Towards Better Understanding of User Authorization Query Problem via Multi-variable Complexity Analysis

JASON CRAMPTON, Royal Holloway, University of London, United Kingdom
GREGORY Z. GUTIN, Royal Holloway, University of London, United Kingrom
DIPTARPIYO MAJUMDAR, Royal Holloway, University of London, United Kingrom

User authorization queries in the context of role-based access control have attracted considerable interest in the last 15 years. Such queries are used to determine whether it is possible to allocate a set of roles to a user that enables the user to complete a task, in the sense that all the permissions required to complete the task are assigned to the roles in that set. Answering such a query, in general, must take into account a number of factors, including, but not limited to, the roles to which the user is assigned and constraints on the sets of roles that can be activated. Answering such a query is known to be NP-hard. The presence of multiple parameters and the need to find efficient and exact solutions to the problem suggest that a multi-variate approach will enable us to better understand the complexity of the user authorization query problem (UAQ).

In this paper, we establish a number of complexity results for UAQ. Specifically, we show the problem remains hard even when quite restrictive conditions are imposed on the structure of the problem. Our FPT results show that we have to use either a parameter with potentially quite large values or quite a restricted version of UAQ. Moreover, our second FPT algorithm is complex and requires sophisticated, state-of-the-art techniques. In short, our results show that it is unlikely that all variants of UAQ that arise in practice can be solved reasonably quickly in general.

CCS Concepts: • **Security and Privacy** → **Security Services**; *Access Control*; Authorization.

Additional Key Words and Phrases: User Authorization Query, Parameterized Complexity, W-hardness, Representative Families

1 INTRODUCTION

In Role-Based Access Control (RBAC), permissions are not assigned to users directly. A user is assigned roles and roles are assigned permissions. Thus, a user u is authorized for those permissions that are assigned to at least one role to which u is assigned.

A user interacts with an RBAC system by activating some subset of the roles to which she is assigned. In certain situations, it is useful to be able to identify the particular subset of roles a user needs to activate in order to complete a task that requires specific permissions. A user authorization query seeks to find a set of roles that is suitable for a given set of permissions. A substantial body of work has established that user authorization queries are, in general, hard to solve [4, 11].

Generally speaking, it is important from the end-user perspective to answer access control queries as quickly as possible. Moreover, it is important that the answers to those queries are correct, otherwise a user may not be able to complete a task (because too few permissions are assigned to the solution's role set) or security breaches may occur (in the case of too many permissions). Thus, it is desirable to find algorithms to solve user authorization queries that are exact and as fast as possible. Existing NP-hardness results [4, 11] show that such algorithms are unlikely to exist for all user authorization queries. However, such queries have several numerical parameters, some of which may be small in all instances of practical interest. Hence, it is worth exploring such queries from the perspective of fixed-parameter tractability (a short introduction to parameterized algorithms and complexity is given in Section 2).

As for other practical problems, SAT and other solvers for User Authorization Query (UAQ) have been tested and compared with other solvers using benchmarks. Recently, Armando et al. [1] suggested a methodology to evaluate existing benchmarks for UAQ and to guide the design of new ones. The methodology is based on the use of fixed-parameter tractable (FPT) algorithms, i.e., algorithms which run in time $f(k)N^c$, where N is the size of the problem instance, k is a parameter or sum of several parameters, f is a function depending only on k, and c is a constant. Note that when k is fixed, then the algorithm's running time becomes polynomial of constant degree. Often c is small, but for an FPT algorithm to be practical, k should be relatively small and f relatively slow growing. Armando et al. observed that "the proposed methodology will yield different, improved results as soon as new complexity results will become available. Thus the benchmarks proposed and used in this paper could be improved consequently." In this paper, we prove some parameterized tractability results for restricted versions of UAQ that can be applied in that methodology.

On the other hand, we show that queries for some quite constrained RBAC configurations remain hard, even from the parameterized complexity point of view, i.e., it is highly unlikely that there are FPT algorithms for them. Our parameterized intractability results indicate that to obtain FPT algorithms for UAQ with small parameters we have to consider quite restricted versions of UAQ.

Two well-known access control problems – UAQ and the Workflow Satisfiability Problem (WSP) – are "equally" intractable from the classical complexity point of view (i.e., NP-complete). However, from a more fine-grained parameterized complexity point of view, the time complexities of the two problems, even in their most basic forms, are quite different. While Wang and Li [23] and subsequent research (e.g., [5, 7, 13]) showed that basic and more advanced versions of WSP are FPT, our hardness results in this paper clearly demonstrate that several parameterized basic versions of UAQ are still intractable. Moreover, we show in Section 4 that even a quite restrictive version of UAQ can require advanced algorithmic methods to design an FPT algorithm, which was not the case with WSP. Thus, it is likely that the search for practical algorithms to solve UAQ will be less successful than it was for WSP (see, e.g., [3, 6, 15]).

1.1 User Authorization Query Problem and Its Reduction

An RBAC *configuration* has the form $\rho = ((R, \leq), P, RP)$, where $(R, \leq)$ is a *role hierarchy* modeled as a partially ordered set of roles, and $RP \subseteq R \times P$ is an assignment of roles to permissions. A role $r \in R$ is *authorized* for the set of permissions $P(r) = \{p \in P \mid (p, r') \in RP, r' \leq r\}$. Then a user u, authorized for a set of roles $R' \subseteq R$, is authorized for the set of permissions $P(R') = \bigcup_{r \in R'} P(r)$. The *role-permission* graph (RPG) is the bipartite graph $(R \cup P, RP)$ with partite sets R and P and edge set RP. (It is assumed that R and P are disjoint.) A (dynamic) *separation of duty constraint* has the form $\langle X, t \rangle$, where $1 \leq t \leq |X|$ for $X \subseteq R$. The semantics of an SoD constraint is that no user can activate t or more roles from X (thus restricting the sets of roles that may provide a solution to the user authorization query). We can now formulate a decision version of the problem studied in this paper.

USER AUTHORIZATION QUERY

Input: An RBAC policy $((R, \leq), P, RP)$, $P_{lb}, P_{ub} \subseteq P$, a set of constraints $\mathcal{D}$, and integers k_r, k_p.

Question: Does there exist a *solution*, which is set of roles $R_s \subseteq R$ such that $|R_s| \leqslant k_r$, R_s satisfies all constraints in $\mathcal{D}$, $P_{lb} \subseteq P(R_s) \subseteq P_{ub}$ and $|P(R_s) \setminus P_{lb}| \leqslant k_p$? If the answer is yes, find a solution.

Note that there are search versions of the problem [1, 24]. Wickramaarachchi et al. [24], for example, do not specify k_p and k_r. Instead, a problem instance includes an objective, which takes one of two values – *min* or *max* – and specifies the required nature of a solution set. Specifically, given $((R, \leq), P, RP)$, P_{lb}, P_{ub}, $\mathcal{D}$ and *min* (respectively, *max*), find $R' \subseteq R$ such that

(1) $P_{lb} \subseteq P(r) \subseteq P_{ub}$,
(2) R' satisfies all constraints in $\mathcal{D}$, and
(3) for any R'' that satisfies conditions 1 and 2, we have $|R'| \leq |R''|$ (respectively, $|R'| \geq |R''|$).

We say that USER AUTHORIZATION QUERY (UAQ) is *non-hierarchical* if no partial order is defined on the set of roles. Moffet [19] has argued that role hierarchies are not necessarily appropriate structures for access control (see also [20]). Moreover, it is always possible to eliminate a role hierarchy by assigning the (inherited) permissions of junior roles explicitly to more senior roles. Hereafter, we will consider only non-hierarchical UAQs.

We will use the following reduction from the UAQ instance to an equivalent instance, i.e., both instances are either yes-instances or no-instances.

REDUCTION RULE 0. *(i) delete any role r such that $P(r) \cap P_{lb} = \emptyset$ (since r cannot contribute to a set of roles that satisfies the instance);*
(ii) delete any role r such that $P(r) \setminus P_{ub} \neq \emptyset$ (since r contributes permissions that are not allowed);
(iii) for any deleted role r and any SoD constraint $\langle X, t \rangle$ with $r \in X$, replace the constraint by $\langle X \setminus \{r\}, t \rangle$;
(iv) for any deleted role r, remove all pairs of the form (r, p) from RP;
(v) delete any constraint $\langle X, t \rangle$ in which $|X| < t$.

One consequence of applying the above reduction rule is that we may assume without loss of generality that $P_{ub} = P$. Thus, in this paper a UAQ instance will be written as a tuple $(R, P, RP, P_{lb}, \mathcal{D}, k_r, k_p)$.

1.2 Our results

In Section 3, we present parameterized intractability results for UAQ when there are no SoD constraints. These results are based on a hierarchy of parameterized intractability classes, which we describe in more detail in Section 2. Informally, parameterized problems which admit FPT algorithms form the tractable class FPT. There is an infinite number of parameterized intractability classes W[i], $i \geq 1$, such that FPT$\subseteq$W[1]$\subseteq$W[2]$\subseteq \ldots$. A problem belongs to W[i] if it can be reduced to one of the hardest problems in W[i]. It is widely believed that FPT$\neq$W[1] (and hence FPT$\neq$W[i] for any i). In particular, a parameterized problem proved to be W[i]-hard is highly unlikely to admit an FPT algorithm.

We prove that if $P_{\ell b} = P$ then UAQ parameterized by k_r is W[2]-hard and if $k_r = |R|$ (i.e., there is no restriction on the size of UAQ solution) then UAQ parameterized by k_p is W[2]-hard. Note that these results strengthen the classical NP-hardness results of Du and Joshi [11] (for $\mathcal{D} = \emptyset$ and $P_{\ell b} = P$) and of Chen and Crampton [4] (for $\mathcal{D} = \emptyset$ and $k_r = |R|$). In the same section, we also prove that UAQ parameterized by $|P_{\ell b}| + k_p$ is W[1]-hard even under the following restrictions: (i) every role is authorized for at most three permissions from P and at most two permissions from $P \setminus P_{\ell b}$, and (ii) every two roles are authorized for at most two common permissions. This result

shows that we have parameterized intractability even for UAQ without SoD constraints, with quite a restrictive structure imposed on RPG, and parameterized by a parameter which can be quite large due to $|P_{\ell b}|$ (see, e.g., Table 1 in [1]).

Let R_2 be the set of roles that are authorized for at least one permission outside $P_{\ell b}$ and let $\hat{r} = |R_2|$. In Section 3 we also prove that UAQ with $\mathcal{D} = \emptyset$ and $k_r = |R|$ admits an algorithm of running time $O^*(2^{\hat{r}})$. Note that this algorithm is asymptotically faster than an algorithm of runtime $O^*(2^{|R|})$ introduced in [25] and studied in [22, 24], so it can be used for producing new UAQ benchmarks with $\mathcal{D} = \emptyset$ and $k_r = |R|$.

Let $\hat{k} = |P \setminus P_{\ell b}|$. In Section 4, we study UAQ such that $|P(r) \cap P(r')| \leqslant 1$ for all $r \neq r' \in R$. We prove that this form of UAQ, parameterized by $k_r + \hat{k}$, is W[1]-hard even when every SoD constraint $\langle X, t \rangle$ in $\mathcal{D}$ has $|X| = t = 2$. Note that in this result the parameter can be significantly larger than in the one above for $P_{\ell b} = P$ but we do allow SoD constraints. This result can be easily extended to the case when for fixed integers $\alpha \geq 2$ and $\beta \geq 2$, no α roles are collectively authorized for β permissions.

In Section 5, we study a more restrictive UAQ problem satisfying the following conditions for fixed integers $\alpha \geq 2$ and $\beta \geq 2$: (i) for every set of α roles, $\{r_1, \ldots, r_\alpha\}$, $|P(r_1) \cap P(r_2) \cap \cdots \cap P(r_\alpha)| < \beta$; (ii) there is a constant c such that for every SoD constraint $\langle X, t \rangle$ we have $|X| \leq c$; and (iii) for every pair $\langle X_1, t_1 \rangle, \langle X_2, t_2 \rangle$ of SoD constraints, we have $X_1 \cap X_2 = \emptyset$. We prove that this problem parameterized by $k_r + \hat{k}$ is FPT and admits an algorithm of running time $O^*(2^{O(k_r^\alpha + \hat{k})})$. The design of this algorithm incorporates several algorithmic tools: reduction rules, branching rules and advanced dynamic programming which uses representative families on matroids. Hence, we believe that it is unlikely that (simple) FPT algorithms exist when the parameter of interest is small, even for significantly restricted versions of UAQ, in sharp contrast to WSP, for which efficient FPT algorithms exist for most instances of practical interest [5, 15].

For ease of reference, we summarize the notation we use in the paper and our results in Tables 1 and 2, respectively.

2 PRELIMINARIES

Terminology and Notation. For a graph $G = \langle V(G), E(G) \rangle$ and vertex $x \in V(G)$, $N_G(x) = \{y \in V(G) \mid xy \in E(G)\}$ is the set of vertices adjacent to x ("neighbors"). For a set $S \subseteq V(G)$, $N(S) = \bigcup_{x \in S} N_G(x) \setminus S$. We will omit the subscript G when the graph is clear from the context.

A graph G is *bipartite* if its vertices can be partitioned into two sets A and B such that for all $ab \in E$, $a \in A$ and $b \in B$. We will generally write a bipartite graph G in the form $G = (A \uplus B, E)$. A bipartite graph $G = (A \uplus B, E)$ is *complete* if $ab \in E$ for every $a \in A$ and $b \in B$. Such a graph will also be denoted (up to isomorphism) by $K_{\alpha, \beta}$, where $\alpha = |A|$ and $\beta = |B|$. A graph is $K_{\alpha, \beta}$-*free* if it contains no induced subgraph isomorphic to $K_{\alpha, \beta}$.

For a positive integer k, we write $[k]$ to denote $\{1, 2, \ldots, k\}$.

Parameterized Complexity. An instance of a parameterized problem Π is a pair (I, k) where I is the *main part* and k is the *parameter*; the latter is usually a non-negative integer. A parameterized problem is *fixed-parameter tractable* (FPT) if there exists a computable function f such that instances (I, k) can be solved in time $O(f(k)|I|^c)$ where $|I|$ denotes the size of I and c is an absolute constant. The class of all fixed-parameter tractable decision problems is called FPT and algorithms which run in the time specified above are called FPT algorithms. As in other literature on FPT algorithms, we will often omit the polynomial factor in $O(f(k)|I|^c)$ and write $O^*(f(k))$ instead.

Consider two parameterized problems Π and Π'. We say that Π has a *parameterized reduction* to Π' if there are functions $k \mapsto k'$ and $k \mapsto k''$ from $\mathbb{N}$ to $\mathbb{N}$ and a function $(I, k) \mapsto (I', k')$ such that

Table 1. Summary of notation used in the paper

Notation	Meaning
R	Set of roles
P	Set of permissions
$RP \subseteq R \times P$	Role-permission assignment relation
$P(r) \subseteq P$	Set of permissions assigned to role r
$\langle X, t \rangle, X \subseteq R, t \in \mathbb{N}$	Separation-of-duty constraint
$\mathcal{D}$	Set of separation-of-duty constraints
$P_{\ell b} \subseteq P$	Required set of permissions in UAQ solution
$k_r \in \mathbb{N}$	Maximum number of roles in UAQ solution
$k_p \in \mathbb{N}$	Maximum number of permissions outside $P_{\ell b}$ in UAQ solution
$\hat{r} \in \mathbb{N}$	Number of roles assigned to at least one permission not in $P_{\ell b}$
$\hat{k} \in \mathbb{N}$	Number of permissions not in $P_{\ell b}$
$G = (V, E)$	Graph G with vertex set V and edge set E
$N(v), v \in V$	Set of neighbors of v in $G = \langle V, E \rangle$
$N(S), S \subseteq V$	Set of neighbors of vertices in S
$A \uplus B$	Union of disjoint sets A and B
$G = (A \uplus B, E)$	Bipartite graph G, $xy \in E$ iff $x \in A$ and $y \in B$
$K_{\alpha, \beta} = (A \uplus B, E)$	Complete bipartite graph, $|A| = \alpha$, $|B| = \beta$
$[k]$	Set of integers $\{1, \ldots, k\}$
$O^*(f(k))$	$O(f(k)p(k))$, where p is some polynomial

Table 2. Summary of our results

Input restrictions				Parameter	Complexity
k_r	k_p	RPG	$\mathcal{D}$		
$|R|$	–	–	$\emptyset$	k_p	W[2]-hard
–	0	–	$\emptyset$	k_r	W[2]-hard
$|R|$	–	–	$\emptyset$	$\hat{r}$	FPT
–	–	$|P(r)| \leq 3$ $1 \leq |P(r) \cap P_{\ell b}| \leq 3$ $|P(r) \cap P(r')| \leq 2$	$\emptyset$	$|P_{\ell b}| + k_p$	W[1]-hard
–	–	$K_{2,2}$-free	$\{\langle X_1, 2 \rangle, \ldots, \langle X_m, 2 \rangle\}$ $|X_i| = 2$	$k_r + \hat{k}$	W[1]-hard
–	–	$K_{\alpha, \beta}$-free	$\{\langle X_1, t_1 \rangle, \ldots, \langle X_m, t_m \rangle\}$ $|X_i| \leq c$ $X_i \cap X_j = \emptyset$	$k_r + \hat{k}$	FPT

(1) $(I, k) \mapsto (I', k')$ is computable in $k''(|I| + k)^{O(1)}$ time, and
(2) (I, k) is a yes-instance of Π if and only if (I', k') is a yes-instance of Π'.

While FPT is a parameterized complexity analog of P in classic complexity theory, there are many parameterized hardness classes, forming a nested sequence of which FPT is the first member: FPT$\subseteq$ W[1]$\subseteq$ W[2] $\subseteq \ldots$. It is well known that if the Exponential Time Hypothesis holds then FPT $\neq$ W[1].[1] Hence, W[1] is generally viewed as a parameterized intractability class, which is an analog of NP in classical complexity. Consider the following two parameterized problems. In the CLIQUE problem parameterized by k, given a graph G and a natural number k, we are to decide whether G has a complete subgraph on k vertices. In the DOMINATING SET problem parameterized by k, given a graph $G = (V, E)$ and a natural number k, we are to decide whether G has a set S of vertices such that every vertex in $V \setminus S$ is adjacent to some vertex in S. A parameterized problem Π is in W[1] (W[2], respectively) if it there is parameterized reduction from Π to CLIQUE (DOMINATING SET, respectively). Thus, every W[1]-hard problem Π_1 (W[2]-hard problem Π_2, respectively) is not 'easier' than CLIQUE (DOMINATING SET, respectively), i.e., CLIQUE (DOMINATING SET, respectively) has a parameterized reduction to Π_1 (Π_2, respectively).

More information on parameterized algorithms and complexity can be found in recent books [8, 10].

3 PARAMETERIZED HARDNESS OF UAQS WITHOUT SOD CONSTRAINTS

In this section, we consider USER AUTHORIZATION QUERY problems with $\mathcal{D} = \emptyset$. We will first consider two SIMPLE USER AUTHORIZATION QUERY problems defined as follows:

SIMPLE UAQ OF TYPE 1: $k_r = |R|$ and $\mathcal{D} = \emptyset$. Thus, an instance of k_p-SIMPLE UAQ problem can be written as a tuple $(R, P, RP, P_{\ell b}, k_p)$.

SIMPLE UAQ OF TYPE 2: $P_{\ell b} = P$ and $\mathcal{D} = \emptyset$. Thus, an instance of k_r-SIMPLE UAQ problem can be written as a tuple (R, P, RP, k_r).

Note that SIMPLE UAQ OF TYPE 1 is a natural simplification of UAQ, which prior research has established is NP-hard [4], thus establishing that more complex variants of the problem are also hard. Below we will strengthen this result, showing (i) that it is "hard" from the FPT perspective (unlike many versions of WSP [5]), and (ii) that FPT versions of the problem do exist when a different small parameter is considered. Type 2 problems essentially ask whether it is possible to find a small set of roles that are collectively authorized for a set of permissions. An algorithm to solve this problem may well have use in applications such as role mining.

Theorem 1. *The* SIMPLE UAQ PROBLEM OF TYPE 1 *(OF TYPE 2, respectively) parameterized by* k_p *(by* k_r, *respectively) is* W*[2]-hard.*

To prove this theorem, we will use the following problem, which is W[2]-complete [9].

RED BLUE DOMINATING SET
Input: A bipartite graph $G = (A \uplus B, E)$
Parameter: k
Question: Is there a subset S of A of size k such that $N(S) = B$?

The proof of the theorem is based on parameterized reductions from RED BLUE DOMINATING SET to the Simple UAQ problems.

PROOF OF THEOREM 1. Let $(G = (A \uplus B, E), k)$ be an instance of RED BLUE DOMINATING SET problem.

---

[1]The Exponential Time Hypothesis is a conjecture that there is no algorithm solving 3-CNF Satisfiability in time $2^{o(n)}$, where n is the number of variables.

Type 1. Let $L = \{p_v \mid v \in A\}$. Set $R = A$, $P_{\ell b} = B$, $P = B \uplus L$, $RP = E \cup \{(v, p_v) \mid v \in A\}$ and $k_p = k$. Let $S \subseteq A$. Note that by definitions of Red Blue Dominating Set and Simple UAQ problem of Type 1, S is a solution of an Red Blue Dominating Set instance, i.e. $N(S) = B$ and $|S| \leq k$, if and only if S is a solution of the corresponding UAQ instance, i.e., $P_{lb} \subseteq P(S)$ and $|P(S) \setminus P_{lb}| = |p_v : v \in S| \leq k = k_p$. Thus, $(G = (A \uplus B, E), k)$ is a yes-instance of Red Blue Dominating Set if and only if $(R, P, RP, P_{\ell b}, k_p)$ is a yes-instance of Simple UAQ of Type 1. Since Red Blue Dominating Set is W[2]-hard this reduction shows that Simple UAQ of Type 1 is also W[2]-hard.

Type 2. Set $R = A$, $P = B$, $RP = E$ and $k_r = k$. This gives us an instance (R, P, RP, k_r) of Simple UAQ of Type 2. By the definitions of Red Blue Dominating Set and Simple UAQ of Type 2 and setting $R_s = S$, it follows that $(G = (A \uplus B, E), k)$ is a yes-instance of Red Blue Dominating Set if and only if (R, P, RP, k_r) is a yes-instance of Simple UAQ of Type 2. Since Red Blue Dominating Set is W[2]-hard this reduction shows that Simple UAQ of Type 2 is also W[2]-hard. □

Theorem 1 asserts that Simple UAQ of Type 1 parameterized by k_p is W[2]-hard, implying that it is highly unlikely to be FPT. This result is somewhat unexpected, given that WSP (which appears to be a more complex problem) is FTP for most instances that are likely to arise in practice [5, 7].

However, we are able to prove that Simple UAQ of Type 1 is FPT when parameterized by a different parameter. Let $R_1 = \{r \in R \mid P(r) \subseteq P_{lb}\}$, $R_2 = R \setminus R_1$, and $\hat{r} = |R_2|$. In other words, $\hat{r}$ is the number of roles that are assigned to at least one permission outside P_{lb}. We then have the following result.

Theorem 2. Simple UAQ of Type 1 *admits an algorithm of running time* $O^*(2^{\hat{r}})$. *Thus,* Simple UAQ of Type 1 *parameterized by $\hat{r}$ is* FPT.

Proof. Let $\mathcal{I}$ be an instance of Simple UAQ of Type 1. We consider every subset S_2 of R_2. If $|P(S_2 \cup R_1)| \leq k_p + |P_{lb}|$ and $P(S_2 \cup R_1) \supseteq P_{lb}$, then $\mathcal{I}$ is a yes-instance. If no such S_2 exists, then $\mathcal{I}$ is a no-instance.

Checking whether $|P(S_2 \cup R_1)| \leq k_p + |P_{lb}|$ and $P(S_2 \cup R_1) \supseteq P_{lb}$ can be done in polynomial time. Thus, the overall running time is $O^*(2^{\hat{r}})$. □

It is not clear how useful this result will be in practice. Further research is required to determine the likelihood of this parameter being small in real-world instances.

In the rest of this section, we will prove the following:

Theorem 3. *UAQ parameterized by $|P_{\ell b}| + k_p$ is W[1]-hard even under the following restrictions:*
- *for all $r \in R$, $|P(r)| \leq 3$ and $1 \leq |P(r) \cap P_{lb}| \leq 3$, and*
- *for all $r, r' \in R$, $|P(r) \cap P(r')| \leq 2$.*

The first restriction requires that every role is authorized for at most three permissions and at most two permissions not in P_{lb}. The second restriction requires that every pair of roles is authorized for at most two common permissions.

It is worth noting that the first restriction is unlikely to be satisfied by real-world RBAC instances. RBAC is based on an assumption that the complexity of managing access control systems can be reduced by introducing (a relatively small number of) roles as an abstraction acting as an intermediate layer between users and permissions and the roles will be assigned to many permissions and many users. (In other words the number of relationships that needs to be maintained is $O(|R|(|U| + |P|))$ rather than $O(|U||P|)$.) In short, imposing such a small upper bound on the number of permission assignments to each role means that this result is unlikely to be applicable to many real-world instances. (The second restriction is less problematic, at least in the non-hierarchical setting, as

one would expect sets of permissions assigned to different roles to be approximately disjoint, with users being assigned to several roles.)

To prove this theorem, we provide a parameterized reduction from MULTICOLORED BICLIQUE, which is known to be W[1]-hard [8], defined as follows.

MULTICOLORED BICLIQUE
Input: A bipartite graph $G = (A \uplus B, E)$, $A = A_1 \uplus \cdots \uplus A_k$, $B = B_1 \uplus \cdots \uplus B_k$
Parameter: k
Question: Is there $A' \subseteq A, B' \subseteq B$ such that for every $i \in [k]$, $|A_i \cap A'| = |B_i \cap B'| = 1$, and $A' \cup B'$ induces a complete bipartite graph?

CONSTRUCTION 1. *Consider an instance of* MULTICOLORED BICLIQUE, *where $G = (A \uplus B, E)$ is a bipartite graph, $A = A_1 \uplus A_2 \uplus \cdots \uplus A_k$, and $B = B_1 \uplus B_2 \uplus \ldots \uplus B_k$. We construct an instance of* USER AUTHORIZATION QUERY *as follows:*

- $R = \{r_{uv} \mid uv \in E\}$,
- $P_{\ell b} = \{p_{i,j} \mid (i,j) \in [k] \times [k]\}$,
- $P = P_{\ell b} \cup A \cup B$,
- *for every $uv \in E$, if $u \in A_i$, $v \in B_j$, then $P(r_{uv}) = \{u, v, p_{i,j}\}$,*
- $k_r = k^2$, $k_p = 2k$.

In the construction we create a role for every edge of G, a permission for every vertex in G, and a permission for every ordered pair in $[k] \times [k]$. Moreover, the vertices of G form $P \setminus P_{lb}$. The intuition behind the construction is that the permissions associated with a role r_e encode an edge $e = uv \in E$, and the pair (A_i, B_j) to which u and v respectively belong. Note that the USER AUTHORIZATION QUERY instance generated by Construction 1 satisfies the criteria in Theorem 3.

Consider the instance of MULTICOLORED BICLIQUE shown in Figure 1(a) comprising sets $A = \{a_1, a_2\} \uplus \{a_3, a_4, a_5\}$ and $B = \{b_1, b_2\} \uplus \{b_3, b_4\}$. The instance has a solution, also shown in Figure 1(a). Construction 1 generates a UAQ instance with solution $\{r_{22}, r_{23}, r_{32}, r_{33}\}$, corresponding to the edges in the biclique solution. The permissions associated with these roles respectively include p_{11}, p_{12}, p_{21} and p_{22} (the set $P_{\ell b}$), reflecting the fact that the MULTICOLORED BICLIQUE solution must contain an edge from every block in the partition of A to every block in the partition of B. (Hence, there must be k^2 such permissions - four in this example.) Moreover, each role is assigned two further permissions, corresponding to the endpoints of the edges defining the roles. (Hence, there are a further $2k$ permissions - four in this example.) Note that in Figure 1(b) we used dashed and dotted lines in the UAQ solution to differentiate between the edges that "encode" the required relationships between the blocks in the MULTICOLORED BICLIQUE instance and the edges in the MULTICOLORED BICLIQUE solution, respectively.

We have the following:

Lemma 1. *Let $(R, P, RP, P_{\ell b}, \emptyset, k_r, k_p)$ be a UAQ instance created from an instance (G, k) of* MULTICOLORED BICLIQUE. *Then, (G, k) is a yes-instance if and only if $(R, P, RP, P_{\ell b}, \emptyset, k_r, k_p)$ is.*

PROOF. First we give the forward direction ($\Rightarrow$) of the proof. Let (G, k) be a yes-instance of MULTICOLORED BICLIQUE and suppose that $S \subseteq V$ be a solution of (G, k). This means that for every $i \in [k]$, $|S \cap A_i| = |S \cap B_i| = 1$. We construct the role set $R_s \subseteq R$ using $S \subseteq V$ as follows. We put r_{uv} into R_s if and only if uv is an edge in $G[S]$. Observe that $P(R_s) \supseteq P_{\ell b}$ since S intersects every A_i and every B_j. Since every role of R_s corresponds to an edge in $G[S]$, and there are k^2 edges in $G[S]$, we have $|R_s| = k^2$. Since there are $2k$ vertices in S, R_s is authorized for $2k$ permissions in $P \setminus P_{\ell b}$.

Now, we give the backward direction ($\Leftarrow$) of the proof. Let $R_s \subseteq R$ be a set of at most k^2 roles that is authorized for all permissions in $P_{\ell b}$ and at most $2k$ permissions in $P \setminus P_{\ell b}$. Consider the

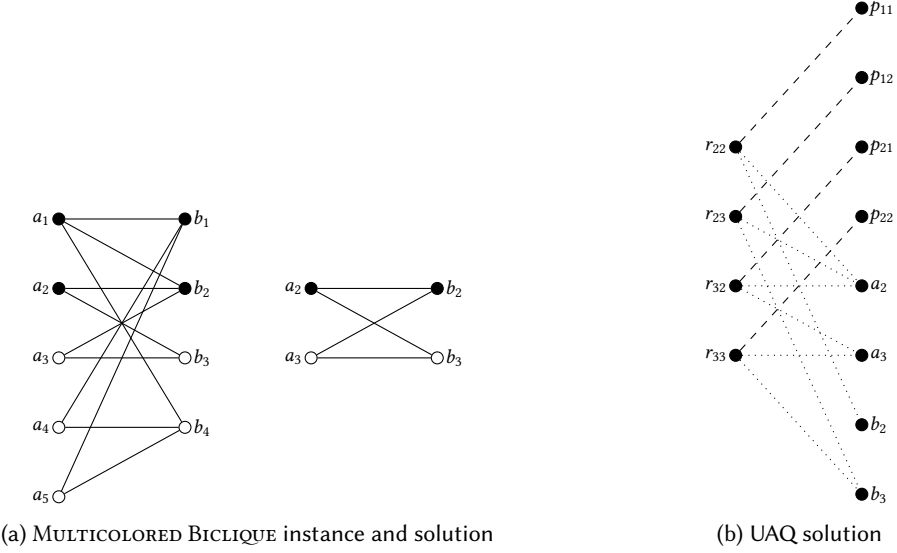(a) MULTICOLORED BICLIQUE instance and solution   (b) UAQ solution

Fig. 1. MULTICOLORED BICLIQUE instance and corresponding UAQ solution using Construction 1

set S of permissions from $P \setminus P_{\ell b}$ authorized by R_s. We claim that $G[S]$ is a complete bipartite graph such that for all $i \in [k]$, $|S \cap A_i| = |S \cap B_i| = 1$. Consider an arbitrary permission $p_{i,j} \in P_{\ell b}$. Only a role r_{uv} such that $u \in A_i, v \in B_j$ is authorized for this permission. Hence, for every $i \in [k]$, $|S \cap A_i|, |S \cap B_i| \geq 1$. Furthermore, the roles in R_s are assigned at most $2k$ permissions from $P \setminus P_{\ell b}$. So, $|S \cap A_i| = |S \cap B_i| = 1$. For any role $r_{uv} \in R_s$, $uv \in E$. So, $G[S]$ induces a complete bipartite graph. Hence, (G, k) is a yes-instance for MULTICOLORED BICLIQUE. □

Proof of Theorem 3. By Construction 1, there exists a polynomial time reduction from MULTICOLORED BICLIQUE parameterized by k to UAQ parameterized by $k^2 + 2k$ where $k_r = k^2$, $k_p = 2k$. Also by this construction, every role is authorized for at most two permissions from $P \setminus P_{\ell b}$ and at most three permissions from P. Since every role is authorized for exactly three permissions and no pair of roles is authorized for the same set of permissions, every pair of roles is authorized for at most two common permissions. By construction, $|P_{\ell b}| = k^2$. Hence, by Lemma 1 and the fact that MULTICOLORED BICLIQUE is W[1]-hard, UAQ parameterized by $|P_{\ell b}| + k_p$ is W[1]-hard even when every role is authorized at most three permissions from P and at most two permissions from $P \setminus P_{\ell b}$ and every two roles are authorized for at most two common permissions. □

4 W[1]-HARDNESS WHEN RPG IS $K_{2,2}$-FREE

We now consider restricting UAQ to instances in which $|P(r) \cap P(r')| \leq 1$ for all $r, r' \in R$. In other words, any two roles are authorized for at most one common permission. Technically, this is equivalent to saying the bipartite graph RPG contains no subgraph isomorphic to $K_{2,2}$. Hence we will call such instances of UAQ $K_{2,2}$-*free*. We show that UAQ for $K_{2,2}$-free instances is W[1]-hard. We believe that this result is useful because it demonstrates that UAQ remains a hard problem even when we impose a strong restriction (and one that few RBAC instances are likely to satisfy) on RPG. The practical consequence of this result is that UAQ is hard for most real-world RBAC instances with constraints.

In Section 5, we relax the condition on RPG somewhat (making UAQ $K_{\alpha, \beta}$-free for $\alpha, \beta > 2$) but impose restrictions on the set of constraints. We show that UAQ is FPT with these restrictions. This

positive result means that UAQ in real-world RBAC instances *without* constraints (i.e., instances that trivially satisfy any restrictions on the set of constraints) may be solvable in a reasonable amount of time. However, the FPT algorithm required to solve UAQ is extremely complicated and requires sophisticated programming techniques. Moreover, the non-polynomial terms in the running time are significant. In short, we believe it is unlikely that a simple and practical FPT algorithm exists for instances of this type.

In the remainder of the paper we will write $\hat{k}$ to denote the number of permissions that are not in $P_{\ell b}$ (i.e., $|P \setminus P_{\ell b}|$).

Theorem 4. *Consider a $K_{2,2}$-free UAQ instance. Then, UAQ parameterized by $k_r + \hat{k}$ is W[1]-hard even when every SoD constraint $\langle X, t \rangle$ in $\mathcal{D}$ has $|X| = t = 2$.*

The proof is based on a parameter-preserving reduction from MULTICOLORED BICLIQUE to UAQ. We construct a UAQ instance as follows. (This construction is similar to the one used in [14].)

CONSTRUCTION 2. *Given $G = (V, E)$, where $V = A_1 \uplus \cdots \uplus A_k \uplus B_1 \uplus \cdots \uplus B_k$:*

- $R = \{r_v \mid v \in V\} \cup \{s\}$;
- $P = \{p_v \mid v \in V\} \cup [2k] \cup \{q\}$, *and* $P_{\ell b} = P$;
- *for every* $i \in [k]$, *if* $v \in A_i$, *then* $P(r_v) = \{p_v, i\}$;
- *for every* $i \in [2k] \setminus [k]$, *if* $v \in B_{i-k}$, *then* $P(r_v) = \{p_v, i\}$;
- $P(s) = \{p_v \mid v \in V\} \cup \{q\}$;
- *given* $u \in A, v \in B$, $\langle \{r_u, r_v\}, 2 \rangle \in \mathcal{D}$ *iff* $uv \notin E$;
- $k_r = 2k + 1$ *and* $k_p = 0$.

Construction 2 defines a role and a permission for each vertex. Additionally, we define role s and permissions $1, \ldots, 2k$ and q. The additional role and permissions are used to encode the structure of the bipartite graph in terms of the sets into which A and B are partitioned, via the *RP* relation. Finally, we use separation-of-duty constraints to ensure that we prohibit solutions to UAQ that are not consistent with the structure of the bipartite graph in the MULTICOLORED BICLIQUE instance.

Observe that the resulting UAQ instance is non-hierarchical. We now show it is $K_{2,2}$-free.

Lemma 2. *Let $(G = (A \uplus B, E), k)$ be an instance of MULTICOLORED BICLIQUE. Then the UAQ instance derived using Construction 2 is $K_{2,2}$-free.*

PROOF. By construction, $R = \{r_v \mid v \in V\} \cup \{s\}$. Consider r_u, r_v for two distinct vertices $u, v \in V$. The following cases can arise.

Case 1: If $u, v \in A_i$ for some $i \in [k]$, then $P(r_u) = \{p_u, i\}, P(r_v) = \{p_v, i\}$. Similarly, if $u, v \in B_{i-k}$ for some $i \in [2k] \setminus [k]$, then $P(r_u) = \{p_u, i\}, P(r_v) = \{p_v, i\}$. Then, $|P(r_u) \cap P(r_v)| \leq 1$.

Case 2: If $u \in A_i, v \in A_j$ for $i, j \in [k], i \neq j$, then $P(r_u) \cap P(r_v) = \{p_u, i\} \cap \{p_v, j\} = \emptyset$. Similarly, if $u \in B_{i-k}, v \in B_{j-k}$ for two distinct $i, j \in [2k] \setminus [k]$, then $P(r_u) \cap P(r_v) = \emptyset$. Similarly, if $u \in A_i, v \in B_{j-k}$ for $i \in [k], j \in [2k] \setminus [k]$, then $P(r_u) \cap P(r_v) = \emptyset$.

Case 3: Let $u \in V$. Recall that $P(s) = \{p_u \mid u \in V\} \cup \{q\}$, and $P(r_u) = \{p_u, i\}$ for some $i \in [2k]$. Hence, $P(r_u) \cap P(s) = \{p_u\}$.

The above cases are exhaustive and, in each case, every pair of distinct roles is assigned at most one common permission. □

Lemma 3. *$(R, P, RP, \mathcal{D}, k_r, k_p)$ is a yes-instance for UAQ if and only if (G, k) is a yes-instance for MULTICOLORED BICLIQUE.*

PROOF. First we give the forward direction ($\Rightarrow$) of the proof. Let (G, k) be a yes-instance of MULTICOLORED BICLIQUE. Then, there exists $S \subseteq V$ such that $|S| = 2k$, for all $i \in [k]$, $|A_i \cap S| =$

$|B_i \cap S| = 1$, and $G[S]$ induces a complete bipartite graph. We construct R^* from S as follows: $R^* = \{r_u \mid u \in S\} \cup \{s\}$. By construction $|R^*| = 2k + 1 = k_r$. Since for every $i \in [k]$, S contains exactly one vertex from A_i, R^* is authorized for all permissions in $[k]$. Since for every $i \in [2k] \setminus [k]$, S contains exactly one vertex from B_{i-j}, R^* is authorized for all permissions in $[2k] \setminus [k]$. The role s is authorized for all permissions in $P \setminus [2k]$. Hence, R^* is authorized for all permissions in P.

Consider an arbitrary SoD constraint $\langle \{r_u, r_v\}, 2 \rangle \in \mathcal{D}$. By construction, we have $u \in A, v \in B$, and $uv \notin E$. But, then $\{u, v\} \nsubseteq S$. However, by construction, $G[S]$ induces a complete bipartite graph in G with bipartition $S_1 \uplus S_2$ such that $S_1 \subseteq A, S_2 \subseteq B$. Hence, for every $u \in S_1, v \in S_2, uv \in E$. Thus, all the SoD constraints are satisfied for R^*. Hence, $(R, P, RP, \mathcal{D}, k_r, k_p)$ is a yes-instance of UAQ.

Now, we give the backward direction ($\Leftarrow$) of the proof. Let $(R, P, RP, \mathcal{D}, k_r, k_p)$ be a yes-instance of UAQ. Then, there exists a set R^* of at most k_r roles that are authorized for all permissions of P. Observe that $s \in R^*$ since $q \in P$. Moreover, R^* is authorized for all permissions in $[2k]$. Every $i \in [k]$ can be authorized by a unique role r_v such that $v \in A_i$. Similarly, every $i \in [2k] \setminus [k]$ can be authorized by a unique role r_v such that $v \in B_{i-k}$. We construct S from R^* as follows. We put $v \in S$ if $r_v \in R^*$. Since $R^* \setminus \{s\}$ contains $2k$ roles, S contains $2k$ vertices. Hence, $|S| = 2k$ and $|S \cap A_i| = |S \cap B_i| = 1$.

We have $S = S_1 \uplus S_2$, where $S_1 = S \cap A$ and $S_2 = S \cap B$. Note that R^* satisfies all constraints in $\mathcal{D}$ and consider a constraint $\langle \{r_u, r_v\}, 2 \rangle \in \mathcal{D}$ such that w.l.o.g. $u \in A$ and $v \in B$. Since R^* satisfies this constraint, $|R^* \cap \{r_u, r_v\}| \leq 1$ and hence $|\{u, v\} \cap S| \leq 1$. Hence, for every $u \in S_1$ and $v \in S_2$, we have $uv \in E$ implying that $G[S]$ is a complete bipartite graph. Therefore, (G, k) is a yes-instance for Multicolored Biclique.   □

Proof of Theorem 4. Based on the above-mentioned construction and Lemma 3, an instance (G, k) of Multicolored Biclique can be transformed into an equivalent instance $(R, P, RP, \mathcal{D}, k_r, k_p)$ of UAQ where every SoD constraint is of the form $\langle X, 2 \rangle$ such that $|X| = 2$. By construction, $P_{lb} = P$. The parameter is transformed from k to $k_r + \hat{k} = 2k + 1$. Hence, this is a parameterized reduction and the fact that Multicolored Biclique is W[1]-hard, UAQ is W[1]-hard when parameterized by $k_r + \hat{k}$.   □

Let $\alpha \geq 2$ and $\beta \geq 2$ be fixed integers and assume that every α roles are authorized for at most $\beta - 1$ common permissions. In other words, RPG is $K_{\alpha,\beta}$-free. We call such UAQ the $K_{\alpha,\beta}$-free UAQ problem. Note that if RPG is $K_{2,2}$-free then it is $K_{\alpha,\beta}$-free for every $\alpha \geq 2$ and $\beta \geq 2$. Thus, $K_{2,2}$-free UAQ is a special case of $K_{\alpha,\beta}$-free UAQ. Therefore, parameterized intractability of Theorem 4 can be extended to arbitrary integers $\alpha \geq 2$ and $\beta \geq 2$. As the main result of the next section shows, this parameterized intractability can be attributed to the fact that here SoD constraints can have overlapping role sets.

5 FIXED-PARAMETER TRACTABILITY WHEN RPG IS $K_{\alpha,\beta}$-FREE AND CONSTRAINTS ARE NON-INTERSECTING

Let $\alpha \geq 2$ and $\beta \geq 2$ be fixed integers. In this section, we consider the User Authorization Query problem restricted by the following conditions:

   (i) for every set of α roles, $\{r_1, \ldots, r_\alpha\}$, $|P(r_1) \cap P(r_2) \cap \cdots \cap P(r_\alpha)| < \beta$;
   (ii) there is a constant c such that for every SoD constraint $\langle X, t \rangle$ we have $|X| \leq c$;
   (iii) for every pair $\langle X_1, t_1 \rangle, \langle X_2, t_2 \rangle$ of SoD constraints, we have $X_1 \cap X_2 = \emptyset$.

We call this the (α, β)-User Authorization Query problem ((α, β)-UAQ). Note that the RPG of an instance of (α, β)-UAQ is $K_{\alpha,\beta}$-free.

Recall that $\hat{k} = |P \setminus P_{\ell b}|$. We consider (α, β)-UAQ parameterized by $k_r + \hat{k}$ and prove that it is FPT by designing an algorithm with running time $O^*(2^{O(k_r^\alpha + \hat{k})})$ to solve the problem. The algorithm makes use of matroids and dynamic programming. Moreover, we have to perform some preprocessing on an (α, β)-UAQ instance to produce the input to the algorithm. Note that the algorithm is not intended to inspire implementations, but rather offer a constructive proof of the main result in this section.

In Section 5.1, we use reduction and branching rules to reduce the size of the original instance. In particular, this preprocessing phase reduces the size of $P_{\ell b}$ to $O(\beta k_r^\alpha)$. This preprocessing takes $O^*(\alpha^{k_r})$ time and also computes a partial solution R_1 to the input instance. In every reduction and branching rule, if we delete a role r from the input and reduce k_r, then we add r to R_1. Otherwise, if k_r remains unchanged, we just delete r from the input instance.

After the preprocessing phase, it is possible to use a dynamic programming algorithm to determine whether there *exists* a solution to the original instance. We can use the same algorithm to determine the *size* of the solution set. However, more advanced techniques are required to compute a *solution set*.

Recall that a dynamic programming (DP) algorithm stores and re-uses solutions to smaller sub-problems. In order to find a solution to (α, β)-UAQ, we need to store in the DP table some candidate partial solutions in each of the DP table entries. However, it is not sufficient to store one partial candidate solution in each DP table entry since we do not know which specific partial candidate solution would extend to a candidate solution for the entire instance.

A naive approach could store all possible candidate partial solutions in the DP table entries. Then the number of candidate partial solutions in a DP table entry could be as large as $|R|^{k_r}$; so this approach would not result in an FPT algorithm. Thus, we have to store only a "small" subset of partial candidate solutions in each DP table entry. To be able to do this, we use an advanced algorithmic technique known as the *method of representative families* on matroids. This method ensures that if the input instance is a yes-instance, then a solution will be found, despite storing only $O^*(2^{k_r})$ partial candidate solutions in a table entry, stored in a table containining $O^*(2^{O(k_r^\alpha)+\hat{k}})$ entries.

In Section 5.2, we provide the terminology, notation and results on matroids and representative families that are necessary for describing and analyzing our DP algorithm based on the method of representative families on matroids. We also describe how to construct a special partition matroid used in the DP algorithm. This matroid enables us to identify those candidate solutions that do not violate any constraints in the instance.

Finally, in Section 5.3, we describe and analyze our DP algorithm which takes a simplified instance as input and finds a solution R_2 of this instance such that $R_1 \cup R_2$ is a solution of the original instance. We also show that our overall algorithm for (α, β)-UAQ is FPT parameterized by $k_r + \hat{k}$.

5.1 Preprocessing

In this section, we define several reduction and branching rules. A reduction rule for an instance (I, k) of a parameterized problem is *safe* if it reduces (I, k) to (I', k') such that (I, k) is a yes-instance if and only if (I', k') is. A branching rule is *safe* if for input (I, k) it outputs a number q of instances $(I'_1, k'_1), \ldots, (I'_q, k'_q)$, $q \geq 1$, such that (I, k) is a yes-instance if and only if (I'_i, k'_i) is for some $i \in [q]$. We prove that each rule is safe and can be implemented either in polynomial time (for the reduction rules) or in FPT time (for the branching rule). Moreover, we construct a partial solution R_1 of the problem, comprising those roles that must be in any solution of the problem; R_1 is initialized as the empty set.

Assume that we have an initial instance $\mathcal{I} = (R, P, RP, P_{lb}, \mathcal{D}, k_r, k_p)$ of (α, β)-UAQ. Note that the initial instance of the problem (already simplified by Reduction Rule 0) will not be reducible by Reduction Rule 0. However, a variant of that rule is still required since it may be applied to instances generated by other reduction rules. The following Reduction Rule 1 is the variant of Reduction Rule 0 used in this section.

Reduction and branching rules are applied to an initial instance of the problem in the order the rules are described in this section. For every rule we take an input instance and output an instance or a number of instances (for a branching rule) such that if the output instance or one of the output instances is different from the input instance then all the previous rules are applied to the new instance (or, each of the new instances).

Reduction Rule 1. *(i) delete any role r such that $P(r) \cap P_{lb} = \emptyset$ (since r cannot contribute to a set of roles that satisfies the instance);*
(iii) delete any role which is in a SoD constraint $\langle X, 1 \rangle$;
(iv) for any deleted role r and any SoD constraint $\langle X, t \rangle$ with $r \in X$, replace the constraint by $\langle X \setminus \{r\}, t \rangle$;
(v) for any deleted role r, remove all pairs of the form (r, p) from RP;
(vi) delete any constraint $\langle X, t \rangle$ in which $|X| < t$.

We use the following *update procedure* for some of our subsequent rules.

PROCEDURE UPDATE. *Let $r \in R$ be an arbitrary role. Then, Procedure UPDATE(r) executes the following steps.*

- *Add r to R_1;*
- *Set $R \leftarrow R \setminus \{r\}$, $k_r \leftarrow k_r - 1$, $k_p \leftarrow k_p - |P(r) \setminus P_{\ell b}|$, $P_{\ell b} \leftarrow P_{\ell b} \setminus P(r)$, $P \leftarrow P \setminus P(r)$;*
- *Remove from RP all pairs which include permissions from $P(r)$; and*
- *For every constraint $\langle X, t \rangle \in \mathcal{D}$, if $r \in X$, then replace the constraint $\langle X, t \rangle$ by $\langle X \setminus \{r\}, t - 1 \rangle$.*

Reduction Rule 2. *Suppose that there is a permission $p \in P_{\ell b}$ that is assigned to a unique role $r \in R$. If $|P(r) \setminus P_{\ell b}| \geq k_p + 1$, then $(R, P, RP, P_{lb}, \mathcal{D}, k_r, k_p)$ is a no-instance. Otherwise (i.e. when $|P(r) \setminus P_{\ell b}| \leq k_p$), perform Procedure UPDATE(r).*

Lemma 4. *Reduction Rule 2 is safe and can be implemented in polynomial time.*

PROOF. For convenience we will denote the reduced instance by $(R', P', RP', P'_{\ell b}, \mathcal{D}', k'_r, k'_p)$. When $r \in R$ is the unique role to which $p \in P_{\ell b}$ is assigned, then r must be in any solution. But, if r is authorized for more than k_p permissions from $P \setminus P_{\ell b}$, then r cannot belong to any solution. Thus, the input instance is a no-instance. So, we assume that $|P(r) \setminus P_{\ell b}| \leq k_p$.

The backward direction ($\Leftarrow$) of the reduction rule is trivial. Let $R^* \subseteq R'$ be a set of at most k'_r roles such that $P(R^*) \supseteq P'_{\ell b}$ and $|P(R^*) \setminus P'_{\ell b}| \leq k'_p$. Also, R^* satisfies all SoD constraints in $\mathcal{D}'$. Then, $R^* \cup \{r\}$ is a set of at most $k'_r + 1 = k_r$ roles such that $P(R^* \cup \{r\}) \supseteq P_{\ell b}$ and $|P(R^* \cup \{r\}) \setminus P_{\ell b}| \leqslant k'_p + |P(r) \cap (P \setminus P_{\ell b})| = k_p$. Also, if a constraint $\langle X, t \rangle \in \mathcal{D}$ contains r, then this constraint $\langle X, t \rangle$ will have been replaced by $\langle X \setminus \{r\}, t - 1 \rangle \in \mathcal{D}'$. Since, $|(X \setminus \{r\}) \cap R^*| \leq t - 1$, we have that, $|X \cap (R^* \cup \{r\})| \leq t$. Hence, all constraints in $\mathcal{D}$ are also satisfied.

Now, we give the forward direction ($\Rightarrow$) of the proof. Let $(R, P, PR, P_{\ell b}, \mathcal{D}, k_r, k_p)$ be a yes-instance. Since $p \in P_{\ell b}$ is assigned to a unique role $r \in R$, r must be in any solution of $(R, P, PR, P_{\ell b}, \mathcal{D}, k_r, k_p)$. If R^* is a set of at most k_r roles that is a solution for $(R, P, PR, P_{\ell b}, \mathcal{D}, k_r, k_p)$, then $R^* \setminus \{r\}$ is a set of at most k'_r roles such that $P(R^* \setminus \{r\}) \supseteq P'_{\ell b}$ and $|P(R^* \setminus \{r\}) \setminus P'_{\ell b}| \leq k'_p$. Let $\langle X, t \rangle \in \mathcal{D}$ be an SoD constraint such that $r \in X$. Then, $\langle X, t \rangle$ will have been replaced by $\langle X \setminus \{r\}, t - 1 \rangle \in \mathcal{D}'$. Hence, $\langle X \setminus \{r\}, t - 1 \rangle$ is satisfied by $R^* \setminus \{r\}$ and all SoD constraints in $\mathcal{D}'$ are also satisfied. □

Branching Rule 1. *Let $b = \beta k_r^q + \sum\limits_{a=1}^{q-1} k_r^a$. For $q = 1, 2, \ldots, \alpha - 2$, in this order, apply Branching Rule 1.q repeatedly until it no longer causes any changes to the RPG.*

- **Branching Rule 1.q:** *If there exists a set $L \subseteq R$ with $(\alpha - q)$ roles such that $|(\bigcap\limits_{r \in L} P(r)) \cap P_{\ell b}| > b$, then apply the following branching: for every role r in L such that $|P(r) \setminus P_{\ell b}| \leq k_p$, perform Procedure UPDATE(r).*

Lemma 5. *Let $R^* \subseteq R$ be a set of at most k_r roles that satisfies all constraints in $\mathcal{D}$, where $P(R^*) \supseteq P_{\ell b}$ and $|P(R^*)| \leq |P_{\ell b}| + k_p$. Consider an application of Branching Rule 1.q, $1 \leq q \leq \alpha - 2$. If L is a set of roles from R that satisfies the condition in Branching Rule 1.q, then $R^* \cap L \neq \emptyset$. Furthermore, $R^* \cap L$ is authorized for at most k_p permissions from $P \setminus P_{\ell b}$.*

PROOF. Let $\hat{P}_{\ell b} = (\bigcap_{r \in L} P(r)) \cap P_{\ell b}$. Let $R^* \subseteq R$ be a set of at most k_r roles that satisfies all constraints in $\mathcal{D}'$, is authorized for all permissions in $P_{\ell b}$, and at most k_p permissions from $P \setminus P_{\ell b}$. Consider two cases.

Case 1: $q = 1$. Then $|L| = \alpha - 1$. Suppose that $R^* \cap L = \emptyset$. Consider any role $r \in R^*$. Since RPG is $K_{\alpha,\beta}$-free, $|P(r) \cap \hat{P}_{\ell b}| \leq \beta - 1$. Then, R^* can be authorized for at most $(\beta - 1)k_r$ permissions from $\hat{P}_{\ell b}$. This implies that there exists some permission $p \in \hat{P}_{\ell b} \subseteq P_{\ell b}$ such that R^* cannot authorize p. This is a contradiction. Hence, $L \cap R^* \neq \emptyset$.

Case 2: $2 \leq q \leq \alpha - 2$. Suppose that $R^* \cap L = \emptyset$. Since $|\hat{P}_{\ell b}| > b$, by the pigeon hole principle, there exists a role $y \in R^*$ such that y is authorized for at least $b/k_r + 1 = \beta k_r^{q-1} + k_r^{q-2} + \cdots + k_r + 1 + 1$ permissions in $\hat{P}_{\ell b}$. Then, consider $P(L \cup \{y\})$. There are $(\alpha - q + 1)$ roles in $L \cup \{y\}$ and $L \cup \{y\}$ is authorized for at least $\beta k_r^{q-1} + k_r^{q-2} + \cdots + k_r + 1 + 1$ common permissions in $P_{\ell b}$. Then, Branching Rule 1.$(q - 1)$ is also applicable. But, we apply Branching Rule 1.q only when for every $i \in [q - 1]$, Branching Rule 1.i is not applicable. This is a contradiction. Hence, R^* has nonempty intersection with L.

As R^* is authorized for at most k_p permissions from $P \setminus P_{\ell b}$, $L \cap R^*$ is authorized for at most k_p permissions in $P \setminus P_{\ell b}$. This completes the proof. □

Reduction Rule 3. *Suppose that there is a role $s \in R$ such that s is authorized for at least $h = \beta k_r^{\alpha-1} + k_r^{\alpha-2} + \cdots + k_r^2 + k_r + 1$ permissions in $P_{\ell b}$. If s is authorized for more than k_p permissions from $P \setminus P_{\ell b}$, then $(R, P, RP, P_{\ell b}, \mathcal{D}, k_r, k_p)$ is a no-instance. Otherwise, perform Procedure UPDATE(s).*

Lemma 6. *Reduction Rule 3 is safe and can be implemented in polynomial time.*

PROOF. It suffices to prove that any solution of $(R, P, RP, P_{\ell b}, \mathcal{D}, k_r, k_p)$ must contain s. Let $A = P(s)$. Suppose that there exists a solution $S \subseteq R$ such that $s \notin S$ and $|S| \leq k_r$. Recall that RPG is $K_{\alpha,\beta}$-free. By assumption, S is authorized for all permissions in A. Since $|A| \geq h$ (by the precondition of the reduction rule), there exists $s' \in S$ such that s' is auhorized for at least $\lceil h/k_r \rceil = \beta k_r^{\alpha-2} + k_r^{\alpha-3} + \cdots + k_r^2 + 1 + 1$ permissions from $P_{\ell b}$. Then, R has $r_1, r_2 \in S$ such that $P(r_1) \cap P(r_2)$ contains at least $\beta k_r^{\alpha-2} + k_r^{\alpha-3} + \cdots + k_r^2 + 1 + 1$ permissions. This means that the precondition of Branching Rule 1.q becomes applicable for $q = \alpha - 2$. This is a contradiction. Hence, $s \in S$. □

After Reduction Rule 2, Branching Rule 1, and Reduction Rule 3 have been performed, there is still a possibility that there is a role r such that $|P(r) \setminus P_{\ell b}| > k_p$. However, such roles cannot be included in any solution. Hence, we need to apply the following reduction rule.

Reduction Rule 4. *Delete any role r such that $|P(r) \setminus P_{lb}| > k_p$ (since r cannot be included in any yes-instance). Remove any pair from RP that contains a deleted role. If there is a deleted role $r \in X$ for a constraint $\langle X, t \rangle$, then replace this constraint by $\langle X \setminus \{r\}, t \rangle$.*

Lemma 7. *Suppose that $\mathcal{I} = (R, P, RP, P_{\ell b}, \mathcal{D}, k_r, k_p)$ is an (α, β)-* User Authorization Query *instance for which Reduction Rules 1, 2, 3, and 4, and Branching Rule 1 are not applicable. If $|P_{\ell b}| > \beta k_r^\alpha + k_r^{\alpha-1} + \cdots + k_r^2 + k_r$, then $\mathcal{I}$ is a no-instance.*

Proof. Let $\mathcal{I} = (R, P, RP, P_{\ell b}, \mathcal{D}, k_r, k_p)$ be an irreducible yes instance but $|P_{\ell b}| > \beta k_r^\alpha + k_r^{\alpha-1} + \cdots + k_r^2 + k_r$. Suppose that $S \subseteq R$ is an arbitrary set such that $|S| \leq k_r$ and it satisfies all the SoD constraints. Since Reduction Rule 3 is not applicable to $\mathcal{I}$,

$$\left| \bigcup_{s \in S} P(s) \cap P_{\ell b} \right| \leq \beta k_r^\alpha + k_r^\alpha + k_r^{\alpha-1} + \cdots + k_r^2 + k_r.$$

Then, S is not authorized for all permissions in $P_{\ell b}$. This contradicts the fact that S is a feasible solution to $\mathcal{I}$. Therefore, $\mathcal{I}$ is a no-instance. □

Observe that we have not yet used any special characteristics of the set $\mathcal{D}$ of SoD constraints. Hence, Lemma 7 holds true for all classes of SoD constraints. Thus we obtain the following result.

Theorem 5. *The preprocessing described above runs in time $O^*(\alpha^{k_r})$ and for an input instance of (α, β)-UAQ, outputs an equivalent instance of (α, β)-UAQ such that $|P_{\ell b}| = O(\beta k_r^\alpha)$.*

Proof. Observe that all the reduction rules above can be implemented in polynomial time. For every $q \in [\alpha - 2]$, Branching Rule 1.q is applied only when for all $x \in [q - 1]$, Branching Rule 1.x is not applicable. This branching rule is applied on a set L with at most $\alpha - 1$ vertices. From Lemma 5, if the precondition to Branching Rule 1 is satisfied, then L has nonempty intersection with a solution of size at most k_r. Hence, this branching rule provides a $(\alpha - 1)$-way branching with depth at most k_r. Hence, the number of leaves in this bounded search tree is $O^*(\alpha^{k_r})$.

We first apply Reduction Rules 1, 2, Branching Rule 1, and Reduction Rules 3, and 4 in sequence. When these reduction and branching rules are no longer applicable, we check whether $|P_{\ell b}| > \beta k_r^\alpha + k_r^\alpha + k_r^{\alpha-1} + \cdots + k_r^2 + k_r$. If $|P_{\ell b}| > \beta k_r^\alpha + k_r^\alpha + k_r^{\alpha-1} + \cdots + k_r^2 + k_r$, then we use Lemma 7 to output that the given input instance is a no-instance. Otherwise, $|P_{\ell b}| \leq \beta k_r^\alpha + k_r^\alpha + k_r^{\alpha-1} + \cdots + k_r^2 + k_r$.

Hence, the reduced instance has $|P_{\ell b}| = O(\beta k_r^\alpha)$. Since α is fixed, the input instance can be transformed into an equivalent instance in $O^*(\alpha^{k_r})$ time. This completes the proof. □

5.2 Encoding Candidate Solutions as a Matroid

This section consists of two parts: in Section 5.2.1, we provide terminology, notation and results on matroids and representative families, which are necessary for describing and analyzing our DP algorithm; and in Section 5.2.2, we construct a matroid encoding role sets that do not violate SoD constraints. We will call this matroid a *constraint satisfaction matroid (CSM)*. This constraint satisfaction matroid is used in our DP algorithm.

5.2.1 Matroids and Representative Families.

Definition 1. *A family of sets $\mathcal{I}$ over a finite universe U is called a* matroid *if it satisfies the following three axioms:*

(1) $\emptyset \in \mathcal{I}$,
(2) *if $A \in \mathcal{I}$ and $B \subseteq A$, then $B \in \mathcal{I}$, and*
(3) *if $A, B \in \mathcal{I}$ such that $|A| < |B|$, then there exists $x \in B \setminus A$ such that $A \cup \{x\} \in \mathcal{I}$.*

Let U be a universe of n elements, and let r be an integer such that $r \leq n$. Then it is not hard to verify that $(U, \mathcal{F})$ where $\mathcal{F} = \{A \subseteq U : |A| \leq r\}$ satisfies the axioms of Definition 1. Hence, $(U, \mathcal{F})$ is a matroid; $(U, \mathcal{F})$ is called a *uniform matroid*.

For a matroid $M = (U, \mathcal{I})$, any set $A \in \mathcal{I}$ is called an *independent set*. It follows from Definition 1 that all maximal independent sets of a matroid M have the same size, denoted $rank(M)$, and called the *rank* of M. Clearly, the rank of the uniform matroid $(U, \mathcal{F}) = \{A \subseteq U : |A| \leq r\}$ is r.

A matroid $M = (U, \mathcal{I})$ is said to be *representable over a field* $\mathbb{F}$ if there is a matrix $\hat{M}$ over $\mathbb{F}$ and a bijection $\pi : U \rightarrow col(\hat{M})$, where $col(\hat{M})$ is the set of columns of $\hat{M}$, such that $A \subseteq U$ is an independent set in M if and only if $\{\pi(a) : a \in A\}$ is linearly independent over $\mathbb{F}$. Clearly, the rank of M is the rank of the matrix $\hat{M}$. A matroid representable over a field $\mathbb{F}$ is a *linear matroid* over $\mathbb{F}$. A uniform matroid with U of size n can be represented over any field $GF(p)$ for $p > n$ (see e.g. [8]).

Definition 2. *Let* $M_1 = (U_1, \mathcal{I}_1), M_2 = (U_2, \mathcal{I}_2), \ldots, M_t = (U_t, \mathcal{I}_t)$ *be a collection of matroids such that for every* $i \neq j$, $U_i \cap U_j = \emptyset$. *Then, the* direct sum $M = M_1 \oplus M_2 \oplus \cdots \oplus M_t$ *of these matroids is a matroid* $M = (U, \mathcal{I})$ *such that* $U = U_1 \cup U_2 \cup \cdots \cup U_t$, *and for every subset* S *of* U, $S \in \mathcal{I}$ *if and only if for all* $i \in [t]$, $S \cap U_i \in \mathcal{I}_i$.

A *partition matroid* *is a matroid formed from a direct sum of uniform matroids.*

We use the following definitions and results [12, 16] to prove Theorem 6 in Section 5.3.

Definition 3. *Let* $M = (U, \mathcal{I})$ *be a matroid and* $\mathcal{A}$ *be a family of independent sets of size* p *in* M. *For sets* $A, B \subseteq U$, *we say that* A *fits* B *if* $A \cap B = \emptyset$, *and* $A \cup B \in \mathcal{I}$.

A subfamily $\hat{\mathcal{A}} \subseteq \mathcal{A}$ *is a* q-representative family *of* $\mathcal{A}$ *if the following holds: for every* $B \subseteq U$ *with* $|B| \leq q$, *there is an* $A \in \mathcal{A}$ *such that* A *fits* B *if and only if there is an* $\hat{A} \in \hat{\mathcal{A}}$ *such that* $\hat{A}$ *fits* B. *We write* $\hat{\mathcal{A}} \subseteq_{rep}^q \mathcal{A}$ *to denote that* $\hat{\mathcal{A}}$ *is a* q-representative family of $\mathcal{A}$.

Informally, a family of sets $\mathcal{A}$ that fits $\{B \subseteq U : |B| = q\}$ provides a way of encoding all sets of cardinality q in a matroid. Thus, a q-representative family is a compact method of encoding all such sets. Our dynamic programming algorithm makes use of q-representative families and the following results [12, 16] to reduce the number of entries in the DP table and ensure our algorithm is FPT.

Lemma 8. *Let* $M = (U, \mathcal{I})$ *be a matroid and* $\mathcal{F} \subseteq \mathcal{I}$ *such that for every* $A \in \mathcal{F}$, $|A| = p$. *If* $\mathcal{F}_1 \subseteq_{rep}^q \mathcal{F}$ *and* $\mathcal{F}_2 \subseteq_{rep}^q \mathcal{F}_1$, *then* $\mathcal{F}_2 \subseteq_{rep}^q \mathcal{F}$.

Lemma 9. *Let* $M = (U, \mathcal{I})$ *be a linear matroid of rank* n. *Suppose that* M *can be represented by an* $n \times |U|$-matrix $\hat{M}$ *over a field* $\mathbb{F}$ *such that* $\mathbb{F} = GF(s)$ *or* $\mathbb{F} = \mathbb{Q}$. *Furthermore, let* $\mathcal{F} = \{S_1, \ldots, S_t\}$ *be a family of independent sets in* M, *each of cardinality* p. *Then, there is a deterministic algorithm that computes* $\hat{\mathcal{F}} \subseteq_{rep}^q \mathcal{F}$ *with* $O(\binom{p+q}{p}^2 tp^3 n^2 + t\binom{p+q}{p}^\omega np) + (n + |U|)^{O(1)}$ *field operations over* $\mathbb{F}$ *such that* $|\hat{\mathcal{F}}| \leq \binom{p+q}{p}$ *and* $\omega < 2.37$ *is the matrix multiplication exponent.*

5.2.2 Constraint Satisfaction Matroid for (α, β)-*UAQ.* Let $\mathcal{D}' = \{\langle X_1, t_1 \rangle, \ldots, \langle X_m, t_m \rangle\}$ be the collection of all SoD constraints in the reduced instance. Recall that for all $i, j \in [m]$ with $i \neq j$, $X_i \cap X_j = \emptyset$. For every $i \in [m]$, define $M_i(\mathcal{D}') = (X_i, \mathcal{I}_i)$, where $\mathcal{I}_i = \{A \subseteq X_i : |A| \leq t_i - 1\}$. Observe that $M_i(\mathcal{D}')$ is a uniform matroid of rank $t_i - 1$ and each element of the matroid is a set of roles (in X_i) to which a single user could be assigned.

Let $\{r_{m+1}, r_{m+2}, \ldots, r_\delta\}$ be the set of roles that do not appear in any X_i. (That is, $\{\{r_{m+1}, \ldots, r_\delta\} \cup X_1 \cup \cdots \cup X_m = R$.) For r_j, $m + 1 \leq j \leq \delta$, we create a constraint $\langle X_j, t_j \rangle$ such that $X_j = \{r_j\}$ and $t_j = 2$, and construct the uniform matroid $M_j(\mathcal{D}') = (X_j, \{\emptyset, \{r_j\}\})$.

Now let $M(\mathcal{D}') = M_1(\mathcal{D}') \oplus \cdots \oplus M_\delta(\mathcal{D}')$. By construction, $M(\mathcal{D}') = (R, \mathcal{I})$ is a partition matroid, where $\mathcal{I} = \{B \subseteq R : |B \cap X_i| \leq t_i - 1, i \in [\delta]\}$. Notice that each set in this matroid is a set of roles that could be assigned to a single user (since the constraint sets are assumed to be disjoint).

It is known that a partition matroid $M = (U, \mathcal{I})$ is linear and that it possible to compute a $(|U| \times |U|)$-matrix that represents a partition matroid over $GF(p)$ for any $p > |U|$ in time polynomial in $|U|$ [8]. Hence, we have the following result.

Lemma 10. *The matroid $M(\mathcal{D}') = (R, \mathcal{I})$ is a linear matroid of rank at most $|R|$. An $|R| \times |R|$-matrix $\hat{M}$ over $GF(p)$ for some $p > |R|$ representing $M(\mathcal{D}')$ can be constructed in time polynomial in $|R|$.*

We call $M(\mathcal{D}')$ the constraint satisfaction matroid (CSM) for $\mathcal{D}'$. Observe that $M(\mathcal{D}')$ is represented by an $(|R| \times |R|)$-matrix. This matrix provides a compact representation of all members in $\mathcal{I}$: a subset of roles $R' \in \mathcal{I}$ if and only if the columns in $M(\mathcal{D}')$ representing the elements of R' are linearly independent. Hence, given an arbitrary $R' \subseteq R$, there exists an algorithm that runs in $O(|R|^{O(1)})$ time and correctly outputs whether $R' \in \mathcal{I}$ or not (see [8]).

5.3 Algorithm Description and Analysis

The *whole algorithm* for (α, β)-UAQ starts from the preprocessing described in Section 5.1 and returns a partial solution R_1 or concludes that the input instance is a no-instance. If R_1 is returned, then the algorithm constructs a constraint satisfaction matroid as in Section 5.2, and then uses the DP algorithm described below to produce a partial solution R_2 or concludes that the input instance is a no-instance. If R_2 is returned, then the whole algorithm returns $R_1 \cup R_2$. We prove correctness of the whole algorithm and evaluate its running time in Theorem 6.

Our algorithm considers possible sets of extra permissions that could be included in a solution. Such a set must be a subset of $P \setminus P_{\ell b}$ and have cardinality no greater than k_p. Accordingly, we define $P_{good} = \{Y \subseteq P \setminus P_{\ell b} : |Y| \leq k_p\}$.

Let $W \subseteq P_{\ell b}, Y \in P_{good}$ and $0 \leq i \leq k_r$. We define the following set:

$$\mathcal{B}[W, Y, i] = \{R' \subseteq R \mid W \subseteq P(R') \subseteq P_{\ell b} \cup Y, |R'| = i, R' \in \mathcal{I}\},$$

where $\mathcal{I}$ is the family of independent sets of $M(\mathcal{D}')$.

Note that if there exist $i \in \{0, \ldots, k_r\}$ and $Y \in P_{good}$ such that $\mathcal{B}[P_{\ell b}, Y, i]$ is non-empty, then any member of $\mathcal{B}[P_{\ell b}, Y, i]$ provides a solution to the reduced (by the preprocessing) problem. Let $\hat{\mathcal{B}}[P_{\ell b}, Y, i] \subseteq_{rep}^{k_r - i} \mathcal{B}[P_{\ell b}, Y, i]$. Observe that by Definition 3, $\hat{\mathcal{B}}[P_{\ell b}, Y, i] \neq \emptyset$ if and only if $\mathcal{B}[P_{\ell b}, Y, i] \neq \emptyset$.

The following observation is not hard to verify.

Proposition 1. *Let $Y \in P_{good}$ and $W \subseteq P_{\ell b}$. Then, the following statements hold true for $i = 0$:*

- *if $W = \emptyset$, then $\hat{\mathcal{B}}[W, Y, 0] = \mathcal{B}[W, Y, 0] = \{\emptyset\}$;*
- *if $W \neq \emptyset$, then $\hat{\mathcal{B}}[W, Y, 0] = \mathcal{B}[W, Y, 0] = \emptyset$.*

PROOF. If $W = \emptyset$, then $\mathcal{B}[W, Y, 0] = \{\emptyset\}$ (since no role is required to authorize an empty set of permissions). Hence, from Definition 3, we obtain that $\hat{\mathcal{B}}[W, Y, 0] = \{\emptyset\}$. If $W \neq \emptyset$, then at least one role is required to authorize the permission set W. Therefore, $\mathcal{B}[W, Y, 0] = \emptyset$. Hence, by Definition 3, we obtain that $\hat{\mathcal{B}}[W, Y, 0] = \emptyset$. □

We now describe our DP algorithm.

(1) First, for all $\emptyset \neq W \subseteq P_{\ell b}$ and $Y \in P_{good}$, we initialize $\hat{\mathcal{B}}[\emptyset, Y, 0] = \{\emptyset\}$ and $\hat{\mathcal{B}}[W, Y, 0] = \emptyset$.
(2) Then, for all $i = 1, 2, \ldots, k_r$, for every $Y \in P_{good}$ and for every $W \subseteq P_{\ell b}$, we compute $\hat{\mathcal{B}}[W, Y, i]$ as follows:
   (a) if for all $r \in R$ such that $P(r) \subseteq Y \cup P_{\ell b}$, we have $\hat{\mathcal{B}}[W \setminus P(r), Y, i - 1] = \emptyset$, then we set $\hat{\mathcal{B}}[W, Y, i] = \emptyset$;

(b) otherwise there exists $r \in R$ such that $P(r) \subseteq P_{\ell b} \cup Y$ and $\hat{\mathcal{B}}[W \setminus P(r), Y, i-1] \neq \emptyset$. Then, we compute $\mathcal{X}[W, Y, i]$ as

$$\{A \cup \{r\} \mid P(r) \subseteq P_{\ell b} \cup Y, \hat{\mathcal{B}}[W \setminus P(r), Y, i-1] \neq \emptyset, A \in \hat{\mathcal{B}}[W \setminus P(r), Y, i-1], r \in R \setminus A\} \cap \mathcal{I} \quad (1)$$

and $\hat{\mathcal{B}}[W, Y, i]$ as $\hat{\mathcal{B}}[W, Y, i] \subseteq_{rep}^{k_r - i} \mathcal{X}[W, Y, i]$.

If $\hat{\mathcal{B}}[P_{\ell b}, Y, i] \neq \emptyset$ for some $Y \in P_{good}$ then return any role set in $\hat{\mathcal{B}}[P_{\ell b}, Y, i]$ as R_2 and halt.
(3) Return "no-instance".

Note that $\mathcal{X}[W, Y, i]$ could contain up to $|R|\binom{k_r}{i-1}$ sets. But by Lemma 9, the size of $\hat{\mathcal{B}}[W, Y, i]$ is at most just $\binom{k_r}{i}$. So, we invoke Lemma 9, and store $\hat{\mathcal{B}}[W, Y, i]$ such that $\hat{\mathcal{B}}[W, Y, i] \subseteq_{rep}^{k_r - i} \mathcal{X}[W, Y, i]$ (informally, $\hat{\mathcal{B}}[W, Y, i]$ is a $(k_r - i)$-representative family of $\mathcal{X}[W, Y, i]$).

Lemma 11. *For every $W \subseteq P_{\ell b}$, $Y \in P_{good}$ and $0 \leq i \leq k_r$, we have $\hat{\mathcal{B}}[W, Y, i] \subseteq_{rep}^{k_r - i} \mathcal{B}[W, Y, i]$.*

PROOF. Let $W \subseteq P_{\ell b}$, and $Y \in P_{good}$. We prove this lemma by induction on i.

Base Case: The case $i = 0$ holds true due to Proposition 1.

Induction Hypothesis: Let $i \geq 1$, and assume by induction hypothesis that for all $j < i$, and for all $W' \subseteq W$, $\hat{\mathcal{B}}[W', Y, j] \subseteq_{rep}^{k_r - j} \mathcal{B}[W', Y, j]$.

Induction Step: Let $i \geq 1$. Recall that by the DP algorithm description $\hat{\mathcal{B}}[W, Y, i] \subseteq_{rep}^{k_r - i} \mathcal{X}[W, Y, i]$. Thus, by Lemma 8, if $\mathcal{X}[W, Y, i] \subseteq_{rep}^{k_r - i} \mathcal{B}[W, Y, i]$ holds true then $\hat{\mathcal{B}}[W, Y, i] \subseteq_{rep}^{k_r - i} \mathcal{B}[W, Y, i]$. Therefore, in the rest of the proof it suffices to show that $\mathcal{X}[W, Y, i] \subseteq_{rep}^{k_r - i} \mathcal{B}[W, Y, i]$.

Let $X_1 = \{r_1, \ldots, r_i\} \in \mathcal{B}[W, Y, i]$ and $X_1' = X_1 \setminus \{r_i\}$. Let X_2 be a set of at most $k_r - i$ roles such that $X_1 \cap X_2 = \emptyset$, and $X_1 \cup X_2 \in \mathcal{I}$ and let $X_2^* = X_2 \cup \{r_i\}$. Since r_i authorizes $P(r_i)$, and X_1' authorizes all permissions in $W \setminus P(r_i)$ and $P(X_1') \setminus P_{\ell b} \subseteq Y$, we have that $X_1' \in \mathcal{B}[W \setminus P(r_i), Y, i-1]$.

By induction hypothesis, $\hat{\mathcal{B}}[W \setminus P(r_i), Y, i-1] \subseteq_{rep}^{k_r - i + 1} \mathcal{B}[W \setminus P(r_i), Y, i-1]$. Hence, by Definition 3, there exists $X_1^* \in \hat{\mathcal{B}}[W \setminus P(r_i), Y, i-1]$ such that $X_1^* \cup X_2^* \in \mathcal{I}$ and $X_1^* \cap X_2^* = \emptyset$. By (1), we have $X_1^* \cup \{r_i\} \in \mathcal{X}[W, Y, i]$. The set of roles $X_1^* \cup \{r_i\}$ is such that $(X_1^* \cup \{r_i\}) \cap X_2 = \emptyset$ and $X_1^* \cup \{r_i\} \cup X_2 \in \mathcal{I}$. Hence, $\mathcal{X}[W, Y, i] \subseteq_{rep}^{k_r - i} \mathcal{B}[W, Y, i]$. This completes the proof. □

Theorem 6. *The whole algorithm solves (α, β)-UAQ in time $O^*(2^{O(k_r^\alpha + \hat{k})})$.*

PROOF. By Lemma 11, we have $\hat{\mathcal{B}}[W, Y, i] \subseteq_{rep}^{k_r - i} \mathcal{B}[W, Y, i]$. Hence, the algorithm correctly computes partial solutions and if there exists $R_2 \in \hat{\mathcal{B}}[P_{\ell b}, Y, i]$ for some $Y \in P_{good}$ and for some $0 \leq i \leq k_r$, then $R_1 \cup R_2$ is a solution to the UAQ instance. (If no such R_2 exists for any appropriate Y then there is no solution to the instance.) Thus, the algorithm is correct.

The time taken to compute a solution is determined by the running times of the pre-processing phase ($O(\alpha^{k_r})$), the computation of the matroid (polynomial in $|R|$) and the dynamic programming phase.

From Theorem 5, we have $|P_{\ell b}|$ is $O(k_r^\alpha)$. Hence, there are $2^{|P_{\ell b}|} = 2^{O(k_r^\alpha)}$ subsets of $P_{\ell b}$ and there are at most $2^{\hat{k}}$ subsets in $P \setminus P_{\ell b}$. Hence, there are $O^*(2^{O(k_r^\alpha + \hat{k})})$ sets of the form $\mathcal{B}[W, Y, i]$. By (1),

$$|\mathcal{X}[W, Y, i]| \leq |R| \max_{r \in R} |\hat{\mathcal{B}}[W \setminus P(r), Y, i-1]| = O(2^{k_r}|R|).$$

Computing $\mathcal{X}[W, Y, i]$ takes time polynomial in $|\mathcal{X}[W, Y, i]|$ and thus time $O^*(2^{O(k_r)})$.

We then compute $\hat{\mathcal{B}}[W, Y, i]$ such that $\hat{\mathcal{B}}[W, Y, i] \subseteq_{rep}^{k_r - i} \mathcal{X}[W, Y, i]$ and store it in $\hat{\mathcal{B}}[W, Y, i]$. Recall that from Lemma 10, our matroid is represented by an $|R| \times |R|$-matrix. By Lemma 9, computing

$\hat{\mathcal{B}}[W, Y, i]$ takes time

$$O^*(2^{O(\omega k_r)}|\mathcal{X}[W, Y, i]|(|R| + i)^{O(1)}) = O^*(2^{O(k_r)}).$$

Hence, computing every table entry takes time

$$O^*(2^{O(k_r^\alpha + \hat{k})}2^{O(k_r)}) = O^*(2^{O(k_r^\alpha + \hat{k})}).$$

Therefore, we can complete the dynamic programming phase (and hence the whole algorithm) in $O^*(2^{O(k_r^\alpha + \hat{k})})$ time. □

6 RELATED WORK

The study of User Authorization Query began with work by Du and Joshi [11] in the context of inter-domain role mapping. They showed that finding a minimal set of roles R' such that $P(R') = P_{\ell b}$ is NP-hard via a reduction from Minimal Set Cover and proposed a polynomial-time algorithm for computing approximate solutions. Crampton and Chen showed that several versions of UAQ, including ones where $P(R')$ may be a superset of P_{lb} were NP-hard. This early work did not consider SoD constraints.

Wickramaarachchi et al [24, 25] extended the definition of UAQ to include SoD constraints and developed exact algorithms to solve UAQ, based on techniques used to solve CNF-SAT and MAXSAT. Armando *et al.* and Lu *et al.* made improvements to these algorithms [2, 17, 18]. Recent work has provided a comprehensive comparative analysis of methods for solving User Authorization Query and developed a set of benchmarks for evaluating User Authorization Query solvers [1]. These results suggest that a reduction of User Authorization Query to PMaxSat is currently the most effective way of solving User Authorization Query.

The focus of the above work was finding approximate and exact algorithms to solve User Authorization Query. Mousavi and Tripunitara were the first to consider the parameterized complexity of UAQ [22], and included constraints in the specification of the problem. They showed the problem of deciding whether an instance has a solution is FPT if P_{ub} is the small parameter, essentially by considering all subsets of P_{ub}. While it may be reasonable in certain cases, in general P_{ub} is not necessarily small.

In summary, existing work on User Authorization Query has mainly attempted to exploit existing algorithms for related problems in order to solve User Authorization Query, without attempting to understand the inherent difficulty of User Authorization Query. In particular, there has been little effort to better understand the complexity of User Authorization Query in terms of each of its parameters. The exception to this is the work of Mousavi, which does explore how the complexity of the problem is affected by the different parameters, although most of this work used traditional methods of complexity analysis [21].

7 CONCLUDING REMARKS

Our work provides the first thorough attempt to investigate User Authorization Query using multi-variate complexity analysis. Our results suggest that it may be difficult to obtain a practical FPT algorithm for User Authorization Query in general. However, we have also shown that if an RBAC configuration satisfies certain properties then we may be able to use an FPT algorithm to solve instances of User Authorization Query for that configuration.

One surprising conclusion of our work is the sharp contrast in FPT results for the workflow satisfiability problem (WSP) and User Authorization Query. Informally, given a set of tasks T, a set of users U, an authorization relation $A \subseteq U \times T$ (where u is authorized to perform t if and only if $(u, t) \in A$), and a set of constraints C, an instance of WSP asks whether there exists a mapping $\pi : U \to T$ such that all constraints in C are satisfied and $(\pi(t), t) \in A$ for all t. WSP constraints

can, for example, require that the same user is not assigned to two particular tasks (a simple form of separation of duty), although considerably more complex constraints are possible.

It is relatively easy to show that WSP is NP-hard, even when constraints are limited to the simple separation of duty constraints described above [23]. Nevertheless, subsequent research has shown that WSP is FPT (when the number of tasks is the small parameter) for all user-independent constraints [5], which include the aforementioned simple separation of duty constraints as well as most other constraints that are likely to arise in practice. Moreover, the FPT algorithms for WSP are not just of theoretical interest. Experimental evaluations have shown that these algorithms provide a more efficient solution for WSP than brute force algorithms and methods using SAT solvers [5, 6, 15].

On the face of it, WSP appears to be more complex than User Authorization Query, not least because the constraints in a WSP instance may be much more varied than those appearing in a UAQ instance. And both problems require us to compute a solution set that is constrained by a binary relation (the RPG in User Authorization Query and the authorization relation in WSP) and a set of constraints. However, our results show that User Authorization Query remains a hard problem for many RBAC configurations.

Informally, the source of the complexity seems to arise from the consequences of choosing a particular element in a potential solution. In the case of WSP, choosing a user to perform a specific task only means that we have to check attempts to allocate that user to other tasks, in order to determine whether such an allocation would violate a constraint. This means that we can compute all partitions of the set of steps such that each block in the partition could be assigned to a particular user; roughly speaking, this is the basis of the FPT algorithms for WSP. In contrast, selecting a role, so that a particular permission in $P_{\ell b}$ is activated, means (i) that potentially many other permissions may be simultaneously activated, and (ii) other roles may become ineligible for consideration because of the SoD constraints.

It would be convenient if $|R|$ were the small parameter. We could compute a solution to User Authorization Query simply by considering all possible subsets of R. But the application of Reduction Rule 0 only eliminates roles that are assigned to a permission outside P_{ub}. There is no reason to assume that the size of the role set after these roles have been eliminated will be small, given that multiple roles may be assigned to the same permission in $P_{ub} \setminus P_{\ell b}$.

Although the results obtained in this paper are mainly negative, and those that are positive require strong restrictions on the User Authorization Query instances, we believe that the work provides useful insights into the difficulty of solving User Authorization Query. In particular, we believe these results supplement the recent work of Armando et al. [1] and may provide useful input into evaluating User Authorization Query solvers and producing new benchmarks for User Authorization Query.

It is well-known that practitioners prefer to use general-purpose solvers for solving practical problems rather than specialised algorithms. It has been shown in the literature, see e.g. [1] for UAQ and [15] for the Workflow Satisfiability Problem, that appropriately chosen general-purpose solvers perform reasonably well on moderate-size instances of tractable problems. Thus, it would be interesting to see whether the state-of-the-art PMaxSAT solver used in [1] performs well on instances of the UAQ problem of Section 5.

REFERENCES

[1] Alessandro Armando, Giorgia Gazzarata, and Fatih Turkmen. 2020. Benchmarking UAQ Solvers. In *25th ACM Symposium on Access Control Models and Technologies, SACMAT 2020, Proceedings*. ACM, 145–152.

[2] Alessandro Armando, Silvio Ranise, Fatih Turkmen, and Bruno Crispo. 2012. Efficient run-time solving of RBAC user authorization queries: Pushing the envelope. In *Second ACM Conference on Data and Application Security and Privacy, CODASPY 2012*. ACM, 241–248. https://doi.org/10.1145/2133601.2133631

[3] Clara Bertolissi, Daniel Ricardo dos Santos, and Silvio Ranise. 2018. Solving Multi-Objective Workflow Satisfiability Problems with Optimization Modulo Theories Techniques. In *Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies, SACMAT 2018, Indianapolis, IN, USA, June 13-15, 2018*, Elisa Bertino, Dan Lin, and Jorge Lobo (Eds.). ACM, 117–128. https://doi.org/10.1145/3205977.3205982

[4] Liang Chen and Jason Crampton. 2009. Set Covering Problems in Role-Based Access Control. In *Computer Security - ESORICS 2009, 14th European Symposium on Research in Computer Security, Saint-Malo, France, September 21-23, 2009. Proceedings (Lecture Notes in Computer Science, Vol. 5789)*, Michael Backes and Peng Ning (Eds.). Springer, 689–704.

[5] David Cohen, Jason Crampton, Andrei Gagarin, Gregory Gutin, and Mark Jones. 2014. Iterative Plan Construction for the Workflow Satisfiability Problem. *J. Artif. Intell. Res.* 51 (2014), 555–577. https://doi.org/10.1613/jair.4435

[6] David A. Cohen, Jason Crampton, Andrei Gagarin, Gregory Gutin, and Mark Jones. 2016. Algorithms for the workflow satisfiability problem engineered for counting constraints. *J. Comb. Optim.* 32, 1 (2016), 3–24.

[7] Jason Crampton, Andrei Gagarin, Gregory Gutin, Mark Jones, and Magnus Wahlström. 2016. On the Workflow Satisfiability Problem with Class-Independent Constraints for Hierarchical Organizations. *ACM Trans. Priv. Secur.* 19, 3 (2016), 8:1–8:29. https://doi.org/10.1145/2988239

[8] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. 2015. *Parameterized Algorithms*. Springer.

[9] R.G. Downey and M.R. Fellows. 1999. *Parameterized Complexity*. Springer. https://doi.org/10.1007/978-1-4612-0515-9

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

[11] Siqing Du and James B. D. Joshi. 2006. Supporting authorization query and inter-domain role mapping in presence of hybrid role hierarchy. In *11th ACM Symposium on Access Control Models and Technologies, SACMAT 2006, Lake Tahoe, California, USA, June 7-9, 2006, Proceedings*, David F. Ferraiolo and Indrakshi Ray (Eds.). ACM, 228–236.

[12] Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. 2016. Efficient Computation of Representative Families with Applications in Parameterized and Exact Algorithms. *J. ACM* 63, 4 (2016), 29:1–29:60.

[13] Gregory Gutin and Daniel Karapetyan. 2020. Constraint Branching in Workflow Satisfiability Problem. In *Proceedings of the 25th ACM on Symposium on Access Control Models and Technologies, SACMAT 2020*. 93–103.

[14] Ashwin Jacob, Diptapriyo Majumdar, and Venkatesh Raman. 2019. Parameterized Complexity of Conflict-Free Set Cover. In *Computer Science - Theory and Applications - 14th International Computer Science Symposium in Russia, CSR 2019, Novosibirsk, Russia, July 1-5, 2019, Proceedings*. 191–202.

[15] Daniel Karapetyan, Andrew J. Parkes, Gregory Z. Gutin, and Andrei Gagarin. 2019. Pattern-Based Approach to the Workflow Satisfiability Problem with User-Independent Constraints. *J. Artif. Intel. Res.* 66 (2019), 85–122.

[16] Daniel Lokshtanov, Pranabendu Misra, Fahad Panolan, and Saket Saurabh. 2018. Deterministic Truncation of Linear Matroids. *ACM Trans. Algorithms* 14, 2 (2018), 14:1–14:20.

[17] Jianfeng Lu, Jianmin Han, Wei Chen, and JinWei Hu. 2012. Safety and availability checking for user authorization queries in RBAC. *International Journal of Computational Intelligence Systems* 5, 5 (2012), 860–867.

[18] Jianfeng Lu, James B. D. Joshi, Lei Jin, and Yiding Liu. 2015. Towards complexity analysis of User Authorization Query problem in RBAC. *Computers & Security* 48 (2015), 116–130.

[19] Jonathan D. Moffett. 1998. Control Principles and Role Hierarchies. In *Proceedings of the Third ACM Workshop on Role-Based Access Control* (Fairfax, Virginia, USA). Association for Computing Machinery, New York, NY, USA, 63–69. https://doi.org/10.1145/286884.286900

[20] Jonathan D. Moffett and Emil C. Lupu. 1999. The Uses of Role Hierarchies in Access Control. In *Proceedings of the Fourth ACM Workshop on Role-Based Access Control* (Fairfax, Virginia, USA). Association for Computing Machinery, New York, NY, USA, 153–160. https://doi.org/10.1145/319171.319186

[21] Nima Mousavi. 2014. *Algorithmic Problems in Access Control*. Ph.D. Dissertation. University of Waterloo.

[22] Nima Mousavi and Mahesh V. Tripunitara. 2012. Mitigating the Intractability of the User Authorization Query Problem in Role-Based Access Control (RBAC). In *Network and System Security - 6th International Conference, NSS 2012. Proceedings (Lecture Notes in Computer Science, Vol. 7645)*, Li Xu, Elisa Bertino, and Yi Mu (Eds.). Springer, 516–529.

[23] Qihua Wang and Ninghui Li. 2010. Satisfiability and Resiliency in Workflow Authorization Systems. *ACM Trans. Inf. Syst. Secur.* 13, 4 (2010), 40. https://doi.org/10.1145/1880022.1880034

[24] Guneshi T. Wickramaarachchi, Wahbeh H. Qardaji, and Ninghui Li. 2009. An efficient framework for user authorization queries in RBAC systems. In *14th ACM Symposium on Access Control Models and Technologies, SACMAT 2009, Stresa, Italy, June 3-5, 2009, Proceedings*, Barbara Carminati and James Joshi (Eds.). ACM, 23–32.

[25] Yue Zhang and James B. D. Joshi. 2008. UAQ: A framework for user authorization query processing in RBAC extended with hybrid hierarchy and constraints. In *13th ACM Symposium on Access Control Models and Technologies, SACMAT 2008, Estes Park, CO, USA, June 11-13, 2008, Proceedings*, Indrakshi Ray and Ninghui Li (Eds.). ACM, 83–92. https://doi.org/10.1145/1377836.1377850