Role-Mining Optimization with Separation-of-Duty Constraints and Security Detections for Authorizations

Wei Sun 1,*, Shiwei Wei 2, Huaping Guo 3 and Hongbing Liu 1

1 Center of Network Information and Computing, Xinyang Normal University, Xinyang 464000, China; liuhbing@xynu.edu.cn
2 School of Computer and Technology, Guilin University of Aerospace Technology, Guilin 541000, China; swwei@guat.edu.cn
3 School of Computer and Information Technology, Xinyang Normal University, Xinyang 464000, China; hpguo@xynu.edu.cn
* Correspondence: sunny810715@xynu.edu.cn

Received: 21 August 2019; Accepted: 16 September 2019; Published: 19 September 2019

Abstract: Role-based access control (RBAC), which has been regarded as one of the most popular access-control mechanisms, is featured by the separation-of-duty constraints, mutually exclusive constraints, and the least-privileges principle. Role mining, a bottom-up role-engineering technology, is an effective method to migrate from a non-RBAC system to an RBAC system. However, conventional role-mining approaches not only do not consider the separation of duty constraints, but also cannot ensure the security of a constructed RBAC system when the corresponding mined results violate the separation of a duty constraint and/or the least-privileges principle. To solve these problems, this paper proposes a novel method called role-mining optimization with separation-of-duty constraints and security detections for authorizations (RMO_SODSDA), which mainly includes two aspects. First, we present a role-mining-optimization approach for satisfying the separation of duty constraints, and we constructed different variants of mutually exclusive constraints to correctly implement the given separation of duty constraints based on unconstrained role mining. Second, to ensure the security of the constructed system and evaluate authorization performance, we reduced the authorization-query problem to a maximal-satisfiability problem. The experiments validate the effectiveness and efficiency of the proposed method.

Keywords: role engineering; role mining; separation of duty constraints; user authorization query

1. Introduction

With the rapid development of network-information technology and the comprehensive application of cloud computing and big data techniques, there is much storage and there are many exchanges in large-scale and complex information-management systems [1]. The mechanism to ensure the security of system data and user information has attracted much attention. An increasing number of enterprises and organizations have adopted role-based access control (RBAC) as their main access-control mechanism, as it makes security administration more flexible and manageable [2–7]. With the successful implementations of RBAC systems, designing an effective, complete set of roles and constructing a good RBAC system have become critical tasks. As a solution to facilitate the process of migrating from a non-RBAC system to an RBAC system, role-engineering technology [8,9] has been proposed. There are two main approaches: top-down [10] and bottom-up [11–16]. In the former approach, roles are derived by carefully analyzing particular organization functions. While this approach can reflect the functional requirements of the organization well, it is time-consuming and
labor-intensive. In the latter approach, roles are aggregated by applying data-mining techniques from existing user-permission assignments, and the architectural structure of RBAC can be automatically constructed. The bottom-up approach, also known as role mining, has gained considerable interest and popularity in recent years.

Vaidya et al. [17] converted role mining into a matrix-decomposition problem and presented the definition of a basic role mining problem (basic RMP), which attempts to find a minimal set of roles from bottom-user permission assignments and completely cover the original assignments. However, it is difficult to derive an optimal role set in practical applications. Afterward, a variant of basic RMP called $\delta$-approx RMP was proposed in the work of [17], which attempts to find a minimal set of roles by allowing a limited number of mismatches. As a result, it is easy to derive a suboptimal role set. Meanwhile, in min-noise RMP [17], the number of roles is bounded, and the number of mismatches between original assignments and assignments in the constructed system is minimized. According to different optimization objectives for role mining, many other approaches have been proposed, such as edge RMP [18], usage RMP [19], and user-oriented exact RMP [20].

A key characteristic of RBAC is that it allows the specification and enforcement of various constraint policies [21–23], such as cardinality constraints and separation-of-duty constraints (also called separation-of-duty policies), which can reflect security policies of different enterprises and organizations and ensure system security. The cardinality constraint limits the maximum number of roles related to users or to permissions, the maximum number of permissions a role can have, or the maximum number of users to which a role can be assigned [14]. Another significant constraint discussed in this paper is the separation-of-duty constraint, which is widely recognized to be a fundamental principle in computer security [14]. Typically, the constraint states that at least $k$ users are required to complete a special task that requires $n$ number of permissions. One approach requires that no $(k-1)$ users together have all the permissions needed to complete the task [24]. It has been widely used in the banking industry and military systems. In the approach to migrate from a non-RBAC system to an RBAC system, however, an inadequately addressed key challenge is that none of the conventional role-mining methods consider the separation-of-duty constraint.

Furthermore, both the mutually-exclusive-role constraint and the least-privileges principle play important roles [25,26] in evaluating system security and authorization performance in a constructed RBAC system. Because of free and random requests of accessing resources and fuzzy system configurations, it is impossible to find an optimal set of roles that cover all permissions for a given access request. The authorization-query problem [27] proved to be satisfiable, strictly complies with the separation-of-duty constraint, and attempts to derive a role set that can cover as many permissions as possible. Maximal satisfiability (MAX-SAT) is a core research problem of theoretical computer science that has had successful applications in various fields, such as artificial intelligence and model detection [28]. Nevertheless, conventional role-mining methods lack security detections for authorizations in the constructed RBAC system. The following problems exist: on the one hand, different results may conflict with each other and violate the mutually exclusive constraint; on the other hand, different results may include permissions outside the permission request and violate the least-privileges principle.

To address the above problems, this paper proposes a novel method called role-mining optimization with separation-of-duty constraints and security detections for authorizations (RMO_SODSDA). The main contributions of this paper are as follows:

1. The separation-of-duty constraint was proved to be effectively enforced by mutually exclusive constraints [24]. In order to satisfy these constraints, we presented a constrained-role optimization approach, we constructed several variants of mutually exclusive constraints to correctly implement the given separation of duty, and we evaluated the effectiveness of RMO_SODSDA.

2. As a branch of MAX-SAT, the partial MAX-SAT problem [29] aims to satisfy all hard clauses and as many soft clauses as possible. It is clearly more versatile. In order to ensure system
security, based on the previously constructed RBAC system, we converted the authorization-query problem into the partial MAX-SAT and evaluated RMO_SODSDA efficiency.

The rest of the paper is organized as follows. In Section 2, we discuss related work and present preliminaries and basic problem descriptions to understand the rest of the paper. Section 3 proposes a novel method (RMO_SODSDA), including two aspects: (1) presenting role optimization with separation-of-duty constraints and demonstrating how to construct different kinds of mutually-exclusive-role constraints, and (2) reducing the authorization-query problem to partial MAX-SAT and demonstrating how to ensure system security. We show the results of the experimental evaluations in Section 4 and present discussions in Section 5. Section 6 concludes the paper and discusses future work.

2. Related Work

2.1. Role-Mining Methods

Many methods have been proposed for role mining. According to whether or not constraints are considered in the optimization process, existing efforts mainly fall into the following two categories: (1) role mining with no constraint, and (2) role mining with constraints.

2.1.1. Role Mining with No Constraint

Vaidya et al. converted role mining into a matrix-decomposition problem and presented the definition of basic RMP. Basic RMP has been proven to be NP-complete, for which several existing studies have already been done to find efficient solutions. In view of the lack of semantics in mining roles, Molloy et al. [30] introduced semantic information, represented roles’ characteristics with a formal concept lattice, and proposed an optimization algorithm to find roles based on formal semantic analysis that can reflect the requirements of the organization’s functions and enhance the interpretability of mining roles. To optimize the RBAC system, Zhang et al. [31] presented an algorithm using graph-optimization theory, which reduces the administration required for role-related assignments through minimizing the number of edges in the graph. However, it does not research the redundancy of the mining results. Ene et al. [32] used heuristics and graph theory to find as few optimized roles as possible, which can reduce graph representations and the redundancy of mining roles. Lu et al. [19] proposed a unified optimization framework for role mining and presented a number of greedy algorithms that could solve basic RMP, δ-approx RMP, min-noise RMP, and edge RMP based on methods of integer linear programming and Boolean matrix decomposition. Although constraints are essential for the RBAC model, none of the above role-mining methods took constraints into account.

2.1.2. Role Mining with Constraints

In order to avoid abuse of privileges, Kumar et al. [33] proposed the constrained role miner algorithm that limits the number of permissions assigned to a role. Hingankar et al. [34] proposed a biclique-cover method to derive a set of roles that limit the maximum number of users related to a role. For example, the managing-director role in a company must be assigned to only one person. John et al. proposed two alternative approaches for restricting the number of roles assigned to a user: the role priority-based approach (RPA) and the coverage of permissions-based approach (CPA). The RPA prioritizes roles based on their number of permissions, and assigns optimal roles to a user according to the priority order. The CPA chooses roles by iteratively picking the role with the largest number of permissions that are yet uncovered, and then ensures that no user contains more than a given number of roles [35]. In order to simultaneously limit the maximum number of roles assigned to a user and a related permission, Harika et al. [36] imposed these constraints during the process of mining roles and proposed a postprocessing framework and a concurrent processing framework.
2.2. Methods of RBAC Authorizations

In order to obtain permissions while satisfying a collection of constraints for a given authorization request, it is necessary to determine whether a set of roles can be activated in a particular system. Zhang et al. presented the user authorization query (UAQ) problem that adopts a greedy algorithm and a mutually-exclusive-role constraint to search for the objects. The UAQ is to discover a set of roles to be activated in a single session for the particular set of permissions requested by the user [27]. This set of roles must satisfy constraints and follow the least-privileges principle. However, it stops searching once the set of roles does not satisfy the requests. Wickramaarachchi et al. [37] introduced the lower and upper bounds for the requested permissions, respectively, converting the UAQ into the MAX-SAT, finding truth assignments that satisfied as many clauses as possible, and demonstrating its effectiveness using SAT solvers. Lu et al. [28] proposed a novel approach based on role–permission reassignments to support the UAQ, which assists administrators to modify system configurations in an automatic manner. Koshimura et al. [38] presented partial MAX-SAT solver QMaxSAT, which encodes soft clauses with the blocking variables and has been proven to be highly effective. Partial MAX-SAT is an extension of MAX-SAT. It sits between the classical SAT problem and the MAX-SAT problem. While the classical SAT problem requires all clauses to be satisfied, MAX-SAT seeks the maximum number of clauses that can be satisfied, and partial MAX-SAT relaxes this requirement by having certain clauses marked as relaxable and others as nonrelaxable [29].

Obviously, from the above analyses, we can see that there are two limitations in the current research on role-mining optimization. One limitation is that none of them consider the separation-of-duty constraint. If a constructed RBAC system lacks several necessary constraints, then a series of security requirements of the organization are not realized. The second limitation is that they lack security detections for role authorization in the constructed RBAC system. Different results may violate the separation-of-duty constraint and/or the least-privileges principle.

2.3. Preliminaries and Basic Problem Descriptions

2.3.1. Basic Elements of Role Engineering

Conventional role engineering contains the following basic elements [5–7]:

1. \( U, P, R, S, DC, UPA, UA, PA, \) and \( RH \) are the basic elements of the RBAC model, which represent a set of users, a set of permissions, a set of roles, a set of sessions, a set of constraints, a many-to-many mapping of user-permission assignments, a many-to-many mapping of role-permission assignments, and a partial-order set on roles, respectively;

2. \( Roles(u), Permissions(r), \) and \( Perms(u) \) represent the mapping function of user \( u \) onto a set of roles, the mapping function of role \( r \) onto a set of permissions, and the mapping function of user \( u \) onto a set of permissions, respectively;

3. \( user\_session(s), roles\_session(s), \) and \( perms\_session(r) \) represent the mapping function of session \( s \) onto a user, the mapping function of session \( s \) onto a set of roles, and the mapping function of role \( r \) onto a set of permissions in a session, respectively.

2.3.2. Separation-of-Duty Constraint

Role-static separation-of-duty constraint (k-n RSSOD) [22] states that, given a task consisting of \( n \) roles \( r_1, r_2, \ldots, r_n \), at least \( k \) users are required together to perform it and have all the \( n \) roles; any set of \( (k - 1) \) users should not be able to have all the roles. It is expressed as \( rssod < [r_1, r_2, \ldots, r_n], k > \), where each \( r_i \) is a role, and \( n \) and \( k \) are integers, such that \( 2 \leq k \leq n \). It can be formalized as follows:

\[
\forall (u_1, u_2, \ldots, u_{k-1}) \in U : \{r_1, r_2, \ldots, r_n\} \bigcup_{i=1}^{k-1} Roles(u_i)
\]  

(1)
2.3.3. Basic RMP Problem

Given a set of users \( U \), a set of permissions \( P \), and a user-permission assignments matrix \( UPA \), the basic RMP [17] aims to find a set of roles \( R \), a user-role assignments matrix \( UA \), and a role–permission assignments matrix \( PA \), such that the \( UA \) and \( PA \) are consistent with the \( UPA \), and the number of mining roles \( |R| \) is minimized. It can be formalized as follows:

\[
\begin{align*}
\min & \{ |R| \} \\
\text{s.t.} & \quad UPA \preceq PA = UPA
\end{align*}
\]  

(2)

2.3.4. UAQ Problem

The UAQ [27] is a four-tuple \(< s, P_{lb}, P_{ub}, match>\), where \( s \in S \), \( P_{lb} \), and \( P_{ub} \) are the lower and upper bound for the set of requested permissions, respectively; \( P_{lb} \subseteq P_{ub} \subseteq P \); and \( match \in \{ \text{min}, \text{max}, \text{exact} \} \) indicates the optimization objective. It can be formalized as follows:

\[
\text{match} = \begin{cases} 
\text{min. if } \forall s' \in (S \setminus \{s\}), \exists (r, r') \in R, \\
\forall r \in R, \forall s \in S, \forall r' \in R, \forall s' \in S, \exists (r, r') \in R, \\
\text{max. if } \forall s' \in (S \setminus \{s\}), \exists (r, r') \in R, \\
\forall r \in R, \forall s \in S, \forall r' \in R, \forall s' \in S, \exists (r, r') \in R, \\
\text{exact. if } P_{lb} = \forall r \in R, \forall s \in S, \forall r' \in R, \forall s' \in S, \exists (r, r') \in R.
\end{cases}
\]

(3)

For the readability, we summarize the main symbols used in the paper in Table 1.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>(&lt; s, P_{lb}, P_{ub}, match&gt;)</td>
<td>The four tuple of permissions request</td>
</tr>
<tr>
<td>SSOD</td>
<td>Role-static separation-of-duty constraint</td>
</tr>
<tr>
<td>SMER</td>
<td>Static mutually exclusive roles</td>
</tr>
<tr>
<td>rssod(&lt; r_1, r_2, \ldots, r_m, k&gt;)</td>
<td>The ( k )-( n ) SSOD constraint</td>
</tr>
<tr>
<td>smer(&lt; r_1, r_2, \ldots, r_m, t&gt;)</td>
<td>The ( t )-( m ) SMER constraint</td>
</tr>
<tr>
<td>E</td>
<td>The set of SSOD constraints</td>
</tr>
<tr>
<td>C</td>
<td>The set of SMER constraints</td>
</tr>
<tr>
<td>( p, \neg p )</td>
<td>The literal and its negation with respect to permission ( p )</td>
</tr>
<tr>
<td>( r, \neg r )</td>
<td>The literal and its negation with respect to role ( r )</td>
</tr>
</tbody>
</table>

3. Proposed Method

In this section, we propose a novel method, RMO_SODSDA, which includes two aspects: (1) satisfying separation-of-duty constraints, and (2) ensuring authorization security.

3.1. Satisfying Separation-of-Duty Constraints

Separation-of-duty constraints can be effectively enforced by mutually exclusive constraints. In order to satisfy these constraints, we constructed different variants of mutually exclusive roles to implement RSSOD based on any unconstrained role-mining algorithm for basic RMP, including algorithm design and description, computational complexity, and correctness analysis. First, we present the following definitions.

**Definition 1.** (static and soft mutually exclusive roles, \( t \)-\( m \) SMER) Given \( m \) roles \( r_1, r_2, \ldots, r_m \), a \( t \)-\( m \) SMER states that no user is allowed to have \( t \) or more of these \( m \) roles. It is expressed as \( \text{smer} < r_1, r_2, \ldots, r_m, t > \), where each \( r_i \) is a role, and \( m \) and \( t \) are integers, such that \( 2 \leq k \leq n \). A \( t \)-\( m \) SMER can be formalized as follows:
\[ \forall u \in U : |\{r_1, r_2, \ldots, r_m\} \cap \text{Roles}(u)| < t \]  

(4)

It has been shown that any \(t\)-\(m\) SMER can be equivalently represented using a set of \(t\)-\(t\) SMER [24].

**Definition 2.** (static and hard mutually exclusive roles, \(t\)-\(t\) SMER) Given \(t\) roles \(r_1, r_2, \ldots, r_t\), a \(t\)-\(t\) SMER states that no user is allowed to have these \(t\) roles. It is expressed as \(\text{smer}<\{r_1, r_2, \ldots, r_t\}, t>\), where each \(r_i\) is a role, and \(t\) is an integer, such that \(t \geq 2\). A \(t\)-\(t\) SMER can be formalized as follows:

\[ \forall u \in U : \{r_1, r_2, \ldots, r_t\} \cap \text{Roles}(u) < t \]  

(5)

**Definition 3.** (role-mining problem with RSSOD constraints) Given a set of users \(U\), a set of permissions \(P\), a user-permission assignments matrix \(\text{UPA}\), and a set of RSSOD constraints \(E = \{e_1, e_2, \ldots\}\), find a set of roles \(R\), a user-role assignments matrix \(\text{URA}\), a role–permission assignments matrix \(\text{P A}\), and a set of SMER constraints \(C\), such that the \(\text{URA}\) and \(\text{P A}\) are consistent with the \(\text{UPA}\), \(C\) enforces \(E\), and the number of mining roles \(|R|\) is minimized. It can be formalized as follows:

\[
\begin{align*}
\min & |R| \\
\text{URA} \otimes \text{P A} & = \text{UPA} \\
C & \text{ enforces } E
\end{align*}
\]  

(6)

3.1.1. Preprocessing

Any role-mining algorithm that takes no RSSOD constraint into consideration is used for mining minimal roles in Algorithm 1.

**Algorithm 1.** Preprocessing

| **Input:** | original matrix \(\text{UPA}\) |
| **Output:** | decomposed matrix \(\text{URA}\), \(\text{P A}\), candidate set \(\text{CR}\) of roles |

The Fast Miner method and Boolean matrix decomposition are adopted to derive \(\text{CR}\) and configure RBAC, such that

\[
\begin{align*}
\min & |\text{CR}| \\
\text{URA} \otimes \text{P A} & = \text{UPA}
\end{align*}
\]

3.1.2. Construction of 2–2 SMER Constraints

A 2–2 SMER constraint can be expressed as \(c=<\{r_i, r_j\}, 2>\), which means that \(r_i\) and \(r_j\) are mutually exclusive, that is, the two roles cannot be assigned to any user. Although 2–2 SMER constraints are considered to be quite restrictive, they are sufficient to enforce any enforceable RSSOD constraint. RSSOD constraint \(e_i\) is nonenforceable in one of the following cases: (1) a single role in the \(\text{P A}\) matrix owns all permissions in \(e_i\), and (2) candidate matrix \(\text{URA}\) already has user assignments violating the constraints.
Algorithm 2. Construction of 2–2 SMER constraints

**Input:** UA, CR, set E of k-n RSSOD constraints

**Output:** set C of 2–2 SMER constraints

1. \( C = \emptyset \);
2. **for** each \( e_i \) in E **do**
3. \( P(e_i) \) denotes the permissions covered by \( e_i \);
4. \( \exists S = (r_1, r_2, \ldots) \subseteq CR : (\forall r \in S : Permissions(r) \cap P(e_i) \neq \emptyset) \)
5. **if** \( \exists r \in S : Permissions(r) \supseteq P(e_i) \) **then**
6. \( e_i \) is nonenforceable;
7. **continue**;
8. **end if**
9. **for** each pair of roles \( (r_i, r_j) \) in S **do**
10. **if** \( Permissions(r_i) \nsubseteq Permissions(r_j) \land Permissions(r_j) \nsubseteq Permissions(r_i) \) **then**
11. \( C = \{smer<r_i, r_j>, 2\} \cup C \);
12. **else**
13. \( e_i \) is nonenforceable with 2–2 SMER constraints;
14. **end if**
15. **end for**
16. **end for**

In Algorithm 2, the inputs are the above UA, CR, and set \( E = \{e_1, e_2, \ldots \} \) of k-n RSSOD constraints, and the output is set \( C = \{c_1, c_2, \ldots \} \) of 2–2 SMER constraints, such that \( C \) enforces \( E \). In order to construct 2–2 SMER constraints, it is necessary to find a set of roles \( S \) such that each role in \( S \) at least has permissions in \( e_i \) (line 4). Then, if any role has all the permissions in \( e_i \), we declare \( e_i \) as nonenforceable (lines 5–8); otherwise, we analyze each pair of roles \( (r_i, r_j) \) to determine whether they are valid 2–2 SMER constraints. If \( r_i \) and \( r_j \) are mutually exclusive roles, each one contains at least a mutually exclusive permission that must be absent in the other (lines 9–17).

Computational complexity: constructing 2–2 SMER constraints for an RSSOD can be done in polynomial time. If the number of roles in set \( S \) for the RSSOD is \( m \), then the time for deciding whether the RSSOD constraint is enforceable or not is \( O(m) \), and the time for constructing mutually exclusive roles is \( O(m^2) \); thus, the total time complexity is \( (m^2) \). The smaller \( m \) gets, the more acceptable the algorithm becomes.

3.1.3. Construction of t-t SMER Constraints

However, many RSSOD constraints might become nonenforceable in some cases because of the restrictive 2–2 SMER constraints. Next, we present an approach for implementing RSSOD with t-t SMER constraints \( (t > 2) \), which is less restrictive as compared with only using 2–2 SMER constraints. It is necessary to find a maximum \( t \) for the given RSSOD constraints in Algorithm 3.
Algorithm 3. Construction of $t-t$ SMER constraints

**Input:** $UA$, $CR$, set $E$ of $k-n$ RSSOD constraints

**Output:** set $C$ of $t-t$ SMER constraints

1. $C = \emptyset$;
2. for each $e_i$ in $E$ do
3. \hspace{0.5cm} $P(e_i)$ denotes the permissions covered by $e_i$;
4. \hspace{0.5cm} $\exists (S = (r_1, r_2, \ldots)) \subseteq CR: (\forall r \in S: \text{Permissions}(r) \cap P(e_i) \neq \emptyset)$
5. \hspace{0.5cm} if $|S| < k$ then
6. \hspace{1cm} $e_i$ is nonenforceable;
7. \hspace{0.5cm} continue;
8. \hspace{0.5cm} else if $\exists r \in S: \text{Permissions}(r) \supseteq P(e_i)$ then
9. \hspace{1cm} $e_i$ is nonenforceable;
10. \hspace{0.5cm} continue;
11. \hspace{0.5cm} else if $\exists (r_i, r_j) \in S: (r_i, r_j) \in RH$ or $(r_j, r_i) \in RH$ then
12. \hspace{1cm} $e_i$ is nonenforceable;
13. \hspace{0.5cm} continue;
14. \hspace{0.5cm} else
15. \hspace{1cm} \hspace{0.5cm} $t=3$ to $\left\lceil \frac{|S|-1}{k-1} \right\rceil+1$ do
16. \hspace{1.5cm} \hspace{0.5cm} for any set $\{r_1', r_2', \ldots, r_t'\}$ in $S$ do
17. \hspace{2.0cm} \hspace{0.5cm} if $UA$ satisfies $\text{smer}<\{r_1', r_2', \ldots, r_t'\}, t>$ then
18. \hspace{2.5cm} \hspace{0.5cm} $C = \text{smer}<\{r_1', r_2', \ldots, r_t'\}, t> \cup C$;
19. \hspace{2.0cm} \hspace{0.5cm} else
20. \hspace{2.5cm} \hspace{0.5cm} $e_i$ is nonenforceable with $t-t$ SMER constraints;
21. \hspace{1.5cm} \hspace{0.5cm} end if
22. \hspace{1.5cm} \hspace{0.5cm} end for
23. \hspace{1cm} \hspace{0.5cm} end for
24. \hspace{0.5cm} end if
25. \hspace{0.5cm} end for

To construct $t-t$ SMER constraints for an RSSOD $e_i$, like in Algorithm 2, we should find set $S$ at first. Next, we determine whether $e_i$ is enforceable or not. In Algorithm 3, line 4 determines set $S$ from $CR$. Lines 5–6 indicate that if the number of roles in set $S$ is less than $k$ in $e_i$, it cannot be enforced. Lines 8–9 indicate that if any role in set $S$ has all the permissions in $e_i$, it cannot be enforced. Lines 11–12 show that if RH relationships exist in $S$, $e_i$ cannot be enforced. Lines 15–23 determine whether the $t-t$ SMER constraints are satisfied in matrix $UA$ and insert them into set $C$.

Computational complexity: if the number of roles in $S$ is $m$, the time for determining whether a constraint is enforceable or not is $O(m)$. Constructing $t-t$ SMER constraints depends on the double loops. The execution number of the first loop is $\left\lceil \frac{|S|-1}{k-1} \right\rceil$; in the second loop, for the particular $t$, it is necessary to obtain any $t$ roles from $S$, and the execution time is $O(C_t^m)$. Thus, the total time complexity is $O(m) + O(m \cdot C_t^m) = O(m \cdot C_t^m)$. If $m$ is small, the efficiency of the algorithm is acceptable.

3.1.4. Construction of $t-m$ SMER Constraints

Compared with $t-t$ SMER constraints, although $t-m$ SMER constraints ($2 < t < m$) are less restrictive, they are more flexible for applications. The differences between $t-m$ SMER and $t-t$ SMER are presented in Algorithm 4.
Theorem 1. Given a 2–m RSSOD constraint $e$, the execution number in the first loop is $\left\lceil \frac{n-1}{m-1} \right\rceil + 1$.

Statement 2. Constraints constructed by our algorithms can precisely enforce $k-n$ RSSOD if $t \leq m$.

Proof. We also analyze the statement from the two sides of sufficiency and necessity.

Sufficiency. When constraint $c_1 = \text{smer} < \{r_1, r_2, \ldots, r_m\}, 2>$ holds, that is, $\forall u \in U : |\text{Roles}(u) \cap \{r_1, r_2, \ldots, r_m\}| < 2$, this indicates that at least $m$ users are required together to have all roles of set $\{r_1, r_2, \ldots, r_m\}$. Any set of $(m-1)$ users should not be able to have all $m$ roles, that is, $\forall (u_1, u_2, \ldots, u_{m-1}) \in U : \cup_{i=1}^{m-1} \text{Roles}(u_i) \not\supseteq \{r_1, r_2, \ldots, r_m\}$. According to RSSOD, $c_1$ can enforce $e_1$.

Necessity. It is proved by contradiction. Assume that there exist $(m-1)$ users who can cover all the $m$ roles, that is, $\exists (u_1, u_2, \ldots, u_{m-1}) \in U : \cup_{i=1}^{m-1} \text{Roles}(u_i) \supseteq \{r_1, r_2, \ldots, r_m\}$. In other words, there exists some user who has two of the $n$ roles, that is, $\exists u \in U : |\text{Roles}(u) \cap \{r_1, r_2, \ldots, r_m\}| = 2$, which is contradictory to constraint $\text{smer} < \{r_1, r_2, \ldots, r_m\}, 2>$. Thus, the assumption is false. \(\square\)

Statement 3. Given a 2–m RSSOD constraint $e_2$, $t$-m SMER constraint $c_2$ can precisely enforce $e_2$ if and only if $t = m$.

Proof. We also analyze the statement from the two sides of sufficiency and necessity.

Sufficiency. When constraint $c_2 = \text{smer} < \{r_1, r_2, \ldots, r_m\}, m>$ holds, that is, $\forall u \in U : |\text{Roles}(u) \cap \{r_1, r_2, \ldots, r_m\}| < m$, this indicates that no user has all roles of the set $\{r_1, r_2, \ldots, r_m\}$. In other words, at least two users are required together to have all the $m$ roles. According to RSSOD, $c_2$ can enforce $e_2$.

Necessity. It is proved by contradiction. Assume that there exists some user who has all the $m$ roles, that is, $\exists u \in U : |\text{Roles}(u) \cap \{r_1, r_2, \ldots, r_m\}| = m$, which is contradictory to the constraint $\text{smer} < \{r_1, r_2, \ldots, r_m\}, m>$. Thus, the assumption is false. \(\square\)

Theorem 1. Given a $k-n$ RSSOD constraint ($k > 2$) and a candidate role set $CR$, where $k \leq n \leq |CR|$, $t$-m SMER constraints constructed by our algorithms can precisely enforce $k-n$ RSSOD if $t \leq \left\lceil \frac{n-1}{m-1} \right\rceil + 1$. 

\[\text{Algorithm 4. Construction of } t \text{-m SMER constraints}\]

\begin{verbatim}
Input: LIA, CR, set E of k-n RSSOD constraints
Output: set C of t-m SMER constraints

......
1. for t=3 to $\left\lceil \frac{n-1}{m-1} \right\rceil + 1$ do
2. m=(k-1)×(t-1)+1;
3. for any set $\{r_1', r_2', \ldots, r_m'\}$ in CR do
4. if LA satisfy $\text{smer} < \{r_1', r_2', \ldots, r_m'\}, t>$ then
5. $C_\text{t} = \{r_1', r_2', \ldots, r_m'\}, t> \cup C$;
6. else
7. $e_i$ is nonenforceable with t-m SMER constraints;
8. end if
9. end for
10. end for
......
\end{verbatim}

Computational complexity: constructing $t$-m SMER constraints depends on the double loops. The execution number in the first loop is $\left\lceil \frac{n-1}{m-1} \right\rceil + 2$; in the second loop, for the particular $m$, it is necessary to obtain any $m$ roles from $CR$. Thus, the execution time of the double loops is $O(2^n)$. The efficiency of the algorithm decreases obviously as the number of $CR$ increases.

3.1.5. Correctness Analysis

Statement 1. Given an $m$-m RSSOD constraint $e_1$, t-m SMER constraint $c_1$ can precisely enforce $e_1$ if and only if $t = 2$.

Proof. We analyze the statement from the two sides of sufficiency and necessity.

Sufficiency. When constraint $c_1 = \text{smer} < \{r_1, r_2, \ldots, r_m\}, 2>$ holds, that is, $\forall u \in U : |\text{Roles}(u) \cap \{r_1, r_2, \ldots, r_m\}| < 2$, this indicates that at least $m$ users are required together to have all roles of set $\{r_1, r_2, \ldots, r_m\}$. Any set of $(m-1)$ users should not be able to have all $m$ roles, that is, $\forall (u_1, u_2, \ldots, u_{m-1}) \in U : \cup_{i=1}^{m-1} \text{Roles}(u_i) \not\supseteq \{r_1, r_2, \ldots, r_m\}$. According to RSSOD, $c_1$ can enforce $e_1$.

Necessity. It is proved by contradiction. Assume that there exist $(m-1)$ users who can cover all the $m$ roles, that is, $\exists (u_1, u_2, \ldots, u_{m-1}) \in U : \cup_{i=1}^{m-1} \text{Roles}(u_i) \supseteq \{r_1, r_2, \ldots, r_m\}$. In other words, there exists some user who has two of the $n$ roles, that is, $\exists u \in U : |\text{Roles}(u) \cap \{r_1, r_2, \ldots, r_m\}| = 2$, which is contradictory to constraint $\text{smer} < \{r_1, r_2, \ldots, r_m\}, 2>$. Thus, the assumption is false. \(\square\)

Statement 2. Given a 2–m RSSOD constraint $e_2$, t-m SMER constraint $c_2$ can precisely enforce $e_2$ if and only if $t = m$.

Proof. We also analyze the statement from the two sides of sufficiency and necessity.

Sufficiency. When constraint $c_2 = \text{smer} < \{r_1, r_2, \ldots, r_m\}, m>$ holds, that is, $\forall u \in U : |\text{Roles}(u) \cap \{r_1, r_2, \ldots, r_m\}| < m$, this indicates that no user has all roles of the set $\{r_1, r_2, \ldots, r_m\}$. In other words, at least two users are required together to have all the $m$ roles. According to RSSOD, $c_2$ can enforce $e_2$.

Necessity. It is proved by contradiction. Assume that there exists some user who has all the $m$ roles, that is, $\exists u \in U : |\text{Roles}(u) \cap \{r_1, r_2, \ldots, r_m\}| = m$, which is contradictory to the constraint $\text{smer} < \{r_1, r_2, \ldots, r_m\}, m>$. Thus, the assumption is false. \(\square\)

Theorem 1. Given a $k$-n RSSOD constraint ($k > 2$) and a candidate role set $CR$, where $k \leq n \leq |CR|$, t-m SMER constraints constructed by our algorithms can precisely enforce $k-n$ RSSOD if $t \leq \left\lceil \frac{n-1}{m-1} \right\rceil + 1$. 


Proof. According to Definition 1, any user has \((t−1)\) roles at most, and any \((k−1)\) users have \((k−1)\) \(\times (t−1)\) roles at most. It is also proved by contradiction. Without loss of generality, \(t=\left[\frac{kn−1}{k−1}\right]+1+1\), we assume that the RSSOD constraint holds, that is, the number of roles assigned to \((k−1)\) users is \((k−1)\times \left[\frac{kn−1}{k−1}\right]+2−1\) \(\approx n−1+k−1=n+k−2>n\), which breaches the \(k\)-\(n\) RSSOD. Thus, the assumption is false and the theorem is proved. \(\square\)

Example 1. Constructing \(t\)-\(m\) SMER constraints based on our method can enforce \(\text{rssod}\{r_1,r_2,r_3,r_4,r_5\},k\) with different \(k\) in Table 2, where \(2\leq k \leq 5\).

<p>| Table 2. Static and soft mutually exclusive roles ((t)-(m) SMERs) enforce (k)-(n) role-static separation-of-duty constraint (RSSOD). |</p>
<table>
<thead>
<tr>
<th>RSSOD</th>
<th>SMERs</th>
</tr>
</thead>
<tbody>
<tr>
<td>(e_1=\text{rssod}{r_1,r_2,r_3,r_4,r_5},2)</td>
<td>(C_1=[\text{smer}{r_1,r_2,r_3,r_4,r_5},5])</td>
</tr>
<tr>
<td>(e_2=\text{rssod}{r_1,r_2,r_3,r_4,r_5},3)</td>
<td>(C_2=[\text{smer}{r_1,r_2,r_3},2,\text{smer}{r_1,r_2,r_4},2,\text{smer}{r_1,r_2,r_5},2,\text{smer}{r_1,r_3,r_4},2,\text{smer}{r_1,r_3,r_5},2,\text{smer}{r_1,r_4,r_5},2])</td>
</tr>
<tr>
<td>(e_3=\text{rssod}{r_1,r_2,r_3,r_4,r_5},4)</td>
<td>(C_3=[\text{smer}{r_1,r_2,r_3,r_4},2,\text{smer}{r_1,r_2,r_4,r_5},2,\text{smer}{r_1,r_2,r_3,r_5},2,\text{smer}{r_1,r_3,r_4,r_5},2])</td>
</tr>
<tr>
<td>(e_4=\text{rssod}{r_1,r_2,r_3,r_4,r_5},5)</td>
<td>(C_4=[\text{smer}{r_1,r_2,r_3,r_4,r_5},2])</td>
</tr>
</tbody>
</table>

3.2. Ensuring Authorization Security

In order to ensure the security of authorizations, we converted the authorization-query problem into partial MAX-SAT as follows: (1) authorization logics and constraints are transformed into hard clauses; (2) the updating algorithm is adopted to transform the requested permissions with different matches into soft clauses; and (3) recursive and encoding algorithms are utilized to satisfy all hard clauses and as many soft clauses as possible. First, we present the definition with respect to the partial MAX-SAT.

Definition 4. (Boolean logical descriptions of partial MAX-SAT):

1. Basic elements: some elements are presented using Boolean variables \(x_1, x_2, \ldots, x_i, \ldots, x_j\), or \(\neg x_i\) (the negation of \(x_i\)), which is called a literal logic; the instance formalized as \(x_1 \lor x_2 \lor \ldots \lor x_n\) using \(n\) literals \(x_i\) is called a clause; and sentence \(c_1 \land c_2 \land \ldots \land c_m\) using \(m\) clauses \(c_i\) is called the conjunctive normal form, which is expressed as \(\text{cnf}(c_1,c_2,\ldots,c_m)\);
2. Hard clause: \(c_1, c_2, \ldots, c_m\) are hard clauses and satisfiable, if and only if there exist truth assignments such that the value of \(\text{cnf}(c_1,c_2,\ldots,c_m)\) is 1. All hard clauses are set as HC;
3. Soft clause: the sentences that do not satisfy hard clauses are soft clauses. All soft clauses are set as SC.

To transform the authorization-query request into the partial MAX-SAT, we represent permission \(p\) and role \(r\) with literal \(p\) and \(r\) respectively, and we describe them and their logical negations as follows:

1. \(\overline{p}: \forall p \in P, \exists r \in R, \exists s \in S, p \in \bigcup_{s \in \text{roles session}(s)} \text{perms session}(r)\);
2. \(\neg \overline{p}: \forall p \in P, \exists r \in R, \exists s \in S, p \notin \bigcup_{s \in \text{roles session}(s)} \text{perms session}(r)\);
3. \(\overline{r}: \forall r \in R, \exists s \in S, r \in \text{roles session}(s)\);
4. \(\neg \overline{r}: \forall r \in R, \exists s \in S, r \notin \text{roles session}(s)\).

3.2.1. Preprocessing

To transform static-authorization logics and dynamic mutual role constraints into hard clauses, we describe the transforming rules as follows:
∀\ p ∈ P, ∃(r, r') ∈ R : ((r, r') ∈ RH) ∧ ((r', p) ∈ PA) ⇒ ¬\ t ∨ \ p;
(1)

∀\ p ∈ P, ∃(r', r_1, r_2, ...) ∈ R : ((r', p) ∈ PA) ∧ (\ (r_1, r') ∈ RH) ∧ ... ⇒ ¬\ p ∨ r_1 ∨ r_2 ∨ ...;
(2)

∀\ r ∈ R, ∃r' ∈ R, ∃(p_1, p_2, ...) ∈ P : ((r, r') ∈ RH) ∧ ((r', p_1) ∈ PA) ∧ (\ (r', p_2) ∈ PA) ∧ ... ⇒ \ r_1 ∨ ¬\ p_1 ∨ ¬\ p_2 ∨ ...;
(3)

∀\ r ∈ R, ∃s ∈ S : (user_session(s), r) \notin UA ⇒ ¬\ r;
(4)

∀\ r' ∈ \{r_1, r_2, ..., r_m\} ⊆ R, ∃s ∈ S, ∃t \geq 2 : (dmer(\{r_1, r_2, ..., r_m\}, t) \in C) ∧ 
\left(\left|\{r_1, r_2, ..., r_m\} \cap \text{roles}_{session}(s)\right| < t\right) ⇒ r'.
(5)

3.2.2. Updating Clauses

To reflect the least-privileges principle, we say that permissions in \(P_{lb}\) must be satisfied and available in a single session, which can be formalized as follows:

\(∀\ p ∈ P, ∃s ∈ S, ∀r ∈ R : p ∈ P_{lb} ⇒ p ∈ \bigcup_{\text{roles}_{session}(s)} \text{perms}_{session}(r)\).

Similarly, permissions not in \(P_{ub}\) must not be satisfied or available in a single session, which can be formalized as follows:

\(∀\ p ∈ P, ∃s ∈ S, ∀r ∈ R : p ∈ (P\setminus P_{ub}) ⇒ p ∉ \bigcup_{\text{roles}_{session}(s)} \text{perms}_{session}(r)\).

Different cases can be easily transformed into the partial MAX-SAT. Essentially, the partial MAX-SAT problem is a special extension of the MAX-SAT problem, whereby some clauses are relaxable and others are hard clauses. It aims to satisfy all hard clauses as well as a maximum number of soft ones. To translate the authorization-query problem to a partial MAX-SAT instance, all permissions in \(P_{lb}\) should be inserted into \(HC\), and all permissions in \((P_{ub}\setminus P_{lb})\) should be inserted into \(SC\). Next, to satisfy the \(max\) match, any permission in \((P_{ub}\setminus P_{lb})\) is initially set to be true, as only permission variables can be used in the permissions request and the partial MAX-SAT is to satisfy the largest number of soft clauses; to satisfy the \(min\) match, on the other hand, any permission in \((P_{ub}\setminus P_{lb})\) is initially set to be false. Thus, hard and soft clauses are updated in Algorithm 5.

**Algorithm 5.** Updating clauses.

Input: the UAQ <s, \(P_{lb}, P_{ub}, \text{match}\)>, preprocessed results \(HC, SC\)

Output: updated results \(HC, SC\)

1. for each \(p\) in \(P_{lb}\) do
2. \(HC = HC \cup \{p\}\);
3. end for
4. for each \(p\) in \(P\setminus P_{ub}\) do
5. \(HC = HC \cup \{¬p\}\);
6. end for
7. for each \(p\) in \(P_{ub}\setminus P_{lb}\) do
8. if match = \(min\) then
9. \(SC = SC \cup \{¬p\}\);
10. else if match = \(max\) then
11. \(SC = SC \cup \{p\}\);
12. else
13. \(SC = SC \cup \cup\);
14. end if
15. end for
In Algorithm 5, the two cases are reflected in lines 1–6. If permission \( p \) is in \( P_{lb} \), we represent \( p \) with literal \(-p\) and insert it into \( HC \). If permission \( p \) is not in \( P_{ub} \), we insert literal \( -p \) into \( HC \). If the requested permissions are not in \( P_{lb} \), but in \( P_{ub} \), lines 7–15 transform them into soft clauses according to the different matches. We insert literal \( -p \) or \( -p \) into \( SC \) when the match objective equals to \( \min \) or \( \max \), and we do not update \( SC \) when the match objective equals to \( \text{exact} \).

### 3.2.3. Assigning Truth for Clauses

To satisfy all hard clauses in \( HC \), it is necessary to search for truth assignments and make sure all clauses are true. The recursive process is described in Algorithm 6.

---

**Algorithm 6.** Assigning truth for hard clauses.

**Input:** \( \text{len}(c) \)

**Output:** true or false

1. Randomly choose clause \( c \) from \( HC \) and use \( \text{len}(c) \) as the length of \( c \);
2. if \( \text{len}(c) = 1 \) then
   3. \( c = 1 \);
   4. \( HC = HC - \{c\} \);
5. for each clause, including \( c' \lor c \) in \( HC \), do
   6. \( HC = HC - \{c' \lor c\} \);
7. end for
8. for each clause, including \( c'' \lor -c \) in \( HC \), do
   9. \( HC = HC - \{c'' \lor -c\} \);
10. end for
11. return true;
12. else
13. Assigning truth for hard clauses (\( \text{len}(c)-1 \));
14. return false;
15. end if

---

In Algorithm 6, lines 2–11 indicate that if chosen clause \( c \) consists of only one literal logic, we simplify \( HC \) according to the two different cases in each iteration: when a clause including \( c' \lor c \) exists, we remove it from \( HC \); when a clause including \( c'' \lor -c \) exists, we remove it from \( HC \) and insert \( c'' \) into \( HC \). Next, if \( \text{len}(c) > 1 \), line 13 recursively calls Algorithm 6.

**Theorem 2.** For each given clause \( c \), there is a truth assignment for \( c \) using Algorithm 6.

**Proof.** On the basis of Sections 3.2.1 and 3.2.2, we can search for the truth assignment from line 2 of the algorithm using the method of mathematical induction, and detailed descriptions are omitted owing to limited space. □

### 3.2.4. Encoding and Querying Clauses

To satisfy as many clauses as possible, soft clauses are encoded and queried in the following steps:

**Step 1.** For given set of soft clauses \( SC = \{sc_1, sc_2, \ldots, sc_n\} \), introduce \( n \) new blocking variables \( b_i (1 \leq i \leq n) \);

**Step 2.** Construct new clause set \( SC^b = \{sc_1 \lor b_1, sc_2 \lor b_2, \ldots, sc_n \lor b_n\} \), where clause \( sc_i \lor b_i \) is considered as \( b_i \) copies of \( sc_i \);

**Step 3.** Search for minimal integer \( k \), such that \( SC^b \) is true and \( \sum_{i=1}^{n} b_i \leq k \);

**Step 4.** Consider \( HC \cup SC^b \) as parameters, take them into the QMaxSAT [39], and output all true clauses from \( SC \).
Example 2. Given set $P$ of permissions, set $R$ of roles, role–hierarchy relationships $RH$, role–permission assignment relationships $PA$, and set $C$ of constraints, where $P = \{p_0,p_1,p_2,p_3,p_4,p_5,p_6,p_7\}$, $R = \{r_0,r_1,r_2\}$, $RH = \{r_0 \geq r_2\}$, $PA = \{(r_0,p_0),(r_0,p_1),(r_0,p_2),(r_0,p_3),(r_0,p_4),(r_0,p_5),(r_1,p_6),(r_1,p_7),(r_2,p_2),(r_2,p_6)\}$, and $C = \{dmer\langle r_0,r_1 \rangle, 2, s \rangle\). In the preprocessing stage, hard clauses are constructed by the transforming rules, and results are listed in Table 3.

Table 3. Hard clauses.

<table>
<thead>
<tr>
<th>Authorization Logics and Constraints</th>
<th>HC</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\forall p \in P, \exists (r,r') \in R: \langle (r,r') \in RH \rangle \land \langle (p,r') \in PA \rangle$</td>
<td>$\neg r_0 \lor p_0 \lor \neg r_0 \lor p_1, \neg r_0 \lor p_2, \neg r_0 \lor p_4$</td>
</tr>
<tr>
<td>$\forall p \in P, \exists (r',r_1,r_2,\ldots) \in R: \langle (p,r') \in PA \rangle \land \langle (r_1,r') \in RH \rangle \land \langle (r_2,r') \in RH \rangle \ldots$</td>
<td>$\neg p_0 \lor r_0 \lor \neg p_1 \lor r_0 \lor \neg p_2 \lor r_0 \lor r_2 \lor \neg p_3 \lor r_1, \neg p_4 \lor r_0 \lor \neg p_5 \lor r_0 \lor r_2 \lor \neg p_6 \lor r_1, \neg r_0 \lor r_0 \lor \neg r_1 \lor r_2 \lor \neg r_2 \lor r_1$</td>
</tr>
<tr>
<td>$\forall r \in R, \exists r' \in R, \exists \langle p_1, p_2, \ldots \rangle \in P: \langle (r,r') \in RH \rangle \land \langle (p_1,r') \in PA \rangle \land \langle (p_2,r') \in PA \rangle \ldots$</td>
<td>$\neg r_0 \lor \neg p_0 \lor \neg r_1 \lor p_2 \lor \neg r_2 \lor p_4 \lor \neg r_3 \lor p_5 \lor r_1, \neg r_0 \lor r_3 \lor \neg r_2 \lor r_7 \lor \neg r_2 \lor p_6$</td>
</tr>
<tr>
<td>$\forall s \in S, \exists \langle r_1, r_2, \ldots \rangle \in R: \langle \text{dmer} \langle r_1, r_2, \ldots, 2, s \rangle \rangle \in C$</td>
<td>$\neg r_0 \lor \neg r_1$</td>
</tr>
</tbody>
</table>

Suppose there are three authorization requests: (1) $P_{ub} = \Phi, P_{ab} = \{p_2,p_3,p_6\}$, $match = \max$; (2) $P_{ub} = \{p_2,p_3,p_6\}, P_{ab} = P$, $match = \min$; (3) $P_{ub} = \{p_2,p_3,p_6\}, P_{ab} = \{p_2,p_3,p_6\}$, $match = \text{exact}$. Then, HC and SC are, respectively, updated in Algorithms 5 and 6; the results are listed in Table 4, and different sets of activated roles are listed in Table 5.

Table 4. Updated clauses.

<table>
<thead>
<tr>
<th>Permissions Request</th>
<th>HC</th>
<th>SC</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\langle \langle \Phi, [p_2,p_3,p_6], \max \rangle$</td>
<td>$HC \cup {\neg p_0, \neg p_1, \neg p_4, \neg p_5, \neg p_7} \cup {p_2, p_3, p_6}$</td>
<td></td>
</tr>
<tr>
<td>$\langle \langle [p_2,p_3,p_6], \min \rangle$</td>
<td>$HC \cup {p_2, p_3, p_6}$</td>
<td>${\neg p_0, \neg p_1, \neg p_4, \neg p_5, \neg p_7}$</td>
</tr>
<tr>
<td>$\langle \langle [p_2,p_3,p_6], \text{exact} \rangle$</td>
<td>$HC \cup {p_2, p_3, p_6}$</td>
<td>${\neg p_0, \neg p_1, \neg p_4, \neg p_5, \neg p_7}$</td>
</tr>
</tbody>
</table>

Table 5. Activated roles.

<table>
<thead>
<tr>
<th>Permissions Request</th>
<th>Literal Roles</th>
<th>Role Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\langle \langle \Phi, [p_2,p_3,p_6], \max \rangle$</td>
<td>$\neg r_0, \neg r_1, r_2$</td>
<td>${r_2}$</td>
</tr>
<tr>
<td>$\langle \langle [p_2,p_3,p_6], \min \rangle$</td>
<td>$\neg r_0, r_1, r_2$</td>
<td>${r_1,r_2}$</td>
</tr>
<tr>
<td>$\langle \langle [p_2,p_3,p_6], \text{exact} \rangle$</td>
<td>$\neg r_0, \neg r_1, \neg r_2$</td>
<td>$\Phi$</td>
</tr>
</tbody>
</table>

4. Experiment Evaluations

In this section, two groups of experiments were carried out. The first group of experiments was to evaluate the effectiveness of RMO_SODSDA, and the second was to evaluate its efficiency. All experiments were implemented on a standard desktop PC with an Intel i5~7400 CPU, 4 GB RAM, and 160 GB hard disk, running a 64-bit Windows 7 operating system. All simulations were compiled and executed in Eclipse IDE for a Java Developers environment.
4.1. RMO_SODSDA Effectiveness

4.1.1. Datasets and Experiment Setup

Datasets: we considered nine real-world datasets from the work of [14]. Meanwhile, in the unconstrained preprocessing stage, regular mining tool RMiner [39] was used for mining roles. The results including the number of roles and execution time are shown in Table 6.

Table 6. Datasets.

| Dataset       | |U| | |P| | |CR| | Execution Time (s) |
|---------------|---------|---------|---------|---------|
| Americas-large| 3485    | 10127   | 423     | 78.78   |
| Americas-small| 3477    | 1587    | 213     | 6.31    |
| Apj           | 2044    | 1164    | 456     | 5.60    |
| Customer      | 10961   | 284     | 276     | 4.66    |
| Domino        | 79      | 231     | 20      | 0.01    |
| Emea          | 35      | 3046    | 34      | 0.02    |
| Firewall1     | 365     | 709     | 69      | 0.11    |
| Firewall2     | 325     | 590     | 10      | 0.15    |
| Healthcare    | 46      | 46      | 15      | 0.01    |

Experiment setup: constraints were constructed synthetically. According to the length of constraint enforcement, we studied four different kinds of \(k-n\) RSSOD: 2–2 RSSOD, 2–3 RSSOD, 3–5 RSSOD, and 5–10 RSSOD, where \(n\) permissions were randomly chosen from the set of all permissions. According to the amount of constraint enforcement, we chose 30 and 50 as the number of each kind of \(k-n\) RSSOD, respectively.

However, after analyzing these datasets, we found that the Domino, Firewall1, Firewall2, and Healthcare datasets could not be used to represent the performance of role-mining algorithms with RSSOD constraints. Some user sets in these datasets violate the constraints, and they have all permissions assigned to them. Thus, valid RSSOD constraints cannot be generated from them.

4.1.2. Evaluation Measures

We considered the average number of constructing SMER constraints, execution time, and percentage of RSSOD constraints that can be successfully enforced as evaluation measures.

4.1.3. Experiment Results

To evaluate the effectiveness of RMO_SODSDA, we considered preprocessing results \(UA\), \(CR\), and the \(k-n\) RSSOD constraints as inputs, and repeatedly implemented the experiments 20 times. The experiment results are shown in Tables 7–10, where \(E_{k-n}RSSOD\) represents the set of RSSOD constraints, and \(C_{t-m}SMER\) represents the set of SMER constraints.

Table 7. Results of SMER constructed from 2–2 RSSOD.

| Dataset       | \(|E_{22}RSSOD|= 30\) | \(|E_{2-2}RSSOD|= 50\) |
|---------------|---------------------|---------------------|
|               | \(C_{2-2}SMER\) | \(C_{t-t}SMER\) | \(C_{t-m}SMER\) | \(C_{2-2}SMER\) | \(C_{t-t}SMER\) | \(C_{t-m}SMER\) |
| Americas-large| 984(95%)           | 984(95%)           | 982(96%)         | 2461(92%)        | 2568(92%)        | 2574(93%)        |
| Americas-small| 441(80%)           | 439(80%)           | 436(80%)         | 1279(84%)        | 1272(84%)        | 1285(84%)        |
| Apj           | 40(80%)            | 39(80%)            | 37(80%)          | 124(84%)         | 124(84%)         | 124(84%)         |
| Customer      | 19(95%)            | 15(80%)            | 11(80%)          | 48(96%)          | 36(78%)          | 33(76%)          |
| Emea          | 57(80%)            | 54(80%)            | 51(80%)          | 142(84%)         | 138(84%)         | 135(83%)         |
Table 8. Results of SMER constructed from 2–3 RSSOD.

| Dataset      | $|E_{2-3 RSSOD}| = 30$ | $|E_{2-3 RSSOD}| = 50$ |
|--------------|-----------------|-----------------|
| Americas-large | C$_{2-2}$ SMER  | 3307(100%)       | 3306(100%)       |
|              | C$_{t-t}$ SMER  | 3306(100%)       | 3304(100%)       |
|              | C$_{t-m}$ SMER  | 993(80%)         | 991(80%)         |
| Americas-small|                | 995(80%)         | 993(80%)         |
|              |                | 991(80%)         | 3365(85%)        |
| Apj          |                | 146(100%)        | 121(100%)        |
| Customer     |                | 32(60%)          | 19(100%)         |
| Emea         |                | 172(95%)         | 163(95%)         |

Table 9. Results of SMER constructed from 3–5 RSSOD.

| Dataset      | $|E_{3-5 RSSOD}| = 30$ | $|E_{3-5 RSSOD}| = 50$ |
|--------------|-----------------|-----------------|
| Americas-large | C$_{2-2}$ SMER  | 11,335(91%)     | 11,335(91%)     |
|              | C$_{t-t}$ SMER  | 11,335(91%)     | 11,335(91%)     |
| Americas-small|                | 553(95%)        | 572(95%)        |
| Apj          |                | 1210(50%)       | 1199(50%)       |
| Customer     |                | 46(17%)         | 372(100%)       |
| Emea         |                | 696(100%)       | 698(100%)       |

Table 10. Results of SMER constructed from 5–10 RSSOD.

| Dataset      | $|E_{5-10 RSSOD}| = 30$ | $|E_{5-10 RSSOD}| = 50$ |
|--------------|-----------------|-----------------|
| Americas-large | C$_{2-2}$ SMER  | 24,908(70%)     | 24,908(70%)     |
|              | C$_{t-t}$ SMER  | 24,908(70%)     | 24,908(70%)     |
| Americas-small|                | 1307(15%)       | 1307(15%)       |
| Apj          |                | 1459(85%)       | 1614(85%)       |
| Customer     |                | 1591(100%)      | 1594(100%)      |
| Emea         |                | 1591(100%)      | 1594(100%)      |

Firstly, by analyzing the results in Tables 7–10, it was observed that there was a certain number of RSSOD constraints that could not be enforced when we constructed 2–2 SMER, $t$-$t$ SMER, or $t$-$m$ SMER constraints. The less that inefficient RSSOD constraints were enforced (in other words, the less the percentage in the bracket), the fewer the constructing SMER constraints.

Second, Tables 7–10 show that the process of constructing 2–2 SMER constraints is more difficult than the other two processes, because the 2–2 SMER was more restricted than the others. We take the Customer dataset as an example. In Table 9, percentages of successfully constructing 2–2 SMER constraints are 17% with 30 RSSOD and 14% with 50 RSSOD, while percentages of constructing the other two are both 100%. In Table 10, percentages of successfully constructing 2–2 SMER constraints are only 5% and 2%.

A further observation is that the number of constructing SMER constraints tended to increase as the number of RSSOD constraints increased in the five datasets. We take the Americas-large dataset as an example. In Table 7, when the number of RSSOD changed from 30 to 50, the number of constructing SMER increased from 984 to 2461 and 2568, respectively.

Finally, it can be observed from Tables 7–10 that the number of constructing SMER increased as the length of RSSOD constraints increased. We take the Americas-large dataset as an example. If the number of RSSOD constraints equaled to 30, then the number of constructing SMER remarkably increased from 984 to 24,908. If the number of RSSOD constraints equaled to 50, then the number of constructing SMER changed remarkably from 2461 to 62,058.

Next, the average time for constructing SMER constraints from different kinds of RSSOD constraints is shown in Tables 11–14, from which it is observed that execution time increased as the number and length of RSSOD constraints increased. Taking the Americas-large dataset as an example, the number of RSSOD changed from 30 to 50, and the length of RSSOD changed from 2 to 10. The time for
constructing 2–2 SMER constraints was 139, 366, 505, 1203, 2022, 5762, 6164, and 14,667 s, respectively. Similar to analyses for the number of constructing constraints, detailed discussions of execution time were omitted in this paper owing to limited space.

Table 11. Execution time with 2–2 RSSOD.

| Dataset      | $|E_{2-2 \text{ RSSOD}}| = 30$ | $|E_{2-2 \text{ RSSOD}}| = 50$ |
|--------------|-------------------------------|-------------------------------|
|              | $C_{2-2 \text{ SMER}}$ | $C_{t-t \text{ SMER}}$ | $C_{t-m \text{ SMER}}$ | $C_{2-2 \text{ SMER}}$ | $C_{t-t \text{ SMER}}$ | $C_{t-m \text{ SMER}}$ |
| Americas-large | 139.7 s          | 144.7 s          | 289.5 s          | 366.2 s          | 379.5 s          | 759.1 s          |
| Americas-small| 10.1 s           | 10.1 s           | 20.2 s           | 29.2 s           | 30.2 s           | 60.4 s           |
| Apj          | 0.3 s            | 0.4 s            | 0.8 s            | 0.9 s            | 1.2 s            | 2.4 s            |
| Customer     | 0.01 s           | 0.2 s            | 0.4 s            | 0.01 s           | 0.5 s            | 1.1 s            |
| Emea         | 0.02 s           | 0.02 s           | 0.04 s           | 0.05 s           | 0.06 s           | 0.12 s           |

Table 12. Execution time with 2–3 RSSOD.

| Dataset      | $|E_{2-3 \text{ RSSOD}}| = 30$ | $|E_{2-3 \text{ RSSOD}}| = 50$ |
|--------------|-------------------------------|-------------------------------|
|              | $C_{2-2 \text{ SMER}}$ | $C_{t-t \text{ SMER}}$ | $C_{t-m \text{ SMER}}$ | $C_{2-2 \text{ SMER}}$ | $C_{t-t \text{ SMER}}$ | $C_{t-m \text{ SMER}}$ |
| Americas-large | 505.5 s          | 509.5 s          | 452.9 s          | 1203.9 s         | 1212.8 s         | 1078.1 s         |
| Americas-small| 30.5 s           | 31.8 s           | 28.3 s           | 89.9 s           | 96.3 s           | 85.6 s           |
| Apj          | 1.5 s            | 1.4 s            | 1.2 s            | 3.6 s            | 3.1 s            | 2.7 s            |
| Customer     | 0.5 s            | 0.3 s            | 0.3 s            | 1.3 s            | 0.7 s            | 0.6 s            |
| Emea         | 0.07 s           | 0.07 s           | 0.06 s           | 0.02 s           | 0.1 s            | 0.1 s            |

Table 13. Execution time with 3–5 RSSOD.

| Dataset      | $|E_{3-5 \text{ RSSOD}}| = 30$ | $|E_{3-5 \text{ RSSOD}}| = 50$ |
|--------------|-------------------------------|-------------------------------|
|              | $C_{2-2 \text{ SMER}}$ | $C_{t-t \text{ SMER}}$ | $C_{t-m \text{ SMER}}$ | $C_{2-2 \text{ SMER}}$ | $C_{t-t \text{ SMER}}$ | $C_{t-m \text{ SMER}}$ |
| Americas-large | 2022.7 s         | 2054.3 s         | 1314.7 s         | 5762.9 s         | 5861.9 s         | 3751.6 s         |
| Americas-small| 117.7 s          | 143.6 s          | 91.9 s           | 280.9 s          | 337.9 s          | 216.2 s          |
| Apj          | 6.7 s            | 7.1 s            | 4.5 s            | 17.9 s           | 19.7 s           | 12.6 s           |
| Customer     | 1.2 s            | 5.0 s            | 3.2 s            | 2.7 s            | 12.1 s           | 7.7 s            |
| Emea         | 0.3 s            | 0.3 s            | 0.2 s            | 0.8 s            | 0.8 s            | 0.5 s            |

Table 14. Execution time with 5–10 RSSOD.

| Dataset      | $|E_{5-10 \text{ RSSOD}}| = 30$ | $|E_{5-10 \text{ RSSOD}}| = 50$ |
|--------------|-------------------------------|-------------------------------|
|              | $C_{2-2 \text{ SMER}}$ | $C_{t-t \text{ SMER}}$ | $C_{t-m \text{ SMER}}$ | $C_{2-2 \text{ SMER}}$ | $C_{t-t \text{ SMER}}$ | $C_{t-m \text{ SMER}}$ |
| Americas-large | 6164.6 s         | 6267.3 s         | 7771.9 s         | 14,667.3 s       | 15,008.8 s       | 18,612.1 s       |
| Americas-small| 260.7 s          | 371.4 s          | 460.5 s          | 5.9 s            | 900.5 s          | 1116.7 s         |
| Apj          | 19.7 s           | 23.3 s           | 28.9 s           | 47.0 s           | 59.4 s           | 73.6 s           |
| Customer     | 0.9 s            | 28.1 s           | 34.8 s           | 2.3 s            | 70.5 s           | 87.4 s           |
| Emea         | 0.7 s            | 0.7 s            | 0.8 s            | 1.7 s            | 0.7 s            | 0.8 s            |

4.2. RMO_SODSDA Efficiency

4.2.1. Datasets and Experiment Setup

Datasets: according to Table 6, the number of the chosen roles in the second group of experiments increased from 40 to 300 with a step of 20. In addition, the QMaxSAT solver was chosen because it performed quite well in the latest MAX-SAT evaluation and because of its efficiency on sets of encoding clauses. Clauses updated based on truth assignments were written in Java.

Experiment setup: for the given UAQ $<s, P_{lb}, P_{ub}, match>$, $s$ is a session in $S$, $P_u$ represents permissions assigned to user $u$ in our constructed RBAC system, and $P_{request} \subseteq P_u$. This can be obtained using Sat4j 2.3.4 (April 2013), an open-source satisfiability library in Java [40]. We considered
three scenarios, as follows: (1) $P_{lb} = P_{request}$, $P_{ub} = Pu$, match = min; (2) $P_{lb} = \Phi$, $P_{ub} = P_{request}$, match = max; and (3) $P_{lb} = P_{ub} = P_{request}$, match = exact.

4.2.2. Evaluation Measures

To evaluate authorization performance of the RMO_SODSDA, we implemented the experiments and compared query time as the number of roles increased according to the three scenarios.

4.2.3. Experiment Results

We randomly selected 10 pairs of values of $P_{lb}$ and $P_{ub}$ for the experiments and took the median value. The results of our experiments are shown in Figure 1 with error bars, where the lateral axis represents varying values of the number of roles, and the vertical axis represents changes of query time including truth assignments for HC, encoding for SC, and execution time by using the QMaxSAT solver.

![Figure 1. Comparison of the different scenarios.](image-url)

Figure 1 demonstrates that query time increased slightly as the number of roles increased in the exact and max cases. Particularly, the time was less than 10 milliseconds with up to 300 roles. In the min case, query time irregularly floated up and down with the increasing number of roles. While time taken for different match requests remained the same in the preprocessing and clause-updating stages, the time taken using the QMaxSAT solver was longer in the min case than in the other two cases. However, total query time was still below one second in any case.

In order to further validate the efficiency of the RMO_SODSDA, we considered a similar set of experiments as the ones performed in the work of [37]. More precisely, our experiments were set as follows: $30 \leq |R| < 300$, $|U| = 20$, $|P| = 70$, $|C| = 5$. Similar to the experiments in the previous UAQ setting section, we took the median value over 10 randomly selected pairs of values of $P_{lb}$ and $P_{ub}$ for the experiments. We compared the performance of our method with the results based on the MAX-SAT solver in Figures 2–4 with error bars, where the lateral axis represents varying values of the number of roles, and the vertical axis represents changes of query time.

It can be observed from Figures 2 and 3 that the query time of our method did not obviously vary. It was less than 20 milliseconds with up to 290 roles. In contrast, the query time in the work of [37] tended to grow linearly up to around 160 milliseconds. The total query time of our method was lower because all hard and soft clauses were completely encoded in the preprocessing stage, while in the work of [37], all clauses were encoded in the querying stage. Therefore, our method outperformed the MAX-SAT solver in the exact and max cases.

It can be observed from Figure 4 that the query time of our method did not obviously vary when $|R|$ changed from 30 to 190. However, the query time irregularly floated up and down when $|R|$ changed from 210 to 290, up to around 240 milliseconds in this paper and around 400 milliseconds in the work of [38]. Therefore, our method did not outperform the MAX-SAT solver in the min case.
lots of different variants of SMER constraints to correctly implement the given RSSOD constraints, although it was time-consuming. We constructed lots of different variants of SMER constraints to correctly implement the given RSSOD constraints, although it was time-consuming. We constructed lots of different variants of SMER constraints to correctly implement the given RSSOD constraints, although it was time-consuming.

(2) We compared the execution time of RMO_SODSDA from Tables 11–14 with that of the unconstrained role mining from Table 6 in Table 15. Obviously, the time of our method was much higher than that of any unconstrained role mining method. We constructed lots of different variants of SMER constraints to correctly implement the given RSSOD constraints, although it was time-consuming.

Figure 2. Query comparison when match = exact. MAX-SAT, maximal satisfiability.

Figure 3. Query comparison when match = max.

Figure 4. Query comparison when match = min.

5. Discussion

From the above analyses of the effectiveness and efficiency of RMO_SODSDA, we found the following:

(1) In the Americas-large, Apj, and Emea datasets, percentage of RSSOD constraints that can be successfully enforced exceeded 70%. However, when the length of RSSOD constraints increased to 3–5 RSSOD and 5–10 RSSOD, the percentage in the Americas-small and Customer datasets was below 50% and 17%, respectively. Thus, the Americas-large, Apj, and Emea datasets were more appropriate for the effectiveness evaluations of RMO_SODSDA.

(2) We compared the execution time of RMO_SODSDA from Tables 11–14 with that of the unconstrained role mining from Table 6 in Table 15. Obviously, the time of our method was much higher than that of any unconstrained role mining method. We constructed lots of different variants of SMER constraints to correctly implement the given RSSOD constraints, although it was time-consuming.
### Table 15. Comparison of execution time.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>RMO_SODSDA(s)</th>
<th>Unconstrained Role Mining(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Americas-large</td>
<td>[139.7, 18612.1]</td>
<td>78.78</td>
</tr>
<tr>
<td>Americas-small</td>
<td>[10.1, 1116.7]</td>
<td>6.31</td>
</tr>
<tr>
<td>Apj</td>
<td>[0.3, 73.6]</td>
<td>5.60</td>
</tr>
<tr>
<td>Customer</td>
<td>[0.01, 87.4]</td>
<td>4.66</td>
</tr>
<tr>
<td>Emea</td>
<td>[0.02, 0.8]</td>
<td>0.02</td>
</tr>
</tbody>
</table>

(3) For the given cases of permissions request in this paper, the query time of RMO_SODSDA was below one second in any case. However, the method did not outperform the MAX-SAT solver in the min case.

(4) We considered a similar set of experiments as the ones from the MAX-SAT solver to evaluate the efficiency of RMO_SODSDA, and we found that the query time does not obviously vary in the exact and max cases when the number of roles exceeds 300. Thus, we did not consider and analyze further the similar results in these cases.

(5) Similar to the number of roles chosen from the MAX-SAT solver, we chose the similar set of constraints as another parameter for our method, and only the RSSOD constraint was taken into consideration. Particularly, it was observed from Table 6 that the number of mining roles from the Americas-small and Customer datasets was close to 230 and 270, respectively. However, according to the first discussion in this section, the percentages of RSSOD constraints that can be successfully enforced were very low. As the results in Figure 4, the query time irregularly floated up and down once the two datasets were chosen for the performance evaluations.

### 6. Conclusions

A novel method, RMO-SODSDA, was proposed in this paper. We first presented constrained role-engineering optimization with separation-of-duty constraints, and constructed several SMER variants to correctly implement the RSSOD, including algorithm design and description, and the computational complexity for constructing the different SMER constraints. Then, in order to evaluate the security of the constructed RBAC system for a given authorization-query problem, an efficient approach for ensuring system security was presented that converts the authorization-query problem into a maximal-satisfiability problem. Lastly, experiment results demonstrated the effectiveness and efficiency of the RMO_SODSDA. There are still a few interesting issues to be handled. One issue is to study the relationships between different cardinality constraints and consider how to implement multiple constraints for role-mining optimization in future work.

**Author Contributions:** Conceptualization, W.S. and S.W.; methodology, W.S. and H.L.; validation, W.S., S.W., and H.L.; formal analysis, W.S.; data curation, S.W.; writing—original draft preparation, W.S.

**Funding:** This work was partially supported by the Natural Science Foundation of China (61501393), the Natural Science Foundation of Henan Province of China (182300410145, 182102210132), and the Key Scientific Research Project of Henan Province University (20B520031).

**Conflicts of Interest:** The authors declare no conflict of interest.

### References


25. Huang, H.; Shang, F.; Liu, J.; Du, H. Handling least privilege problem and role mining in RBAC. *J. Comb. Optim.* 2015, 30, 63–86. [CrossRef]


