# Satisfiability and Resiliency in Workflow Systems

Qihua Wang and Ninghui Li

Center for Education and Research in Information Assurance and Security
and Department of Computer Science
Purdue University

**Abstract.** We propose the role-and-relation-based access control ($\mathsf{R}^2\mathsf{BAC}$) model for workflow systems. In $\mathsf{R}^2\mathsf{BAC}$, in addition to a user's role memberships, the user's relationships with other users help determine whether the user is allowed to perform a certain step in a workflow. For example, a constraint may require that two steps must not be performed by users who have a conflict of interest. We also study the workflow satisfiability problem, which asks whether a set of users can complete a workflow. We show that the problem is $\mathbf{NP}$-complete for $\mathsf{R}^2\mathsf{BAC}$, and is $\mathbf{NP}$-complete for any workflow model that supports certain simple types of constraints (e.g., constraints that state certain two steps must be performed by two different users). After that, we apply tools from parameterized complexity theory to better understand the complexities of this problem. We show that the problem is fixed-parameter tractable when the only relations used are $=$ and $\neq$, and is fixed-parameter intractable when user-defined binary relations can be used. Finally, we study the resiliency problem in workflow systems, which asks whether a workflow can be completed even if a number of users may be absent. We formally define three levels of resiliency in workflow systems, namely, static resiliency, decremental resiliency and dynamic resiliency, and study computational problems related to these notions of resiliency.

## 1 Introduction

Workflow systems are used in numerous domains, including production, purchase order processing, and various management tasks. Workflow authorization systems have gained popularity in the research community [1,3,5,10,12]. A workflow divides a task into a set of well-defined sub-tasks (called *steps* in the paper). Security policies in workflow systems are usually specified using authorization constraints. One may specify, for each step, which users are authorized to perform it. In addition, one may specify the constraints between users who perform different steps in the workflow. For example, one may require that two steps must be performed by different users for the purpose of separation of duty [4]. Oftentimes, constraints in workflow authorization systems need to refer to relationships among users. For example, the rationale under a separation of duty policy that requires 2 users to perform the task is that this deters and controls fraud, as the collusion of 2 users are required for a fraud to occur. However, when two users are close relatives, then collusion is much more likely. To achieve the objective of deterring and controlling fraud, the policy should require that two different steps in a workflow must be performed by users who are not in conflict of interest with each other. In different environments, the conflict-of-interest relation need to be defined differently.

J. Biskup and J. Lopez (Eds.): ESORICS 2007, LNCS 4734, pp. 90–105, 2007.

For instance, inside an organization's system, relationships such as close relatives (e.g., spouses and parent-child) can be maintained and users who are close relatives may be considered to be in conflict of interest. In a peer-review setting, conflict of interest may be based on past collaborations, common institutions, etc. For another example, one university may have a policy that a graduate student's study plan must be first approved by the student's advisor and then by the graduate officer in the student's department. To specify such a constraint, one needs to define and refer to the advisor-student binary relation.

In this paper, we introduce the role-and-relation-based access control ($\mathsf{R^2BAC}$) model for workflow systems. The model is role-based in the sense that individual steps of a workflow are authorized for roles. The model is relation-based in the sense that user-defined binary relations can be used to specify constraints and an authorized user is prevented from performing a step unless the user satisfies these constraints. $\mathsf{R^2BAC}$ is a natural step beyond Role-Based Access Control (RBAC) [9], especially in the setting of workflows. As a role defines a set of users, which can be viewed as a unary relation among the set of all users, a binary relation is the natural next step.

One fundamental problem in any workflow authorization systems is the *workflow satisfiability problem* (WSP), which asks whether a workflow can be completed in a certain system configuration. We show that WSP is $\mathbf{NP}$-complete in $\mathsf{R^2BAC}$. Furthermore, we show that the intractability is inherent in any workflow authorization systems that support some simple kinds of constraints. In particular, we show that WSP is $\mathbf{NP}$-hard in any workflow system that supports *either* constraints that require two steps must be performed by different users *or* constraints that require one step must be performed by a user who also performs at least one of several other steps. Such intractability results are somewhat surprising and discouraging, because the constraints involved are simple and natural. It is also unsatisfying as such results do not shed light on the computation cost one has to pay by introducing additional expressive features such as user-defined binary relations, since the complexity of WSP is $\mathbf{NP}$-complete with or without them. Finally, the practical significance of such intractability results is unclear, as in real-world workflow systems, the number of steps should be small.

To address these issues, we apply tools from *parameterized complexity* [6] to WSP. Parameterized complexity is a measure of computational complexity of problems with multiple input parameters. Parameterized complexity enables us to perform finer-grained study on the computational complexity of WSP. We show that if only equality and inequality relations are used and the number of steps in the workflow is treated as a parameter, WSP is fixed-parameter tractable. More specifically, the problem can be solved in $O(f(k)n)$, where $f$ is a function, $k$ is the number of steps in the workflow, and $n$ is the size of the problem. As the number of steps is relatively small in practice, this result shows that it is possible to solve WSP efficiently, when only equality and inequality relations are used. Also, we show that if user-defined relations are allowed, WSP is fixed-parameter intractable. More specifically, WSP is $W[1]$-hard and is in the complexity class $W[2]$; both of $W[1]$ and $W[2]$ are parameterized complexity classes within $\mathbf{NP}$. This illustrates that while supporting user-defined binary relations increases the expressive power, it also introduces a computational cost. We note that a naive algorithm solving WSP in $\mathsf{R^2BAC}$ takes time $O(kn^{k+1})$, which may be acceptable when $k$ is small. The complexity $O(kn^{k+1})$ is not considered fixed-parameter tractable because

one cannot separate $n$ and $k$ in the complexity to the form of $f(k)n^{\alpha}$, where $f(k)$ is independent of $n$ and $\alpha$ is a constant independent of $k$. We also note that it is also possible to develop algorithms with heuristic optimizations that can solve WSP efficiently for practical instances; the study of such algorithms is beyond the scope of this paper.

In many situations, it is not enough to ensure that a workflow can be completed in the current system configuration. In particular, when the workflow is designed to complete a critical task, it is necessary to make sure that the workflow can be completed even if certain users become absent in emergency situations. In other words, *resiliency* is important in workflow systems. The notion of resiliency policies in access control has been recently introduced [8]. Unlike traditional security policies about access control, which focus on ensuring that access is properly *restricted* so that users who should not have access do not get access, resiliency policies aim at ensuring that access is properly *enabled* so that the system is resilient to the absence of users. The goal of resiliency policies is to guarantee that even if a number of users become absent in certain emergent situation, the remaining users can still finish the crucial tasks. An example resiliency policy is as follows: Upon the absence of up to four users, there must still exist three mutually disjoint sets of users such that the users in each set together have all permissions to carry out a critical task. Such a policy would be needed when one needs to be able to send up to three teams of users to different sites to perform a certain task, perhaps in response to some emergent events.

A challenging problem with both theoretical and practical interest is resiliency in workflow systems. Resiliency in workflow systems differs from the resiliency policies proposed in [8] in two aspects. First, due to the existence of authorization constraints, even if a set of users together are authorized to perform all steps in a workflow, it is still possible that they cannot complete the task. Second, as a workflow consists of a sequence of steps and finishing all these steps may take a relatively long time, it is possible that certain users become absent at some point and come back later. In other words, the set of available users may change during the execution of a workflow. Therefore, more refined notions of resiliency for workflow systems are needed. In this paper, we introduce three levels of resiliency in workflow systems and study the complexity of checking resiliency.

The contributions of this paper are as follows:

- We propose the role-and-relation-based access control ($\mathsf{R}^2\mathsf{BAC}$) model for workflow systems. $\mathsf{R}^2\mathsf{BAC}$ naturally extends RBAC to use binary relations to specify authorization constraints and capture many security requirements commonly encountered in workflows.
- We show that WSP in $\mathsf{R}^2\mathsf{BAC}$ is $\mathbf{NP}$-complete in general. We also show that WSP remains $\mathbf{NP}$-hard for any workflow model that supports one of two simple kinds of constraints. Such results are inherent to features of workflow authorization systems and are independent from specific modeling approaches.
- We apply tools from the parameterized complexity theory to WSP and show that it is fixed-parameter tractable when only equality and inequality relations are allowed. However, when user-defined binary relations can be used, WSP becomes fixed-parameter intractable. This clearly illustrates the computational cost incurred by having user-defined binary relations and gives algorithmic insights and ideas about solving WSP in the fixed-parameter tractable (but $\mathbf{NP}$-complete) case.

To the best of our knowledge, this paper is the first to use parameterized complexity in access control policy analysis. As a number of policy analysis problems in access control have been shown to be **NP**-complete, we believe that parameterized complexity theory can be fruitfully applied to these problems to shed insight on the causes of hardness in these problems as well as to give new algorithmic insights.

– We formally define three levels of resiliency in workflow systems. In *static resiliency*, up to $t$ users are absent before the execution of an instance of a workflow. We show that checking whether a set of users is statically resilient for a workflow is **NP**-hard and is in $\mathbf{coNP^{NP}}$, a complexity class in the Polynomial Hierarchy. In *decremental resiliency*, users may become absent during the execution of an instance of a workflow, absent users will never come back for the same workflow instance, and at most $t$ users may be absent in the end. *Dynamic resiliency* differs from decremental resiliency in that absent users may come back later and work on the same workflow instance, and at most $t$ users may be absent at any given point of time. We show that checking whether a set of users is decremental resilient or dynamic resilient for a workflow is **PSPACE**-complete.

The remainder of the paper is organized as follows. We introduce the R$^2$BAC model in Section 2. After that, we study the workflow satisfiability problem in Section 3 and study parameterized complexity of the problem in Section 4. We then define and study resiliency problems in workflow systems in Section 5. We discuss related work in Section 6 and conclude in Section 7.

## 2 The Role-and-Relation-Based Access Control Model for Workflow Systems

In this section, we introduce the Role-and-Relation-Based Access Control (R$^2$BAC) model for workflow systems. We start with a motivating example.

*Example 1.* In an academic institution, submitting a grant proposal to an outside sponsor via the sponsor program services (SPS) is modeled as a workflow with five steps[1] (see Figure 1).

1. `Preparation`: A faculty member prepares a proposal and sends it to the business office of his or her department.
2. `Budget`: An account clerk prepares the budget, checks the proposal, and submits it to the SPS office.
3. `Expert Review`: A regulation expert in the SPS office reviews the proposal to check whether the proposal satisfies various regulations, e.g., those governing export control and human subject research.
4. `Account Review`: An account manager reviews the proposal and the budget.
5. `Submission`: An account manager submits the proposal to the outside sponsor.

[1] This is a simplified version of the process in the authors' institution, which also requires signatures of the department head and the dean's office.

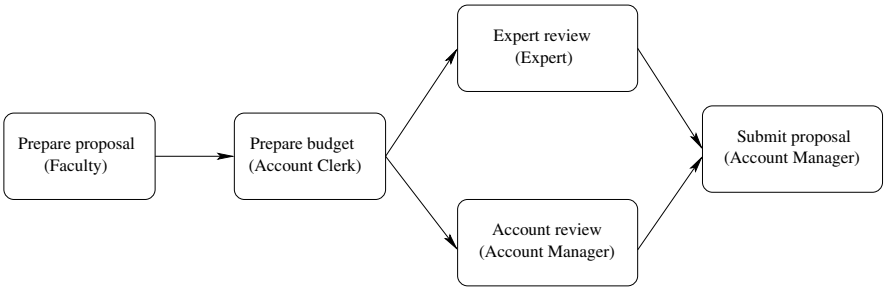


**Fig. 1.** A workflow for grant proposal submission to outside sponsor via the sponsor program services (SPS)

In the workflow, steps `expert review` and `account review` may be performed concurrently while all other steps must be carried out sequentially. The step `preparation` can be performed by any personnel who can serve as a primary investigator, while the step `budget` must be carried out by an account clerk. A regulation expert is authorized to review the proposal in the step `expert review`. The privilege to perform steps `account review` and `submission` is granted to account managers.

The workflow has the following constraints.

1. Steps `preparation`, `budget`, `expert review` and `account review` must be performed by four different users.
2. The account clerk who signs the proposal must be in the same department as the faculty member who prepares the proposal.
3. The persons who review the proposal must not have a conflict of interest with the one submitting the proposal.
4. The account manager who reviews the proposal is responsible to submit it to the outside sponsor.

In the above, Constraint 2 reflects certain procedural and duty requirements, while Constraint 1 enforces the principle of separation of duty. Constraint 3 follows the spirit of separation of duty and goes beyond that. Rather than simply requiring that the two steps must be performed by different people, the constraint requires that the people who perform the two steps must not have a conflict of interest. Constraint 4 enforces a binding-of-duty policy [5] by requiring two tasks be performed by the same user.

As security and practical requirements vary from tasks to tasks, the specification of constraints plays a crucial role in the expression of workflow. As demonstrated in Example 1, binary relations play an important role in expressing authorization constraints. Most existing workflow authorization models support only a few pre-defined binary relations, which limits the expressive power of these models. For example, the model proposed in [10] supports only six pre-defined binary relations $\{=, \neq, <, \leq, >, \geq\}$ between users and roles. Hence, there is no way to express relations like "in the same department" or "is a family member". The model in [5] supports user-defined relations. Our role-and-relation-based access control ($\mathsf{R}^2\mathsf{BAC}$) model for workflow systems

extends the model in [5] by explicitly combining roles and relations and by supporting more sophisticated forms of constraints using these relations.

We now introduce formal definitions for $\mathsf{R}^2\mathsf{BAC}$. Note that $\mathcal{U}$, $\mathcal{R}$ and $\mathcal{B}$ are names of all possible users, roles and binary relations in the system, respectively.

**Definition 1 (Configuration).** A *configuration* is given by a tuple $\langle U, UR, B\rangle$, where $U \subseteq \mathcal{U}$ is a set of users, $UR \subseteq U \times \mathcal{R}$ is the user-role membership relation and $B = \{\rho_1, \cdots, \rho_m\} \subseteq \mathcal{B}$ is a set of binary relations such that $\rho_i \subseteq U \times U$ $(i \in [1, m])$. For convenience, we assume that when $\rho$ is in $B$, $\overline{\rho}$ is also in $B$, where $(u_1, u_2) \in \overline{\rho}$ if and only if $(u_1, u_2) \notin \rho$. Also, $\overline{\overline{\rho}}$ is the same as $\rho$. Furthermore, we assume that $B$ contains two predefined binary relations "$=$" and "$\neq$", which denote equality and inequality, respectively.

A configuration $\langle U, UR, B\rangle$ defines the environment in which a workflow is to be run. In particular, $B$ should define all the binary relations that appear in any constraint in workflows to be run in the environment.

**Definition 2 (Workflow and Constraints).** A *workflow* is represented as a tuple $\langle S, \preceq, SA, C\rangle$, where $S$ is a set of steps, $\preceq \subseteq S \times S$ defines a partial order among steps in $S$, $SA \subseteq \mathcal{R} \times S$, and $C$ is a set of constraints, each of which takes one of the following forms:

1. $\langle \rho(s_1, s_2)\rangle$: the user who performs $s_1$ and the user who perform $s_2$ must satisfy the binary relation $\rho$.
2. $\langle \rho(\exists X, s)\rangle$: there exists a step $s' \in X$ such that $\langle \rho(s', s)\rangle$ holds, i.e., the user who performs $s'$ and the user who performs $s$ satisfy $\rho$.
3. $\langle \rho(s, \exists X)\rangle$: there exists a step $s' \in X$ such that $\langle \rho(s, s')\rangle$ holds.
4. $\langle \rho(\forall X, s)\rangle$: for each step $s' \in X$, $\langle \rho(s', s)\rangle$ must hold.
5. $\langle \rho(s, \forall X)\rangle$: for each step $s' \in X$, $\langle \rho(s, s')\rangle$ must hold.

Intuitively, in a workflow $\langle S, \preceq, SA, C\rangle$, that $s_i \preceq s_j$ $(i \neq j)$ indicates that step $s_i$ must be performed before step $s_j$. Steps $s_i$ and $s_j$ may be performed *concurrently*, if neither $s_i \preceq s_j$ nor $s_j \preceq s_i$. $SA$ is called *role-step authorization* and $(r, s) \in SA$ indicates that members of role $r$ is authorized to perform step $s$.

*Example 2.* Consider the workflow for submitting a grant proposal in Example 1. Let $s_{prepare}$, $s_{budget}$, $s_{xp\_review}$, $s_{ac\_review}$ and $s_{submit}$ denote the five steps in the workflow. The constraints of the workflow can be represented in tuple-based specification as follows.

1. $\langle \neq (s_{budget}, s_{prepare})\rangle$, $\langle \neq (s_{xp\_review}, \forall\{s_{prepare}, s_{budget}\})\rangle$, $\langle \neq (s_{ac\_review}, \forall\{s_{prepare}, s_{budget}, s_{xp\_review}\})\rangle$
   These require that the first four steps in the workflow must be performed by four different users.
2. $\langle \rho_{same\_dept}(s_{budget}, s_{prepare})\rangle$
   $(u_x, u_y) \in \rho_{same\_dept}$ when $u_x$ and $u_y$ are in the same department. The constraint requires that the person who signs the proposal must be in the same department as the person who prepares it.

3. $\langle \overline{\rho}_{conflict\_interest}(\forall\{s_{xp\_review}, s_{ac\_review}\}, s_{prepare})\rangle$
   $(u_x, u_y) \in \rho_{conflict\_interest}$ when $u_x$ and $u_y$ have a conflict of interest. The constraint requires that the person who reviews the proposal must not have a conflict of interest with the person who prepares it.
4. $\langle =(s_{submit}, s_{ac\_review})\rangle$
   The constraint requires that `account review` and `submission` must be performed by the same person.

**Definition 3 (Plans and Partial Plans).** A *plan* $P$ for workflow $W = \langle S, \preceq, SA, C\rangle$ is a subset of $\mathcal{U} \times S$ such that, for every step $s_i \in S$, there is exactly one tuple $(u_a, s_i)$ in $P$, where $u_a \in \mathcal{U}$. A *partial plan* $PP$ for $W$ is a subset of $\mathcal{U} \times S$ such that, for every step $s_i \in S$, there is at most one tuple $(u_a, s_i)$ in $PP$, where $u_a \in \mathcal{U}$. And $(u_a, s_i) \in PP$ implies that, for every $s_j \preceq s_i$, there exists $u_b \in \mathcal{U}$ such that $(u_b, s_j) \in PP$.

Intuitively, a plan assigns exactly one user to every step in a workflow, while a partial plan does this for only a portion of the steps in the workflow. Furthermore, if a step is in a partial plan, then its prerequisite steps must also be in the partial plan.

**Definition 4 (Valid Plan).** Given a workflow $W = \langle S, \preceq, SA, C\rangle$, and a configuration $\Gamma = \langle U, UR, B\rangle$, we say that a user $u$ is an *authorized user* of a step $s \in S$ under $\Gamma$ if and only if there exists a role $r$ such that $(u, r) \in UR$ and $(r, s) \in SA$. We say that a plan $P$ *is valid for* $W$ under $\Gamma$ if and only if for every $(u, s) \in P$, $u$ is an authorized user of $s$, and no constraint in $C$ is violated. We say that $W$ is *satisfiable* under $\Gamma$ if and only if there exists a plan $P$ that is valid for $W$ under $\Gamma$.

Note that there can be multiple valid plans for a workflow $W$ under a configuration. In fact, it is the existence of multiple valid plans that makes it possible for $W$ to be completed even if a number of users are absent. In situations where the configuration changes during the execution of a workflow instance (e.g. users become absent), we will have to change our plan at runtime and thus constraints need to be checked at runtime as well. If a constraint $c$ contains $\forall$, then it is checked whenever a step restricted by $c$ is to be executed. Other kinds of constraints are checked before the last step restricted by the constraint is to be executed.

**Definition 5 (Valid Partial Plan).** Given a workflow $\langle S, \preceq, SA, C\rangle$ and a configuration $\langle U, UR, B\rangle$, let $s_1, \cdots, s_m$ be a sequence of steps such that $s_i \not\preceq s_j$ when $i > j$. A partial plan $PP$ is *valid* with respect to the sequence $s_1, \cdots, s_i$ if it assigns one user to each step in $s_1, \cdots, s_i$ and no constraint that is checked before the execution of $s_i$ is violated by $PP$.

## 3 The Workflow Satisfiability Problem

One fundamental problem in workflow authorization systems is the Workflow Satisfiability Problem (WSP), which checks whether a workflow $W$ is satisfiable under a configuration $\Gamma$. Note that, given configuration $\langle U, UR, B\rangle$, checking whether $W$ is satisfiable under $\Gamma$ is equivalent to checking whether there is a valid plan for $W$ under $\Gamma$. In this section, we study the computational complexity of WSP.

### 3.1 Computational Complexity of WSP for R$^2$BAC

**Theorem 1.** WSP is $\mathbf{NP}$-complete in R$^2$BAC.

The proof of Theorem 1 consists of two parts. The first part is Lemma 1, which shows that WSP is in $\mathbf{NP}$ in R$^2$BAC. In the second part, Lemma 2 and Lemma 3 show that WSP is $\mathbf{NP}$-hard in two restricted cases. Due to page limit, proofs to the following lemmas are given in our technical report [11].

**Lemma 1.** WSP is in $\mathbf{NP}$ in R$^2$BAC.

Intuitively, a nondeterministic Turing can guess a plan and check whether the plan is valid in polynomial time.

**Lemma 2.** WSP is $\mathbf{NP}$-hard in R$^2$BAC, if the workflow uses constraints of the form $\langle \neq (s_1, s_2) \rangle$.

To prove the above lemma, we use a reduction from the $\mathbf{NP}$-complete GRAPH K-COLORABILITY problem. In the reduction, vertices in a graph are mapped to steps in the workflow, while colors are mapped to users. In the GRAPH K-COLORABILITY problem, the number of vertices is normally much larger than the number of colors. Hence, the number of steps in the constructed workflow is much larger than the number of users, which is rarely the case in practice. Such a phenomenon indicates that classical complexity framework is inadequate to study the complexity of WSP in a real-word setting. This motivates us to apply the tool of parameterized complexity to perform finer-grained study of the complexity of WSP, which will be discussed in Section 4.

**Lemma 3.** WSP is $\mathbf{NP}$-hard in R$^2$BAC, if the workflow uses constraints of the form $\langle = (s, \exists X) \rangle$.

The proof of this lemma uses a reduction from the $\mathbf{NP}$-complete HITTING SET problem to WSP.

Although WSP is intractable in general in R$^2$BAC, the problem is in $\mathbf{P}$ for certain special cases. Lemma 4 states a tractable case for WSP.

**Lemma 4.** WSP is in $\mathbf{P}$ in R$^2$BAC, if the workflow only has constraints in the forms of $\langle = (s_1, s_2) \rangle$, $\langle = (s, \forall X) \rangle$ or $\langle = (\forall X, s) \rangle$.

### 3.2 The Inherent Complexity of Workflow Systems

In Section 3.1, we show that WSP is $\mathbf{NP}$-hard in R$^2$BAC in general. In this section, we stress that the intractability of WSP is inherent to certain fundamental features of workflow authorization systems and independent from modeling approaches. We say that a workflow system supports the feature of *user-step authorization* if it allows one to specify (either directly or indirectly) which users are allowed to perform which steps in the workflow. User-step authorization is probably the most fundamental feature and almost all workflow systems found in existing literatures support such feature. A *user-inequality constraint* states that certain two steps cannot be performed by the same user, i.e., $\langle \neq (s_1, s_2) \rangle$ in R$^2$BAC. An *existence-equality constraint* states that a certain step must be performed by a user who performs at least one step in a given set of steps, i.e., $\langle = (s, \exists X) \rangle$ in R$^2$BAC.

**Theorem 2.** Checking whether a set of users can complete a workflow is **NP**-hard for any workflow system that satisfies either (or both) of the followings:

- The system supports user-step authorization and user-inequality constraints.
- The system supports user-step authorization and existence-equality constraints.

The proof to Theorem 2 follows from the proofs of Lemmas 2 and 3. Please refer to [11] for details.

Note that user-inequality constraints are widely used in existing literatures to enforce separation of duty in workflow systems. Many workflow models [3,10,5] support such type of constraints. Existence-equality constraints are a natural way to enforce the general form of binding of duty policies, which require a step be performed by one of those users who have performed some prerequisite steps.

## 4 Beyond Intractability of WSP

In Section 3, we have shown that WSP is **NP**-complete in $\mathsf{R}^2\mathsf{BAC}$ for the general case as well as the two special cases where only a simple form of constraints are used. Such results are, however, unsatisfying, as they do not shed light on the computation cost associated with introducing additional expressive features such as user-defined binary relations, since the complexity of WSP is **NP**-complete in all the three cases. Such a phenomenon indicates that classical computational complexity does not precisely capture the computational difficulty of different cases of WSP. Furthermore, the practical significance of such intractability results is unclear. The input to WSP consists of many aspects, such as the number of steps in the workflow, the number of constraints and the number of users in the configuration etc. In practice, some aspects of the input will not take a large value. For instance, even though the number of users may be large, the number of steps in the workflow is expected to be small. An interesting question arises is whether WSP can be solved efficiently given the restriction that the number of steps is small.

To address these issues, we apply tools from the theory of parameterized complexity [6] to WSP.

### 4.1 Why Parameterized Complexity?

Parameterized complexity is a measure of complexity of problems with multiple input parameters. The theory of parameterized complexity was developed in the 1990s by Rod Downey and Michael Fellows. It is motivated, among other things, by the observation that there exist hard problems that (most likely) require exponential runtime when complexity is measured in terms of the input size only, but that are computable in a time that is polynomial in the input size and exponential in a (small) parameter $k$. Hence, if $k$ is fixed at a small value, such problems can still be considered 'tractable' despite their traditional classification as 'intractable'.

In classical complexity, a decision problem is specified by two items of information: (1) the input to the problem, and (2) the question to be answered. In parameterized complexity, there are three parts of a problem specification: (1) the input to the problem,

(2) the aspects of the input that constitute the parameter, and (3) the question to be answered. Normally, the parameter is selected because it is likely to be confined to a small range in practice. The parameter provides a systematic way of specifying restrictions of the input instances. Some **NP**-hard problems can be solved by algorithms that are exponential only in a fixed parameter while polynomial in the size of the input. Such an algorithm is called a *fixed-parameter tractable* algorithm. More specifically, an algorithm for solving a problem is a fixed-parameter tractable algorithm, if when given any input instance of the problem with parameter $k$, the algorithm takes time $O(f(k)n^{\alpha})$, where $n$ is the size of the input, $k$ is the parameter, $\alpha$ is a constant (independent of $k$), and $f$ is an arbitrary function.

If a problem has a fixed-parameter tractable algorithm, then we say that it is a fixed-parameter tractable problem and belongs to the class **FPT**. For example, the **NP**-complete VERTEX COVER asks, given a graph $G$ and an integer $k$, whether there is a size-$k$ set $V'$ of vertices, such that every edge in $G$ is adjacent to at least one vertex in $V'$. This problem is in **FPT** when taking $k$ as the parameter, as there exists a simple algorithm with running time of $O(2^k n)$, where $n$ is the size of $G$. Note that not all intractable problems are in **FPT**. For instance, the **NP**-complete DOMINATING SET problem is fixed-parameter intractable. Given a graph $G$ and an integer $k$, DOMINATING SET asks whether there is a size-$k$ set $V'$ of vertices such that every vertex in $G$ is either in $V'$ or is connected to a vertex in $V'$ by an edge. For DOMINATING SET, there is no significant alternative to trying all size-$k$ subsets of vertices in $G$ and there are $O(n^k)$ such subsets, where $n$ is the number of vertices.

Finally, we would like to point out that a problem in **FPT** does not necessarily mean that it can be efficiently solved as long as the parameter is small. Note that $f(k)$ may be a function that grows very fast over $k$. For instance, an $O(k^{k^k} n)$ algorithm is not practical even if $k$ is as small as 5, just as we cannot claim that a problem in **P** can be solved efficiently when the best algorithm takes time $O(n^{100})$. However, showing that a problem is in **FPT** has significant impact as experiences have shown that improvement on fixed-parameter tractable algorithms are oftentimes possible. For instance, when VERTEX COVER was first observed to be solvable in $O(f(k)n^3)$, $f(k)$ was such a function that the algorithm is utterly impractical even for $k = 1$. An $O(2^k n)$ algorithm was proposed later, and then an algorithm with running time $O(kn + (4/3)^k k^2)$ was revealed. Right now, VERTEX COVER is well-solved for input of any size, as long as the parameter value is $k \leq 60$. Parameterized complexity offers a fresh angle into designing algorithms for such problems.

In this paper, we only study which subcases of WSP are in **FPT** and which are not. Improvement on the fixed-parameter tractable algorithms for the **FPT** cases is beyond the scope of this paper.

### 4.2 Fixed Parameter Tractable Cases of WSP

As the number of steps in a workflow is likely to be small in practice, we select the number of steps as the parameter for WSP. We first show that a special case of WSP in which only the $\neq$ relation is allowed is in **FPT**. The proof gives a fixed-parameter tractable algorithm and illustrates the intuition why this problem is in **FPT**.

**Lemma 5.** WSP in $\mathsf{R}^2\mathsf{BAC}$ is in $\mathbf{FPT}$, if $\neq$ is the only binary relation used by constraints in the workflow. In particular, given a workflow $W$ and a configuration $\Gamma$, WSP can be solved in time $O(k^{k+1}n)$, where $k$ is the number of steps in $W$ and $n$ is the size of the entire input to the problem.

*Proof.* A constraint using binary relation $\neq$ requires a certain step to be performed by a user who does not perform certain other step(s). Since there are $k$ steps in $W$, if step $s$ is authorized to no less than $k$ users in $U$, then we can always find an authorized user of $s$, who is not assigned to any other steps in $W$. In other words, we only need to consider those steps that are authorized to less than $k$ users in $U$, and there are at most $k$ such steps. We construct partial plans for these steps by trying all combinations of authorized users and there are no more than $k^k$ such combinations. Verifying whether a plan is valid can be done in $O(kn)$, as there are $O(n)$ constraints and each constraints restricts at most $k$ steps. Therefore, checking whether $U$ can complete $W$ can be done in time $O(k^{k+1}n)$.

**Theorem 3.** WSP is in $\mathbf{FPT}$ in $\mathsf{R}^2\mathsf{BAC}$, if = and $\neq$ are the only binary relations used by constraints in the workflow.

This Theorem subsumes Lemma 5. Please refer to [11] for its proof.

### 4.3 WSP Is Fixed Parameterized Intractable in General

A natural question to ask is whether WSP is still in $\mathbf{FPT}$ when user-defined binary relations are allowed in the workflow. We show that the answer is "no". Similar to proving a problem is intractable in classical complexity framework, we prove that a problem is fixed-parameter intractable by reducing another fixed-parameter intractable problem to the target problem. To preserve fixed-parameter tractability, we need to use a kind of reduction different from the classical ones used in $\mathbf{NP}$-completeness proofs. We say that $L$ reduces to $L'$ by a *fixed-parameter reduction* if given an instance $\langle x, k\rangle$ for $L$, one can compute an instance $\langle x' = g_1(\langle x, k\rangle), k' = g_2(k)\rangle$ in time $O(f(k)|x|^{\alpha})$ such that $\langle x, k\rangle \in L$ if and only if $\langle x', k'\rangle \in L'$, where $g_1$ and $g_2$ are two functions and $\alpha$ is a constant. Note that many classical reductions are not fixed-parameter reduction as they do not carry enough structure, and lead to lose of control for the parameter.

Under parameterized complexity, each problem falls somewhere in the hierarchy: $\mathbf{P} \subseteq \mathbf{FPT} \subseteq W[1] \subseteq W[2] \subseteq \cdots \subseteq W[P] \subseteq \mathbf{NP}$. If a problem is $W[1]$-hard, then it is believed to be fixed-parameter intractable. To understand the classes $W[t]$, we can start by viewing a 3CNF formula as a (boolean) decision circuit, consisting of one input for each variable and structurally a large *and* gate taking inputs from a number of small *or* gates. (Some wires in the circuit may include a negation.) The *or* gates are small in that each of them takes 3 inputs, and the *and* gate is large in that it takes an unbounded number of inputs. The *weft* of a decision circuit is the maximum number of large gates on any path from the input variable to the output line. The *weighted satisfiability* problem for decision circuits asks whether a decision circuit has a weight $k$ satisfying assignment (i.e., a satisfying assignment in which at most $k$ variables are assigned true). The class $W[t]$ includes all problems that are fixed parameter reducible to the weighted satisfiability problem for decision circuits of weft $t$.

The following theorem states that WSP is fixed-parameter intractable in $\mathsf{R^2BAC}$ when user-defined binary relations are allowed in the workflow.

**Theorem 4.** WSP is $W[1]$-hard in $\mathsf{R^2BAC}$ if user-defined binary relations are used in constraints.

The proof to the above theorem is given in [11]. In the proof, we reduce the $W[1]$-complete INDEPENDENT SET problem to WSP.

We conclude from Theorem 3 and Theorem 4 that supporting user-defined binary relations introduces additional complexity to WSP in $\mathsf{R^2BAC}$. Parameterized complexity reveals such a fact that is hidden by classical complexity framework and allows us to better understand the source of complexity of WSP in $\mathsf{R^2BAC}$. We point out that a naive algorithm solving WSP for $\mathsf{R^2BAC}$, which enumerates all possible plans and verifies each of them, takes time $O(kn^{k+1})$, which may be acceptable when $k$ is small. We also note that it is possible to develop algorithms with heuristic optimizations that can solve WSP efficiently for practical instances; the study of such algorithms is beyond the scope of this paper.

Finally, we provide an upperbound for WSP in $\mathsf{R^2BAC}$ in the parameterized complexity framework. Please refer to [11] for the proof to Theorem 5.

**Theorem 5.** WSP in $\mathsf{R^2BAC}$ is in $W[2]$.

It remains open whether WSP is $W[1]$-complete or $W[2]$-complete.

## 5 Resiliency in Workflow Systems

We have studied the workflow satisfiability problem (WSP) in previous sections. In many situations, it is not enough to ensure that a workflow is satisfiable in the current system configuration. In particular, when the workflow is designed to complete a critical task, it is necessary to guarantee that even if certain users are absent unexpectedly, the workflow can still be completed. Resiliency is a property of those system configurations that can satisfy the workflow even with absence of some users.

In this section, we define and study resiliency in workflow systems. The workflow model we use is $\mathsf{R^2BAC}$. Before giving formal definitions of resiliency in workflow systems, let us consider several possible scenarios.

1. The execution of instances of a workflow is done in a relatively short period of time, say within fifteen minutes. Although it is possible that certain users are absent before the execution of a workflow instance, it is unlikely that available users become absent during the execution of the workflow instance. In other words, the set of users who are available for a workflow instance is stable.
2. The execution of instances of a workflow takes a relatively long period of time, say within one day. Some users may not come to work on the day when a workflow instance is executed. Furthermore, some users may have to leave at some point (e.g. between the execution of two steps) before the workflow instance is completed and will not come back to work until the next day. In such a situation, the set of users available to the workflow instance becomes smaller and smaller over time. Such a scenario would also be possible in potentially hazardous situations such as battlefield and fire-fighting.

3. The execution of instances of a workflow takes a long period of time. For example, only a single step of the workflow is performed each day. Since the set of users who come to work may differ from day to day, the set of available users may differ from step to step.

We capture the above three scenarios by proposing three levels of resiliency in workflow systems. They are *static (level-1) resiliency*, *decremental (level-2) resiliency* and *dynamic (level-3) resiliency*. In static resiliency, a number of users are absent before the execution of a workflow instance, while remaining users will not be absent during the execution; in decremental resiliency, users may be absent before or during the execution of a workflow instance, and absent users will not become available again; in dynamic resiliency, users may be absent before or during the execution of a workflow instance and absent users may become available again. In all cases, we assume that the number of absent users at any point is bounded by a parameter $t$. We now give formal definitions of the three levels of resiliency.

**Definition 6 (Static Resiliency).** Given a workflow $W$ and an integer $t \geq 0$, a configuration $\langle U, UR, B\rangle$ is *statically resilient* for $W$ up to $t$ absent users if and only if for every size-$t$ subset $U'$ of $U$, $W$ is satisfiable under $\langle (U - U'), UR, B\rangle$.

Intuitively, a configuration is statically resilient for a workflow if the workflow is still satisfiable after removing $t$ users from the configuration.

**Definition 7 (Decremental Resiliency).** Given a workflow $W = \langle S, \preceq, SA, C\rangle$ and an integer $t$, a configuration $\langle U, UR, B\rangle$ is *decrementally resilient* for $W$ up to $t$ absent users, if and only if Player 1 can always win the following two-person game when playing optimally.

Initialization: $PP \leftarrow \emptyset$, $U_0 \leftarrow U$, $S_0 \leftarrow S$, $t_0 \leftarrow t$ and $i \leftarrow 1$.

Round $i$ of the Game:

1. Player 2 selects a set $U'_{i-1}$ such that $|U'_{i-1}| \leq t_{i-1}$.
   $U_i \leftarrow (U_{i-1} - U'_{i-1})$ and $t_i \leftarrow (t_{i-1} - |U'_{i-1}|)$.
2. Player 1 selects a step $s_{a_i} \in S_{i-1}$ such that $\forall s_b (s_b \prec s_{a_i} \Rightarrow s_b \notin S_{i-1})$.
   Player 1 selects a user $u \in U_i$.
   $PP \leftarrow PP \cup \{(u, s_{a_i})\}$ and $S_i \leftarrow (S_{i-1} - \{s_{a_i}\})$.
   If $PP$ is not a valid partial plan with respect to the sequence $s_{a_1}, \cdots, s_{a_i}$, then Player 1 loses.
3. If $S_i = \emptyset$, then Player 1 wins; otherwise, let $i \leftarrow (i + 1)$ and the game goes on to the next round.

In each round, Player 2 may remove a certain number of users and then Player 1 has to pick a remaining step that is ready to be performed and assign an available user to it. The total number of users Player 2 may remove throughout the game is bounded by $t$. A configuration is decrementally resilient for a workflow if there is always a way to complete the workflow no matter when and which users are removed, as long as the total number of absent users is bounded by $t$.

Also, in Definition 7, we assume that Player 1 plays optimally, which implies that in each round, Player 1 has to consider not only the next step but also all future steps.

**Definition 8 (Dynamic Resiliency).** Given a workflow $W = \langle S, \preceq, SA, C\rangle$ and an integer $t$, a configuration $\langle U, UR, B\rangle$ is *dynamically resilient* for $W$ up to $t$ absent users, if and only if Player 1 can always win the following two-person game when playing optimally.

Initialization: $PP \leftarrow \emptyset$, $S_0 \leftarrow S$ and $i \leftarrow 1$.
Round $i$ of the Game:

1. Player 2 selects a set $U'_{i-1}$ of up to $t$ users.
   $U_i \leftarrow (U - U'_{i-1})$.
2. Player 1 selects a step $s_{a_i} \in S_{i-1}$ such that $\forall s_b (s_b \prec s_{a_i} \Rightarrow s_b \notin S_{i-1})$.
   Player 1 selects a user $u \in U_i$.
   $PP \leftarrow PP \cup \{(u, s_{a_i})\}$ and $S_i \leftarrow (S_{i-1} - \{s_{a_i}\})$.
   If $PP$ is not a valid partial plan with respect to the sequence $s_{a_1}, \cdots, s_{a_i}$, then Player 1 loses.
3. If $S_i = \emptyset$, then Player 1 wins; otherwise, let $i \leftarrow (i+1)$ and the game goes on to the next round.

Intuitively, Player 2 may temporarily remove up to $t$ users from the configuration at the beginning of each round. Then, Player 1 has to select a remaining step that is ready to be performed and assign an available user to it. After that, the configuration is restored and the next round of the game starts.

By definition, dynamic (level-3) resiliency is stronger than decremental (level-2) resiliency, which is in turn stronger than static (level-1) resiliency.

### 5.1 Computational Complexities of Checking Resiliency

In this section, we study computational problems related to resiliency in workflow systems. Due to page limit, proofs are given in [11].

**Theorem 6.** Checking whether a configuration $\Gamma$ is statically resilient for a workflow $W$ up to $t$ users, which is called the *Static Resiliency Checking Problem* (SRCP), is $\mathbf{NP}$-hard and is in $\mathbf{coNP^{NP}}$.

It remains open whether SRCP is $\mathbf{coNP^{NP}}$-complete or not. Readers who are familiar with computational complexity theory will recognize that $\mathbf{coNP^{NP}}$ is a complexity class in the Polynomial Hierarchy. Because the Polynomial Hierarchy collapses when $\mathbf{P} = \mathbf{NP}$, showing that an $\mathbf{NP}$-hard decision problem is in the Polynomial Hierarchy, although is not equivalent to showing that the problem is $\mathbf{NP}$-complete, has the same consequence: the problem can be solved in polynomial time if and only if $\mathbf{P}$ =$\mathbf{NP}$.

**Theorem 7.** Checking whether a configuration $\Gamma$ is decremental resilient for a workflow $W$ up to $t$ users, which is called the *Decremental Resiliency Checking Problem* (CRCP), is $\mathbf{PSPACE}$-complete.

Checking whether $\Gamma$ is dynamically resilient for a $W$ up to $t$ users, which is called the *Dynamic Resiliency Checking Problem* (DRCP), is $\mathbf{PSPACE}$-complete.

To prove the above theorem, we reduce the $\mathbf{PSPACE}$-complete QUANTIFIED SATISFIABILITY problem to CRCP and DRCP. Intuitively, we use user-step assignments in workflow to simulate truth assignments for boolean variables.

## 6 Related Work

Bertino et al. [3] introduced a language to express workflow authorization constraints as clauses in a logic programming language. The language supports a number of predefined relations for constraint specification. Bertino et al. [3] also proposed searching algorithms to assign users to complete a workflow. This work does not support user-defined binary relations, nor does it formally study computational complexity of the workflow satisfiability problem. Tan et al. [10] studied the consistency of authorization constraints in workflow systems. The model in [10] supports six predefined binary relations: $\{=, \neq, <, \leq, >, \geq\}$, but not user-defined relations. Atluri and Huang [1] proposed a workflow authorization model that focuses on temporal authorization. This model does not support constraints about users performing different steps in a task. Atluri and Warner [2] proposed a model that supports conditional delegation in workflow systems. Delegation is a potential mechanism to achieve resiliency. In this paper, we consider resiliency without using delegation. We plan to extend our definitions on resiliency to take delegation into account and study how to use delegation to achieve resiliency in workflow systems. Furthermore, in [12], Warner and Atluri considered authorization constraints that span multiple instances of a workflow. Their model supports predefined relations with emphasis on inter-instance constraints. Inter-instance problems in workflow systems is an interesting research area. The models in [2,12] do not support user-defined relations. Finally, Kang et al. [7] investigated access control mechanisms for inter-organizational workflow. Their workflow model authorizes steps to roles and supports dynamic constraints. However, they do not explicitly point out how constraints are specified and what kinds of constraints are supported besides separation of duty. Their paper mainly focuses on infrastructure design and implementation.

The workflow authorization model proposed by Crampton [5] is probably the one that is most closely related to $\mathsf{R}^2\mathsf{BAC}$. The model in [5] supports user-defined binary relations; however, it does not support quantifiers in constraints, so that constraints of the form $\langle \rho(\exists X, s) \rangle$ cannot be expressed in that model. Crampton [5] also studied the workflow satisfiability problem and presented a polynomial time algorithm for their model. However, the algorithm is incorrect.[2] As we have pointed out in Theorem 2, the workflow satisfiability problem is $\mathbf{NP}$-hard in general for any workflow model that supports user-inequality constraints. Since the model in [5] supports such type of constraints, a polynomial time algorithm for the satisfiability problem in their model could not exist.

None of the work mentioned above have given the computational complexity results of the Workflow Satisfiability Problem, whereas we give a clear characterization using parameterized complexity. Also, the resiliency problem in workflow has not been studied before in the literature.

The concept of resiliency policies in access control is first formally proposed by Li et al. [8]. To our knowledge, this paper is the first to define and study resiliency problems in workflow systems. There are major difference between resiliency in workflow systems and the resiliency policies proposed in [8], and we have discussed the differences in Section 1.

[2] We have verified the bug with the author of [5]. Please refer to [11] for more details.

## 7 Conclusion

We have proposed a role-and-relation-based model ($R^2BAC$) for workflow systems, and have shown that the workflow satisfiability problem in $R^2BAC$ is **NP**-complete. We have also shown that the problem remains intractable for any workflow model that supports certain simple types of constraints such as user-inequality constraints and existence-equality constraints. We then apply tools from parameterized complexity to better understand the complexities of the problem. Furthermore, we have formally defined three levels of resiliency in workflow systems, namely, static resiliency, decremental resiliency and dynamic resiliency. We have also shown that checking whether a system configuration is statically resilient for a workflow is **NP**-hard and is in the Polynomial Hierarchy, and the same problems for decremental resiliency and dynamic resiliency are **PSPACE**-complete.

## References


1. Atluri, V., Huang, W.: An authorization model for workflows. In: Martella, G., Kurth, H., Montolivo, E., Bertino, E. (eds.) ESORICS 96. LNCS, vol. 1146, pp. 44–64. Springer, Heidelberg (1996)
2. Atluri, V., Warner, J.: Supporting conditional delegation in secure workflow management systems. In: SACMAT '05: Proceedings of the tenth ACM symposium on Access control models and technologies, pp. 49–58. ACM Press, New York (2005)
3. Bertino, E., Ferrari, E., Atluri, V.: The specification and enforcement of authorization constraints in workflow management systems. ACM Transactions on Information and System Security 2(1), 65–104 (1999)
4. Clark, D.D., Wilson, D.R.: A comparision of commercial and military computer security policies. In: Proceedings of the 1987 IEEE Symposium on Security and Privacy, pp. 184–194. IEEE Computer Society Press, Los Alamitos (May 1987)
5. Crampton, J.: A reference monitor for workflow systems with constrained task execution. In: SACMAT 2005. Proceedings of the Tenth ACM Symposium on Access Control Models and Technologies, Stockholm, Sweden, pp. 38–47. ACM Press, New York (June 2005)
6. Downey, R., Fellows, M.: Parameterized Complexity. Springer, Heidelberg (1999)
7. Kang, M.H., Park, J.S., Froscher, J.N.: Access control mechanisms for inter-organizational workflow, pp. 66–74 (2001)
8. Li, N., Tripunitara, M.V., Wang, Q.: Resiliency policies in access control. In: CCS. Proc. ACM Conference on Computer and Communications Security, ACM Press, New York (November 2006)
9. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. IEEE Computer 29(2), 38–47 (1996)
10. Tan, K., Crampton, J., Gunter, C.: The consistency of task-based authorization constraints in workflow systems. In: CSFW. Proceedings of the 17th IEEE Computer Security Foundations Workshop, pp. 155–169. IEEE Computer Society Press, Los Alamitos (2004)
11. Wang, Q., Li, N.: Satisfiability and resiliency in workflow systems. Technical Report CERIAS-TR-2007-28, Center for Education and Research in Information Assurance and Security, Purdue University (2007)
12. Warner, J., Atluri, V.: Inter-instance authorization constraints for secure workflow management. In: SACMAT. Proc. ACM Symposium on Access Control Models and Technologies, pp. 190–199. ACM Press, New York (2006)