Securing the Intermediate Data of Scientific Workflows in Clouds with ACISO

Yawen Wang¹, Yunfei Guo¹, Zehua Guo², Wenyan Liu¹, Chao Yang¹
¹National Digital Switching System Engineering and Technological Research Center, Zhengzhou 450002, China
²Beijing Institute of Technology, Beijing 100081, China

Corresponding author: Zehua Guo (guolizihao@hotmail.com).

This work was supported by the National Key Research and Development Program of China under Grant 2018YFB0804004 and 2018YFB1003700, the Foundation for Innovative Research Groups of the National Natural Science Foundation of China under Grant 61521003, the Beijing Natural Science Foundation under Grant Z170003 and the Beijing Institute of Technology Research Fund Program for Young Scholars.

ABSTRACT Many scientific workflow applications are moving to clouds. A scientific workflow is a complicated scientific computing task consisting of many sub-tasks, and each sub-task execution can generate the intermediate data used for the successor sub-task execution. The correct execution of scientific workflows depends on the security of the intermediate data, which is transmitted frequently between virtual machines during the process of the workflow execution. In multi-tenant clouds, the intermediate data contains three attributes: availability, confidentiality and integrity. If the intermediate data is lost, stolen, or tampered with by malicious tenants, the intermediate data’s attribute will be damaged, causing workflow interruption, the leakage of secret information or incorrect workflow results. For these problems, we propose ACISO scheme to secure the intermediate data by improving its availability, confidentiality, and integrity. In the scheme, availability, confidentiality and integrity strategy pools are constructed by various erasure codes, encryption algorithms and hash functions, respectively. Then we present a security strategy optimal allocation model named SSOA, which aims to maximize the overall intermediate data security strength while meeting the constraints of the workflow makespan and storage overhead. Normally, a scientific workflow contains a large number of the intermediate data, so solving this model is NP hard. Therefore, we propose a heuristic solution to solve SSOA. The simulation results show that ACISO can effectively improve the availability, confidentiality, and integrity of the intermediate data of the scientific workflows.

INDEX TERMS Cloud security, intermediate data security, security strategy allocation, scientific workflows
workflow sub-task to VM mapping is produced in advance and executed once. Dynamic task scheduling algorithms make workflow sub-task to VM assignment decisions at runtime. These decisions are based on the current state of the system and the workflow execution. The main advantage of static task scheduling algorithms is their ability to generate high-quality schedules by using global workflow-level, optimization techniques and to compare different solutions before choosing the best suited one [39]. But the adaptability of static algorithms is worse than dynamic algorithms.

The task scheduling has many benefits but also brings some security threats. The task scheduling makes the intermediate data that sub-tasks depend on to be frequently transferred between different VMs, in multi-tenant clouds, it provides good chances for malicious users to attack against these data. The security requirements for the intermediate data in multi-tenant clouds contain three types: availability, confidentiality, and integrity [14].

Typically, the cloud based scientific workflow system usually preserves the workflow intermediate data because of three reasons. First, scientists may need to re-analyze the results or apply new analyses on the intermediate data [2]. Second, for collaboration, the intermediate results are shared among scientists from different institutions, and the intermediate data can be reused [2]. Third, the preserved intermediate data can quickly restore the workflow execution status when the workflow is abnormally interrupted [24]. The cloud based scientific workflow system usually uses VMs to store the intermediate data [39]. If these VMs fail, the stored intermediate data will be lost. Therefore, the intermediate data availability requires that no intermediate data lost even if some VMs fail. The intermediate data confidentiality requires that even if a malicious attacker gets the intermediate data, the content cannot be obtained. The intermediate data integrity requires that the system can detect whether the transmitted intermediate data has been tampered with by others.

A typical method to enhance intermediate data availability is to use the erasure coding [37]. However, erasure coding will increase the size of the intermediate data. The improvement of intermediate data confidentiality can be achieved by applying encryption algorithms. Normally, the efficiency of algorithm with high encryption strength is low. For example, the IDEA algorithm [15] has a higher encryption strength than RC4 algorithm [16], but the encryption efficiency of IDEA is lower [17]. Hash functions can be used for protecting intermediate data integrity. The principle of hash functions is to map a large space to a small space, which will inevitably lead to conflicts. The more security strength of the applied hash functions, the lower probability of conflicts, the lower processing efficiency. The increased size of data, data encryption time and hash processing time necessarily result in execution delays for sub-tasks relying on these intermediate data.

In this paper, first, we construct availability, confidentiality and integrity strategy pools based on various erasure codes, encryption algorithms and hash functions. Then on the basis of the strategy pools, we propose availability, confidentiality and integrity strategy optimization (ACISO) scheme to enhance these three attributes of the intermediate data. ACISO scheme is applied after the workflow system gives the task scheduling strategy, and our work is limited to the scenario where static task scheduling algorithms are used.

The main contributions are summarized as follows:

- We jointly consider the attributes (i.e., availability, confidentiality, and integrity) of the intermediate data of scientific workflows, and use various erasure codes, encryption algorithms, and hash functions to build availability, confidentiality and integrity strategy pools, respectively. Then the security strategies are dynamically selected to protect the intermediate data.
- We transform the intermediate data security problem into the security strategy optimized allocation (SSOA) problem and formulate the SSOA, which aims to maximize the overall intermediate data security strength under the constraints of the workflow makespan and storage overhead. Since solving the SSOA is NP-hard [18], we propose a heuristic solution based on the PSO. Considering traditional PSO could fall into local optimal results, we improve the velocity update formula of PSO to balance the global and local search capabilities of particles.
- We evaluate the performance of ACISO. The simulation is conducted with WorkflowSim toolkit [19] and several actual scientific workflows. The simulation results demonstrate that ACISO can effectively strengthen the availability, confidentiality and integrity attributes of the intermediate data of scientific workflows.

The paper is organized as follows: Section 2 introduces the execution process and threats of the scientific workflows in clouds. Section 3 presents the security strategies for the intermediate data. Section 4 presents ACISO scheme. Section 5 explains simulations. Section 6 discusses the related work, followed by Section 7 that concludes this work.

II. BACKGROUND AND MOTIVATION

A. THE EXECUTION OF A SCIENTIFIC WORKFLOW IN CLOUDS

A workflow is often modeled as a DAG [4], which can be represented as $W = (A, D)$, where $A = \{a_1, a_2, \cdots, a_m\}$ denotes the set of sub-tasks making up the workflow, $a_i \in A$ denotes $i$-th sub-task in the workflow $W$. $D = \{d_1, d_2, \cdots, d_n\}$ denotes the set of intermediate data between workflow sub-tasks, $d_j \in D$ denotes $j$-th intermediate data in the workflow $W$. pred$(a_i)$ represents all the predecessor sub-tasks of $a_i$, the sub-task $a_i$ can only start to be executed
after collecting the intermediate data from all predecessor sub-tasks.

A typical scientific workflow system in clouds consists of three layers: presentation layer, workflow management layer, and operational layer [20]. Fig. 1 shows the structure of a typical scientific workflow system in clouds.

Users design their workflows via the visual interface in presentation layer and submit the workflow to workflow management layer, which is responsible for managing and monitoring workflows. The workflow engine is a central subsystem enabling workflow execution [41], which firstly translates each submitted workflow into an internal executable workflow representation [21]. Then workflow engine selects the sub-task for executing and assigns it to virtual machines (VMs). Exceptions (e.g., workflow execution interruptions caused by resource failures [10]) may occur, so the execution state of each sub-task will be monitored in the workflow management layer. Furthermore, scientists may need to re-analyze the intermediate data of scientific workflows [2], so the generated intermediate data should be managed in the workflow management layer.

In the operational layer, workflow sub-tasks are executed in VMs. As shown in Fig. 1, sub-tasks \( a_1 \), \( a_2 \) and \( a_3 \) are placed in VM 1, and sub-task \( a_4 \) is placed in VM \( n \). Since sub-tasks \( a_2 \) and \( a_4 \) are placed in different VMs, the intermediate data \( d_2 \) between \( a_2 \) and \( a_4 \) will be transmitted from VM 1 to VM \( n \) through the network.

**FIGURE 1.** The structure of a typical scientific workflow system in clouds [20]

**B. THE THREAT ANALYSIS OF WORKFLOW INTERMEDIATE DATA**

Task scheduling is the key step of the scientific cloud workflow execution, which can directly affect the system load and performance [22]. The task scheduling of the scientific cloud workflow is realized by transmitting the intermediate data between VMs. The cloud computing system is a complex computing environment where multiple tenants coexist. When the data is placed in a VM, it is difficult for malicious users to obtain this data due to the firewall and access rights. But when the data is being transmitted through the link, the data will be exposed to the malicious users.

The workflow intermediate data in clouds could experience three typical threats: data loss, traffic eavesdropping, and malicious media [23]. We explain the three threats as follows:

1. **Data loss:** workflow intermediate data is usually stored in VMs. If these VMs fail, the stored intermediate data will be lost, which will threaten the availability of the intermediate data.

2. **Traffic eavesdropping:** it could help the adversaries illegally collect information transmitted through the network. Many workflows belong to important scientific computing tasks, such as high-energy physics, bioinformatics, atmospheric science, and so on [25]. Therefore, workflow intermediate data usually involves core secrets in some areas, it will cause huge losses if the data information is stolen. This attack will threaten the confidentiality of intermediate data.

3. **Malicious medium:** it refers to the action of intercepting and tampering with the data during the transmission process, some adversaries even insert harmful data to impair the data security. This attack will threaten the integrity of the intermediate data.

**III. SECURITY STRATEGIES FOR THE INTERMEDIATE DATA**

In this section, we will introduce the security strategies which can strengthen the availability, confidentiality and integrity attributes of the intermediate data of scientific workflows.

**A. OVERVIEW OF THE SECURITY STRATEGY APPLICATION**

Traditional workflow model does not contain the information about the security. In order to contain the information about security requirements, we extend traditional workflow model \( W = (A, D) \) to \( W = (A, D, S) \), where \( S = \{sr_{as}, sr_{cs}, sr_{ls}\} \) represents the security requirement of users for the intermediate data of the workflow \( W \), where \( sr_{as} \), \( sr_{cs} \) and \( sr_{ls} \) are availability requirement, confidentiality requirement and integrity requirement, respectively. The minimum and maximum values for the three types of security requirements are 0 and 1, respectively.

After the user’s workflow execution request is submitted to the cloud platform, the cloud platform will formulate a scheduling strategy according to the workflow parameters including sub-task runtime, dependent relationship between sub-tasks and the size of the intermediate data. It is assumed that there is a workflow consists of sub-tasks \( a_1, a_2, a_3, \ldots \). \( a_2 \) depends on intermediate data \( d_1 \) generated from \( a_1 \), and \( a_3 \) depends on intermediate data \( d_2 \) generated from \( a_1 \), as shown in Fig. 2. If the task scheduling strategy formulated by cloud systems is that sub-task \( a_1 \) and \( a_2 \) are placed into VM A, and sub-task \( a_3 \) is placed into VM B, we call...
intermediate data $d_2$ as “mobile data”, since the sub-tasks relying on this data are in different VMs, the data needs to be moved to perform subsequent sub-tasks. We call intermediate data $d_1$ as “fixed data”, since this data and the sub-tasks relying on it are in the same VM.

![Diagram](image)

**FIGURE 2. The overview of the security strategy application**

For fixed data, the main consideration is its availability attribute, which needs to be partitioned by availability strategies, and each data block is sent to a different VM for storage. In order to ensure the confidentiality and integrity of the workflow intermediate data during the transmission, each data block is encrypted with a different key, and then the hash value is calculated and stored in the metadata. The costs of the fixed data, which are brought by the security strategies, are primarily storage costs.

For mobile data, availability, confidentiality and integrity attributes need to be considered together. On the one hand, similar to fixed data, the mobile data will be divided into multiple data blocks and sent to multiple VMs for backup. On the other hand, these data blocks need to be sent to destination VM for subsequent sub-task execution. After the data has been reached to the destination VM, hash check is first performed to verify the integrity, then the data is decrypted, and finally the erasure decoding is performed to recover the data. Subsequent sub-tasks can only start to be executed after the data is successfully reached to the destination VM. Therefore, the costs of the mobile data, which are brought by the security strategies, are primarily time costs and storage costs.

The introduction of the availability strategy not only avoids the data loss caused by VM failures, but also further improves the confidentiality and integrity of the intermediate data. For the confidentiality, an attacker needs to crack multiple data blocks to recover the original data. For the integrity, as long as the fault tolerance is not exceeded, even if an attacker has tampered with some data blocks, it does not affect the correct data recovery.

However, using one kind of security strategy cannot balance the intermediate data security and workflow execution efficiency. Different security strategies work for different workflow intermediate data. Because there must be multiple paths in the workflow, and the workflow makespan is decided by the length of the longest path [43]. The performance of a security strategy is usually proportionally to its processing time. Therefore, the intermediate data in the long path requires efficient security strategies with short processing time, since the time overhead generated in the long path has a large impact on workflow makespan. While the intermediate data in the short path can use a high secure strategies because the time overhead generated in the short path has a small impact on the whole workflow makespan.

Therefore, we propose to respectively construct the availability, confidentiality, and integrity strategy pools with various erasure codes, encryption algorithms, and hash functions.

**B. INTERMEDIATE DATA AVAILABILITY STRATEGY POOL**

We have discussed that a VM failure can cause all the intermediate data stored in it to be lost. A common idea is to improve data fault tolerance through data redundancy.
Therefore, we introduce erasure codes to design an intermediate data availability strategy.

Erasure codes are often used to ensure high availability of data in cloud storage systems [37]. \((n,k)\) erasure code can divide a data file into \(k\) equal-length data blocks in bytes. Then \(n\) \((m=n-k)\) check data blocks with equal length are generated by encoding. Afterwards, these \(n\) data blocks are sent to different equipment for storage. Any \(k\) of \(n\) data blocks can be used to restore the original data file. We use \(D_1, D_2\) and \(D_3\) to represent original data blocks, and use \(C_1\) and \(C_2\) to represent check data blocks. Fig. 3 (a) shows the process of erasure code encoding. In this figure, \(X\) is coding matrix, \(Y\) is the original data, and \(Z\) is the encoded data block. \(B_1 D_1 + B_2 D_2 + B_3 D_3 = C_1, B_2 D_1 + B_2 D_2 + B_3 D_3 = C_2.\) Assume that \(D_1\) and \(C_1\) are lost, matrix \(X\) becomes \(X'\) by deleting the first and fourth lines. Fig. 3 (b) shows the process of erasure code decoding. In this figure, since \(X'\) is reversible, so the original data \(Y\) can be restored by \(X'^{-1} Z'.\)

**FIGURE 3.** The illustration of the principle of \((5,3)\) RS erasure codes (a) The process of erasure code encoding (b) The process of erasure code decoding

In this paper, we use 3 RS erasure codes with different parameters as intermediate data availability strategies, as shown in Table I. \(S^{as} = \{s_1^{as}, s_2^{as}, ..., s_{kn(AS)}^{as}\}\) is used to indicate the availability strategy pool, and \(s_k^{as} \in S^{as}\) denotes an availability strategy. \(N(AS)\) denotes the number of availability strategies in the pool.

We use the fault tolerance of the availability strategy as the security strength. In Table I, \(s_1^{as}\) is \((5,4)\) RS erasure code, which transforms the original data into 5 data blocks, and restore the original data with any 4 data blocks. Thus, the fault tolerance of \(s_1^{as}\) is 1. Similarly, the fault tolerances of \(s_2^{as}\) and \(s_3^{as}\) are 2 and 3, respectively. The normalized security strength of \(s_1^{as}, s_2^{as}\) and \(s_3^{as}\) are 1.00, 0.67, 0.33, respectively. After applying the availability strategies, the data size will increase. If \(s_1^{as}\) is applied, the encoded data size ratio is 5/4, increasing the storage overhead. In Table I, the AS Speed is the processing speed of the availability strategy, which combines encoding and decoding processes and can be calculated as follows:

\[
\text{AS Speed} = \frac{\text{data size}}{\text{encoding time + decoding time}}\]

**TABLE I**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Algorithm</th>
<th>Security strength</th>
<th>Data size ratio</th>
<th>AS Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>(s_1^{as})</td>
<td>IDEA</td>
<td>1.00</td>
<td>5/2</td>
<td>12.11 Mb/s</td>
</tr>
<tr>
<td>(s_2^{as})</td>
<td>DES</td>
<td>0.67</td>
<td>5/3</td>
<td>13.92 Mb/s</td>
</tr>
<tr>
<td>(s_3^{as})</td>
<td>Rijndael</td>
<td>0.33</td>
<td>5/4</td>
<td>15.03 Mb/s</td>
</tr>
</tbody>
</table>

**C. INTERMEDIATE DATA CONFIDENTIALITY STRATEGY POOL**

We have discussed that the attackers can steal secret data through traffic eavesdropping. For this problem, the intermediate data can be encrypted before transmission, then the workflow execution program decrypts the data to obtain the original content after receiving the data. In this way, even if an attacker has succeeded in intercepting the intermediate data, the original content cannot be obtained. There are many types of encryption algorithms, the security strengths and processing speeds of different encryption algorithms are different. In [18] the authors list the security strengths of 5 encryption algorithms and test their processing speeds, as shown in Table II. In Table II, the CS Speed is the processing speed of the confidentiality strategy, which combines encryption and decryption processes and can be calculated as follows:

\[
\text{CS Speed} = \frac{\text{data size}}{\text{encryption time + decryption time}}\]

The confidentiality strategy pool is represented by \(S^{cs} = \{s_1^{cs}, s_2^{cs}, ..., s_{kn(cs)}^{cs}\}\), \(s_k^{cs} \in S^{cs}\) indicates the confidentiality strategy, \(N(cs)\) denotes the number of confidentiality strategies in the pool.

**TABLE II**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Algorithm</th>
<th>Security strength</th>
<th>CS Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>(s_1^{cs})</td>
<td>IDEA</td>
<td>1.00</td>
<td>17.34 Mb/s</td>
</tr>
<tr>
<td>(s_2^{cs})</td>
<td>DES</td>
<td>0.90</td>
<td>18.21 Mb/s</td>
</tr>
<tr>
<td>(s_3^{cs})</td>
<td>Rijndael</td>
<td>0.64</td>
<td>39.88 Mb/s</td>
</tr>
<tr>
<td>(s_4^{cs})</td>
<td>Blowfish</td>
<td>0.36</td>
<td>39.96 Mb/s</td>
</tr>
<tr>
<td>(s_5^{cs})</td>
<td>RC4</td>
<td>0.30</td>
<td>87.07 Mb/s</td>
</tr>
</tbody>
</table>

**D. INTERMEDIATE DATA INTEGRITY STRATEGY POOL**

We have discussed that a kind of attack called malicious medium, which can intercept and tamper with the intermediate data transmitted through the network. For this problem, hash functions can be adopted to calculate the feature code of the intermediate data before transmission, then the data and the feature code are transmitted together. After the workflow execution program receives the data, the feature code will be used for integrity verification. There are many types of hash functions with different security.
strengths and processing speeds. In [18], the authors list the security strengths of 5 hash functions and test their processing speeds, as shown in Table III. In Table III, the IS Speed is the processing speed of the integrity strategy, which combines encoding and verification processes and can be calculated as follows:

$$\text{speed} = \frac{\text{data size}}{\text{encoding time} + \text{verification time}}$$  \hspace{1cm} (3)

The integrity strategy pool is represented by $S^I = \{s^I_1, s^I_2, \ldots, s^I_{N(is)}\}$, $s^I_k \in S^I$ indicates the integrity strategy, $N(is)$ denotes the number of integrity strategies in the pool.

<table>
<thead>
<tr>
<th>TABLE III</th>
<th>INTEGRITY STRATEGY POOL OF THE INTERMEDIATE DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbol</td>
<td>Algorithm</td>
</tr>
<tr>
<td>$s^I_1$</td>
<td>TIGER</td>
</tr>
<tr>
<td>$s^I_2$</td>
<td>RIFDMD-160</td>
</tr>
<tr>
<td>$s^I_3$</td>
<td>SHA-1</td>
</tr>
<tr>
<td>$s^I_4$</td>
<td>RIFDMD-128</td>
</tr>
<tr>
<td>$s^I_5$</td>
<td>MD5</td>
</tr>
</tbody>
</table>

IV. ACISO SCHEME

In this section, we will introduce the ACISO scheme, which mainly includes SSOA model and the heuristic solution. To improve the readability, we sum up the main notation used in this section, as shown in Table IV.

A. SSOA MODEL

In this section, we transform the intermediate data security problem into the security strategy optimal allocation problem and establish SSOA model, which aims to maximize the overall intermediate data security strength while meeting several constraints.

The content is based on the premise that the cloud platform has given the task scheduling strategy, that is, the system knows which the intermediate data is fixed data and which the intermediate data is mobile data. And the VM assigned to each sub-task is also a known condition. $vm(a_k)$ indicates the serial number of the VM performing sub-task $a_k$. If sub-task $a_3$ is placed into VM 5, $vm(a_3) = 5$. Furthermore, the original workflow makespan $ms_o$ is also known.

1) CONSTRAINTS OF MODEL INDEPENDENT VARIABLES

A binary variable $x_{i,k}^j$ is used to indicate whether the $k$-th security strategy is selected from the security strategy pool $S^I$, $j \in \{cs, is, as\}$ for intermediate data $d_i$.

$$x_{i,k}^j = \begin{cases} 1, & \text{if } k \text{ th strategy is selected} \\ 0, & \text{otherwise} \end{cases}$$  \hspace{1cm} (4)

Since a kind of security requirement for each intermediate data can be implemented by only one security strategy, it comes to the following constraint:

$$\sum_{k=1}^{N(j)} x_{i,k}^j \leq 1, i = 1, 2, \ldots, |D|, j \in \{cs, is, as\}$$  \hspace{1cm} (5)

$|D|$ represents the amount of workflow intermediate data.

<table>
<thead>
<tr>
<th>TABLE IV</th>
<th>DEFINITIONS OF NOTATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_k$</td>
<td>The $k$-th sub-task of the workflow;</td>
</tr>
<tr>
<td>$a_{id}$</td>
<td>The last sub-task of the workflow;</td>
</tr>
<tr>
<td>$d_k$</td>
<td>The $k$-th intermediate data of the workflow;</td>
</tr>
<tr>
<td>size($d_k$)</td>
<td>The size of intermediate data $d_k$;</td>
</tr>
<tr>
<td>$st_{cs}$, $st_{as}$, $st_{is}$</td>
<td>The requirement for the intermediate data availability, confidentiality and integrity;</td>
</tr>
<tr>
<td>$s^a_{ks}$, $s^a_{ks}$</td>
<td>The $k$-th availability, confidentiality and integrity strategy;</td>
</tr>
<tr>
<td>$S^{a_{cs}}, S^{a_{as}}, S^{a_{is}}$</td>
<td>The availability, confidentiality and integrity strategy pool;</td>
</tr>
<tr>
<td>$N(as), N(cs), N(is)$</td>
<td>The number of security strategies in the availability, confidentiality and integrity strategy pool;</td>
</tr>
<tr>
<td>$r(s^a_{ks})$</td>
<td>The proportion of encoded data after applying $s^a_{ks}$;</td>
</tr>
<tr>
<td>$st(s)$</td>
<td>The security strength of strategy $s$;</td>
</tr>
<tr>
<td>$sp(s)$</td>
<td>The processing speed of strategy $s$;</td>
</tr>
<tr>
<td>$x_{i,k}^j$</td>
<td>The $k$-th security strategy is selected from $S^I$, $j \in {cs, is, as}$ for intermediate data $d_i$;</td>
</tr>
<tr>
<td>$st_{k}$</td>
<td>The security strength of the availability strategy, confidentiality strategy and integrity strategy applied in intermediate data $d_k$;</td>
</tr>
<tr>
<td>$st_{k}$</td>
<td>The normalized overall security strength of intermediate data $d_k$;</td>
</tr>
<tr>
<td>$\eta_t$</td>
<td>The maximum workflow makespan delay ratio which can be tolerated;</td>
</tr>
<tr>
<td>$\eta_x$</td>
<td>The maximum storage overhead which can be tolerated;</td>
</tr>
<tr>
<td>$vm(a_k)$</td>
<td>The serial number of the VM performing sub-task $a_k$;</td>
</tr>
<tr>
<td>$tl(vm(a_k))$</td>
<td>The time when the VM $vm(a_k)$ begins to be idle;</td>
</tr>
<tr>
<td>$tt_{p,i}$</td>
<td>The time of transmitting intermediate data $d_i$;</td>
</tr>
<tr>
<td>$bw(vm(a_p), vm(a_i))$</td>
<td>The network bandwidth between $vm(a_p)$ and $vm(a_i)$;</td>
</tr>
<tr>
<td>$tf(a_k)$</td>
<td>The finish time of sub-task $a_k$;</td>
</tr>
<tr>
<td>$ts(a_k)$</td>
<td>The start time of sub-task $a_k$;</td>
</tr>
<tr>
<td>$pred(a_k)$</td>
<td>The direct predecessor sub-task set of sub-task $a_k$;</td>
</tr>
<tr>
<td>$</td>
<td>D</td>
</tr>
<tr>
<td>$to^{a_{cs}}, to^{a_{as}}, to^{a_{is}}$</td>
<td>The time overhead of availability, confidentiality and integrity strategy for $d_k$;</td>
</tr>
<tr>
<td>$to^k$</td>
<td>The total time overhead of security strategies for $d_k$;</td>
</tr>
<tr>
<td>$ms_o$</td>
<td>The original workflow makespan;</td>
</tr>
<tr>
<td>$ms_k$</td>
<td>The workflow makespan after applying ACISO scheme;</td>
</tr>
<tr>
<td>$so_o$</td>
<td>The intermediate data storage overhead;</td>
</tr>
<tr>
<td>$so_k$</td>
<td>The intermediate data storage overhead after applying ACISO scheme;</td>
</tr>
</tbody>
</table>

2) CONSTRAINTS OF WORKFLOW MAKESPAN

If there is an intermediate data $d_i$ that needs to be transmitted, the availability strategy should be applied to $d_i$ firstly, then the time overhead $to^{a_{cs}}$ can be denoted as:
\[
    tso^{as} = \sum_{k=1}^{N(as)} sp(ss_k^{as}) \cdot size(d_i) \cdot x_{t,k}^{as}
\]

(size(d_i) represents the size of d_i. After applying the availability strategy, the data size will change to size(d_i')).

\[
    size(d_i') = \sum_{k=1}^{N(as)} size(d_i) \cdot r(ss_k^{as}) \cdot x_{t,k}^{as}
\]

Then confidentiality strategy and integrity strategy are implemented in turn, the confidentiality and integrity strategy time overhead to_t^{cs} and to_t^{is} can be calculated by (8).

\[
to_t^j = \sum_{k=1}^{N(j)} sp(ss_k^j) \cdot size(d_i') \cdot x_{t,k}^j , j \in \{cs, js\}
\]

Therefore, the sum of the security strategy time costs of the intermediate data d_i can be represented as (9).

\[
to_t = to_t^{as} + to_t^{cs} + to_t^{is}
\]

The generated time costs of intermediate data d_i will delay the execution of the sub-tasks which depend on d_i. For any sub-task of the workflow, it can be executed only when it receives intermediate data from all direct predecessors and the assigned VM is idle, which can be represented as (10).

\[
ts(a_i) = \max \{ tl(vm(a_i)) , \max_{a \in pred(a_i)} \{ tf(a_p) + tp_{p,i} + tt_{p,i} \} \}
\]

where \( t\ell(vm(a_i)) \) represents the time when the VM \( vm(a_i) \) begins to be idle, \( pred(a_i) \) denotes the direct predecessor sub-task set of sub-task \( a_i \), \( tf(a_p) \) indicates the finish time of \( a_p \), and \( ts(a_i) \) indicates the start time of \( a_i \). \( tp_{p,i} \) represents the processing time of security strategies applied in the intermediate data, which can be calculated by (11).

\[
    tp_{p,i} = \begin{cases} 
    0, & \text{if } vm(a_p) = vm(a_i) \\
    to_p, & \text{otherwise} 
\end{cases}
\]

\( tt_{p,i} \) represents the time of transmitting the intermediate data from \( vm(a_p) \) to \( vm(a_i) \), which can be calculated by (12).

\[
    tt_{p,i} = \begin{cases} 
    size(\text{the data between } a_p \text{ and } a_i), & \text{otherwise} \\
    \frac{size(\text{the data between } a_p \text{ and } a_i)}{bw(vm(a_p), vm(a_i))}, & \text{otherwise} 
\end{cases}
\]

where \( bw(vm(a_p), vm(a_i)) \) denotes the network bandwidth between \( vm(a_p) \) and \( vm(a_i) \). The workflow makespan \( ms_a \) after applying ACISO scheme must meet the following constraints,

\[
    (ms_a - ms_o)/ms_o < \eta_t
\]

where \( ms_a = tf(a_{Last}) \), \( a_{Last} \) is the last sub-task of the scientific workflow, \( ms_o \) represents the workflow makespan without applying ACISO scheme, \( \eta_t \) indicates the maximum workflow makespan delay ratio which can be tolerated.

In practice, the constraint of workflow makespan should be a user-defined deadline. But in the paper, we use the delay ratio as the constraint of the workflow makespan, which can clearly reflect the increased time overhead of ACISO and show the results in simulations. If the constraint of workflow makespan is user-defined deadline, the workflow scheduling algorithm needs to ensure that the theoretical workflow makespan is earlier than the deadline, then ACISO is applied according to the user’s security requirements. If ACISO cannot find a solution, the system will inform users to extend the deadline or reduce the security requirements.

### 3) CONSTRAINTS OF STORAGE COSTS

The workflow intermediate data is usually stored in VMs [39]. We regard the stored intermediate data size as the storage overhead. If ACISO scheme is not applied, the storage overhead \( s_o \) can be calculated by (14).

\[
    s_o = \sum_{i=1}^{\lvert D \rvert} size(d_i)
\]

If ACISO scheme is applied, the storage overhead will increase, which can be calculated by (15).

\[
    s_o = \sum_{i=1}^{\lvert D \rvert} x_{i,k}^{as} \cdot r(ss_k^{as}) \cdot size(d_i)
\]

Usually the storage space of VMs is limited, so the storage overhead \( s_o \) needs to be constrained, as shown in (16).

\[
    (s_o - s_o^a)/s_o < \eta_s
\]

\( \eta_s \) represents the maximum storage overhead increase rate that can be tolerated.

### 4) CONSTRAINTS OF INTERMEDIATE DATA SECURITY STRENGTH

The security strength of the availability, confidentiality and integrity strategy applied to the intermediate data can be represented as (17).

\[
    st_{i,j} = \sum_{k=1}^{N(j)} st(ss_k^j) \cdot x_{i,k}^j , j \in \{as, cs, is\}
\]

In order to ensure the security strength of each strategy applied to the intermediate data \( d_i \) meets the requirements, it comes to the following constraint.

\[
    st_{i,j} \geq sr_j, j \in \{as, cs, is\}, i = 1, 2, \ldots, \lvert D \rvert
\]

The proposed ACISO scheme can be divided into multiple security levels for a user to choose, for example, the low security level, medium security level and high security level. In the low security level, \( sr_j = 0.2, j \in \{as, cs, is\} \). In the medium security level, \( sr_j = 0.5, j \in \{as, cs, is\} \). In the high security level, \( sr_j = 0.8, j \in \{as, cs, is\} \). \( st^{as}, st^{cs}, st^{is} \) are the availability requirement, confidentiality requirement and integrity requirement, respectively.
If the submitted workflow contains a lot of sensitive information, we recommend the high security level. If the submitted workflow contains a little sensitive information, we recommend the medium security level. If the submitted workflow contains no sensitive information, we recommend the low security level.

5) OBJECTIVE FUNCTION

The normalized overall security strength of the intermediate data is defined as (19),

\[ st_i = \frac{1}{\sqrt{N}} \sum_{k=1}^{N} s_t^{as} \cdot s_t^{cs} \cdot s_t^{is} \]

(19)

The purpose of SSOA is to maximize the overall security strength of all intermediate data, so the objective function is defined as (20),

\[ f(x) = -\sqrt{\prod_{i=1}^{D} st_i} \]

(20)

6) SSOA PROBLEM FORMULATION

The optimization equations of SSOA are shown in (21)-(25), (21) is the objective function, (22) is the constraint of the model independent variables, (23) is the constraint of the intermediate data security strength, (24) is the constraint of the workflow makespan, and (25) is the constraint of the storage overhead. \( x \) is a matrix, as shown in (26),

\[
\begin{aligned}
\text{min } & f(x) \\
\text{s.t. } & g_1(x) = -\sum_{k=1}^{N(j)} x_{i,k}^{j} + 1 \geq 0, \quad j \in \{as, cs, is\}, \quad i = 1, 2, \ldots, |D| \\
& g_{|D|+1}(x) = -st_j + st_i^{j} \geq 0, \quad j \in \{as, cs, is\}, \quad i = 1, 2, \ldots, |D| \\
& g_{|D|+2}(x) = -(ms_o - ms_o + \eta_o > 0) \\
& g_{|D|+3}(x) = -(ms_o - ms_o + \eta_o > 0) \\
\end{aligned}
\]

(21)-(25)

B. PROBLEM TRANSFORMATION

The SSOA is a constraint optimization problem, so we use external point penalty function method [40] to build auxiliary function \( F(x, \sigma) \), as shown in (27), which can transform the constrained optimization problem into an unconstrained optimization problem.

\[ F(x, \sigma) = f(x) + \sigma \sum_{i=1}^{2|D|+2} [\max\{0, -g_i(x)\}]^2 \]

(27)

\( \sigma \) is a very large positive number. In (27), if \( x \) is a viable point, the value of the auxiliary function \( F(x, \sigma) \) is equal to the value of the object function \( f(x) \). If \( x \) is not, the value of the auxiliary function \( F(x, \sigma) \) is equal to the value of the object function \( f(x) \) plus a large positive number. Therefore, (21)-(25) can be transformed into (28),

\[ \text{min } F(x, \sigma) \]

(28)

The actual workflow is usually composed of thousands of intermediate data, and there are 75 combinations of security strategies (3 availability strategies, 5 confidentiality strategies and 5 integrity strategies) for each intermediate data. So solving this problem is NP-hard [18]. Therefore, the heuristic algorithm is required to solve this model.

C. HEURISTIC SOLUTION

In this section, we propose a heuristic solution based on improved PSO to solve SSOA model. PSO has the advantages of high search efficiency and fast convergence, which has been the most common algorithm used for task scheduling in scientific cloud workflow systems [27]-[29].

PSO is derived from the foraging behavior of the bird flock. It is assumed that a group of birds search for food randomly, and there is only one piece of food. At the beginning, all the birds do not know where the food is, but are able to remember where is closest to the food in their flight route, and they also know where is closest to the food in the flight route of the entire bird flock. Therefore, each bird combines its own experience and the group experience to decide where to fly next. Birds follow the same logic to iteratively change their flying route until finding the food.

PSO regards the birds as particles, the flight space as the solution space, and foods as the optimal solution. Particles have two properties: position and velocity. The position of the particle corresponds to the solution of the original problem, which is described by (26), and the velocity of the particle determines where the particle will fly next and how far it will fly. The quality of the particle position is measured by the fitness value, which is calculated by taking the particle position into the objective function. The core of PSO is the speed update formula and the position update formula, as shown in (29) and (30), respectively.

\[
\begin{align*}
\text{In this section, we propose a heuristic solution based on improved PSO to solve SSOA model. PSO has the advantages of high search efficiency and fast convergence, which has been the most common algorithm used for task scheduling in scientific cloud workflow systems [27]-[29].}
\end{align*}
\]
\[ \mathbf{v}_i^{k+1} = w \mathbf{v}_i^k + c_1 \text{rand}_1 (\mathbf{p}_{\text{best},i}^k - \mathbf{p}_i^k) + c_2 \text{rand}_2 (\mathbf{g}_{\text{best}} - \mathbf{p}_i^k) \] (29)

\[ \mathbf{p}_i^{k+1} = \mathbf{p}_i^k + \mathbf{v}_i^{k+1} \] (30)

In (29), \( \mathbf{v}_i^k \) represents the moving velocity of the \( i \)-th particle at the \( k \)-th iteration, \( w \) denotes the inertia coefficient, \( c_1, j = 1,2 \) indicates the acceleration coefficient. \( \text{rand}_1, j = 1,2 \) is a random number between 0 and 1. \( \mathbf{p}_i^k \) denotes the position of the \( i \)-th particle at the \( k \)-th iteration, \( \mathbf{p}_{\text{best},i}^k \) denotes the optimal position of the \( i \)-th particle, and \( \mathbf{g}_{\text{best}} \) denotes the optimal position in the particle swarm.

We define the fitness of the \( i \)-th particle at the \( k \)-th iteration as (31),

\[ \text{fit}(\mathbf{p}_i^k) = F(\mathbf{p}_i^k, \sigma) \] (31)

The smaller the value of the fitness, the better the position of the particle.

In order to avoid the defect of local convergence, we introduce adaptive inertia coefficient [30] to improve the traditional PSO. In the improved algorithm, we use \( \text{update}(i,k) \) to indicate whether the \( i \)-th particle is successfully updated at the \( k \)-th iteration, which can be calculated by (32),

\[ \text{update}(i,k) = \begin{cases} 1, & \text{fit}(\mathbf{p}_{\text{best},i}^k) < \text{fit}(\mathbf{p}_{\text{best},i}^{k-1}) \\ 0, & \text{fit}(\mathbf{p}_{\text{best},i}^k) = \text{fit}(\mathbf{p}_{\text{best},i}^{k-1}) \end{cases} \] (32)

\( \mathbf{p}_{\text{best},i}^k \) denotes the optimal position of the \( i \)-th particle at the \( k \)-th iteration. Then, we use \( \text{rate}(k) \) to represent the proportion of the particles whose current positions are better than before, which can be calculated by (33),

\[ \text{rate}(k) = \frac{1}{\text{NP}} \sum_{i=1}^{\text{NP}} \text{update}(i,k) \] (33)

\( \text{NP} \) denotes the total number of particles. After that \( w(k) \) representing the inertia coefficient at the \( k \)-th iteration can be calculated by (34),

\[ w(k) = (w_{\text{max}} - w_{\text{min}}) \text{rate}(k) + w_{\text{min}} \] (34)

\( w_{\text{min}} \) and \( w_{\text{max}} \) respectively indicate the minimum and maximum values allowed. The purpose of (32)-(34) is to adaptively adjust the particle moving speed to balance the global and local search capabilities of the particles, which can avoid the algorithm falling into the local optimum. When \( \text{rate}(k) \) is large, the whole particle swarm is far from the global optimum. At this time, the particle needs to perform the search at a large velocity. When \( \text{rate}(k) \) is small, the whole particle swarm is close to the global optimum. At this time, the particle search speed needs to be reduced to accurately locate the global optimum. Therefore, (29) can be transformed into (35),

\[ \mathbf{v}_i^{k+1} = w(k) \mathbf{v}_i^k + c_1 \text{rand}_1 (\mathbf{p}_{\text{best},i}^k - \mathbf{p}_i^k) + c_2 \text{rand}_2 (\mathbf{g}_{\text{best}} - \mathbf{p}_i^k) \] (35)

Algorithm 1: The heuristic solution

Input: NI: the number of iterations, NP: the number of particles

Output: \( \mathbf{g}_{\text{best}} \)

for \( i=0 \) to \( \text{NP} \)
\[ p[i]=\text{initialize the position of each particle} \]
\[ f[i]=\text{Calculate the fitness of each particle by (31)}; \]
\[ \mathbf{p}_{\text{best}}[i]=p[i]; //\text{initialize the best position for each particle} \]
\[ \text{fit}_{\text{pbest}}[i]=f[i]; \]
end
\[ w=\text{Calculate the inertia coefficient by (34)}; \]
\[ \mathbf{g}_{\text{best}}=\text{Select the particle position with the highest fitness}; \]
for \( j=0 \) to \( \text{NI} \)
\[ v[i]=\text{Calculate the moving vector by (35)}; \]
\[ p[i]=\text{Update the position by (30)}; \]
\[ f[i]=\text{Calculate the fitness by (31)}; \]
\[ \text{if} f[i]<\text{fit}_{\text{pbest}}[i] \]
\[ \mathbf{p}_{\text{best}}[i]=p[i]; \]
\[ \text{fit}_{\text{pbest}}[i]=f[i]; \]
end
\[ w=\text{Calculate the inertia coefficient by (34)}; \]
\[ \mathbf{g}_{\text{best}}=\text{Select the particle position with the highest fitness}; \]
end
return \( \mathbf{g}_{\text{best}} \).

The steps of the heuristic solution are shown in Algorithm 1. In this algorithm, the position of each particle is described by (26). At the beginning of the algorithm, the position and moving vector of each particle are initialized with random numbers. The fitness of the particle is calculated by (31), the larger the value, the better the position of the particle. During the iterative process, (35) and (30) are used to continually update the position of each particle so that the particles as a whole move closer to the global optimal position. The inertia coefficient is calculated by (34). When the number of iterations reaches the maximum number of iterations, the algorithm will end and the current \( \mathbf{g}_{\text{best}} \) will be the approximate optimal solution.

When applying ACISO, if there are already encryption operations in the network protocol layers such as SSL, ACISO can be used for selecting encryption algorithm, and the network protocol layer uses the selected algorithm for the actual encryption operation.

V. SIMULATION

A. SIMULATION SETUP

We use the open source software WorkflowSim [19] to conduct several tests to evaluate ACISO: (1) Evaluating the convergence of ACISO. (2) Evaluating total security strength brought by ACISO. (3) Evaluating the availability...
of the intermediate data protected by ACISO. (4) Evaluating the confidentiality and integrity of the intermediate data protected by ACISO.

WorkflowSim is designed for simulating task scheduling in the cloud computing environment. In this software, each workflow is described by a XML file, which includes some parameters such as the number of sub-tasks, sub-task execution time, the dependent relationship between sub-tasks and the size of the intermediate data.

The workflows used in this paper are CyberShake, Epigenomics, Montage and Inspiral, which are published by Pegasus project [26], the structures and parameters of them are shown in Fig. 4 and Table V, respectively. In the test, we first use HEFT [31] which has been already implemented in WorkflowSim to make the scheduling strategy for the workflows. Based on the scheduling result, ACISO is adopted to ensure the security of the intermediate data.

Furthermore, we define the security strength of the intermediate data that is transmitted between VMs but without any security strategy protection as 0.01.

![Fig. 4. The structures of workflow Epigenomics, Inspiral, CyberShake and Montage](image)

**TABLE V**

<table>
<thead>
<tr>
<th>Workflow</th>
<th>Number of sub-tasks</th>
<th>Number of intermediate data</th>
<th>Average data size</th>
<th>Average sub-task runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epigenomics (997)</td>
<td>997</td>
<td>1234</td>
<td>388.59 MB</td>
<td>3858.67 s</td>
</tr>
<tr>
<td>Inspiral (1000)</td>
<td>1000</td>
<td>1233</td>
<td>8.90 MB</td>
<td>227.25 s</td>
</tr>
<tr>
<td>CyberShake (1000)</td>
<td>1000</td>
<td>1988</td>
<td>102.29 MB</td>
<td>22.71 s</td>
</tr>
<tr>
<td>Montage (1000)</td>
<td>1000</td>
<td>2485</td>
<td>3.21 MB</td>
<td>11.36 s</td>
</tr>
</tbody>
</table>

B. COMPARISON ALGORITHMS

We compare ACISO with the following two schemes:

NSS (no security strategy) scheme: This scheme does not offer the security strategy for the intermediate data of scientific workflows.

CISO (confidentiality and integrity strategy optimization) scheme: In [17], the authors use diverse encryption algorithms and hash functions to enhance the confidentiality and integrity of the workflow intermediate data. Then the security strategy allocation is transformed into an optimization problem. In the optimization model, the object is to maximize the overall intermediate data security strength with the constraint of the workflow makespan and the minimal security requirement. At last, the integer linear program method is adopted to solve the optimization problem and get the optimal security strategy allocation result. However, in [17], the authors do not list the parameters of the used security strategies, and thus we can only use the security strategies shown in Table II and Table III to construct the optimization model.

C. SIMULATION RESULTS

1) CONVERGENCE

In this section, we will evaluate the convergence of ACISO. It is assumed that the workflow makespan can be delayed by no more than 5%, the storage overhead can be increased by no more than 60%, and the security strength of each type of security requirements for each intermediate data cannot be less than 0.2. The number of particles is set to 2000. We employ WorkflowSim to build a simulated cloud workflow execution environment with 10 available VMs, and use the number of iterations as variables to calculate the security strength of the intermediate data. Each simulation is performed 1000 times and the average value is regarded as the final result, as shown in Fig. 5. From the figure we can find that when the number of iterations exceeds 1000, the overall security strengths of the intermediate data of the four workflows show a steady trend. Therefore, in the subsequent simulations, the number of particles is set to 2000 and the number of iterations is set to 1000. Furthermore, there are obvious gaps in the security strength of the four workflows. The security strength of Inspiral stabilizes at 0.78 while
CyberShake stabilizes at 0.51. The cause of this difference is related to the ratio of the average sub-task execution time and the average intermediate data size in the workflow. The smaller the ratio, the greater the proportion of the intermediate data transfer time in the total workflow execution time. Thus, under the same workflow makespan constraints, such workflows can only be protected by the security strategies with high efficiency but low security strength.

3) SECURITY STRENGTH UNDER DIFFERENT WORKFLOW MAKESPAN
It is assumed that the number of available VMs is 10, the storage overhead can be increased by no more than 60%, and the strength of each type of security requirement for each intermediate data cannot be less than 0.2. We respectively use 5%, 10%, 15% and 20% as constraints of the workflow makespan and calculate the corresponding security strength of the intermediate data. Each experiment is performed 1000 times and the average value is regarded as the final result, as shown in Fig. 7. The softer the workflow makespan constraints, the stronger security strategies applied in the intermediate data, until all the mobile data can apply the strongest security strategies.

4) AVAILABILITY OF INTERMEDIATE DATA
In this section, we define the availability strength of the workflow as the probability that each intermediate data can be accessed under a certain probability of data loss. We suppose that the number of available VMs is 10, the workflow makespan delay cannot be more than 5%, the storage overhead can be increased by no more than 60%, and the strength of each type of security requirement for each intermediate data cannot be less than 0.2. Epigenomics workflow is used for the test. The evaluation results are shown in Fig. 8. From the figure we can find that ACISO can effectively improve the availability strength of the workflow. It is because that erasure encoding can enhance the fault tolerant ability of the...
intermediate data. Even if some data blocks are lost, it will not affect the recovery of the entire data.

![FIGURE 8. Availability strength of ACISO, CISO and NSS](image)

4) CONFIDENTIALITY AND INTEGRITY OF INTERMEDIATE DATA

In this section, we also suppose that the number of available VMs is 10, the workflow makespan delay cannot be more than 5%, the storage overhead can be increased by no more than 60%, and the strength of each type of security requirement for each intermediate data cannot be less than 0.2. We employ ACISO and CISO to get optimal security strategy allocation results for Epigenomics workflow, respectively. The allocation proportion of the encryption algorithms and hash functions are shown in Fig. 9 (a) and (b), respectively. From the figure we can find that the proportion of the encryption algorithms and hash functions with high security strength in CISO is higher than that in ACISO. It is because that the erasure encoding used in ACISO will increase the processing time of the intermediate data. Under the same workflow makespan constraint, ACISO can only focus on the highly efficient encryption algorithm and hash functions.

![FIGURE 9. The allocation proportion of security strategies (a) The confidentiality strategies (b) The integrity strategies](image)

In order to evaluate the confidentiality and integrity strength of the intermediate data protected by ACISO, we define the confidentiality strength as the probability that attackers cannot acquire the content of any intermediate data, and define the integrity strength as the probability that attackers cannot tamper with the any intermediate data without being discovered. We define the decryption probability of the data encrypted by $ss_k^c$ as $p/st(ss_k^c)$, where $p$ is a constant, and define the probability that the attacker tampers with the data protected by $ss_k^e$ without changing the hash value as $p/st(ss_k^e)$, in this test, $p$ is 0.001. We calculate the confidentiality and integrity strength of the intermediate data of Epigenomics, the calculation result is shown in Fig. 10. From the figure we can find that the confidentiality strength of the intermediate data protected by ACISO is improved significantly, since erasure encoding make it more difficult for adversaries to decrypt the intermediate data. The integrity strength is also improved, because even if an attacker successfully tampers with a data block, this block is not necessarily used for data recovery.

![FIGURE 10. Confidentiality and integrity strength of ACISO, CISO and NSS](image)

VI. RELATED WORK

Most workflows can be categorized as data-intensive tasks, and a large amount of intermediate data will be generated during the execution. In [2], the authors believe that the intermediate data of scientific workflows should be stored for repeated analysis, and present a scientific workflow intermediate data storage strategy in which both the generation costs and storage costs are considered. In order to improve the efficiency of task scheduling, in [32], k-means algorithm is used to optimize the initial data placement strategy. Then a multi-level task replication strategy is proposed to reduce the amount of intermediate data transmission by analyzing the dependencies between data sets and sub-tasks. In addition to optimizing data transfer between workflow sub-tasks, there are some researches on reducing the amount of data transferred between data centers during workflow execution. In [33], a data center selection method is presented. The method first minimizes the number of data centers and then optimizes the network bandwidth between data centers to achieve efficient transmission of workflow intermediate data between different data centers. In [34], graph segmentation method is introduced to optimize the transmission cost of workflow intermediate data between different data centers. In
[44], task scheduling and the data assignment problems are treated together, which can avoid unnecessary data transfers.

The characteristics of cloud computing environment resource sharing and multi-tenant coexistence have made the security of the data in the cloud widely concerned. Xiao et al. [14] divide cloud data security problem into data availability, data confidentiality and data integrity, and list existing defense strategies. In [35], a secure cloud data outsourcing mode is proposed, in which data segmentation, data encryption and data compression are combined together to minimize the cost of data processing while maintaining cloud data confidentiality. In [36], the authors present a data placement method which can automate the choice of whether to store the data in the cloud or client according to the data privacy requirements. Bessani et al. [37] design a trusted cloud storage system called TruXy to achieve secure storage of intermediate data. In [47], mimic defense strategy is proposed, in which data segmentation, data encryption and data compression are combined together to minimize the cost of data processing while maintaining cloud data confidentiality. In [36], the authors present a data placement method which can automate the choice of whether to store the data in the cloud or client according to the data privacy requirements. Bessani et al. [37] design a trusted cloud storage system called TruXy to achieve secure storage of intermediate data.

Some works have studied security-aware workflow scheduling algorithm. Zeng et al. [45] introduce immovable dataset concept which constrains the movement of certain datasets due to security and cost considerations and propose a new scheduling model in the context of cloud systems. Ding et al. [9] propose a primary-backup workflow scheduling strategy to realize fault-tolerant elastic task scheduling. Yao et al. [10] consider that the process of rescheduling in cloud systems had great similarities to the immune system which kept the body stable by removing the intrusive antigens, so an immune system inspired failure-aware rescheduling algorithm for the workflow task in cloud systems was designed to achieve fault-tolerant workflow execution. In [12], considering the cost and security together, the authors presented an adaptive, just-in-time workflow scheduling algorithm, which uses both spot and on-demand instances to reduce costs and provide fault tolerance.

Some works have studied the workflow data security. In [38], the authors build a workflow security cost model to reasonably reduce the security overhead of sensitive data protection, then propose a data placement strategy to strengthen the data security by dynamically placing intermediate data for scientific workflows. However, the optimization of the data placement can only reduce the probability that the workflow intermediate data is exposed to the attackers, cannot fundamentally ensure the security of the intermediate data. Nepal et al. [46] propose a trusted storage cloud system named TruXy to achieve secure storage of workflow intermediate data. In [47], mimic defense technology is used to ensure the security of the process from the workflow sub-task execution to the intermediate data generation. In [17], the diverse security strategies are applied to enhance the confidentiality and integrity of workflow intermediate data. However, the authors do not consider the intermediate data availability and do not list the parameters of all the used security strategies. In [18], workflow scheduling and security strategy allocation are jointly considered. The authors firstly propose a workflow scheduling algorithm, which can reduce the workflow makespan. Then the authors present to use the workflow sub-tasks’ laxity time to encrypt the intermediate data. The proposed ACISO scheme is more flexible than [18], since it is separated from the workflow scheduling. Thus, users can choose any type of workflow scheduling algorithm, such as such as minimizing workflow execution costs, minimizing workflow makespan, maximizing VM utilization, and maximizing reliability.

VII. CONCLUSION AND FUTURE WORK

In this paper, we propose ACISO scheme to enhance the availability, confidentiality and integrity of the workflow intermediate data. In order to improve the availability of intermediate data, the erasure encoding technology is used to construct the availability strategy. After applying availability strategies, the intermediate data will be divided into multiple data blocks to store in different VMs. In order to improve the confidentiality and integrity, multiple types of encryption algorithms and hash functions are introduced to construct confidentiality strategy pool and integrity strategy pool. On the basis of these three kinds of security strategy pools, we propose SSOA model to transform the intermediate data security problem into the security strategy optimized allocation problem and propose a heuristic solution to solve the model. The simulation results show that ACISO can effectively improve the availability, confidentiality, and integrity of the intermediate data of scientific workflows. However, currently, our work cannot be applied to the scenario where dynamic task scheduling algorithms are used. We will consider the combination of ACISO with dynamic task scheduling algorithms in the future.

REFERENCES


