. Finally, an intuitive and exclusive mathematical model for CMD was established, which could express the CMD mechanism mathematically and transform the problems of the attack and defence game of the CMD into corresponding mathematical subproblems, thereby enabling the qualitative assessment of the CMD safety capacities.

This paper first presents the research background followed by the main research content. Section 2 introduces the CMD concepts and mathematical knowledge required for modelling. Section 3 describes the prerequisite knowledge for the modelling. Section 4 describes the modelling process and model mechanisms. Section 5 clarifies the mathematical problems of the attack and defence game of the CMD examined through a simulation experiment, analysis, and evaluation, describes the qualitative assessment of the CMD safety aspects, and analyzes the overhead of CMD. Section 6 presents the concluding remarks.

2. Background Knowledge

2.1. Cyber Mimic Defence. Cyber mimic defence [11] is a revolutionary defence technology of "game-changing" initiated by China. The development of CMD was inspired by the mimicry phenomenon and biological immune system in the biological world. The dynamic heterogeneous redundancy (DHR) architecture was used as the core architecture of the CMD. Finally, the CMD theory was formulated, with "structure determines security" as the core idea. CMD is a nonpoint type defence technology with a dynamic [12] and closed-loop mechanism. The high security and high robustness [13] in the core architecture (DHR) of CMD are endogenous and coexisting. The unknown vulnerabilities and backdoors cannot be easily exploited under the CMD framework. In contrast to the static and single characteristics, the uncertainty may induce the attacker's cognitive dilemma, thereby making it nearly impossible for the attacker to form an attack chain.

Here is a brief introduction to the principle of CMD [14]. For specific vulnerabilities or backdoors, defenders must first be able to identify them before they can accurately defend. However, the current level of technology cannot grasp all unknown vulnerabilities and backdoors in advance. When the same target function is implemented in heterogeneous forms, the probability of them having the same vulnerabilities or backdoors will be greatly reduced. CMD puts these heterogeneous redundant executors to work in parallel without communicating with each other. In other words, they have no cooperative relationship and do not know each other's existence. It is a very rare event that the same vulnerabilities or backdoors are existent and are triggered at the same time in heterogeneous executors. Therefore, if the unknown vulnerabilities or backdoors in the CMD system are triggered, the output results of heterogeneous executors will be inconsistent. At this time, CMD uses the "relatively correct" principle to not only detect abnormal situations and perceive threats but also locate abnormal executors based on certain strategies and then take corresponding measures against them. For the aforementioned parallel heterogeneous executors, CMD then introduces a dynamic scheduling mechanism to make the system dynamic.

Next, through the DHR architecture visually shows the principle of CMD. The DHR architecture is shown in Figure 1.

The DHR architecture introduces a dynamic scheduling mechanism and feedback control mechanism [15] based on the executor heterogeneous redundancy and multimode adjudication, respectively. The operating mechanism of the DHR architecture is as follows:

(1) A functionally equivalent heterogeneous executor pool is constructed for a target business function.

(2) Through the dynamic scheduling strategy, several "online" heterogeneous executors are selected from the functionally equivalent heterogeneous executor pool.

(3) When the system input arrives, it is distributed to each "online" heterogeneous executor through the input distributor to ensure that each executor can be executed separately without coordination and communication.

(4) The output vectors of all the "online" heterogeneous executors produce the final output result through the multimode adjudication strategy. At this time, if an abnormal "online" heterogeneous executor is "perceived," the feedback control mechanism is activated according to the multimode adjudication strategy.

(5) If the feedback control is activated, the abnormal "online" heterogeneous executors are replaced by "offline" executors through the feedback control strategy, and the subsequent processes such as self-cleaning, self-reconstruction, self-reorganisation, log recording, and log analysis are performed in the background.

The DHR architecture, as the core architecture of CMD, illustrates the premise of CMD. CMD can be applied to a target object having the form of "Input-Process-Output," and it is represented by the I [P] O model. In [11], the input distributor and output arbiter were termed as "mimic brackets" (MB), and the scope of protection limited by the MB was defined as the "mimic defence boundary" (MDB). The MDB is usually a heterogeneous execution environment with unknown vulnerabilities, backdoors, viruses, and Trojan horses.

In recent years, research pertaining to the theory and mechanism of CMD has progressed rapidly. At present, the mimic domain name server has been implemented online, and the principle prototypes of the mimic web server [16, 17] and mimic router [18, 19] have been developed. In addition, the CMD technology has been applied in several fields to ensure multimedia security, 5G security [20], SDN network security [21, 22], software diversification [23], mimic storage system realization, mimic encryption, and mimic cloud, mimic firewall, and mimic gateway realization.

2.2. Mathematical Knowledge Required for Modelling

2.2.1. Decomposition Problem of Large Prime Factors. The decomposition problem of large prime factors can be described as follows: if there exist several large prime factors, they can be easily multiplied to obtain a large composite number. However, it is extremely difficult to obtain these large prime factors by factorising the large composite number. This problem has been studied by mathematicians for hundreds of years, and no rapid algorithm is available to solve this problem. The research on the decomposition problem of large prime factors is challenging, but has theoretical and application value [24]. For example, the popular RSA [25] encryption algorithm relies on the decomposition problem of large prime factors.

In addition to the early violent trial division method, certain algorithms to solve the decomposition problem of large prime factors have been proposed by researchers, such as the ρ-method [26], P − 1 method [27], elliptic curve method [28], random square method, quadratic sieve method [29], and number field sieve method [30]. Among these algorithms, the number field sieve method is considered to be the best at present.

2.2.2. Convolution Operation. Convolution is a key operation in analytical mathematics and is performed as follows: first, two independent functions and the definition domain of their parameters are specified. Subsequently, the function values are calculated, and the corresponding function values are multiplied. Finally, all the products are summed. Specifically, convolution involves rolling a binary function into a univariate function in a process commonly known as "dimension reduction."

The convolution operation can be divided into continuous and discrete convolutions:

(1) To perform convolution, a certain relationship must exist among variables x, y, and n. Assume that the relation $x + y = n$ holds. For a particular n, this relation can represent a straight line with a slope of 1 in the Cartesian coordinate system.

(2) The variable τ is defined, and the range of τ is expressed as R. According to the aforementioned relation, $x = \tau$, and $y = n - \tau$.

(3) If the functions of x and y are $f(x)$ and $g(y)$, respectively, they can be written as $f(\tau)$ and $g(n - \tau)$, respectively.

In this case, the discrete and continuous convolution can be represented as in formulas (1) and (2), respectively:

$$(f * g)(n) = \sum_{\tau \in R} f(\tau) g(n - \tau), \tag{1}$$

$$(f * g)(n) = \int_{\tau \in R} f(\tau) g(n - \tau)\, \mathrm{d}\tau. \tag{2}$$

Here, $(f * g)(n)$ is termed as the convolution of functions f and g.

A convolution operation can reflect valuable physical meanings in an engineering system and can be used to calculate the output of such a system.

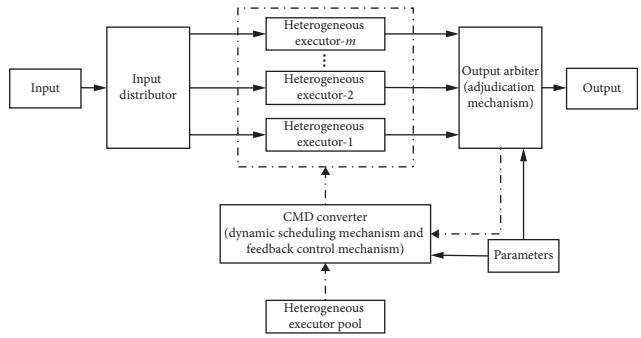

Figure 1: DHR diagram.

2.2.3. Martingale. The martingale concept originated from the mathematical description of the fair gambling process. Specifically, the martingale system is a concept in probability theory, which describes a special class of stochastic processes. Herein, the definitions and meanings of discrete and continuous martingale are presented.

(1) Discrete Martingale. If a discrete-time stochastic process X_n satisfies the following conditions,

(a) $E(|X_n|) < \infty$ and

(b) $E(X_n + 1 \mid X_1,\ldots,X_n) = X_n$,

the stochastic process X_n is a discrete martingale. Specifically, for the stochastic process X_n, if all the values at the present moment and all the previous moments are known, the conditional expectation value at the next moment is equal to the value of the present moment.

If the discrete-time stochastic processes X_n and Y_n satisfy the following conditions,

(a) $E(|Y_n|) < \infty$ and

(b) $E(Y_n + 1 \mid X_1,\ldots,X_n) = Y_n$,

the stochastic process Y_n is a discrete martingale on X_n. Specifically, for the stochastic processes X_n and Y_n, if all the values of the former process at the present moment and previous moments are known, the conditional expectation value of the latter process at the next moment is equal to the value of the latter process at the present moment.

(2) Continuous Martingale. If a continuous-time stochastic process X_t satisfies the following conditions,

(a) $E(|X_t|) < \infty$ and

(b) $E(X_t \mid \{X_m, m \le s\}) = X_s$,

the stochastic process X_t is a continuous martingale. Specifically, for the stochastic process X_t, if all the values up to time s are known, the conditional expectation value at time t $(t > s)$ is equal to the value at time s.

If the continuous-time stochastic processes X_t and Y_t satisfy the following conditions,

(a) $E(|Y_t|) < \infty$ and

(b) $E(Y_t \mid \{X_m,\ m \le s\}) = Y_s$,

the stochastic process Y_t is a continuous martingale on X_t. Specifically, for the stochastic processes X_t and Y_t, if all the values of the former process up to time s are known, the conditional expectation value of the latter process at time t $(t > s)$ is equal to the value of the latter process at time s.

3. Modelling Concept

3.1. Network Attack Analysis. The network attack technology, in combination with computer technology, is undergoing constant development. The attack behaviour has the characteristics of uncertainty, complexity, and diversity and is developing towards a large-scale, collaborative, and multilevel framework. Therefore, the research [31–34] and formal description of network attacks are of considerable significance to both sides. The classical network attack modelling methods involve the use of the attack tree [35], attack graph [36], and attack network [37]. In addition, the attack surface (AS) [38, 39] and mobile attack surface (MAS) [40, 41] theories have emerged in recent years to analyse and examine the law of network attack behaviour.

3.1.1. Attack on the Traditional Static and Single System. The following is described in the context of the traditional static and single system.

Although there are differences in the description of the network attack process by the above methods or theories, in conclusion, a successful network attack is a process comprising several stages, and there may be repeated backtracking subprocesses, as shown in Figure 2.

To summarize, a successful network attack can be simply described as a critical path from the beginning of the attack to the success of the attack based on each stage. In this paper, this critical path is referred to as the successful attack vector (SAV). By further analysing the SAV, the following two characteristics can be defined:

(1) *Target Characteristics.* The SAV has target characteristics similar to a vector, and the target and direction from the beginning to the end point are clear and unique. Consequently, a successful network attack can be easily described as a chain because of the targeting characteristics of the SAV.

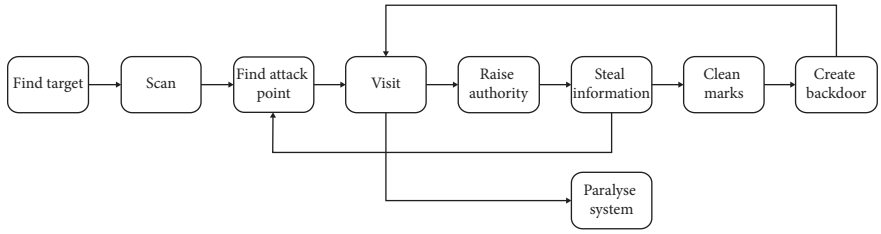

Figure 2: Example of the attack process.

(2) *Process Characteristics.* The SAV can clearly reflect the process characteristics of the network attack. Specifically, the negative influence of a successful attack follows the attack behaviour. Because the final result is not immediately visible, it is latent to the attacked object. Thus, the attacker must move sequentially toward the final goal with considerable patience, which leads to a typical "delayed satisfaction."

3.1.2. Attack on the CMD System. Because the technology of the CMD is not widespread at present, the data of only a few typical attack cases are available. Nevertheless, it is possible to analyse and estimate the attacks against the CMD system.

This paper emphasises that the premise of the CMD is to realize scenarios that satisfy the I [P] O model, which is critical to activate the defence effect of the CMD. In this paper, we divide the attacks against the CMD system into type-ρ and type-φ attacks.

(1) Type-ρ Attack. A type-ρ attack is a kind of attack that deviates from the premise of CMD. The CMD cannot always resist this kind of attack, but it can be applied to supplement the existing security defence technology or means [42]. In this paper, certain simple examples are presented to illustrate the type-ρ attacks:

(a) The attacker successfully intrudes an "online" heterogeneous execution of a certain CMD system and deletes an important file in it. The successful attack does not produce any output that must pass through the CMD system arbiter.

(b) The attacker successfully intrudes an "online" heterogeneous execution of a certain CMD system, cracks the key of an important encrypted file on it, and browses the valuable data in the file. Moreover, the successful attack does not produce any output that must pass through the CMD system arbiter.

(c) A certain CMD system is subjected to a DDoS attack.

(2) Type-φ Attack. In contrast to a type-ρ attack, a type-φ attack is a kind of attack that satisfies the premise of the CMD. In this case, after the attacker successfully attacks the "online" heterogeneous executor in the CMD system, an output that must pass through the CMD system arbiter is produced. This output is the desired attack result for the attacker and the so-called abnormal output for the CMD system. From the perspective of the CMD system, inconsistencies among the abnormal outputs may exist in theory. However, in terms of the rationality, because the CMD adjudication mechanism is implemented, for a type-φ attack to succeed, a "joint escape" in which a consistent abnormal output is generated must be realized, and it is meaningless for the attackers to create an inconsistent abnormal output. Therefore, all the abnormal outputs mentioned in this paper are consistent, which negates the problem of the type-φ attack on the CMD system and facilitates follow-up research. Considering this background, a successful and failed type-φ attack is that in which a "joint escape" can and cannot be realized, respectively.

Considering these aspects and the categorization of the attacks against the CMD system, it can be noted that only the φ-attack problem has practical research significance for both sides of the CMD attack and defence.

3.2. Problem Description. A mathematical model is key to support the continuous improvement of a theoretical system and to promote the related research work. In this context, developing a mathematical model can help describe and analyse the related mechanism of the CMD.

At present, two kinds of CMD modelling are implemented, specifically, modelling the overall mechanism of the CMD and modelling only part of the mechanism of the CMD [43, 44]. This paper focuses on the former approach, and thus, the following problems as well as the existing research results correspond to this approach. Until now, a universally accepted mathematical model for the theory of CMD has not been established, and the mainstream modelling method is mainly based on the "Markov chain." [45, 46]

(1) Realistic mapping of the overall mechanism modelling of CMD:

This problem is defined according to the aforementioned SAV characteristics.

(a) In contrast from that in the traditional static and single system, the target protection object in the CMD system is heterogeneously redundant. The heterogeneous redundant executors are completely independent and do not communicate with one another. If an attacker wants to successfully realize the φ-attack, they must complete several independent SAVs simultaneously, and this requirement is different from that for attacking a traditional static and single system. However, because several independent SAVs are

present, the relevant stages may not be completely consistent or have a one-to-one correspondence. In addition, backtracking and repeated subprocesses may be present in the attack process. Therefore, although the target characteristics of a single SAV facilitate the description of a successful network attack in the chain mode, the description of several independent SAVs in the chain mode simultaneously is complex and challenging. In this case, a single chain node must represent the stages or states from several independent SAVs simultaneously. If these nodes are combined as in the actual situation, the number of chain nodes required may be considerably higher than that in a "traditional chain," as discussed in the subsequent sections. This type of chain structure is easy to network; however, the target characteristic of each independent SAV becomes difficult to identify, which makes the analysis and research highly challenging.

(b) In the core architecture (DHR) of the CMD, one input corresponds to one output. If a type-φ attack fails and an abnormal output is present, it will be detected in the adjudication stage of the DHR. As mentioned previously, an SAV has a process characteristic, and when an attack is conducted, an abnormal output is not necessarily produced. In other words, the attack cannot be represented by only one input-output pair. At present, the DHR architecture cannot directly reflect this process characteristic. Therefore, it is necessary to model the overall mechanism of CMD with a "buffer" that can reflect the delay in the process characteristic. As a simple and intuitive example, we consider the case of uploading files on a server. In the absence of any other security protection measures, three steps can be defined: request, upload, and access. In malicious attacks, the three steps may be as follows: request, upload the virus, and access and activate the virus. Each of these three steps corresponds to a system input. The attack behaviour can be considered to start in Step 1 or 2, although the negative impact of the successful attack occurs in Step 3.

(2) Problem of modelling the overall mechanism of CMD based on the Markov chain:

(a) A Markov chain is a state chain that describes a process of state dynamic transfer. At present, the Markov chain is applied to the CMD theory to describe the process of an attack and defence game. Therefore, the Markov chain modelling is focused on describing the transfer path of the attack and defence game state, owing to which the CMD mechanism appears to be highly abstract. This phenomenon occurs because the state transfer path occupies the main body, and only the state nodes in the chain are used to represent the state of the whole CMD system at a certain time. The CMD mechanism cannot be intuitively reflected in this scenario, which hinders the mechanism description and analysis.

(b) The Markov chain is a generic modelling tool and not specifically applied to CMD modelling. To apply the Markov chain to model the overall mechanism of the CMD, a custom set of state parameters must be used to describe the state of the CMD system. The complexity of this set of custom state parameters directly affects the complexity of the Markov Chain, and the complexity of the real scene directly affects the complexity of the custom state parameter set. Owing to this strong binding with the real scene, the CMD Markov chain can only be used to model simple system scenes, for instance, to model the type of system input that would lead to a certain type of system state and system output.

Considering these problems, in this work, the modelling method based on the Markov chain was not employed, and the objective was to develop an intuitive and exclusive mathematical model for the CMD.

3.3. Core Concept. From the mathematical viewpoint, this paper performs a formal analysis of the heterogeneous redundant executors of the CMD framework:

(1) For a given CMD system, the heterogeneous executor pool can be represented by the mathematical set E, where $E = \{e_1, e_2, e_3, \ldots, e_n\}$ with e_i $(1 \le i \le n)$, and the set elements represent the heterogeneous executors. The set of "online" heterogeneous executors is represented as the mathematical set E_O, where $E_O = \{e_{o1}, e_{o2}, e_{o3}, \ldots, e_{om}\}$ with e_{oj} $(1 \le j \le m)$, and the set elements represent the "online" heterogeneous executors. The set E_O is a subset of E.

(2) In the CMD mechanism, the set of "online" heterogeneous executors is modified according to the strategy. Therefore, the mechanism can be formalised as the process of selecting a series of E_O on set E in "1" according to the strategy, which represents a mathematical process of selecting combinations.

(3) A combination in "2" is termed as a "sample," represented by S. The sample space comprising all possible S values is represented as S_S. S_S is similar to a sample warehouse, termed as the "mimic warehouse" in this paper. Therefore, the essence of the CMD mechanism is to schedule and operate the "samples" on the so-called "mimic warehouse."

(4) The elements of sample S in "3" are formalised as large prime factors, represented as f_j $(1 \le j \le m)$, where m represents the number of large prime factors corresponding to the number of elements of combination S and is distinguished by label j.

Subsequently, the sample S can be formalised as the product of these large prime factors, represented as the following formula:

$$S = \prod_{j=1}^{m} f_j, \quad 1 \le j \le m. \tag{3}$$

The objective of this formal analysis process is to associate the "online" heterogeneous executor set of the CMD system at a certain time with a product of the large prime factors. Analysing the product of the large prime factors corresponds to the analysis of the corresponding "online" heterogeneous executor set.

4. Large-Number Convolutional Mimic Defence Mathematical Model

The proposed mathematical model of the CMD is based on the convolution operation and depends on the decomposition problem of large prime factors. The model is termed as the large-number convolutional mimic defence (LNCMD). The LNCMD model is an intuitive and exclusive mathematical model of CMD. This section describes the modelling process and mechanism of the LNCMD.

4.1. Modelling

4.1.1. Assumptions

(1) The LNCMD model has the same black box characteristics as the CMD system.

(2) Based on the relationship described in the "core concept" section, the input and output of the LNCMD model are not the input and output of the information system in the actual sense, respectively. Therefore, the LNCMD model does not involve any specific input and output values of the actual CMD system. The "problem description" section highlighted the problems faced by the CMD Markov chain owing to its feature of strong binding with the real scene. The aforementioned configuration helps the LNCMD model avoid these problems.

(3) Each strategy in the CMD system, namely, the dynamic scheduling strategy, adjudication strategy, and feedback control strategy, is configured by the defender.

(4) The attack considered in the LNCMD model is a type-φ attack. The abnormal output, successful type-φ attack, and failed type-φ attack are defined according to the aforementioned descriptions.

(5) All the heterogeneous executors in the CMD system have a comprehensive evaluation value. A larger evaluation value corresponds to a more safe executor [47]. It is considered that the comprehensive evaluation value is a large prime number. It is assumed that the development tasks of all the heterogeneous executors in a CMD system are not completed by the defender, and the comprehensive evaluation values of all the heterogeneous executors are known.

4.1.2. Model Framework.

Based on the model components, the LNCMD model logically divides the CMD system into three layers, namely, the extraction, convolution, and judgement layers. The mechanism of the LNCMD model corresponds to the cooperation of these three layers. The logic of the extraction, convolution, and judgement layers is represented by a mathematical function, and the corresponding functions are termed as extraction, convolutional, and judgement functions, respectively.

4.1.3. Model Components and Rules

(1) Private components:

According to the assumptions, the LNCMD model exhibits black box features to the external environment. The private components are visible only inside the LNCMD model and can be read and written only inside the LNCMD model.

(a) Large prime factor pool: this pool stores the comprehensive evaluation values of all the heterogeneous executors in the CMD system. In a practical sense, this pool represents the heterogeneous executor pool of the CMD system.

(b) Convolution kernel vector: this vector stores the comprehensive evaluation values of the "online" heterogeneous executors. The length of the vector is equal to the number n of the "online" heterogeneous executors, and the i^{th} element on the vector corresponds to "online" heterogeneous executor i $(1 \le i \le n)$.

(c) Product variable: this variable stores the product of all the elements of the convolution kernel vector. Although the variable is in the form of a product, the order of each factor of the product strictly follows the order of the original element position on the convolution kernel vector.

(d) Hidden matrix: this matrix is an abstract matrix with the dimensions $n \times m$, where n is the number of "online" heterogeneous executors, and row i corresponds to "online" heterogeneous executor i $(1 \le i \le n)$. Here, m is an uncertain value when a column of the hidden matrix corresponds to a model input, and it increases dynamically with the number of inputs. The element values of the hidden matrix are generated through the model input as the excitation. The j^{th} excitation can only generate the elements of column j, and the elements after column j are not visible. The value range of the hidden matrix elements is [0, 1], which represents the attack progress of 0–100%. In the LNCMD model, this range represents the progress of the attacker decomposing the specified large prime factor from the value of the product variable. In terms

of the actual meaning, this range represents the amount of the SAV covered by the attacker attacking a specific "online" heterogeneous executor.

(e) Layer signal: the values can be 0, 1, or 2, which correspond to the operation of the extraction, convolutional, and judgement functions, respectively. When the layer signal changes from one value to another, it is considered that the extraction, convolutional, and judgement functions are strictly mutually exclusive even if one function is operating, that is, only one layer is allowed to operate at a certain time. The extraction, convolutional, and judgement functions can influence the signal actively. However, due to the dynamic scheduling strategy, the signal may be set passively through to dynamic scheduling. For example, the online time of the "online" executor may reach the limit. The logical turbulence caused by the passive setting above that of the LNCMD model depends on the rationality of the dynamic scheduling strategy and is not related to the LNCMD model.

(2) Nonprivate components:

Nonprivate components are those components of the LNCMD model that communicate with the outside.

(a) *Model Input*. This input is generated by the customer. In terms of the influence, the external environment influences the LNCMD model through the input of the convolution layer function. In terms of the attack and defence game, the attacker tries to decompose the large prime factor from the product as the game action.

(b) *System Strategies*. These strategies are configured by the defender. In terms of the influence, the system strategies, as the system level configuration, play a key role in the LNCMD model, and they are not used as the input of any layer function of the LNCMD model. In terms of the attack and defence game, the defender adopts the game actions by changing various system strategies.

(c) *Model Output*. This output is used to indicate whether the LNCMD model has been attacked according to the adjudication strategy. The output is a Boolean type value, with true-1 and false-0 indicating an attacked and not attacked state, respectively.

4.1.4. Model Symbol Set. The LNCMD model comprises tuples as follows: LNCMD = {*C*, eL, cL, jL}.

(1) *C* represents the components and is denoted as *C* = {Pri, Pub}, where Pri and Pub represent the private components and nonprivate components, respectively.

(a) Pri = {pool, vector, product, matrix, signal}
Here, "pool" is the large prime factor pool, "vector" is the convolution kernel vector, "product" is the product variable, "matrix" is the hidden matrix, and "signal" is the layer signal.

(b) Pub = {input, strategy, output}
Here, "input" is the model input, "strategy" represents the system strategies, and "output" is the model output.

(2) eL represents the extraction layer, with the logic corresponding to the extraction function.

(3) cL represents the convolution layer, with the logic corresponding to the convolutional function.

(4) jL represents the judgement layer, with the logic corresponding to the judgement function.

The LNCMD model is shown in Figure 3.

4.2. Mechanism

4.2.1. Extraction Layer. The logic of the extraction function in the extraction layer is to refresh the convolution kernel vector, set the product variable, refresh the hidden matrix according to the scheduling strategies on the heterogeneous executors—including the dynamic scheduling strategy and feedback control strategy—and, finally, set the layer signal. The extraction function is described in Table 1.

The detailed functions are as follows:

(1) Refresh the convolution kernel vector:

The extraction function first determines the "online" heterogeneous executors 1,. . .,*n* according to the scheduling strategies of the heterogeneous executors. Subsequently, the extraction function extracts the comprehensive evaluation values of these "online" heterogeneous executors from the large prime factor pool. $F_1, \ldots, F_n$ are the comprehensive evaluation values of the extracted heterogeneous executors, where F_i corresponds to "online" heterogeneous executor *i*. Finally, the extraction function places $F_{1,}. \ldots, F_n$ into the convolution kernel vector, where F_i is placed into the position of the i^{th} element of the convolution kernel vector.

The redundancy scale of three is considered as an example, as shown in Figure 4.

(2) Set the product variable:

Calculate the product of all the elements of the convolution kernel vector, and place this product into the product variable.

(3) Refresh the hidden matrix:

Considering the scheduling strategies of the heterogeneous executors, the "online" heterogeneous executors $1, \ldots, n$ are redetermined. Therefore, the extraction function must refresh the hidden matrix at this time by reinitialising the hidden matrix in an abstract sense.

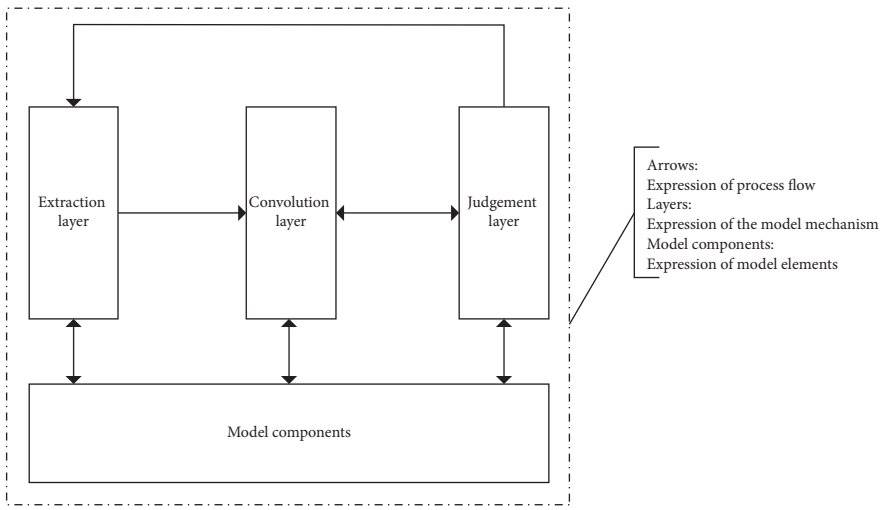

Figure 3: LNCMD diagram.

Table 1: Explanation of the extraction function.

Category	Explanation
Function input	Not applicable
Function output	Not applicable
Process summary	The extraction function operates internally on the private components of the LNCMD model
Function shape	Pseudo-code shape: void extract ();

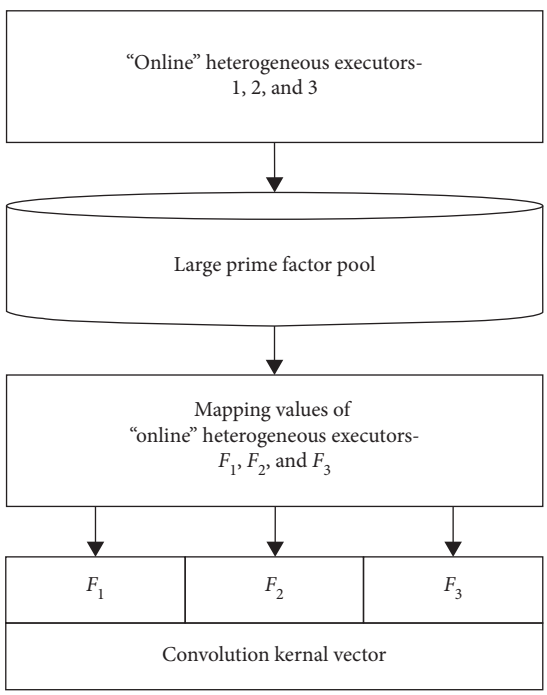

Figure 4: Refresh the convolution kernel vector.

Considering the redundancy scale of three as an example, the action of refreshing the hidden matrix is illustrated in Figure 5.

The significance of refreshing the hidden matrix is as follows: because the combination of the "online" heterogeneous executors changes, the statistical number of the model input times should be recounted from zero; consequently, the exploration or expansion of the hidden matrix columns excited by the model inputs should also be restarted. In a logical sense, this action represents a milestone start.

The overall significance of the two actions of refreshing the convolution kernel vector and hidden matrix is to reflect the scheduling mechanism of the CMD for "online" heterogeneous executors, and this mechanism is reflected by these two actions on the LNCMD model.

(4) Set the layer signal:

The extraction function sets the layer signal as 1, indicating the commencement of the convolutional function operation.

4.2.2. Convolution Layer. The logic of the convolutional function in the convolution layer is to wait for the j^{th} model input, which excites the convolutional function to produce the j^{th} convolution output. The convolutional function is described in Table 2.

The detailed functions are as follows:

(1) Convolution operation:

The convolution operation of the convolutional function is performed between the vector and the matrix, and thus, it is a type of discrete convolution. The discrete convolution formula of the convolutional function is as follows:

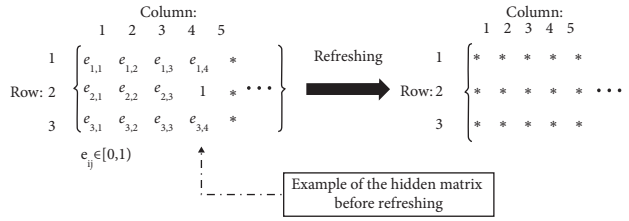

Figure 5: Refresh the hidden matrix.

Table 2: Explanation of the convolutional function.

Category	Explanation
Function input	j^{th} model input
Function output	Arithmetic formula of the j^{th} convolution result
Process summary	The output of the convolutional function is generated by a special convolution operation with the parameter input as the excitation
Function shape	Pseudo-code shape: outputType convolutional (inputType inputJ)

$$\text{convolutional}(\text{input } J) = (f^* g)(\text{input } J) = \sum_{i=1}^{n} f(i) g(\text{try}F_i), \quad n \geq 1. \tag{4}$$

Assuming that the value of the product variable is P, the formula can be explained as follows:

(a) Parameters:

There exist three parameters, i, $tryF_i$, and $inputJ$. Parameter i is the label of the i^{th} "online" heterogeneous executor. The parameter $tryF_i$ is abstract. In an arithmetic sense, this parameter is a trial value of F_i, and F_i is the i^{th} large prime factor of P. In a logical sense, this parameter represents an attempt of the attacker to attack the i^{th} "online" heterogeneous executor. The parameter $inputJ$ is the j^{th} model input.

The aforementioned three parameters satisfy the following relationship: $i + tryF_i = inputJ$. According to this relationship, in the arithmetic sense, $inputJ$ corresponds to a set of points $(i, tryF_i)$ on the "logical straight line" $(i + tryF_i = inputJ)$. The two functions $f(i)$ and $g(tryF_i)$ are combined into one convolutional function ($inputJ$) along the direction of the "logical straight line" $(i + tryF_i = inputJ)$. In a logical sense, this parameter indicates that a model input corresponds to a set of the attackers' attempts to attack each large prime factor (ordered) in P. In a practical sense, this parameter is the input distribution mechanism of the CMD.

(b) Range:

The value range of Σ summation is 1 to n, where n is the number of current "online" heterogeneous executors. This value is the length of the convolution kernel vector, which is the number of large prime factors of P, equal to the number of rows of the hidden matrix.

(c) Logic of subfunctions:

After the $f(i)$ function obtains parameter i, it searches for the i-th element F_i in the convolution kernel vector, and the function returns the reciprocal of F_i.

After the $g(tryF_i)$ function obtains the parameter $tryF_i$, the element value e of the corresponding position of the hidden matrix is calculated, and the return value of the $g(tryF_i)$ function is $1 - e$. The $g(tryF_i)$ function calculates e as follows:

The $g(tryF_i)$ function which calculates e requires a kind of carrier function, which needs to satisfy the following properties:

(1) The function is a one-variable continuous function, which can be expressed as $y = f(x)$
(2) In the domain $(0, +\infty)$, the function values are always greater than 0 and have a unique absolute maximum value
(3) Assuming that the function value of $f(x)$ at x_0 is the unique absolute maximum value, then x_0 should be a large prime number

This paper gives a typical example, selecting the Gaussian distribution probability density function as the carrier function of the $g(tryF_i)$ function. The characteristic of the Gaussian distribution probability density function is that a value closer to the

mathematical expectation (average) μ corresponds to a greater probability density. The $g(tryF_i)$ function uses this property for the mapping; therefore, the closer $tryF_i$ is to F_i, the closer the value e is to 1, and the attack progress is closer to 100%.

The formula of the Gaussian distribution probability density function is as follows:

$$\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right), \quad -\infty < x < +\infty. \tag{5}$$

The formula of the Gaussian distribution probability density function in the $g(tryF_i)$ function is as follows (5):

$$\begin{cases} \dfrac{1}{\sqrt{2\pi}}\exp\left(-\dfrac{(tryF_i - F_i)^2}{2}\right), & 0 < tryF_i < +\infty, \\ 0, & tryF_i = 0, \\ 0, & tryF_i = +\infty. \end{cases} \tag{6}$$

In formula (5), when $tryF_i$ is equal to F_i, the calculated value is $(2\pi)^{-0.5}$. Finally, the formula to calculate the value of e in the $g(tryF_i)$ function is as follows:

$$\frac{1/\sqrt{2\pi}\exp\left(-(tryF_i - F_i)^2/2\right)}{\left(2\pi^{-0.5}\right)}. \tag{7}$$

(d) Calculation result:

The result of the convolution operation of the convolutional function corresponds to the multiplication of the return values of the $f(i)$ and $g(tryF_i)$ functions. Subsequently, a summation formula in the range $i = 1$ to n is defined, which is used to determine the output of the convolutional function as

$$\sum_{i=1}^{n}\frac{1-e_{ij}}{F_i}. \tag{8}$$

(2) Set the layer signal:

The convolutional function sets the layer signal as 2, indicating the operation of the judgement function.

4.2.3. Judgement Layer. The logic of the judgement function in the judgement layer is to receive the output of the convolutional function, perform an adjudication according to the adjudication strategy, and generate the model output. The judgement function is described in Table 3.

The detailed functions are as follows:

Considering the redundancy scale of three as an example, the logic of the judge function can be described as follows:

(1) Generic Pretreatment. The generic preprocessing consists of four steps. Let the value of the product variable be P.

Step 1: define P to formulate a score with a real value of 1 as follows:

$$\frac{P}{P} = \frac{F_1 \times F_2 \times F_3}{F_1 \times F_2 \times F_3} = 1. \tag{9}$$

Step 2: set the function input as convolutionJ. Multiply convolutionJ by the fraction in Step 1 to merge it into a single fractional form. In the arithmetic sense, this step involves merging all the subfractions in convolutionJ with P as the denominator as follows:

$$\begin{aligned} &\text{convolution } J\colon \frac{1-e_{1j}}{F_1} + \frac{1-e_{2j}}{F_2} + \frac{1-e_{3j}}{F_3} \Rightarrow \\ &\text{convolution } J \times \frac{P}{P} \Rightarrow \frac{\left(1-e_{1j}\right)\times F_2 \times F_3 + \left(1-e_{2j}\right)\times F_1 \times F_3 + \left(1-e_{3j}\right)\times F_1 \times F_2}{F_1 \times F_2 \times F_3}. \end{aligned} \tag{10}$$

Table 3: Explanation of the judgement function.

Category	Explanation
Function input	Output of the convolutional function
Function output	Output of the LNCMD model
Process summary	The process of generating a function output with the parameter input in the judgement function is highly dependent on the adjudication strategy
Function shape	Pseudo-code shape: bool judge (inputType convolutionJ)

Step 3: for the numerator of the result fraction in Step 2, extract the common factor based on the factor in the denominator. The following example illustrates the successful extraction of the common factor:

$$\begin{aligned}\text{convolution } J\text{: } \frac{1-e_{1j}}{F_1}+\frac{1-e_{3j}}{F_3} &\Longrightarrow \text{convolution } J\\ &\times\frac{P}{P}\Longrightarrow\frac{\left(1-e_{1j}\right)\times F_2\times F_3+\left(1-e_{3j}\right)\times F_1\times F_2}{F_1\times F_2\times F_3}\\ &\Longrightarrow\frac{F_2\times\left(\left(1-e_{1j}\right)\times F_3+\left(1-e_{3j}\right)\times F_1\right)}{F_1\times F_2\times F_3}.\end{aligned} \tag{11}$$

In the arithmetic sense, the extraction of certain common factors indicates that, in the convolution process of the convolutional function, there exist subfractions of the following form, owing to which certain F_i are extracted as common factors:

$$f(i)g\left(tryF_i\right)=\frac{1}{F_i}\times 0. \tag{12}$$

In other words, certain outputs of the $g(tryF_i)$ function are 0. In the logical sense, the extracted common factor directly corresponds to the large prime factor decomposed by the attacker from P. In the practical sense, the extracted common factor reflects the "online" heterogeneous executor whose SAV is covered by the attacker.

Step 4: define the result formula in Step 3 as the "result."

(2) Process Highly Dependent on the Adjudication Strategy. Because this process can affect the logical flow direction of the LNCMD model according to different adjudication strategies, this process is highly dependent on the adjudication strategy. We assume that the adjudication strategy is the majority voting strategy, described as follows:

(1) Output: the most consistent results are considered as the final result.

(2) Adjudication: when the results are not completely consistent, the system is considered to be attacked, and feedback control is launched. At this time, the information of the executors whose outputs are inconsistent with the final result is sent to the feedback control strategy.

The process strongly dependent on the adjudication strategy includes three sequential subprocesses. The sequence is $a\rightarrow b\&c$, in which the b and c subprocesses can be performed simultaneously.

(a) Judgement process:
According to the adjudication strategy, the attack status of a system can be determined. The judgement process is shown in Figure 6.
When the logic passes through the left branch, the number of common factors extracted in the "result" is 0 or 3, which represents the situation in which the output vectors of the "online" heterogeneous executors are completely consistent. In particular, when the number of common factors extracted in the "result" is 3, the situation is an extreme one in which the attacker has covered the SAVs of all the "online" heterogeneous executors. When the logic passes through the right branch, the number of common factors extracted in the "result" is neither 0 nor 3, which represents the inconsistency of the output vectors of the "online" heterogeneous executors. Furthermore, in this case, two situations may occur, corresponding to the abnormal output vectors being in the majority or minority.
Let the judgement result of this subprocess be "judge."

(b) Output process:
The judgement result "judge" of subprocess a is considered as the output of the model.

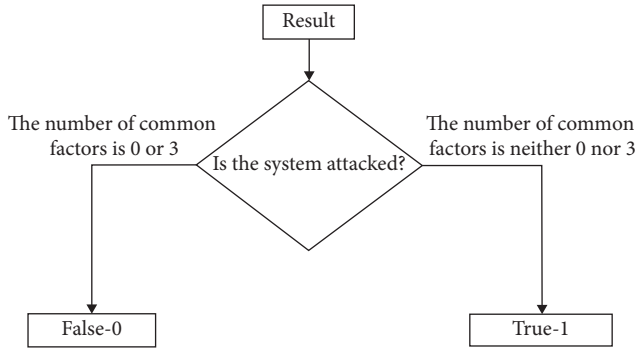

Figure 6: Judgement process flow.

(c) Process of setting the layer signal:
The layer signal settings are defined according to the adjudication strategy. The judge function sets the layer signal to 0, indicating the operation commencement of the extraction function. The judge function sets the layer signal to 1, indicating the operation commencement of the convolutional function. The practical significance is the determination of whether the feedback control must be started, according to the adjudication strategy. This subprocess can be regarded as a snapshot for the partial action of the DHR architecture, which pertains to the launch of the feedback control, as shown in Figure 7.

The process strongly dependent on the adjudication strategy ($a \rightarrow b\&c$) is illustrated in Figure 8.

Because the processing of the judgement function in the judge layer depends strongly on the adjudication strategy, the following description holds.

The aforementioned modelling helped define the role and rules for each strategy in the CMD system in the LNCMD model. The LNCMD model does not directly reflect the logic encapsulated by each strategy. As shown in the figure, the logical process in the dashed box is actually the action of the LNCMD model. Moreover, although the left branch of the "result" variable is the logic encapsulated by the adjudication strategy, it does not belong to the process of the LNCMD model.

4.3. Summary. The foregoing content establishes the LNCMD model and introduces the composition of the model and the mechanism of the model in detail. For the mathematical knowledge introduced in Section 2, it has been integrated into the LNCMD model. The decomposition problem of large prime factors is regarded as a core mathematical problem throughout the entire model mechanism. How to solve this problem is imposed on the attacker, and the defender creates this problem through the LNCMD model. Both sides carry out game behaviour around this mathematical problem. Convolution operation supports the mechanism of the entire convolution layer, and it is an important bridge to carry out the game behaviour around the decomposition problem of large prime factors. Martingale is a special stochastic process. When the LNCMD model is not started to run, the martingale cannot be reflected in it. When the LNCMD model is started to run, it has actual procedural properties. At this time, the martingale can be used to evaluate the safety status of the LNCMD model. This will appear and be described in detail in Section 5.

5. Simulation Experiment and Evaluation

5.1. Simulation Experiment

5.1.1. Simulation Environment Design

(1) Physical Background. Considering the assumed web service programme as the physical background for the simulation experiment and to apply the CMD technology, the redundancy scale of the "online" heterogeneous service programmes was set as three. The DHR architecture to construct this physical background is shown in Figure 9.

(2) LNCMD Model Population. In the modelling, a symbol set for the LNCMD model was established but not populated. This symbol set of the LNCMD model was populated according to the physical background. The specific population details for the LNCMD model are presented in Table 4. The populated LNCMD model is termed as lncmdDemo in this paper.

(3) Experiment Configuration. In the simulation experiment, a feature string backdoor is defined. Let this feature string backdoor be "door" and the corresponding feature string be "flag." In the simulation experiment, it is assumed that the only way for the attacker to complete the SAV is to trigger the "door" by trying the correct "flag." In lncmdDemo, the "flag" that can trigger the "door" is the large prime factors (ordered) of P.

The simulation experiment is performed under 5 configurations. In the simulation experiment, a simulation run with lncmdDemo is performed under each configuration. The 5 configurations are as follows:

Configuration 1: all the three "online" heterogeneous executors have no door, that is, $a\&b\&c = +\infty$

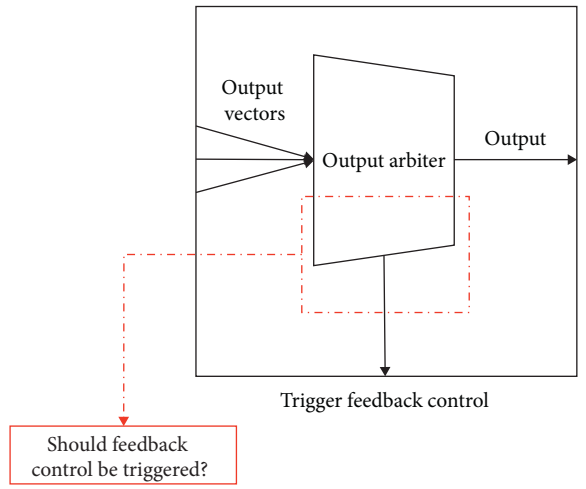

Figure 7: Local snapshot to determine whether the DHR starts the feedback control.

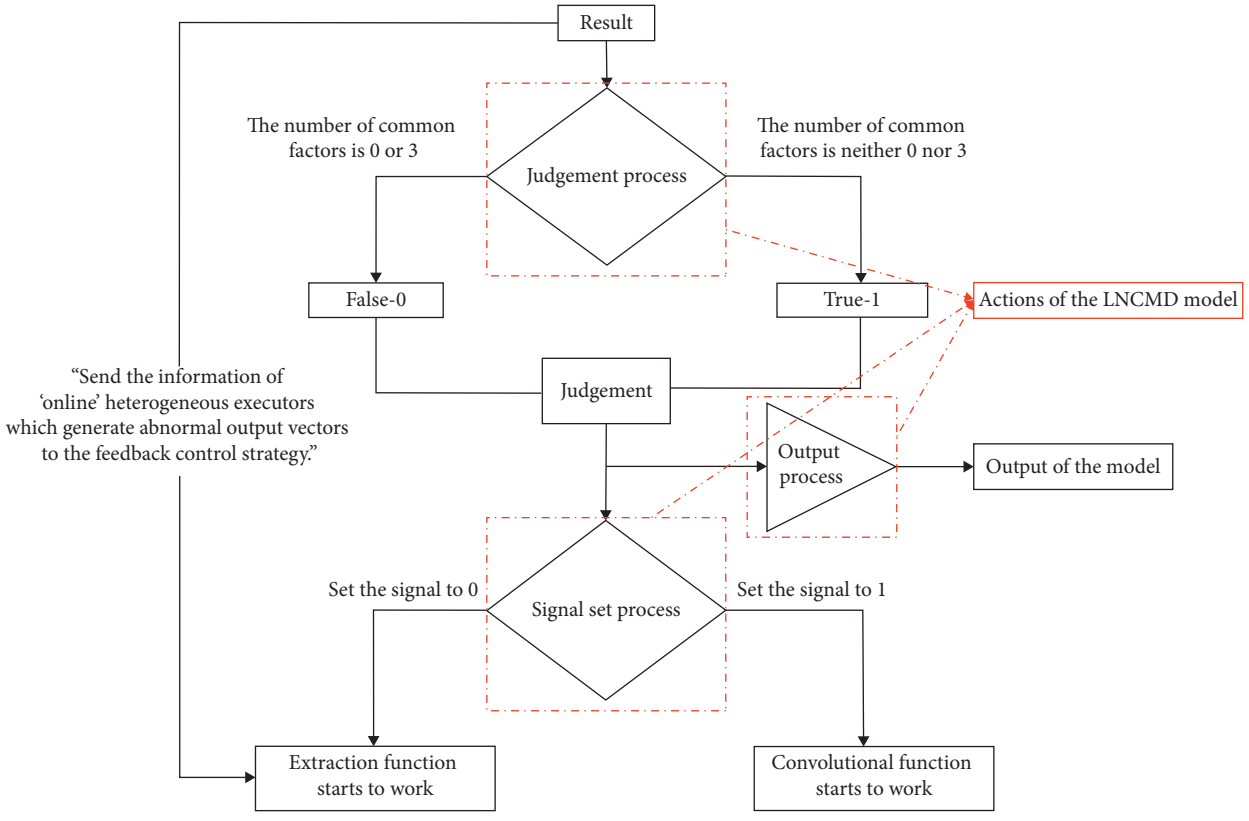

Figure 8: Flow of process highly dependent on the adjudication strategy.

Configuration 2: only the first "online" heterogeneous executor has a door, that is, $a < +\infty$ and $b\&c = +\infty$

Configuration 3: only the first "online" heterogeneous executor does not have a door, and the doors of the other two "online" heterogeneous executors are the same, which means that the flags that trigger the two doors are the same, that is, $a = +\infty$ and $(b = c) < +\infty$

Configuration 4: all the three "online" heterogeneous executors have the same doors, which means that the flags that trigger these doors are the same, that is, $(a = b = c) < +\infty$

Configuration 5: all the three "online" heterogeneous executors have different doors, which means that the flags that trigger these doors are unique, that is,

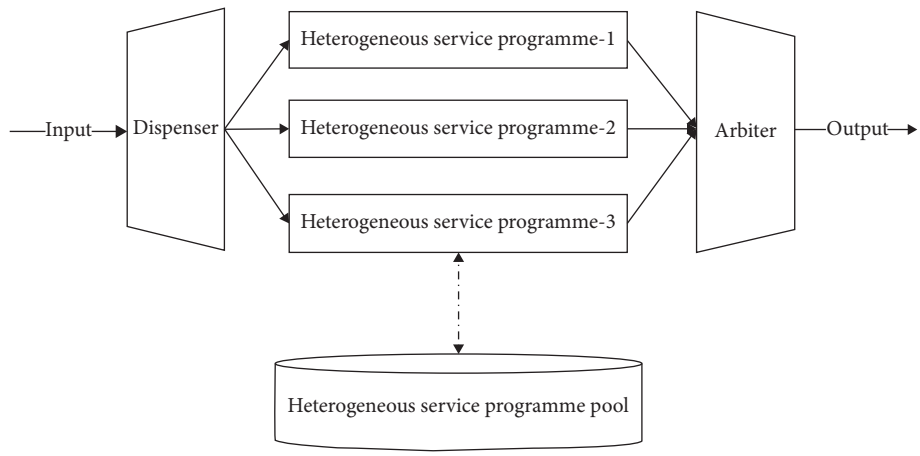

Figure 9: Physical background for the simulation experiment.

Table 4: Population of the LNCMD model.

Populated items	Content
Pool	The mapping values of the heterogeneous executors in the pool are $\{a, b, c, d, e, f, g\}$
Vector	The values in the vector are $[a, b, c]$
Product	The product P is $P = a \times b \times c$
Matrix	The matrix is as follows: The number of rows is 3, and no elements are visible
Signal	The value of the signal is 1
Input	The current status of the input is as follows: Waiting for the customer to write
Strategy	The system strategies are as follows: (1) Dynamic scheduling strategy: all the "online" heterogeneous executors are replaced randomly from the heterogeneous executor pool at fixed intervals (2) Feedback control strategy: the abnormal "online" heterogeneous executors are replaced randomly from the heterogeneous executor pool (3) Adjudication strategy: the aforementioned majority vote strategy is selected
Output	The current status of the output is not applicable

The populated LNCMD model is termed as lncmdDemo in this paper.

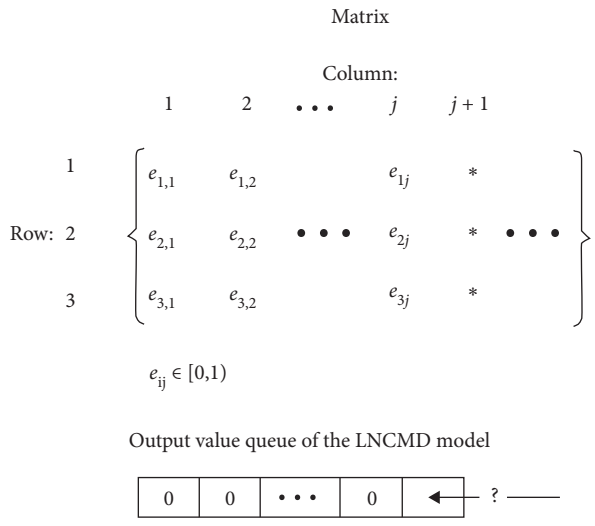

Figure 10: Simulation to observe lncmdDemo under configuration 1.

$$(a \neq b \neq c) < +\infty$$

5.1.2. Simulation Result. Let the output value queue of the LNCMD model be "Q." In this case, the simulation experiment of the LNCMD model can be experimentally observed through the "matrix" and "Q."

The matrix and Q corresponding to lncmdDemo in configuration 1 are shown in Figure 10.

In configuration 1, it is impossible for the attacker to cover the SAV on any "online" heterogeneous executor; that is, the attacker cannot hit any large prime factor (ordered) in P. The model output shows that lncmdDemo was never attacked.

The matrix and Q corresponding to lncmdDemo in configuration 2 are shown in Figure 11.

In configuration 2, the attacker can cover the SAV on only the first "online" heterogeneous executor; that is, the attacker may hit the first large prime factor in P. Once the hit

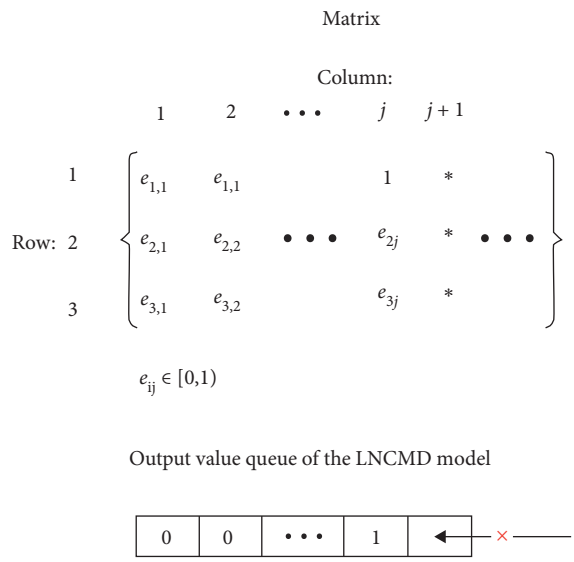

Figure 11: Simulation to observe lncmdDemo under configuration 2.

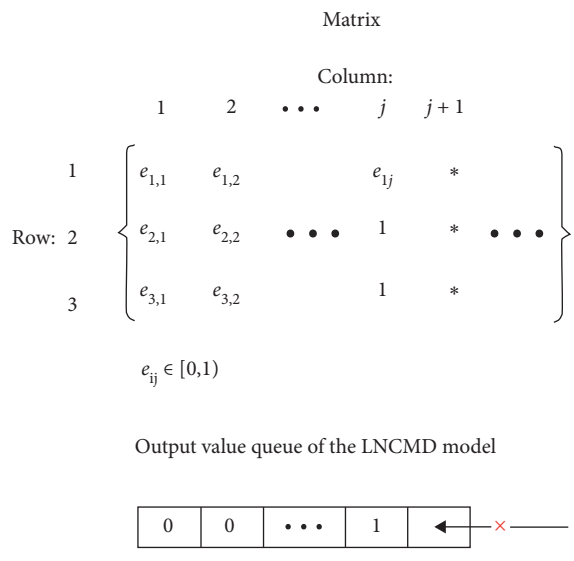

Figure 12: Simulation to observe lncmdDemo under configuration 3.

is successful, the model output will show that lncmdDemo is under attack and cause a "reset" operation.

The matrix and Q corresponding to lncmdDemo in configuration 3 are shown in Figure 12.

In configuration 3, the attacker cannot cover the SAV on the first "online" heterogeneous executor; that is, the attacker may hit the second and third large prime factors in P simultaneously. Once the hit is successful, the model output will show that lncmdDemo is under attack and cause a "reset" operation.

The matrix and Q corresponding to lncmdDemo in configuration 4 are shown in Figure 13.

In configuration 4, the attacker can cover the SAVs on all the "online" heterogeneous executors simultaneously; that is, the attacker may hit all the large prime factors (ordered) in P simultaneously. When the hit is successful, the model output will show that lncmdDemo was not attacked.

The matrix and Q corresponding to lncmdDemo in configuration 5 are shown in Figure 14.

The observation result of lncmdDemo under configuration 5 may be the same as that of any of the previous four configurations. If the attacker cannot cover the SAV on any "online" heterogeneous executor, lncmdDemo will behave as in configuration 1. If the attacker can only cover the SAV on one "online" heterogeneous executor, lncmdDemo will behave as in configuration 2. If the attacker can only cover the SAV on two "online" heterogeneous executors,

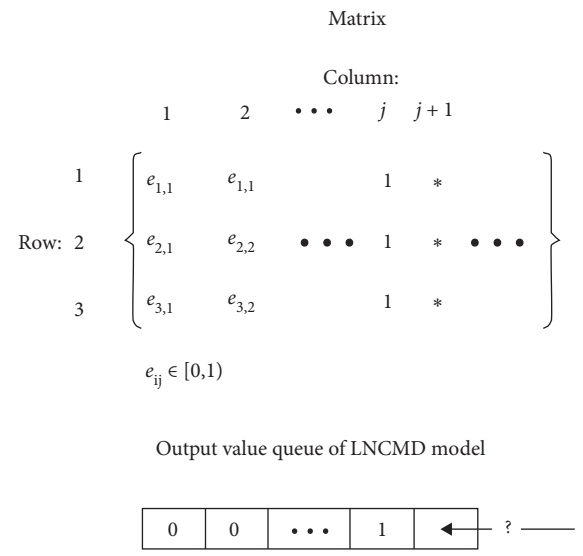

Figure 13: Simulation to observe lncmdDemo under configuration 4.

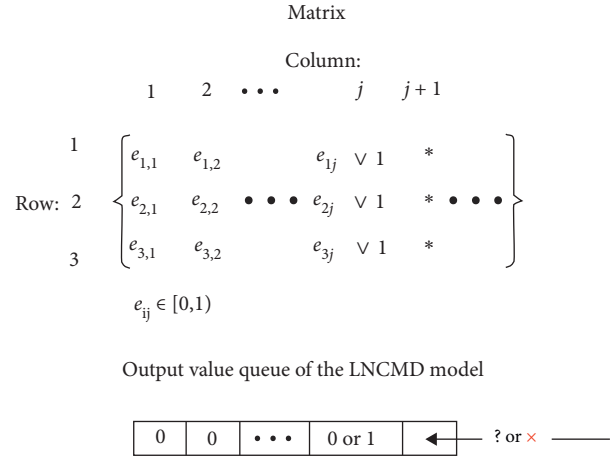

Figure 14: Simulation to observe lncmdDemo under configuration 5.

lncmdDemo will behave as in configuration 3. If the attacker can cover the SAVs on all the "online" heterogeneous executors, lncmdDemo will behave as in configuration 4.

5.1.3. Experimental Analysis and Evaluation. A part of the observation results of lncmdDemo under the five configurations is expected by the defender, and the remaining results are expected by the attacker. In this paper, the expected LNCMD model observation result for one side is termed as the "solution" of this side under the LNCMD model.

(1) For the defender:

(a) Analysis:
The simulation experiment indicates that the observation results of lncmdDemo under configurations 1, 2, and 5 are solutions for the defender.
The observation result of lncmdDemo in configuration 1 reflects that the CMD system always maintains a normal system output. Moreover, the feedback control is not triggered, and thus, no system turbulence is caused by the feedback control. The observation result of lncmdDemo in configuration 1 can be considered as the "optimal solution" for the defender under lncmdDemo.
The observation result of lncmdDemo in configuration 2 reflects that the CMD system always maintains a normal system output, although the feedback control may be triggered, which may induce the system turbulence caused by the feedback control. The observation result of lncmdDemo in configuration 2 can be considered as the "ordinary solution" for the defender.
The observation result of lncmdDemo in configuration 5 is unstable, that is, it may or may not be what the defender expects, and a luck component is involved. The observation result of

lncmdDemo in configuration 5 can be considered as the "worst solution" for the defender.

The observation results of lncmdDemo in configurations 3 and 4 do not correspond to the "solution" for the defender. At this time, the CMD system already exhibits a "joint escape" phenomenon.

(b) Evaluation:

Through the simulation experiment, it can be concluded that chasing the "optimal solution" under the LNCMD model is the ultimate goal of the defender. The defender's "optimal solution" under the LNCMD model can be attained by ensuring that the LNCMD model exhibits the following martingale characteristic.

Hypothesis:

Let the judgement of the attack by the judgement function excited by the j^{th} model input be a stochastic process Y_j.

In the j^{th} model input, let the probability that the attacker cannot decompose any large prime factor (ordered) from the value of the product variable correspond to a stochastic process X_j.

When a stochastic process X_j is used as a condition, it means that an event with X_j as the probability has occurred, so $0 < X_j \leq 1$.

Then, this martingale characteristic is

$$E\left(Y_{(j+1)} \mid X_1, \ldots, X_j\right) = Y_j = 0. \tag{13}$$

At this time, the stochastic process Y_j is a discrete martingale on X_j. Its meaning is that the attacker can never decompose any large prime factor (ordered) from the value of the product variable, which makes the judgement function always judge that the LNCMD model was not attacked. On this basis, for the future $(j+1)$-th model input, the conditional expectation of the event $Y_{(j+1)}$ is that the LNCMD model will not be attacked. In a practical sense, this scenario means that, in the CMD system, the attacker cannot cover the SAV on any "online" heterogeneous executor, and no system turbulence is caused by triggering the feedback control. When the LNCMD model has the above martingale characteristic, it reflects the defender's "optimal solution" under the LNCMD model.

Therefore, the problem of how the defender chases the "optimal solution" under the LNCMD model can be transformed into the problem of ensuring that the LNCMD model exhibits the aforementioned martingale characteristic. The defender considers how to adjust various system strategies to ensure that the LNCMD model exhibits the aforementioned martingale characteristic, which is the mathematical nature of the problem faced by the defender in the CMD system in the practical sense, as clarified by the LNCMD model.

(2) For the attacker:

(a) Analysis:

The simulation experiment indicates that the observation results of lncmdDemo under configurations 3, 4, and 5 are solutions for the attacker.

The observation result of lncmdDemo in configuration 4 reflects that the CMD system has a "joint escape" phenomenon; however, it is impossible to perceive the occurrence of the attack. The observation result of lncmdDemo in configuration 4 can be considered as the "optimal solution" for the attacker.

The observation result of lncmdDemo in configuration 3 reflects that the CMD system exhibits a "joint escape" phenomenon, and the occurrence of the attack can be perceived. The observation result of lncmdDemo in configuration 3 can be considered as the "ordinary solution" for the attacker.

The observation result of lncmdDemo in configuration 5 is unstable, that is, it may or may not be what the attacker expects, and a luck component is involved. The observation result of lncmdDemo in configuration 5 can be considered as the "worst solution" for the attacker.

The observation results of lncmdDemo in configurations 1 and 2 do not correspond to the attacker's "solution." In this scenario, the CMD system always maintains a normal system output.

(b) Evaluation:

The simulation experiment indicated that chasing the "optimal solution" under the LNCMD model is the ultimate goal of the attacker. The attacker wants to obtain the "optimal solution" under the LNCMD model depending on the composition of the large prime factors in the value of the product variable and the hit method for each large prime factor (ordered).

The following simulation experiment is considered as an example:

The large prime factors contained in P for lncmdDemo are a, b, and c. Under configurations 1, 2, and 3, the compositions of the large prime factors in P are $(a\&b\&c = +\infty)$, $(a < +\infty$ and $b\&c = +\infty)$, and $(a = +\infty$ and $(b = c) < +\infty)$, respectively. Under these compositions of the large prime factors, the attacker cannot obtain the "optimal solution" under lncmdDemo, regardless of the hit method employed by the attacker. Under configuration 4, the composition of the large prime factors in P is $(a = b = c) < +\infty$. Under this composition of the large prime factors, the attacker can obtain the "optimal solution" under lncmdDemo. At this time, the hit method used by the attacker determines the rate at

which the "optimal solution" is attained under lncmdDemo. Under configuration 5, the composition of the large prime factors in P is $(a \neq b \neq c) < +\infty$. Under this composition of the large prime factors, the attacker can attain the "optimal solution" under lncmdDemo. At this time, the hit method used by the attacker determines whether the attacker can obtain the "optimal solution" under lncmdDemo.

5.2. Security of CMD. The evaluation of the network security is different from that of the information system performance. The former evaluation is more difficult to describe quantitatively compared to the latter evaluation [48]. Therefore, by modelling the CMD mechanism and using a mathematical model to express the CMD mechanism, this paper establishes a connection between the CMD mechanism and mathematics to ensure that the safety of the CMD can be qualitatively evaluated based on the LNCMD model. The following section describes the qualitative evaluation of the safety of the CMD based on the LNCMD model.

The LNCMD model uses a large prime factor product to represent the "online" heterogeneous executor set of the CMD system at a certain time, and it uses the large prime factor decomposition problem to map an attacker's attack on the CMD system. The concept of the LNCMD model is applied, and the traditional static and single system is represented as one large prime number. For the convenience of the subsequent description, the aforementioned large prime factor product and large prime factor are, respectively, represented as "composite" and "pfactor$_i$" ($1 \leq i \leq n$, n is the number of factors), and the aforementioned large prime number is represented as "prime." At the same time, the "ray" that starts from 0 and grows to $+\infty$ is used to represent a dimension. Considering these aspects, "composite" and "pfactor$_i$" are shown in the upper part of Figure 15, and "prime" is shown in the lower part of Figure 15.

When the "composite" is determined, the attacker must analyse "pfactor$_i$s" that the "composite" comprises. The "composite" is only on a one-dimension "ray," and each "pfactor$_i$" that composes the "composite" is also on a one-dimension "ray." Therefore, a mapping relationship from a multidimensional "ray" to a one-dimension "ray" is formed between the "composite" and "pfactor$_i$." However, the attacker must also analyse this mapping from the multidimensional "ray" to the one-dimension "ray." Furthermore, for "prime," the attacker only needs to analyse the specific value of the "prime." Because the "prime" does not have a multidimensional mapping relationship, the attacker only needs to analyse on a one-dimension "ray."

To impede the attacker from performing this analysis, the complexity can be considered as a general approach. The complexity for a "composite" can be improved by increasing the number or value of "pfactor$_i$," and these two aspects can be combined. For "prime," the complexity can only be improved by increasing the value of "prime." However, according to the CMD mechanism, the "composite" is not immutable. Through the multidimensional dynamic driving one-dimensional dynamic, the "composite" can implement active changes based on the above two aspects or passive changes caused by the attacker's analysis. Regardless of the active or passive changes in the "composite," all the previous efforts of the attacker may be wasted, thereby greatly increasing the complexity on the original basis and rendering the analysis to be conducted by the attacker more difficult.

Next, an arithmetic analysis is carried out to intuitively reflect this complexity relationship. We use time to measure the above complexity, denoted as T. Assuming that the redundancy scale of the "online" heterogeneous executors in the CMD system is three, then there are "pfactor$_1$," "pfactor$_2$," and "pfactor$_3$." At the same time, "pfactor$_i$s " ($1 \leq i \leq 3$) are different from each other; then, their complexity is t_1, t_2, and t_3, respectively. We take the dynamic characteristic of CMD as a weight, denoted as w, w tends to $+\infty$. For the traditional static and single system, let the complexity of "prime" be t. The following formula gives this complexity relationship:

$$\begin{aligned} T(\text{Traditional}) &= t, \\ T(\text{CMD\_Static}) &= t_1 + t_2 + t_3, \\ T(\text{CMD}) &= t_1 \times w + t_2 \times w + t_3 \times w \\ &= (t_1 + t_2 + t_3) \times w \approx +\infty. \end{aligned} \tag{14}$$

For the above complexity, T(Traditional) refers to the traditional static and single system, T(CMD_Static) refers to the CMD system without the dynamic characteristic, and T(CMD) refers to the CMD system. For t, t_1, t_2, and t_3, they should be in the same order of magnitude, but there are differences in size. It is not difficult to see that the complexity of CMD far exceeds the traditional static and single system.

Therefore, based on the analysis of the aforementioned arithmetic significance, this paper performs a qualitative assessment of the safety of the CMD. In contrast to the traditional static and single system, the CMD system raises the difficulty level of guessing a single large prime number to the difficulty level of solving the decomposition problem of the dynamic large prime factor product. Therefore, the CMD is highly secure.

5.3. Overhead of CMD. While the use of CMD technology brings high security and high robustness, additional system overhead is inevitable because the use of any technology has to pay a certain price, but it is entirely possible to control these additional overheads within an acceptable range by certain means. The additional overheads brought by CMD can be analyzed from the following two aspects:

(1) Complexity Overhead. We use space complexity to measure the complexity overhead and use the CMD system to compare with the traditional static and single system. Here, the space complexity is expressed in terms of quantity, and a unit quantity is $O(1)$.

The complexity overhead of the CMD system is

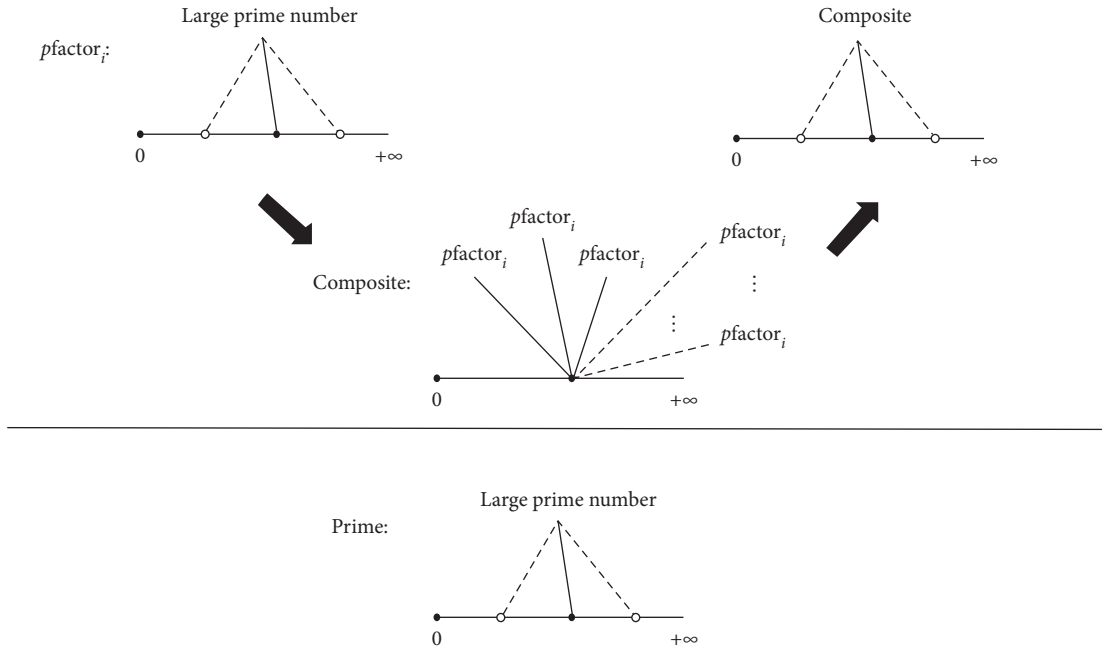

Figure 15: Diagram of "composite & $p\text{factor}_i$ and prime."

$$\begin{cases} O(E) = O(e_1) + O(e_2) + \cdots + O(e_n), \\ S(\text{CMD}) = O(d) + O(E) + O(a) + O(c) + \Delta. \end{cases} \tag{15}$$

$O(E)$ is the space complexity of the heterogeneous executor pool. $O(e_i)$ is the space complexity of the heterogeneous executor. Assuming that there are a total of n heterogeneous executors in the heterogeneous executor pool, then $O(E)$ is the sum of each $O(e_i)$ $(1 \le i \le n)$. $O(d)$ is the space complexity of the input distributor. $O(a)$ is the space complexity of the output arbiter. $O(c)$ is the space complexity of the CMD converter. Δ is the extra space complexity. For example, when the CMD system includes the "cleaning" function for abnormal executors, this part of the space complexity belongs to Δ. Ultimately, the complexity overhead of the CMD system is $S(\text{CMD})$.

The complexity overhead of the traditional static and single system is

$$S(\text{Traditional}) = O(e). \tag{16}$$

$O(e)$ is the space complexity of the executor that achieves the target function; then, the complexity overhead of the traditional static and single system is $S(\text{Traditional})$.

So far, it can be concluded that the additional complexity overhead of the CMD system compared to the traditional static and single system is $S(\text{CMD}) - S(\text{Traditional})$. For the executors in these two types of systems, their space complexity is similar, that is, $O(e) \approx O(e_i)$ $(1 \le i \le n)$.

(2) Performance Overhead. We use the time complexity to measure the performance overhead and use the CMD system to compare with the traditional static and single system. Here, the time complexity is expressed in terms of quantity, and a unit quantity is $O(1)$.

The performance overhead of the CMD system is

$$\begin{cases} O(E) = \max(O(e_1), O(e_2), \ldots, O(e_n)), \\ T(\text{CMD}) = O(d) + O(E) + O(a) + O(c) + \Delta. \end{cases} \tag{17}$$

$O(E)$ is the time complexity of the "online" heterogeneous executor set. $O(e_i)$ is the time complexity of the "online" heterogeneous executor. Assuming that there are a total of n "online" heterogeneous executors in the "online" heterogeneous executor set, because the "online" heterogeneous executors are executed in parallel, $O(E)$ is the maximum value of all $O(e_i)$ $(1 \le i \le n)$. $O(d)$ is the time complexity of the input distributor. $O(a)$ is the time complexity of the output arbiter. $O(c)$ is the time complexity of the CMD converter. Δ is the extra time complexity. For example, when the CMD system includes the "cleaning" function for abnormal executors, this part of the time complexity belongs to Δ. Ultimately, the performance overhead of the CMD system is $T(\text{CMD})$.

The performance overhead of the traditional static and single system is

$$T(\text{Traditional}) = O(e). \tag{18}$$

$O(e)$ is the time complexity of the executor that achieves the target function; then, the performance overhead of the traditional static and single system is $T(\text{Traditional})$.

So far, it can be concluded that the additional performance overhead of the CMD system compared to the traditional static and single system is $T(\text{CMD}) - T(\text{Traditional})$. For the executors in these two types of systems, their time complexity is similar, that is, $O(e) \approx O(e_i)$ $(1 \le i \le n)$.

For the additional overheads of using CMD technology, it is necessary to reduce them to an acceptable range. Referencing the aforementioned various complexities to

optimize the system implementation is the first method, and focusing on the rationality of using CMD technology is the second method. In terms of rationality, assuming that CMD technology is used to protect data, the overheads of only using CMD technology to protect a small amount of critical data are far less than using CMD technology to protect ordinary mass data, but the security will not differ too much. For example, using CMD technology to protect the access control list (ACL) in the firewall, the overhead caused by encryption operation is acceptable. At the same time, because ACL is the critical data, the overall security of the system will also be greatly improved.

6. Conclusion and Future Work

This paper proposes a large-number convolutional mimic defence mathematical model. The LNCMD model is an intuitive and exclusive mathematical model of the CMD. The LNCMD model transforms the problems of the attack and defence game of the CMD into corresponding mathematical problems. For the defender, the LNCMD model transforms the problem of how the defender uses the CMD for security protection into the problem of how the defender adjusts various system strategies to ensure that the LNCMD model has a specific martingale characteristic. For the attacker, the LNCMD model innovatively transforms the problem of the attacker attacking the CMD system into the problem of the attacker factorising the large prime factor product. Therefore, based on the LNCMD model, this paper performs a qualitative assessment that indicates that the CMD is highly secure. The proposed LNCMD model can be implemented directly through programming, and the subsequent step is to programme the LNCMD model to further examine the key technologies of the CMD framework.

Data Availability

The simulation data used to support this study are included within this article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (no. 2016YFB0800100).

References

[1] T. D. Braun, H. J. Siegel, N. Beck et al., "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," *Journal of Parallel and Distributed Computing*, vol. 61, no. 6, pp. 810–837, 2001.

[2] M. Armbrust, A. Fox, R. Griffith et al., "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[3] R. C. Newman, "Cybercrime, identity theft, and fraud: practicing safe internet - network security threats and vulnerabilities," in *Proceedings of the 3rd Annual Conference on Information Security Curriculum Development*, pp. 68–78, Kennesaw, Georgia, 2006.

[4] G. Cai, B. Wang, T. Wang, Y. Luo, and X. Cui, "Research and development of moving target defense technology," *Journal of Computer Research & Development*, vol. 53, no. 5, pp. 968–987, 2016.

[5] X. Luo, Q. Tong, Z. Zhang, and J. Wu, "Mimic defense technology," *Strategic Study of Chinese Academy of Engineering*, vol. 18, no. 6, pp. 69–73, 2016.

[6] L. Wang, J. Tao, M. Kunze, A. C. Castellanos, and W. Karl, "Scientific cloud computing: early definition and experience," in *Proceedings of the the 10th IEEE International Conference on High Performance Computing and Communications*, pp. 825–830, Dalian, China, September 2008.

[7] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux Journal*, vol. 239, 2014.

[8] J. Wu, *Introduction to Cyberspace Mimic Defense*, Science Press, Beijing,China, 2017.

[9] J. Wu, *Principle of Cyberspace Mimic Defense*, Science Press, Beijing, China, 2018.

[10] X. Si, W. Wang, J. Zeng et al., "A review of the basic theory of mimic defense," *Engineering Sciences*, vol. 18, no. 6, pp. 62–68, 2016.

[11] J. Wu, "Research on cyber mimic defense," *Journal of Cyber Security*, vol. 1, no. 4, pp. 1–10, 2016.

[12] D. Ouelhadj and S. Petrovic, "A survey of dynamic scheduling in manufacturing systems," *Journal of Scheduling*, vol. 12, no. 4, p. 417, 2009.

[13] K. Zhou, C. John, and Doyle, *Essentials of Robust Control*, p. 38, Prentice Hall, Upper Saddle River, NJ, USA, 1998.

[14] J. Wu, *Cyberspace Mimic Defense*, Springer, Switzerland Cham, Switzerland, 2020.

[15] G. F. Franklin, J. D. Powell, and E. Abbas, *Feedback Control of Dynamic Systems*, China Machine Press, Beijing, China, 2016.

[16] Q. Tong, Z. Zhang, W. Zhang, and J. Wu, "Design and implementation of mimic defense web server," *Journal of Software*, vol. 28, no. 4, pp. 883–897, 2017.

[17] Z. Zhang, B. Ma, and J. Wu, "The test and analysis of prototype of mimic defense in web servers," *Journal of Cyber Security*, vol. 2, no. 1, pp. 13–28, 2016.

[18] H. Ma, P. Yi, Y. Jiang, and L. He, "Dynamic heterogeneous redundancy based router architecture with mimic defenses," *Journal of Cyber Security*, vol. 2, no. 1, pp. 29–42, 2016.

[19] H. Ma, Y. Jiang, B. Bai, and J. Zhang, "Tests and analyses for mimic defense ability of routers," *Journal of Cyber Security*, vol. 2, no. 1, pp. 43–53, 2017.

[20] X. Ji, K. Huang, L. Jin et al., "Overview on 5G security technology," *Mobile Communications*, vol. 43, no. 1, pp. 34–39+45, 2019.

[21] Z. Wang, H. Hu, and G. Cheng, "Design and implementation of mimic network operating system," *Journal of Computer Research and Development*, vol. 54, no. 10, pp. 2321–2333, 2017.

[22] Z. Gu, X. Zhang, and S. Lin, "Research on security mechanism for SDN control layer based on mimic defense theory," *Application Research of Computers*, vol. 35, no. 7, pp. 2148–2152, 2018.

[23] J. Pang, Y. Zhang, Z. Zhang, and J. Wu, "Applying a combination of mimic defense and software diversity in the

software security industry," *Engineering Sciences*, vol. 18, no. 6, pp. 74–78, 2016.
[24] X. Liu, X. Zou, J. Tan, and J. Wu, "Survey of large integer factorization algorithms," *Application Research of Computers*, vol. 31, no. 11, pp. 3201–3207, 2014.
[25] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.
[26] J. M. Pollard, "A Monte Carlo method for factorization," *Bit*, vol. 15, no. 3, pp. 331–334, 1975.
[27] J. M. Pollard, "Theorems on factorization and primality testing," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 76, no. 3, pp. 521–528, 1974.
[28] H. W. Lenstra, "Factoring integers with elliptic curves," *The Annals of Mathematics*, vol. 126, no. 3, pp. 649–673, 1987.
[29] C. Pomerance, "The quadratic sieve factoring algorithm," in *Proceedings of the of the Eurocrypt 84 Workshop on Advances in Cryptology: Theory & Application of Cryptographic Techniques*, Springer, Paris, France, April 1984.
[30] A. K. Lenstra and H. W. Lenstra, *The Development of the Number Field Sieve*, Springer-Verlag, Berlin, Germany, 1956.
[31] K. Pipyros, L. Mitrou, D. Gritzalis et al., "A cyber attack evaluation methodology," in *Proceedings of the 13th European Conference on Cyber Warfare and Security*, pp. 264–270, Piraeus, Greece, 2014.
[32] Z. Shi, G. Zhao, and J. Liu, "The effect evaluation of the network attack based on the fuzzy comprehensive evaluation method," in *Proceedings of the International Conference on Systems & Informatics*, November 2016.
[33] G. Tuvell, C. Jiang, and S. Bhardwaj, "Off-line mms malware scanning system and method," 2008.
[34] A. Orebaugh, *Ethereal Packet Sniffing*, Syngress Publishing, Amsterdam, Netherlands, 2004.
[35] B. Schneier, "Attack trees: modeling security threats," *Dobb's Journal*, vol. 24, no. 12, pp. 4–6, 1999.
[36] L. P. Swiler and C. Phillips, "A graph-based system for network-vulnerability analysis," in *Proceedings of the Workshop on New Security Paradigms*, pp. 71–79, New York, NY, USA, 1998.
[37] J. P. Mcdermott, "Attack net penetration testing," in *Proceedings of the Workshop on new security paradigms*, ACM, New York, NY, USA, 2001.
[38] P. K. Manadhata and J. M. Wing, "An attack surface metric," *IEEE Transactions on Software Engineering*, vol. 37, no. 3, pp. 371–386, 2011.
[39] M. Howard, J. Pincus, and J. M. Wing, "Measuring relative attack surfaces," in *Computer Security in the 21st Century*, pp. 109–137, Springer, Berlin, Germany, 2005.
[40] P. K. Manadhata, "Game theoretic approaches to attack surface shifting," *Moving Target Defense II*, Springer, New York, NY, USA, 2013.
[41] S. Jajodia, A. K. Ghosh, V. Swarup, C. Wang, and X. S. Wang, *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats*, Springer, Berlin, Germany, 2011.
[42] W. Stallings, *Network and Internetwork Security: Principles and Practice. Network and Internetwork Security: Principles and Practice*, Prentice-Hall, Upper Saddle River, NJ, USA, 1995.
[43] S. Lin, Q. Liu, and X. Wang, "Competitive arbitration model for mimic defense system," *Computer Engineering*, vol. 44, no. 4, pp. 193–198, 2018.
[44] W. Li, Z. Zhang, L. Wang, and J. Wu, "The modeling and risk assessment on redundancy adjudication of mimic defense," *Journal of Cyber Security*, vol. 3, no. 5, pp. 64–74, 2018.
[45] X. Zhang, Z. Gu, S. Wei, and J. Shen, "Markov game modeling of mimic defense and defense strategy determination," *Journal on Communications*, vol. 39, no. 10, pp. 143–154, 2018.
[46] Q. Ren, L. He, and J. Wu, "Analysis of different anti-interference system models based on discrete time markov chain," *Chinese Journal of Network and Information Security*, vol. 4, no. 4, pp. 30–37, 2018.
[47] S. E. Chang and C. B. Ho, "Organizational factors to the effectiveness of implementing information security management," *Industrial Management & Data Systems*, vol. 106, no. 3, pp. 345–361, 2006.
[48] J. Zhang, J. Pang, and Z. Zhang, "Quantification method for heterogeneity on web server with mimic construction," *Journal of Software*, vol. 31, no. 2, pp. 564–577, 2020.