The Authorization Policy Existence Problem

Pierre Bergé, Jason Crampton, Gregory Gutin, Rémi Watrigant

Abstract—Constraints such as separation-of-duty are widely used to specify requirements that supplement basic authorization policies. However, the existence of constraints (and authorization policies) may mean that a user is unable to fulfill her/his organizational duties because access to resources has been denied. In short, there is a tension between the need to protect resources (using policies and constraints) and the availability of resources. Recent work on workflow satisfiability and resiliency in access control asks whether this tension compromises the ability of an organization to achieve its objectives. In this paper, we develop a new method of specifying constraints which subsumes much related work and allows a wider range of constraints to be specified. The use of such constraints leads naturally to a range of questions related to "policy existence", where a positive answer means that an organization's objectives can be realized. We analyze the complexity of these policy existence questions and, for particular sub-classes of constraints defined by our language, develop fixed-parameter tractable algorithms to solve them.[1]

Index Terms—access control; resiliency; satisfiability; computational complexity; fixed-parameter tractability

1 Introduction

Access control is a fundamental aspect of the security of any multi-user computing system, and is typically based on the specification and enforcement of an authorization policy. Such a policy identifies which interactions between users and resources are to be allowed by the system.

Over the last twenty years, authorization policies have become more complex, not least because of the introduction of constraints, which further refine an authorization policy. A separation-of-duty constraint (also known as the "two-man rule" or "four-eyes policy") may, for example, require that no single user is authorized for some particularly sensitive group of resources. Such a constraint is typically used to prevent misuse of the system by a single user.

The use of authorization policies and constraints, by design, limits which users may access resources. Nevertheless, the ability to perform one's duties requires access to particular resources, and overly prescriptive policies and constraints may mean that some resources are inaccessible. In short, "tension" may exist between authorization policies and operational demands: too lax a policy may suit organizational demands but lead to security violations; whereas too restrictive a policy may compromise an organization's ability to meet its business objectives.

Recent work on workflow satisfiability and access control resiliency has recognized the importance of being able to identify whether or not security policies prevent an organization from achieving its objectives [12], [13], [19], [20], [26]. In this paper, we seek to extend existing work in this area. Specifically, we introduce the Authorization Policy Existence Problem (APEP), which may be treated as a decision or optimization problem. Informally, APEP seeks to find an authorization policy, subject to restrictions on individual authorizations (defined by a "base" authorization relation) and restrictions on collective authorizations (defined by a set of authorization constraints). We show that a number of problems in the literature on workflow satisfiability and resiliency are special cases of APEP, thereby showing that APEP is computationally hard.

The framework within which APEP is defined admits a greater variety of constraints than is usually considered in either the standard access control literature [6], [15], [18], [23] or in workflow satisfiability [1], [7], [26]. In this paper we characterize the constraints of interest and extend the definition of user-independent constraints [7] to this framework. We then establish the complexity of APEP for certain types of constraints, using both classical and multivariate complexity analysis. In this paper, we make some progress in this direction by establishing the complexity of APEP for the constraints that we believe will be the most useful in practice. In particular, we establish connections between APEP and both the Workflow Satisfiability Problem and resiliency in access control.

In the next section, we summarize relevant background material and related work. We introduce the APEP in Section 3 and elaborate on the nature of the constraints we consider in Section 4. We then discuss further constraint types and connections between APEP and existing problems in the literature. In Section 6, we investigate the complexity of several variants of the APEP. We conclude the paper with a summary of our contributions and some ideas for future

- *P. Bergé is a PhD student at at LRI, Université Paris-Sud, Université Paris-Saclay, France.*
  E-mail: pierre.berge@lri.fr
- *J. Crampton is with the Department of Information Security, Royal Holloway, University of London, United Kingdom.*
  E-mail: jason.crampton@rhul.ac.uk
- *G. Gutin is with the Department of Computer Science, Royal Holloway, University of London, United Kingdom.*
  E-mail: G.Gutin@rhul.ac.uk
- *R. Watrigant is with Université de Lyon, UCBL1, CNRS, LIP, ENS de Lyon, France.*
  E-mail: remi.watrigant@univ-lyon1.fr

1. An extended abstract of this paper appeared in the Proceedings of the Seventh ACM Conference on Data and Application Security and Privacy [3]. Research of J. Crampton and G. Gutin was partially supported by grant RPG-2018-161 of Leverhulme Trust. Research of G. Gutin was also partially supported by Royal Society Wolfson Research Merit Award.

work.

2 Background and related work

A number of interesting (and computationally hard) problems arise naturally in the context of authorization policies and constraints. However, the relative sizes of the parameters in many of these problems mean that it is fruitful to analyze these problems using multivariate complexity analysis. In this section, we review some of those problems and provide a brief introduction to fixed-parameter tractability.

2.1 Fixed-parameter tractability

Many problems take multiple inputs and the complexity of solving such problems is determined by the sizes of those inputs. In general, a problem may be hard in terms of the total size of the input. However, if we consider the complexity of a problem under an assumption that some of the parameters of the input are small and terms that are exponential in those parameters are acceptable, then we may discover that relatively efficient algorithms exist to solve the problem.

More formally, an algorithm is said to be *fixed-parameter tractable* (FPT) if it solves a decision problem in time $O(f(k)p(n))$, where k is some (small) parameter of the problem, n is the total size of the input, and f and p are, respectively, a computable function and a polynomial. As is customary in the literature on FPT algorithms, we will often write $O(f(k)p(n))$ as $O^*(f(k))$. (That is, O^* suppresses polynomial factors, as well as multiplicative constants.) If a problem can be solved using an FPT algorithm then we say that it is an *FPT problem* and that it belongs to the class FPT [14], [21]. An FPT algorithm for a hard problem is particularly valuable when k is significantly smaller than n for most instances of the problem that arise in practice.

While many NP-complete problems admit FPT algorithms, many other NP-complete problems do not admit such algorithms unless certain parameterized complexity-theory assumptions fail, which is highly unlikely [14], [21]. In particular, the Workflow Satisfiability Problem is highly unlikely to admit an FPT algorithm [26]. However, if all constraints are user-independent, then the problem is FPT and efficient algorithms have been developed to solve practical instances of the problem [7], [11], [16]. Since the APEP is an extension of the Workflow Satisfiability Problem, it is also highly unlikely to admit an FPT algorithm. In this paper we identify important restrictions to the general APEP which do admit such algorithms.

2.2 Workflow satisfiability

A workflow may be modeled as a set of steps in some automated business process. An authorization relation determines which users are authorized to perform which steps, and constraints restrict which subsets of users may perform subsets of steps [4], [9], [26]. Given a set of users U, a set of workflow steps S, an authorization relation $A \subseteq S \times U$, and a set of constraints C, a plan $\pi : S \to U$ is a function allocating users to steps. A valid plan must allocate an authorized user to each step and every constraint must be satisfied. The Workflow Satisfiability Problem (WSP) asks whether there exists a valid plan for a given U, S, A and C. Basin, Burri and Karjoth model a workflow (with "break points") as a process algebra and introduced the notion of an *enforcement process*, which is analogous to a valid plan [1]. This leads naturally to the *enforcement process existence* problem, which is analogous to the workflow satisfiability problem and inspires the name for the problem we study in this paper.

Wang and Li observed that fixed-parameter algorithmics is an appropriate way to study WSP, because the number of steps is usually small and often much smaller than the number of users.[2] Wang and Li [26] proved that WSP is FPT if we consider only separation-of-duty and binding-of-duty constraints [26]. We will write such constraints as $(s, s', \neq)$ and $(s, s', =)$, respectively, where $s, s' \in S$. A plan π satisfies a constraint $(s, s', \neq)$ ($(s, s', =)$, respectively) if $\pi(s) \neq \pi(s')$ ($\pi(s) = \pi(s')$, respectively). WSP with only separation-of-duty constraints (only binding-of-duty constraints, respectively) will be denoted by WSP($\neq$) (WSP(=), respectively). Subsequent research has shown that WSP remains FPT even if additional types of constraints, notably user-independent [7] and class-independent constraints [10], are permitted in the input to WSP. Note that WSP is not FPT in general [26] unless a widely-accepted hypothesis in complexity theory fails; a significant body of research suggests this is highly unlikely [14].

2.3 Static separation-of-duty constraints

Constraints have been studied extensively in the context of role-based access control (RBAC) [15], [18], [23], [25]. In its simplest form, a static separation-of-duty constraint may be defined as a pair of roles $\{r, r'\}$ belonging to the set of roles R. A user-role assignment relation $UR \subseteq U \times R$, where U is the set of users, satisfies the constraint $\{r, r'\}$ if there is no user u such that (u, r) and (u, r') belong to UR.

More generally, Li, Tripunitara and Bizri [18] introduced the notion of a k-out-of-n static separation-of-duty constraint, defined as a pair (R', n), where R' is a subset of R of cardinality k. A user-role assignment relation $UR \subseteq U \times R$ satisfies the constraint (R', n) if there is no set of t users, where $t < k$, that are collectively authorized for R'. That is, for all subsets V of U such that $|V| < k$,

$$\bigcup_{u \in V} \{r : (u, r) \in UR\} \subset R'.$$

Note that the simple separation-of-duty constraint defined by a pair of roles $\{r, r'\}$ is a 2-out-of-2 separation-of-duty constraint.

2.4 Resiliency

Li, Wang and Tripunitara introduced the idea of resiliency in access control [19]. Informally, a resiliency policy is a requirement that even in the absence of a limited number of users the remaining users can be authorized for some set of resources such that given constraints are satisfied. The existence of both a resiliency policy and authorization

2. The SMV loan origination workflow studied by Schaad *et al.*, for example, has 13 steps and identifies five roles [24]. It is generally assumed that the number of users is significantly greater than the number of roles.

constraints may mean that no authorization relation can satisfy all requirements. Li *et al.* introduce a number of problems investigating whether an authorization relation does exist for a given resiliency policy [19].

3 The Authorization Policy Existence Problem

In this paper, we extend existing work on workflow satisfiability, constraints and resiliency, by defining a simple yet very expressive authorization framework. Roughly speaking, we specify a problem concerned with the existence of an appropriate authorization relation.

Given a set of users U and a set of resources R to which access should be restricted, we may define an *authorization relation* $A \subseteq U \times R$, where $(u, r) \in A$ if and only if u is authorized to access r. Given a resource r, we will write $A(r)$ to denote the set of users that are authorized to access resource r. More formally, $A(r) = \{u \in U : (u, r) \in A\}$. Similarly, for $u \in U$, we will write $A(u)$ to denote the set of resources that u is authorized to access, that is $A(u) = \{r \in R : (u, r) \in A\}$. We extend this notation to subsets of R and U in the natural way: for $R' \subseteq R$ and $U' \subseteq U$,

$$A(R') \stackrel{\text{def}}{=} \bigcup_{r \in R'} A(r) \quad \text{and} \quad A(U') \stackrel{\text{def}}{=} \bigcup_{u \in U'} A(u).$$

We introduce two fundamental concepts, which will be used to formulate the Authorization Policy Existence Problem.

- a *base authorization relation* $A_{\mathsf{Bse}} \subseteq U \times R$ such that $A_{\mathsf{Bse}}(r) \neq \emptyset$ for each $r \in R$;
- a set of *authorization constraints* C.

Informally, A_{Bse} specifies restrictions on all valid authorization relations, while C specifies additional restrictions that any valid authorization relation must satisfy.

An authorization constraint may be defined by enumerating the set of all authorization relations that satisfy the constraint. Of course, an extensional definition of this nature is utterly impractical, and all useful constraints will be defined in an intensional manner. A simple example would be a constraint requiring no user is assigned to both resources r and r'. In other words, an authorization relation A satisfies this constraint provided that $\{(u, r), (u, r')\} \not\subseteq A$ for all $u \in U$. We discuss constraints in more detail in Section 4.

More formally, we have the following definitions. Given a base authorization relation A_{Bse} and a set of constraints C, we say an authorization relation $A \subseteq U \times R$ is

- *authorized* with respect to A_{Bse} if $A \subseteq A_{\mathsf{Bse}}$;
- *complete* if $A(r) \neq \emptyset$ for every $r \in R$;
- *eligible* with respect to $c \in C$ if A satisfies c;
- *eligible* with respect to C if A satisfies c for all $c \in C$;
- *valid* with respect to A_{Bse} and $c \in C$ if A is authorized, complete and eligible with respect to c; and
- *valid* with respect to A_{Bse} and C if A is authorized, complete and eligible with respect to C.

We introduce the term Authorization Policy Existence Problem (APEP) as a generic term for questions concerned with finding a valid authorization relation, given A_{Bse} and C. Then APEP comes in two "flavors": (a) **Decision (D-APEP):** Does there exist a valid authorization relation? If so, find a valid authorization relation. (b) **Optimization (O-APEP):** Find a "best" valid authorization relation if one exists (where the objective function has to be specified).

We assume that determining whether an authorization relation satisfies a constraint takes polynomial time. (This is a reasonable assumption for all constraints of relevance to access control.) Let n denote $|U|$, k denote $|R|$ and m denote $|C|$. Then a brute force approach to solving D-APEP (by simply examining every possible authorization relation) takes time $O^*(2^{nk})$.

A few special cases of APEP are worth mentioning. For simplicity we will write A_{Sol} to denote one of the authorization relations that are solutions to APEP.

1) A_{Sol} is required to be a *function* $A_{\mathsf{Sol}} : R \to U$. In this case, we allocate a unique user to each resource. Computing a plan allocating one user to each step in a workflow instance, subject to an authorization policy defined by A_{Bse} and some constraints C, is an example of this type of scenario. In this case, D-APEP corresponds to the Workflow Satisfiability Problem (WSP) [26]. Moreover, the Cardinality-Constrained Minimum User Problem [22], whose solution is a plan using the minimum number of users, is an instance of O-APEP.
2) $A_{\mathsf{Bse}} = U \times R$. In this case, A_{Bse} itself imposes no restrictions on A_{Sol}. All restrictions on A_{Sol} are defined by the constraints C. Defining an authorization policy in the presence of static separation-of-duty constraints is an example of this type of scenario. Li, Tripunitara and Bizri have studied problems of this nature [18].
3) A constraint that requires each resource is assigned to at least t users enables us to model problems related to resiliency in access control [19] and workflow systems [26].
4) If we seek to maximize the cardinality of A_{Sol}, then, informally, a solution to O-APEP provides a "resilient" authorization policy. While this is different from existing notions of resiliency [19], [26], it would seem to be an interesting way of approaching the problem of making an authorization policy resilient to the unavailability of users.

4 Constraints

We now describe constraints in more detail. We generalize the approach used in prior work on constraints for workflow systems [13], [26].

4.1 Binding-of-duty and separation-of-duty constraints

Binding-of-duty and *separation-of-duty* constraints have received considerable attention in the access control literature, and such constraints may be *static* or *dynamic*. Informally, static constraints specify restrictions on policy relations, whereas dynamic constraints specify constraints on particular sequences of events within the context of an access control system. A (static) separation-of-duty constraint, for example, in the context of role-based access control system, might require that no user is authorized for both roles r

and r' [23]. In contrast, a (dynamic) separation-of-duty constraint, in the context of a workflow system, might simply require that two steps s and s' are performed by different users in each instance of the workflow [9], [26]. (This constraint, however, does not prevent the same user being authorized for both those steps.) Within the framework of APEP, we seek to define a more general (and uniform) syntax and semantics for constraints.

We express constraints in terms of the following logical (binary) operators defined via their respective truth tables:

p	q	$p \leftrightarrow q$	$p \rightarrow q$	$p \leftarrow q$	$p \updownarrow q$
0	0	1	1	1	0
0	1	0	1	0	1
1	0	0	0	1	1
1	1	1	1	1	0

Let r and r' be resources in R; let $\circ$ denote one of the logical operators in the set $\{\leftrightarrow, \leftarrow, \rightarrow, \updownarrow\}$; and let Q be one of the first-order quantifiers $\exists$ or $\forall$. Then $(r, r', \circ, Q)$ is a constraint: a constraint of the form $(r, r', \circ, \forall)$ is said to be *universal*, while a constraint of the form $(r, r', \circ, \exists)$ is said to be *existential*. A complete relation A

- satisfies $(r, r', \circ, \exists)$ if there exists $u \in A(r) \cup A(r')$ such that the propositional formula $(u \in A(r)) \circ (u \in A(r'))$ evaluates to true; and
- satisfies $(r, r', \circ, \forall)$ if for all $u \in A(r) \cup A(r')$, the propositional formula $(u \in A(r)) \circ (u \in A(r'))$ evaluates to true.

Note that for any complete relation A and any $r \in R$, $A(r) \neq \emptyset$, so $A(r) \cup A(r') \neq \emptyset$. Thus constraints cannot be vacuously satisfied by a valid relation.

Informally speaking, universal constraints are "stronger" than existential constraints: (for any complete relation) the satisfaction of $(r, r', \circ, \forall)$ implies the satisfaction of $(r, r', \circ, \exists)$, but the converse does not hold.

We now expand these generic definitions for the four constraints defined by $(r, r', \circ, Q)$, where $\circ$ is either $\leftrightarrow$ or $\updownarrow$:

1) $(r, r', \leftrightarrow, \exists)$ is satisfied if there exists $u \in U$ such that $u \in A(r)$ and $u \in A(r')$; that is, $A(r) \cap A(r') \neq \emptyset$.
2) $(r, r', \updownarrow, \exists)$ is satisfied if there exists $u \in U$ such that either (i) $u \in A(r)$ and $u \notin A(r')$ or (ii) $u \notin A(r)$ and $u \in A(r)$; that is, $A(r) \neq A(r')$.
3) $(r, r', \leftrightarrow, \forall)$ is satisfied if for all $u \in A(r) \cup A(r')$, $u \in A(r)$ if and only if $u \in A(r')$; that is, $A(r) = A(r')$.
4) $(r, r', \updownarrow, \forall)$ is satisfied if for all $u \in A(r) \cup A(r')$, either (i) $u \in A(r)$ and $u \notin A(r')$ or (ii) $u \notin A(r)$ and $u \in A(r')$; that is, $A(r) \cap A(r') = \emptyset$.

Thus, constraints of the form $(r, r', \updownarrow, Q)$ correspond closely to the idea of separation-of-duty. Indeed, the satisfaction criterion for $(r, r', \updownarrow, \forall)$ is identical to that for a simple static separation-of-duty constraint. Similarly, constraints of the form $(r, r', \leftrightarrow, Q)$ correspond to the idea of binding-of-duty.

Now consider a constraint of the form $(r, r', \rightarrow, \forall)$. Such a constraint is satisfied if for all $u \in A(r) \cup A(r')$, $(u \in A(r)) \rightarrow (u \in A(r'))$. In other words, $A(r) \subseteq A(r')$. Thus a global constraint of this form could be used to specify a role hierarchy (in which role r' is senior to r). Conversely, a constraint of the form $(r, r', \leftarrow, \forall)$ could be used to specify a role hierarchy in which r' is junior to to r.[3] Thus we can use constraints to insist that a hierarchy is strict: that is, there exists at least one user that is assigned to r' but not r. Specifically, a relation A simultaneously satisfies constraints $(r, r', \rightarrow, \forall)$ and $(r, r', \updownarrow, \exists)$ only if $A(r) \subset A(r')$.

4.2 Cardinality constraints

We may also define *cardinality constraints*, which come in two flavors. In the following, $\lhd$ is a binary relation belonging to the set $\{=, <, >, \leqslant, \geqslant\}$ and t is an integer greater than 0.

- A *global* (cardinality) constraint has the form $(\lhd, t)$. The constraint $(\lhd, t)$ is satisfied by relation A if for all $r \in R$, $|A(r)| \lhd t$.
- A *local* (cardinality) constraint has the form $(R', \lhd, t)$, where $R' \subseteq R$. The constraint $(R', \lhd, t)$ is satisfied by relation A if $|A(R')| \lhd t$.

Then, for example, the global constraint $(=, 1)$ requires a valid relation A to be a function (since the number of users assigned to each resource is precisely 1), while the local constraint $(\{r\}, \leqslant, t)$ is a cardinality constraint in the RBAC96 sense [23] (if resource r is interpreted as a role). Finally, the k-out-of-n static separation-of-duty constraint $\mathsf{ssod}(\{r_1, \ldots, r_k\}, n)$, introduced by Li *et al.* [18], may be represented by the cardinality constraint $(\{r_1, \ldots, r_k\}, \geqslant, n)$.

Remark 1. *If we define a global constraint $(=, 1)$, then the universal constraint $(r, r', \circ, \forall)$ is equivalent to the existential constraint $(r, r', \circ, \exists)$ (in the sense that the former is satisfied if and only if the latter is).*

4.3 User-independent constraints

Research on the WORKFLOW SATISFIABILITY PROBLEM has shown that the notion of *user-independent* (UI) constraints is very important. First, the class of UI constraints includes a very wide range of constraints and almost all constraints that are of relevance to access control. Thus, we believe that many constraints of interest for applications in the future will also be user-independent. Second, WSP is fixed-parameter tractable (FPT) if we restrict attention to UI constraints [7]. (WSP is not FPT if we allow arbitrarily complex constraints [26].) Informally, a constraint is UI in the context of workflow satisfiability if its satisfaction only depends on the relationships that exist between users assigned to steps in a workflow (and not on the specific identities of users) [7]. We now extend the definition of user-independent used in the context of workflow satisfiability.

Let A be an authorization relation and $\sigma : U \rightarrow U$ a permutation of the user set (that is, σ is a bijection). Then, given an authorization relation $A \subseteq U \times R$, we write $\sigma(A) \subseteq U \times R$ to denote the relation $\{(\sigma(u), r) : (u, r) \in A\}$. A constraint c is *user-independent* if for every authorization relation A that satisfies c and every permutation $\sigma : U \rightarrow U$, $\sigma(A)$ satisfies c.

Consider, for example, a constraint of the form $(r, r', \leftrightarrow, \exists)$ and suppose that $A \subseteq U \times R$ satisfies the constraint.

3. Since $A(r)$ and $A(r')$ are non-empty, the constraints $(r, r', \leftarrow, \exists)$ and $(r, r', \rightarrow, \exists)$ are both equivalent to $(r, r', \leftrightarrow, \exists)$.

Then, by definition, there exists a user u such that $(u, r), (u, r') \in A$. Then, for any permutation σ, $(\sigma(u), r), (\sigma(u), r') \in \sigma(A)$, so $\sigma(A)$ also satisfies the constraint. Similar arguments may be used to show that constraints of the form $(r, r', \leftrightarrow, \forall)$, $(r, r', \updownarrow, \exists)$ and $(r, r', \updownarrow, \forall)$ are all UI. Equally, it is clear that global and local constraints, whose satisfaction is defined in terms of the cardinality of sets of the form $A(r)$, are UI, since a permutation (being a bijection) will preserve the cardinality of such sets. In other words, all constraints we consider in this paper are UI.

4.4 Bounded UI constraints

We now define an important class of UI constraints that will be useful for establishing positive results in the remainder of the paper. Given a base relation A_{Bse} and a constraint c, let A be valid with respect to A_{Bse} and c. We say A *requires* v if $\{(u, r) \in A : u \neq v\}$ is not valid. (Since A is valid, this means that $\{(u, r) \in A : u \neq v\}$ is either incomplete or does not satisfy c.) Then we define

$$\mathsf{core}(A : A_{\mathsf{Bse}}, c) \stackrel{\text{def}}{=} \{u \in U : A \text{ requires } u\}$$

to be the *core of A with respect to A_{Bse} and c.*

Consider for instance a constraint c of the form $(r, r', \updownarrow, \forall)$. If there exists an authorization relation A satisfying c, then there is one whose core contains at least two users: indeed, remove iteratively from A any pair $(u, r) \in A$ such that A does not require u. When this is no longer possible, the obtained relation has to allocate two distinct users to r and r', both belonging to the core. The core could contain as many as k users, since the relation may allocate each resource to a different user and removing any user would compromise the completeness of the relation. However, the core cannot contain more than k users, since in any set of at least $k + 1$ users, at least two users must be allocated to the same resource and one of them could be removed without compromising the completeness or the eligibility of the relation. Conversely, for a constraint of the form $(r, r', \leftrightarrow, \forall)$ the core could contain a single user but no core contains more than $k - 1$ users, since r and r' must be assigned to the same user and any additional users may be removed without compromising completeness or eligibility.

Proposition 2. *Let $\mathcal{I} = (A_{\mathsf{Bse}}, C)$ be a satisfiable instance of* D-APEP *with a UI constraint $c \in C$. If A is a valid solution with respect to A_{Bse} and c then*

$$|\mathsf{core}(A : U \times R, c)| \geq |\mathsf{core}(A : A_{\mathsf{Bse}}, c)|$$

Proof. We prove the more general statement that

$$\mathsf{core}(A : A_{\mathsf{Bse}}, c) \subseteq \mathsf{core}(A : \Omega, c)$$

for any $\Omega \supseteq A_{\mathsf{Bse}}$. Suppose A requires u and let $A \setminus u$ denote $\{(v, r) \in A : u \neq v\}$. Then $A \setminus u$ is either incomplete or violates c (since A is authorized, so is $A \setminus u$). If $A \setminus u$ is incomplete, then it is also incomplete for the instance (Ω, c). The same argument holds if $A \setminus u$ violates c, which concludes the proof. □

Definition 3. *We say a UI constraint c is* $f(k, n)$-bounded *if $|\mathsf{core}(A : U \times R, c)| \leqslant f(k, n)$ for all A valid with respect to $U \times R$ and c.*

The definition of $f(k, n)$-bounded constraints and Proposition 2 impose an upper bound on the number of users we need to consider when constructing candidate solutions to an instance (A_{Bse}, C) of D-APEP, and thus plays an important role in determining whether a particular variant of APEP is FPT. In particular, we show in Section 6 that D-APEP is FPT when all constraints are $f(k)$-bounded for some function f.

We have the following elementary results, the proofs of which can be found in the appendix. The results are summarized, for ease of reference, in Table 1.

Proposition 4. *Constraints $(r', r'', \rightarrow, \forall)$, $(r', r'', \leftrightarrow, \forall)$ and $(r', r'', \leftrightarrow, \exists)$ are $(k - 1)$-bounded.*

Proposition 5. *Constraints $(r', r'', \updownarrow, \forall)$ and $(r', r'', \updownarrow, \exists)$ are k-bounded.*

Proposition 6. *Constraints $(R', \leq, t)$ and $(\leq, t)$ are k-bounded.*

Remark 7. *Obviously, Proposition 6 remains true when we replace $\leq$ with $<$. However, it does not hold if we replace $\leq$ by $=$ or $\geq$. Indeed, a constraint such as $(=, t)$ requires that some set of t users cannot be removed. Hence, if t is not bounded by a function of k only, the constraint is not $f(k)$-bounded for any computable function f.*

Proposition 8. *Constraints $(R', =, t)$ and $(R', \geq, t)$ are $2\max\{k, t\}$-bounded, but not $(\max\{k, t\} - 1)$-bounded.*

Constraint Type	Largest Core
$(r, r', \updownarrow, \forall)$, $(r, r', \updownarrow, \exists)$	k
$(r, r', \leftrightarrow, \forall)$, $(r, r', \rightarrow, \forall)$, $(r, r', \leftrightarrow, \exists)$	$k - 1$
$(R', \leq, t)$, $(\leq, t)$	k
$(R', =, t)$, $(R', \geq, t)$	$2\max\{k, t\}$

TABLE 1: Upper bounds on the size of the core

4.5 Notation

In the remainder of this paper, we consider versions of APEP in which we restrict our attention to particular types of constraints. We use the following abbreviations for families of constraints: (i) BoD to denote the family of all existential and universal (binding-of-duty) constraints having the form $(r, r', \leftrightarrow, \exists)$ or $(r, r', \leftrightarrow, \forall)$; (ii) SoD to denote the family of all existential and universal (separation-of-duty) constraints having the form $(r, r', \updownarrow, \exists)$ or $(r, r', \updownarrow, \forall)$; (iii) $\mathsf{BoD_E}$ and $\mathsf{BoD_U}$ to denote, respectively, the family of all existential and universal binding-of-duty constraints; (iv) $\mathsf{SoD_E}$ and $\mathsf{SoD_U}$ to denote, respectively, the family of all existential and universal separation-of-duty constraints; (v) $f(k)$-bounded to denote $f(k)$-bounded constraints; (vi) G_{card} to denote the family of all global cardinality constraints; (vii) L_{card} to denote the family of all local cardinality constraints. Finally, we write, for example, $\text{APEP}\langle\mathsf{BoD}, \mathsf{L}_{card}\rangle$ to restrict the set of instances of APEP in which the set of constraints C contains only binding-of-duty and local cardinality constraints.

5 Further applications of APEP

In Section 6, we establish the computational complexity of a number of variants of APEP. In particular, we prove that

APEP is FPT provided all constraints are $f(k)$-bounded. In this section, we consider a number of extensions to the basic constraints we have already defined and their relevance as extensions to authorization policies. We also discuss a number of problems associated with constraints and policies from the literature and demonstrate why they may be considered to be instances of APEP, thereby establishing that these problems are FPT.

5.1 Constraint types

In Section 4.1 we identified a number of constraint types of the form $(r, r', \circ, Q)$, where r and r' are resources, $\circ$ is a logical binary operator, and Q is a quantifier. For ease of reference we summarize these constraints and the respective conditions for satisfaction in Table 2.

$(r, r', \leftrightarrow, \forall)$	$A(r) = A(r')$
$(r, r', \leftrightarrow, \exists)$	$A(r) \cap A(r') \neq \emptyset$
$(r, r', \updownarrow, \forall)$	$A(r) \cap A(r') = \emptyset$
$(r, r', \updownarrow, \exists)$	$A(r) \neq A(r')$
$(r, r', \rightarrow, \forall)$	$A(r) \subseteq A(r')$

TABLE 2: Constraint types defined in Section 4.1

We chose to introduce the constraints shown in Table 2 because of their obvious connections to known constraints in the literature and to simplify the exposition of the technical material. We now discuss ways in which these constraints could be extended. Notice that the satisfaction of each constraint may be defined in terms of $A(r)$ and $A(r')$.

One obvious extension, then, is to define constraints of the form $((r_1, \ldots, r_m), \circ, Q)$, and to define constraint satisfaction in terms of $A(r_i)$, $1 \leqslant i \leqslant m$. We may define constraint satisfaction in a number of ways, including (but not limited to) the following: (i) for *all* i and j, $1 \leqslant i < j \leqslant m$, $(r_i, r_j, \circ, Q)$ is satisfied; or (ii) for *some* i and j, $1 \leqslant i < j \leqslant m$, $(r_i, r_j, \circ, Q)$ is satisfied. Note, however, that the first of these choices can be realized simply by defining a set of constraints $\{(r_i, r_j, \circ, Q) : 1 \leqslant i < j \leqslant m\}$.

Consider the constraint $((r_1, \ldots, r_m), \updownarrow, \forall)$, and suppose, as another alternative for constraint satisfaction, we require that $\bigcap_{i=1}^{m} A(r_i) = \emptyset$. In other words, there is no user that is assigned to all resources in the set $\{r_1, \ldots, r_m\}$. It is easy to see that such a constraint is k-bounded (since removing a user from a valid relation can only affect completeness, not the eligibility, of the relation). Thus, with this interpretation, $((r_1, \ldots, r_m), \updownarrow, \forall)$ represents a canonical SMER constraint [18] (if the set of resources is interpreted as a set of mutually exclusive roles). We return to SMER constraints in Section 5.3.

Another possible extension is to define constraints of the form $(R', R'', \circ, Q)$ and to define constraint satisfaction in terms of $A(R')$ and $A(R'')$. For example, the constraint $(R', R'', \updownarrow, \forall)$ requires that $A(R') \cap A(R'') = \emptyset$. Again, constraints of this form are k-bounded. In other words, the users assigned to resources in R' are different from the users assigned to resources in R''. This constraint, therefore, allows us to specify that resources should be allocated to disjoint *teams* of users (rather than just individual users). Of course such constraints could be nested: we might define a further constraint $(R''_1, R''_2, \updownarrow, \forall)$ where R''_1 and R''_2 are subsets of R''.

5.2 Resiliency in access control

Suppose we are given an authorization relation $A \subseteq U \times R$ and a set of resources $Q \subseteq R$. Then a resiliency policy is defined by a tuple (Q, s, d, t), where s, d and t are integers [19]. The policy is satisfied if, following the removal of any s users from U, there exist d disjoint teams of users, $U_1, \ldots, U_d$, such that $A(U_i) \supseteq Q$ and $|U_i| \leqslant t$ for each i.

The *resiliency checking problem* [19] asks whether a resiliency policy is satisfiable or not. It has been shown that the hard part of the problem is finding the teams (since we can enumerate all possible user sets that are missing s users), so research has focused on solving the problem for instances in which $s = 0$ [12], [19].

Informally, a solution of the resiliency checking problem may be viewed as a function mapping (different copies of the set of) resources to users. Thus we can transform an instance of the resiliency checking problem (where $s = 0$) into an instance of D-APEP. We define d copies of each resource in Q; we write $r^{(i)}$ to denote the ith copy of resource r in Q. We then define

$$A_{\mathsf{Bse}} = \left\{(u, r^{(i)}) : (u, r) \in A, 1 \leqslant i \leqslant d\right\}.$$

Finally, we define the global constraint $(=, 1)$ and, for all $r_1, r_2 \in Q$ and all i and j such that $1 \leqslant i < j \leqslant d$, we define a constraint $(r_1^{(i)}, r_2^{(j)}, \updownarrow, \forall)$. The authorization relation A_{Bse} ensures that each user is authorized according to the original relation A. The global constraint ensures that each resource is assigned to a single user. The other constraints ensure that a user is only assigned to resources in one copy of Q.

The results in Section 6.2 assert that the resulting problem is FPT. Hence, the resiliency checking problem is also FPT (confirming an earlier result of Crampton *et al.* [12]).

Note, finally, that we can simplify the above construction, using the constraints introduced in Section 5.1: we use $Q^{(i)}$ to denote the ith copy of the set of resources Q and define the set of constraints

$$\left\{(Q^{(i)}, Q^{(j)}, \updownarrow, \forall) : 1 \leqslant i < j \leqslant d\right\}.$$

5.3 Resiliency and separation of duty

The RBAC96 standard discusses constraints based on mutually exclusive roles [23]. Such a constraint is defined by a set of roles R_{mutex} and is satisfied by the user-role assignment relation provided no user is assigned to more than one role in R_{mutex}.

Li, Tripunitara and Bizri introduced the more general *static mutually exclusive role* (SMER) constraints [18], which have the form (R_{mutex}, t), where $t \leqslant |R_{\mathsf{mutex}}|$. (A canonical SMER constraint has $t = |R_{\mathsf{mutex}}|$.) Such a constraint is satisfied provided every user is assigned fewer than t of the roles in R_{mutex}. We can check whether a user-role assignment relation satisfies a SMER constraint in polynomial time [18], informally because we only need to consider each user once.

Li *et al.* went on to distinguish SMER constraints from *static separation of duty* (SSoD) policies [18], which are defined by a set of permissions P and an integer $t \leqslant |P|$. Such a constraint is satisfied if no subset of fewer than t users is collectively authorized for the permissions in P. Checking whether a SSoD policy is satisfied by a given user-role assignment relation is computationally hard, informally

because we need to consider every possible subset of users having cardinality less than t.

Li, Wang and Tripunitara studied the complexity of determining whether it was possible to simultaneously satisfy static separation of duty constraints and a resiliency policy [19]. Unsurprisingly, it is computationally hard to decide this question, given that it is hard to decide whether an authorization relation satisfies a static separation of duty policy [18]. However, they did not consider the possibility of simultaneously satisfying SMER constraints and resiliency policies. Now observe that a SMER constraint is user-independent and is k-bounded. Thus, for example, it is possible to develop an FPT algorithm to determine whether there exists an authorization relation $A \subseteq U \times R$ such that a resiliency policy and a set of SMER constraints are simultaneously satisfied.

6 Complexity of APEP

Before exploring the fine-grained complexity of D-APEP with respect to the different types of constraints, we first state general properties of some special cases.

Firstly, note that adding the function constraint F to any D-APEP instance having only SoD or BoD constraints forces any solution A_{Sol} of D-APEP to be a function. In this case, D-APEP becomes equivalent to WSP. More formally, we say that a parameterized problem A is *parameter-reducible* to a parameterized problem B (and we write $A \leq_{fpt} B$) if there is a polynomial algorithm which transforms an instance (I, k) of A into an instance (I', k') of B such that (I, k) is positive for A iff (I', k') is positive for B, and $k' \leq f(k)$ for some computable function f. Clearly, if A is parameter-reducible to B and B is FPT, then A is FPT. We say that A is *parameter-equivalent* to B (and we write $A =_{fpt} B$) if $A \leq_{fpt} B$ and $B \leq_{fpt} A$. The proof of the following result is straightforward, by the definition of the function constraint F.

Theorem 9. *If* D-APEP *and WSP are parameterized by the number of resources and steps, respectively, we have the following:*

- D-APEP$\langle \mathsf{SoD}, \mathsf{F} \rangle =_{fpt}$ *WSP*$(\neq)$*;*
- D-APEP$\langle \mathsf{BoD}, \mathsf{F} \rangle =_{fpt}$ *WSP*$(=)$*; and*
- D-APEP$\langle \mathsf{BoD}, \mathsf{SoD}, \mathsf{F} \rangle =_{fpt}$ *WSP*$(=, \neq)$.

Moreover, the following result asserts that adding $\mathsf{BoD_U}$ constraints to any instance of D-APEP does not change its (parameterized) complexity.

Theorem 10. *Given any instance* $(U, R, A_{\mathsf{Bse}}, C)$ *of* D-APEP$\langle \mathsf{BoD}, \mathsf{SoD} \rangle$*, one can obtain in polynomial time an instance* $(U, R', A'_{\mathsf{Bse}}, C')$ *of* D-APEP *such that: (i)* C' *does not contain any* $\mathsf{BoD_U}$ *constraint, (ii)* $|A'_{\mathsf{Bse}}| \leq |A_{\mathsf{Bse}}|$*, (iii)* $|R'| \leq |R|$*, and (iv)* $|C'| \leq |C|$*.*

Proof. Let C^* be the set of $\mathsf{BoD_U}$ constraints from C. The idea is to consider $\mathsf{BoD_U}$ constraints as an equivalence relation: for $r, r' \in R$, $r \sim_b r'$ if and only if $(r, r', \leftrightarrow, \forall) \in C^*$. Now, let $R' = \{R_1, \ldots, R_q\}$ be the equivalence classes of $\sim_b$. Obviously, $|R'| \leq |R|$. For all $i \in [q]$ and all $r \in R_i$, let $A'_{\mathsf{Bse}}(r)$ denote $\bigcap_{r' \in R_i} A_{\mathsf{Bse}}(r')$. Once again, it holds that $|A'_{\mathsf{Bse}}| \leq |A_{\mathsf{Bse}}|$. Finally, for every constraint $c = (r, r', \sim, Q)$ in $C \setminus C^*$ with $\sim \in \{=, \neq\}$ and $Q \in \{\forall, \exists\}$, we distinguish two cases:

- if $r \in R_i$ and $r' \in R_j$ with $i \neq j$, then add the constraint $(R_i, R_j, \sim, Q)$ (notice that in this case c is not a $\mathsf{BoD_U}$ constraint);
- if $r, r' \in R_i$ for some $i \in [k]$, then if c is a $\mathsf{SoD_U}$ or $\mathsf{SoD_E}$ constraint, obviously the instance is unsatisfiable (and we can output a trivially negative instance of D-APEP).

Clearly $|C'| \leq |C|$, and C' does not contain any $\mathsf{BoD_U}$ constraint. Finally, one can check that the output instance is satisfiable if and only if the input instance is satisfiable. □

Remark 11. *A corollary of Theorem 10 is that we may exclude* $\mathsf{BoD_U}$ *constraints from consideration when establishing FPT results.*

In the remainder of this section, we establish that D-APEP with bounded constraints is FPT. We also propose extra algorithms for some mixed policies composed of BoD and SoD constraints in order to improve the execution time of the main algorithm for these subcases. Figure 1 summarizes our results for D-APEP with BoD and SoD constraints.

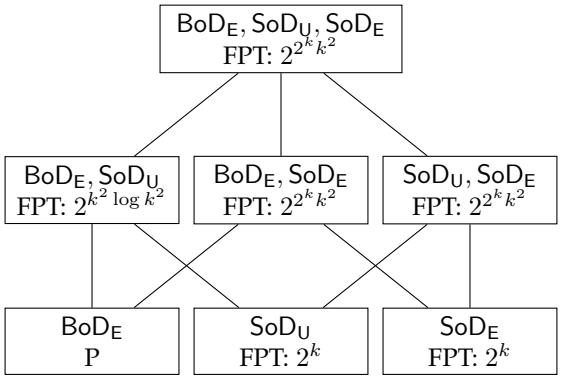


Fig. 1: Complexity of specific cases of D-APEP (polynomial factors are ignored)

6.1 Instances with constraints of a single type

As a direct application of Theorem 10, we are able to show that D-APEP$\langle \mathsf{BoD_U} \rangle$ is polynomial-time solvable: indeed, after using the reduction of the previous result, it is clear that the instance is satisfiable if and only if A'_{Bse} is complete.

Theorem 12. D-APEP$\langle \mathsf{BoD_U} \rangle$ *is solvable in polynomial time.*

We now consider the complexity of D-APEP$\langle \mathsf{SoD_U} \rangle$ and D-APEP$\langle \mathsf{BoD_E} \rangle$.

Theorem 13. D-APEP$\langle \mathsf{SoD_U} \rangle =_{fpt}$ D-APEP$\langle \mathsf{SoD_U}, \mathsf{F} \rangle$.

Proof. It is sufficient to prove that there always exists a feasible solution of D-APEP$\langle \mathsf{SoD_U} \rangle$ which satisfies the function constraint F.

Let us suppose $(U, R, A_{\mathsf{Bse}}, C)$ is satisfiable, so there exists a valid solution A_{Sol} for this instance. We define another relation A' which is a function. For any $r \in R$, we have $|A_{\mathsf{Sol}}(r)| \geq 1$, thus we can pick an arbitrary user u_r in $A_{\mathsf{Sol}}(r)$ and set $A'(r) = \{u_r\}$. One can observe that $A' \subseteq A_{\mathsf{Sol}} \subseteq A_{\mathsf{Bse}}$, and thus A' is indeed a function. For a constraint $(r, r', \updownarrow, \forall) \in C$, we have $A_{\mathsf{Sol}}(r) \cap A_{\mathsf{Sol}}(r') = \emptyset$, but since $A'(r) \subseteq A_{\mathsf{Sol}}(r)$ and $A(r') \subseteq A_{\mathsf{Sol}}(r')$, we

have $A'(r) \cap A'(r') = \emptyset$ as well. Thus, A' is valid for $(U, R, A_{\mathsf{Bse}}, C')$ and is clearly a function. □

Since, by Theorem 9, D-APEP$\langle \mathsf{SoD_U}, \mathsf{F} \rangle =_{fpt}$ WSP($\neq$), and since WSP($\neq$) is NP-hard and FPT parameterized by the number of steps [13], we obtain the following corollary:

Corollary 14. D-APEP$\langle \mathsf{SoD_U} \rangle$ *is NP-hard and FPT parameterized by* k.

In fact, it follows from [13] that D-APEP$\langle \mathsf{SoD_U} \rangle$ can be solved in time $O^*(2^k)$. We now consider $\mathsf{BoD_E}$ constraints.

Theorem 15. D-APEP$\langle \mathsf{BoD_E} \rangle$ *is polynomial-time solvable.*

Proof. We show that an instance $(U, R, A_{\mathsf{Bse}}, C)$ is satisfiable iff A_{Bse} is valid. Obviously, if A_{Bse} is valid, the D-APEP$\langle \mathsf{BoD_E} \rangle$ instance is satisfiable. Conversely observe first that A_{Bse} is obviously authorized and complete. Then, if D-APEP$\langle \mathsf{BoD_E} \rangle$ is satisfiable, there exists $A_{\mathsf{Sol}} \subseteq A_{\mathsf{Bse}}$, which is valid. However, since $A_{\mathsf{Sol}}(r) \subseteq A_{\mathsf{Bse}}(r)$ for any $r \in R$, any constraint $(r, r', \leftrightarrow, \exists)$ satisfied by A_{Sol} is also satisfied by A_{Bse}. In other words, A_{Bse} is eligible. □

6.2 Complexity of D-APEP⟨f(k)-bounded⟩

Let f be an arbitrary function in k and let $\mathcal{I} = (U, R, A_{\mathsf{Bse}}, C)$ be a D-APEP instance containing only $f(k)$-bounded constraints. Without loss of generality, we assume that $f(k) \geq k$ (observe that all constraints considered in this paper are never better than $(k-1)$-bounded). In this section, we introduce a method to decide whether $\mathcal{I}$ is satisfiable or not in time $\mathcal{O}^*\left(2^{2^k f(k) k}\right)$.

Given a set of resources $T \subseteq R$, we define

$$U_T = \{u \in U : \ A_{\mathsf{Bse}}(u) = T\}$$

We call U_T the (user) *family* associated with T. Note that for $T \neq T'$, we have $U_T \cap U_{T'} = \emptyset$. Moreover, $U = \bigcup_{T \subseteq R} U_T$. Thus $\{U_T\}_{T \subseteq R}$ is a partition of U containing at most 2^k sets. The intuition behind this definition is that when considering UI constraints (in particular, $f(k)$-bounded constraints), all users in a set U_T play the same role. The idea of the algorithm is thus to eliminate users in "large" families to upper-bound the number of users by a function of k. To eliminate users, we apply the following reduction rule:

> if there exists $T \subseteq R$ such that $|U_T| > f(k)$, then remove an arbitrary user u^* from U_T.

Successive applications of this rule will result in an instance in which the number of users is at most $f(k)2^k$, a function of k only.

Consider, for example, an instance comprising the base authorization relation A_{Bse} shown in Figure 2 ($k = 3$ and $n = 8$) and a single constraint $(r_1, r_2, \updownarrow, \exists)$. The constraint $(r_1, r_2, \updownarrow, \exists)$ is 3-bounded. There are three families of users, of which $U_{\{r_1, r_2\}}$ has cardinality greater than 3. Applying the reduction rule, we may remove users u_3 and u_4 from $U_{\{r_1, r_2\}}$.

	r_1	r_2	r_3
u_1, u_2	1		1
~~u_3~~, ~~u_4~~, u_5, u_6, u_7	1	1	
u_8	1	1	1

Fig. 2: Use of the reduction rule

Lemma 16. *$\mathcal{I}$ is satisfiable iff $\mathcal{I}'$ is satisfiable, where $\mathcal{I}'$ is the instance obtained by applying the reduction rule to $\mathcal{I}$.*

Proof. Obviously, if $\mathcal{I}'$ is satisfiable, then so is $\mathcal{I}$.

Assume $\mathcal{I}$ satisfiable and $\mathcal{I}'$ unsatisfiable, and let A be a solution for $\mathcal{I}$. Then there exists $T \subseteq R$ such that $|U_T| \geq f(k) + 1 \geq k + 1$, which means that A violates some constraint $c \in C$ (*i.e.* unsatisfiability of $\mathcal{I}'$ does not come from incompleteness). Since c is $f(k)$-bounded, we have $|\mathsf{core}(A : \mathcal{I})| \leq f(k)$ and thus $|\mathsf{core}(A : \mathcal{I}) \cap U_T| \leq f(k)$. But, since $|U_T| \geq f(k) + 1$, and since c is user independent, we may assume, without loss of generality, that there is a user $u^* \in U_T$ such that $u^* \notin \mathsf{core}(A : \mathcal{I})$. However, u^* is a user whose removal makes the instance unsatisfiable, a contradiction. □

Theorem 17. *For any computable function f depending only on k,* D-APEP$\langle$*f(k)-bounded*$\rangle$ *is FPT parameterized by k.*

Proof. Whenever the reduction rule can be applied, we remove one user from the instance. If the rule cannot be applied, then we have an instance with at most $2^k f(k)$ users. Applying a brute force algorithm (by checking every possible relation for the reduced user set), one can check the satisfiability in time $\mathcal{O}^*\left(2^{2^k f(k) k}\right)$. □

Corollary 18. D-APEP$\langle \mathsf{SoD}, \mathsf{BoD} \rangle$ *is FPT parameterized by k.*

Proof. The result follows from Propositions 4, 5, and Theorem 17. □

More generally, as we proved in Section 4.4 that cardinality constraints with symbol $\leq$ or $<$ are k-bounded, such constraints can be added to any D-APEP$\langle \mathsf{SoD}, \mathsf{BoD} \rangle$ instance without degrading the execution time. Concerning cardinality constraints with symbols $\geq$, $>$ or $=$, we have the following corollary of Proposition 8.

Corollary 19. *For any computable function f depending only on k,* D-APEP$\langle$*f(k)-bounded*$, \mathsf{G}_{card}, \mathsf{L}_{card} \rangle$ *is FPT parameterized by k plus the maximum cardinality of all cardinality constraints.*

6.3 Complexity of D-APEP⟨BoD_E, SoD_U⟩

We now prove that a better running time can be obtained when considering only $\mathsf{BoD_E}$ and $\mathsf{SoD_U}$ constraints.

Theorem 20. D-APEP$\langle \mathsf{BoD_E}, \mathsf{SoD_U} \rangle$ *can be solved in time* $\mathcal{O}\left(2^{k^2 \log k^2}\right)$.

Proof. We reduce to WSP($=, \neq$). Let $R = \{r_1, \ldots, r_k\}$. We build a WSP($=, \neq$) instance, denoted by (S', U', A', C'). We set $U' = U$. Then, for any $i \in [k]$, let

$$\Gamma(r_i) = \{j \in [k] : \ (r_i, r_j, \leftrightarrow, \exists) \in C\}.$$

For each resource $r_i \in R$, we introduce a set of steps S^i. If $\Gamma(r_i) = \emptyset$, then $S^i = \{s^i\}$. Otherwise, $S^i = \{s^i_j : j \in \Gamma(r_i)\}$. Then define $S' = \bigcup_{i \in [k]} S^i$. Observe that $|S'| \leq k(k-1)$. We then define the following constraints:

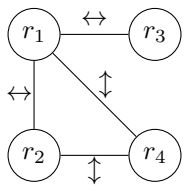
- For any $i \in [k]$, if $\Gamma(r_i) \neq \emptyset$, then for all $j \in \Gamma(r_i)$, we add the constraint $(s^i_j, s^j_i, =)$.

- For any $i, j \in [k]$, if $(r_i, r_j, \updownarrow, \forall) \in C$, then, for every $s \in S^i$ and every $s' \in S^j$, we add the constraint $(s, s', \neq)$.

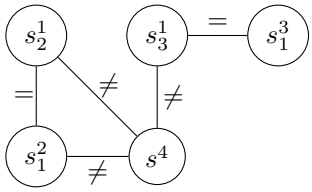
Finally, we define the authorization policy A'. For every $i \in [k]$ and every $u \in U$:

- If $\Gamma(r_i) = \emptyset$, then $(u, s^i) \in A'$ iff $(u, r_i) \in A_{\mathsf{Bse}}$.
- If $\Gamma(r_i) \neq \emptyset$, then $\forall j \in \Gamma(r_i)$, $(u, s^i_j) \in A'$ iff $(u, r_i), (u, r_j) \in A_{\mathsf{Bse}}$.

The construction is illustrated in Figure 3 for a small example. Clearly this construction can be carried out in polynomial time.



(a) C



(b) C'

A_{Bse}	r_1	r_2	r_3	r_4
u_1	1	1	1	
u_2			1	1
u_3	1		1	
u_4	1	1		
u_5		1		1

(c) A_{Bse}

A'	s^1_2	s^1_3	s^2_1	s^3_1	s^4
u_1	1	1	1	1	
u_2					1
u_3		1		1	
u_4	1		1		
u_5					1

(d) A'

Fig. 3: Reducing D-APEP$\langle \mathsf{BoD_E}, \mathsf{SoD_U} \rangle$ to WSP$(=, \neq)$

Let us suppose there is a valid plan $\pi : S' \to U'$ for (S', U', A', C'). We set $A_\pi = \{(u, r_i) : i \in [k], \pi(s) = u$ for some $s \in S^i\}$. One can observe that A_π is authorized and complete. Then, for any $i, j \in [k]$ such that $(r_i, r_j, \leftrightarrow, \exists) \in C$, we must have $\pi(s^i_j) = \pi(s^j_i) = u$ for some $u \in U$, which implies that $u \in A_\pi(r_i) \cap A_\pi(r_j)$. Then, if $(r_i, r_j, \updownarrow, \forall) \in C$, we know that $\pi(s) \neq \pi(s')$ for any $s \in S^i$ and any $s' \in S^j$, which implies that $A_\pi(r_i) \cap A_\pi(r_j) = \emptyset$. This proves that A_π is valid.

Conversely, we suppose $(U, R, A_{\mathsf{Bse}}, C)$ is satisfiable, and let A_{Sol} be a valid solution. For any $i \in [k]$, we have the following:

- If $\Gamma(r_i) = \emptyset$, then define $\pi(s^i)$ to be an arbitrary user in $A_{\mathsf{Sol}}(r_i)$.
- If $\Gamma(r_i) \neq \emptyset$, then, for every $j \in \Gamma(r_i)$, define $\pi(s^i_j)$ as an arbitrary user in $A_{\mathsf{Sol}}(r_i) \cap A_{\mathsf{Sol}}(r_j)$, and set also $\pi(s^i_j) = \pi(s^j_i)$.

One can observe that π is authorized and complete. By construction, every constraint $\left(s^i_j, s^j_i, =\right) \in C'$ is satisfied. Finally, for every $s, s' \in S'$ such that $(s, s', \neq)$, it must be the case that $s \in S^i$ and $s' \in S^j$ such that $(r_i, r_j, \updownarrow, \forall) \in C$, which implies that $A_{\mathsf{Sol}}(r_i) \cap A_{\mathsf{Sol}}(r_j) = \emptyset$. Hence we must have $\pi(s) \neq \pi(s')$, and π is a valid plan. □

6.4 Complexity of MAXAPEP$\langle \mathsf{SoD_U} \rangle$

We now introduce a particular version of O-APEP, which seeks to find a valid authorization relation of maximum cardinality. We write M_{Sol} to denote the cardinality of such a relation. Such a relation is, in some sense, a most resilient authorization relation possible, given the authorization constraints. We call this problem MAXAPEP. (We may also define a decision version APEP to find resilient authorization relations. We may, for example, introduce a global constraint $(\geqslant, t)$, which requires that at least t users are authorized for each resource. These types of problems are related to notions of resiliency in workflow systems [26].)

In this section, (A_{Bse}, C) is an APEP$\langle \mathsf{SoD_U} \rangle$ instance. In Theorem 13, we established that D-APEP$\langle \mathsf{SoD_U} \rangle$ could be reduced to D-APEP$\langle \mathsf{SoD_U}, \mathsf{F} \rangle$. Let Π denote the set of valid solutions to instance $(A_{\mathsf{Bse}}, C \cup \{(=, 1)\})$ (that is, functions $\pi : R \to U$). Given a function $\pi \in \Pi$, we say $A \subseteq U \times R$ *contains* π if and only if for every $r \in R, (\pi(r), r) \in A$. Let M_π denote the maximum size of a valid authorization relations containing π. Theorem 13 established that any solution A_{Sol} of APEP$\langle \mathsf{SoD_U} \rangle$ contains at least one function $\pi \in \Pi$. We write M_{Sol} to denote $\max\{M_\pi : \pi \in \Pi\}$.

6.4.1 Patterns

A function $\pi : R \to U$ defines an equivalence relation $\sim_\pi$ on R, where $r \sim_\pi r'$ iff $\pi(r) = \pi(r')$. The equivalence classes defined by this relation form a partition of R which we call the *pattern* associated with π and denote it by $P(\pi)$. We say two functions π and π' are *equivalent* if $P(\pi) = P(\pi')$. For UI constraints and any two functions π and π' such that $P(\pi) = P(\pi')$, π is eligible iff π' is eligible. Hence, we will say P is *eligible* if and only if, there exists π such that $P = P(\pi)$ and π is eligible for C. Henceforth, we only consider eligible patterns. We write M_P to denote $\max\{M_\pi : P(\pi) = P\}$. There exists an eligible pattern P such that $M_{\mathsf{Sol}} = M_P$.

Let us suppose that we are able, given a pattern P, to construct a valid A, such that $|A| = M_P$, in FPT time $f(k)n^{O(1)}$. There are at most $\mathcal{B}_k$ eligible patterns, where $\mathcal{B}_k$ is the Bell number and $\mathcal{B}_k = O(2^{k \log k})$ [2].[4] Then, MAXAPEP$\langle \mathsf{SoD_U} \rangle$ would be FPT: exploring all the eligible patterns and applying the FPT algorithm to compute M_P for each P is executed in time $O^*(2^{k \log k} f(k))$. As a consequence, our objective now is to design a FPT algorithm to compute A_P such that $|A_P| = M_P$.

6.4.2 Exploring patterns to solve MAXAPEP$\langle \mathsf{SoD_U} \rangle$

Lemma 21. *Let $P = \{T_1, T_2, \ldots, T_d\}$ be a pattern. An authorization relation A_P, such that $|A_P| = M_P$, can be computed in FPT time $O^*(2^k)$.*

Proof. Clearly $d \leq n$. We extend P into P^* in order to have $|P^*| = n$. We set $P^* = \{T_1, \ldots, T_d, \emptyset_1, \ldots, \emptyset_{n-d}\} = \{T_1, \ldots, T_n\}$. We build a weighted bipartite graph $G_P = (P^* \cup U, E, \omega)$, where $(T_i, u) \in E$ if and only if $T_i \subseteq A_{\mathsf{Bse}}(u)$ and for any $i \in [n-d]$ and $u \in U$, $(\emptyset_i, u) \in E$. Assign to $(T_i, u) \in E$ weight $\omega(T_i, u)$ which is the cardinality of the maximum independent set in $A_{\mathsf{Bse}}(u)$ containing T_i:

$$\omega(T_i, u) = \max_{\substack{\forall (r,r') \in X^2,\ (r,r',\updownarrow,\forall) \notin C \\ T_i \subseteq X \subseteq A_{\mathsf{Bse}}(u)}} |X|$$

There are at most n^2 weights to compute. For any $e \in E$, calculation of every $\omega(e)$ can be performed in time $O(2^k)$ by

4. All logarithms in this paper are of base 2.

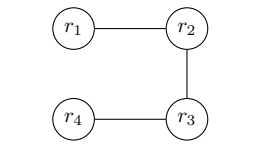
enumerating all subsets of $A_{\mathsf{Bse}}(u)$. Thus the bipartite graph G_P can be built in time $O^*(2^k)$. We solve the ASSIGNMENT PROBLEM on G_P and obtain a maximum weighted matching (MWM), $\mathcal{M}$, in polynomial time using the Hungarian algorithm [17].

For every edge $e \in \mathcal{M}$ we compute an independent set X_e as follows. For each $(T_i, u) \in \mathcal{M}$, choose a maximum independent set $X_{(T_i,u)}$ such that $T_i \subseteq X_{(T_i,u)} \subseteq A_{\mathsf{Bse}}(u)$ and, therefore, $\left|X_{(T_i,u)}\right| = \omega(T_i, u)$. We define the authorization relation $A_{\mathcal{M}}$ such that $A_{\mathcal{M}}(u) = X_{(T_i,u)}$.

For any $u \in U$, $A_{\mathcal{M}}(u) \subseteq A_{\mathsf{Bse}}(u)$. Furthermore, for any $u \in U$, $A_{\mathcal{M}}(u)$ contains resources which are pairwise independent, so $A_{\mathcal{M}}$ is valid. We define the function $\tilde{\pi}$ such that $\tilde{\pi}(r) = u$ if and only if $r \in T_i$ and $(T_i, u) \in \mathcal{M}$. $A_{\mathcal{M}}$ contains $\tilde{\pi}$ whose pattern is P. Thus, $|A_{\mathcal{M}}| \leq M_P$.

We know that there exists a valid function π such that $M_P = M_\pi$ and $P(\pi) = P = P(\tilde{\pi})$. There exists a matching $\mathcal{M}'$ representing π in G_P. If $\pi(T_i) = u$, then $(T_i, u) \in \mathcal{M}'$. If $\pi^{-1}(u)$ is empty, we associate u with an arbitrary vertex $\emptyset_i$. Since $|A_{\mathcal{M}}|$ is equal to the weight of the MWM $\mathcal{M}$ of G_P, we have $M_\pi \leq |A_{\mathcal{M}}|$. Hence, $M_P = |A_{\mathcal{M}}|$. □

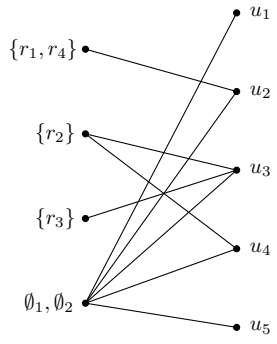
In Figure 4, we use a simple example to illustrate the matching process described in the proof of Lemma 21. We consider the pattern $P = \{\{r_1, r_4\}, \{r_2\}, \{r_3\}\}$ and merge $\emptyset_1$ and $\emptyset_2$ into a single node to keep the bipartite graph readable. The figure shows G_P (derived from A_{Bse}) and the resulting MWM (where the matching is indicated by the thick lines). Then, for example, $\omega(\{r_3\}, u_3) = 2$ because $X = \{r_1, r_3\}$ is the largest independent subset of $A_{\mathsf{Bse}}(u_3)$ containing r_3.



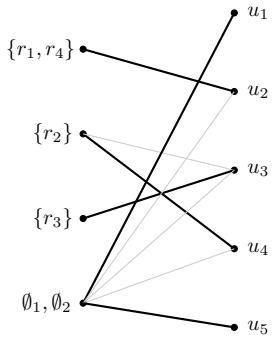
(a) Constraints

	r_1	r_2	r_3	r_4
u_1				1
u_2	1			1
u_3	1	1	1	
u_4	1	1		
u_5				1

(b) Base relation A_{Bse}



(c) G_P



(d) MWM

Fig. 4: Computing a maximum weighted matching for an instance of MAXAPEP$\langle \mathsf{SoD_U} \rangle$

Theorem 22. MAXAPEP$\langle \mathsf{SoD_U} \rangle$ *(and thus* D-APEP$\langle \mathsf{SoD_U} \rangle$*) can be solved in FPT time* $O^*(2^{k+k\log k})$.

Proof. We explore all eligible patterns P. For each one, we construct A_P using Lemma 21, and keep the largest one. The time complexity of this algorithm is $O^*\left(2^{k+k\log k}\right)$. Hence, it is FPT parameterized by k. □

6.5 Complexity of D-APEP$\langle \mathsf{SoD_E} \rangle$

In this section, we solve the MAXAPEP$\langle \mathsf{SoD_E} \rangle$ problem by reducing to the MAX WEIGHTED PARTITION problem [5]: that is, given a ground set K and p functions $f_1, \ldots, f_p$ from 2^K to integers from the range $[-M, M]$, $M \geq 1$, find a partition $\{K_1, \ldots, K_p\}$ of K that maximizes $\sum_{i=1}^{p} f_i(K_i)$. The following result is a corollary of the main theorem on MAX WEIGHTED PARTITION in [5].

Lemma 23. MAX WEIGHTED PARTITION *can be solved in time* $O^*(2^k p^2 M)$.

Theorem 24. D-APEP$\langle \mathsf{SoD_E} \rangle$ *and* MAXAPEP$\langle \mathsf{SoD_E} \rangle$ *can be solved in time* $\mathcal{O}^*(2^k)$.

Proof. We reduce to the MAX WEIGHTED PARTITION problem. The ground set is R, and we construct weight functions indexed by sets, which are elements of a family χ of subsets of U. We say that two resources $r, r' \in R$ are *independent* if and only if $(r, r', \updownarrow, \exists) \notin C$. Moreover, the *degree* $d(r)$ of a resource $r \in R$ is the number of resources r' such that $(r, r', \updownarrow, \exists) \in C$. The index set χ is defined as $\bigcup_{r \in R} \chi_r$, where χ_r consists of all subsets of $A_{\mathsf{Bse}}(r)$ if $|A_{\mathsf{Bse}}(r)| \leq \log k$, or $d(r) + 1$ largest subsets of $A_{\mathsf{Bse}}(r)$ if $|A_{\mathsf{Bse}}| > \log k$ (breaking ties arbitrarily). For every $X \in \chi$, we define a weight function $f_X : 2^R \to [-|A_{\mathsf{Bse}}| - 1, |A_{\mathsf{Bse}}|]$ as follows: for every $T \subseteq R$, $f_X(T)$ is set to $|T||X|$ if T is an independent set and $X \subseteq A_{\mathsf{Bse}}(r)$ for every $r \in T$, and $f_X(T) = -|A_{\mathsf{Bse}}| - 1$ otherwise. Now, we show that any valid authorization relation A_{Sol} of D-APEP$\langle \mathsf{SoD_E} \rangle$ corresponds to a partition of R of cost $|A_{\mathsf{Sol}}|$, and vice versa.

Let $\mathcal{T} = \{T_X : X \in \chi\}$ be a partition of R of nonnegative weight. Note that if $f_X(T_X) \geq 0$ for every $X \in \chi$ then $\sum_{X \in \chi} f_X(T_X) \leq |A_{\mathsf{Bse}}|$. Thus, since $\mathcal{T}$ is of nonnegative weight, $f_X(T_X) \geq 0$ for every $X \in \chi$. Construct an authorization relation A_{Sol} such that for any $X \in \chi$, for any $r \in T_X$, we have $A_{\mathsf{Sol}}(r) = X$. Obviously, since $\mathcal{T}$ is a partition of R, A_{Sol} is complete. Then, by definition of χ, we have $A_{\mathsf{Sol}}(r) \subseteq A_{\mathsf{Bse}}(r)$ and thus A_{Sol} is authorized. Finally, for any $r, r' \in R$ such that $A_{\mathsf{Sol}}(r) = A_{\mathsf{Sol}}(r')$, it must hold that $r, r' \in T_{A_{\mathsf{Sol}}(r)}$, and since $T_{A_{\mathsf{Sol}}(r)}$ is independent, $(r, r', \updownarrow, \exists) \notin C$, and A_{Sol} is eligible. In other words, A_{Sol} is a valid authorization relation, and its weight is $\sum_{X \in \chi} f_X(T_X)$.

For any valid authorization relation A_{Sol}, let $P(A_{\mathsf{Sol}})$ be the partition of R into equivalence classes with respect to the following equivalence relation: $r, r' \in R$ are equivalent if and only if $A_{\mathsf{Sol}}(r) = A_{\mathsf{Sol}}(r')$. We now prove that there always exists a valid authorization relation A'_{Sol} of size at least $|A_{\mathsf{Sol}}|$ such that $A'_{\mathsf{Sol}}(r) \in \chi$ for every $r \in R$. If this is true, then it will mean that we may assume that $P(A_{\mathsf{Sol}}) = \{T_X : X \in \chi\}$, and since A_{Sol} is valid, $\sum_{X \in \chi} f_X(T_X) = |A_{\mathsf{Sol}}|$. For $r \in R$, if $|A_{\mathsf{Bse}}(r)| \leq \log k$, then since χ contains all subsets of $A_{\mathsf{Bse}}(r)$, it holds that $A_{\mathsf{Sol}}(r) \in \chi$. If $|A_{\mathsf{Bse}}(r)| > \log k$ and $A_{\mathsf{Sol}}(r) \notin \chi_r$, recall that χ_r consists of $d(r)+1$ largest subsets of $A_{\mathsf{Bse}}(r)$. Hence, there must exist $X \in \chi_r$ such that $A_{\mathsf{Sol}}(r') \neq X$ for every

r' such that $(r, r', \updownarrow, \exists) \in C$. Hence, replacing $A_{\mathsf{Sol}}(r)$ by X creates another valid authorization relation A'_{Sol} of size at least $|A_{\mathsf{Sol}}|$ and such that $A'_{\mathsf{Sol}}(r) \in \chi_r$. Repeating this modification for every $r \in R$ such that $A_{\mathsf{Sol}}(r) \notin \chi$, we end up with a valid authorization having the desired property.

Using the reduction above together with Lemma 23, we prove the claimed statement. □

7 Conclusion

In this paper we have introduced a general framework within which we can specify problems concerned with finding authorization relations ("policies") that must satisfy certain kinds of constraints. We have shown that there exist FPT algorithms to solve the authorization policy existence problem when all constraints are user-independent and are bounded in an appropriate way. We have also shown that many constraints of practical interest are indeed user-independent and bounded.

Our prior work on implementing FPT algorithms for the workflow satisfiability problem [8] suggests that the theoretical results obtained in this paper should translate to relatively efficient algorithms in practice; certainly far more efficient than a brute-force approach to solving APEP. We hope to confirm this expectation by developing practical implementations of the algorithms described in this paper.

Although we have chosen to consider user-independent constraints, not least because such constraints have been studied extensively in the literature on workflow satisfiability, we could equally well consider resource-independent constraints, because our framework is symmetric in a way that workflow satisfiability questions are not. So, for example, we could define a constraint of the form $(u, u', \updownarrow, \forall)$ which would be satisfied provided the set of resources assigned to u is distinct from the set of resources assigned to u'. In this way, we search for authorization relations that guarantee certain users do not have access to the same resources. Moreover, if the number of users is small relative to the number of resources, which may well be the case in some multi-user systems (such as file systems), then n will be the small parameter and the symmetry of our framework admits FPT algorithms for solving problem instances of this form.

We believe there are many opportunities for future work, not least exploring what types of authorization constraints might be useful in practice and determining whether those constraints are user-independent and bounded.

References

[1] Basin, D. A., Burri, S. J., and Karjoth, G. Obstruction-free authorization enforcement: Aligning security and business objectives. *Journal of Computer Security 22*, 5 (2014), 661–698.

[2] Berend, D., and Tassa, T. Improved bounds on Bell numbers and on moments of sums of random variables. *Probability and Math. Statistics 30*, 2 (2010), 185–205.

[3] Bergé, P., Crampton, J., Gutin, G. Z., and Watrigant, R. The authorization policy existence problem. In *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy, CODASPY 2017, Scottsdale, AZ, USA, March 22-24, 2017* (2017), G. Ahn, A. Pretschner, and G. Ghinita, Eds., ACM, pp. 163–165.

[4] Bertino, E., Ferrari, E., and Atluri, V. The specification and enforcement of authorization constraints in workflow management systems. *ACM Trans. Inf. Syst. Secur. 2*, 1 (1999), 65–104.

[5] Björklund, A., Husfeldt, T., and Koivisto, M. Set partitioning via inclusion-exclusion. *SIAM J. Comput. 39*, 2 (2009), 546–563.

[6] Brewer, D. F. C., and Nash, M. J. The Chinese wall security policy. In *Proceedings of the 1989 IEEE Symposium on Security and Privacy* (1989), pp. 206–214.

[7] Cohen, D., Crampton, J., Gagarin, A., Gutin, G., and Jones, M. Iterative plan construction for the workflow satisfiability problem. *J. Artif. Intell. Res. (JAIR) 51* (2014), 555–577.

[8] Cohen, D. A., Crampton, J., Gagarin, A., Gutin, G. Z., and Jones, M. Algorithms for the workflow satisfiability problem engineered for counting constraints. *J. Comb. Optim. 32*, 1 (2016), 3–24.

[9] Crampton, J. A reference monitor for workflow systems with constrained task execution. In *SACMAT* (2005), E. Ferrari and G.-J. Ahn, Eds., ACM, pp. 38–47.

[10] Crampton, J., Gagarin, A. V., Gutin, G., and Jones, M. On the workflow satisfiability problem with class-independent constraints. In *10th International Symposium on Parameterized and Exact Computation, IPEC 2015, September 16-18, 2015, Patras, Greece* (2015), T. Husfeldt and I. A. Kanj, Eds., vol. 43 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, pp. 66–77.

[11] Crampton, J., Gutin, G., and Karapetyan, D. Valued workflow satisfiability problem. In *Proceedings of the 20th ACM Symposium on Access Control Models and Technologies* (2015), pp. 3–13.

[12] Crampton, J., Gutin, G., and Watrigant, R. Resiliency policies in access control revisited. In *Proceedings of the 21st ACM on Symposium on Access Control Models and Technologies* (2016), ACM, pp. 101–111.

[13] Crampton, J., Gutin, G., and Yeo, A. On the parameterized complexity and kernelization of the workflow satisfiability problem. *ACM Trans. Inf. Syst. Secur. 16*, 1 (2013), 4.

[14] Downey, R. G., and Fellows, M. R. *Fundamentals of Parameterized Complexity*. Springer Verlag, 2013.

[15] Gligor, V. D., Gavrila, S. I., and Ferraiolo, D. F. On the formal definition of separation-of-duty policies and their composition. In *Security and Privacy - 1998 IEEE Symposium on Security and Privacy, Proceedings* (1998), IEEE Computer Society, pp. 172–183.

[16] Karapetyan, D., Gagarin, A., and Gutin, G. Pattern backtracking algorithm for the workflow satisfiability problem with user-independent constraints. In *Frontiers in Algorithmics - 9th International Workshop, Proceedings* (2015), pp. 138–149.

[17] Kuhn, H. W. The Hungarian method for the assignment problem. In *50 Years of Integer Programming 1958-2008 - From the Early Years to the State-of-the-Art*. 2010, pp. 29–47.

[18] Li, N., Tripunitara, M. V., and Bizri, Z. On mutually exclusive roles and separation-of-duty. *ACM Trans. Inf. Syst. Secur. 10*, 2 (2007).

[19] Li, N., Wang, Q., and Tripunitara, M. V. Resiliency policies in access control. *ACM Trans. Inf. Syst. Secur. 12*, 4 (2009).

[20] Mace, J. C., Morisset, C., and van Moorsel, A. P. A. Quantitative workflow resiliency. In *Computer Security - ESORICS 2014 - 19th European Symposium on Research in Computer Security, Wroclaw, Poland, September 7-11, 2014. Proceedings, Part I* (2014), M. Kutylowski and J. Vaidya, Eds., vol. 8712 of *Lecture Notes in Computer Science*, Springer, pp. 344–361.

[21] Niedermeier, R. *Invitation to fixed-parameter algorithms*. Oxford University Press, 2006.

[22] Roy, A., Sural, S., Majumdar, A. K., Vaidya, J., and Atluri, V. Minimizing organizational user requirement while meeting security constraints. *ACM Trans. Management Inf. Syst. 6*, 3 (2015), 12.

[23] Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman, C. E. Role-based access control models. *IEEE Computer 29*, 2 (1996), 38–47.

[24] Schaad, A., Lotz, V., and Sohr, K. A model-checking approach to analysing organisational controls in a loan origination process. In *SACMAT 2006,11th ACM Symposium on Access Control Models and Technologies, Lake Tahoe, California, USA, June 7-9, 2006, Proceedings* (2006), D. F. Ferraiolo and I. Ray, Eds., ACM, pp. 139–149.

[25] Simon, R. T., and Zurko, M. E. Separation of duty in role-based environments. In *10th Computer Security Foundations Workshop (CSFW '97), June 10-12, 1997, Rockport, Massachusetts, USA* (1997), IEEE Computer Society, pp. 183–194.

[26] Wang, Q., and Li, N. Satisfiability and resiliency in workflow authorization systems. *ACM Trans. Inf. Syst. Secur. 13*, 4 (2010), 40.

APPENDIX

Proof of Proposition 4. Let c be one of $(r', r'', \to, \forall)$, $(r', r'', \leftrightarrow, \forall)$ or $(r', r'', \leftrightarrow, \exists)$. Let V be a set of $k-1$ distinct users and consider A, where $A(r') = A(r'') = \{u\}$ for some $u \in V$ and, for any $r_1, r_2 \in R \setminus \{r', r''\}$, $|A(r_1)| = 1$, $A(r_1) \neq A(r')$, and $A(r_1) \neq A(r_2)$. Then $\mathsf{core}(A : U \times R, c) = V$ and $|\mathsf{core}(A : U \times R, c)| = k-1$. Moreover, for any relation A' valid with respect to $U \times R$ and c, any subset of $A'(R)$ of size at least k must contain two users who are both assigned to the same resource; thus one of them can be removed without affecting completeness or satisfiability. Hence, the constraint is $(k-1)$-bounded. □

Proof of Proposition 5. Let V be a set of k distinct users and consider A where $|A(r)| = 1$ for each $r \in R$ and $A(R) = V$. Then $\mathsf{core}(A : U \times R, c) = V$ and $|\mathsf{core}(A : U \times R, c)| = k$. Now, for any for any relation A' valid with respect to $U \times R$ and c, any subset of $A'(R)$ of size at least $k+1$ must contain two users who are both assigned to the same resource, and thus one of them can be removed without violating completeness or satisfiability. Thus $|\mathsf{core}(A' : U \times R, c)| \leqslant |\mathsf{core}(A : U \times R, c)|$, from which the result follows. □

Proof of Proposition 6. Given any valid solution A, the removal of any user cannot make A non-eligible with respect to $c = (R', \leq, t)$, but may violate completeness. Hence, $\mathsf{core}(A : U \times R, c)$ is largest when $|A(r)| = 1$ for all r and $A(R) = k$, in which case we have $|\mathsf{core}(A : U \times R, c)| = k$. (The global cardinality constraint $c = (\leq, t)$ is k-bounded because, in this case too, no removal affects the eligibility of the relation; it can only affect the completeness.) □

Proof of Proposition 8. We only give the proof for $(R', =, t)$, the other one being similar. One can observe that $\mathsf{core}(A : U \times R, c)$ is largest when $|A(r)| = 1$ for any $r \in R \setminus R'$, $|A(R \setminus R')| = |R \setminus R'|$, and $A(R') \cap A(R \setminus R') = \emptyset$. In this case we have $|\mathsf{core}(A : U \times R, c)| \leq |R \setminus R'| + t \leq 2 \max\{k, t\}$.

Concerning the negative result, observe that if $\max\{k, t\} = t$, then no user of $A(R')$ can be removed from any valid solution, and if $\max\{k, t\} = k$, then there exists solutions in which $A(R) \geq k$ and the removal of any user from $A(R)$ either violates a constraint or breaks completeness. □

Pierre Bergé is a PhD student of computer science at LRI, Université Paris-Sud, Université Paris-Saclay, France. He is a graduate of Centrale-Supélec, France. His research interests are combinatorial optimization, graph theory and complexity.

Jason Crampton is a Professor in Information Security at Royal Holloway, University of London, UK. His research focuses on access control, particularly languages for specifying access control policies and cryptographic enforcement of policies.

Gregory Gutin received PhD in mathematics from Tel Aviv University, Israel in 1993. Since 2000 he has been a professor of computer science at Royal Holloway, University of London, UK. His research interests include access control, combinatorial optimization, graph theory and applications, and parameterized algorithms and complexity.

Rémi Watrigant is an assistant professor at Université Claude Bernard Lyon 1, doing research in the LIP laboratory of ENS Lyon. His research interests are algorithmics, mainly for graph optimization problems.