Set Covering Problems in Role-Based Access Control

Liang Chen and Jason Crampton

Information Security Group and Department of Mathematics
Royal Holloway, University of London
{l.chen-2,jason.crampton}@rhul.ac.uk

Abstract. Interest in role-based access control has generated considerable research activity in recent years. A number of interesting problems related to the well known set cover problem have come to light as a result of this activity. However, the computational complexity of some of these problems is still not known. In this paper, we explore some variations on the set cover problem and use these variations to establish the computational complexity of these problems. Most significantly, we introduce the minimal cover problem – a generalization of the set cover problem – which we use to determine the complexity of the inter-domain role mapping problem.

1 Introduction

Role-based access control (RBAC) has been the subject of considerable research in recent years [1,2] and is widely accepted as an alternative to traditional discretionary and mandatory access controls. A number of commercial products, such as Windows Authorization Manager and Oracle 9, implement some form of RBAC.

A number of interesting computational problems arise in the context of RBAC:

- the inter-domain role mapping (IDRM) problem [3,4],
- the user authorization query (UAQ) problem [5,6],
- the enforceability of static separation of duty constraints [7], and
- the generation of role-based static separation of duty (RSSoD) constraints [7].

However, existing work does not always pose the most appropriate problem (as in the IDRM problem of Du and Joshi [3]) or does not determine the computational complexity of the problem (instead presenting either approximate [4] or exhaustive algorithms to compute a solution [7]). All the above problems appear to be related to the *set cover problem* [8]: the decision version of this problem is NP-complete, while the optimization problem is NP-hard.

In this paper, we examine the connections between problems in RBAC and the set cover problem. Our most important contribution is to define the *minimal cover problem* – a generalization of the set cover problem – and use this

M. Backes and P. Ning (Eds.): ESORICS 2009, LNCS 5789, pp. 689–704, 2009.

problem to determine the computational complexity of the IDRM-availability problem [4]. In doing so, we identify some interesting auxiliary problems and establish their computational complexity. We also establish a vocabulary and a suite of techniques for handling similar problems that may arise in the context of RBAC.

In the next section we introduce relevant background material, including a formal model for RBAC and definitions of the set cover problem. Section 3 introduces the minimal cover problem and establishes its relationship to the basic set cover problem, thereby enabling us to derive its computational complexity. In Sect. 4, we discuss applications of our results to RBAC, establishing complexity results for a number of different problems. We also discuss related work in Sect. 4. We conclude the paper with a summary of our results and a discussion of future work.

2 Background

2.1 RBAC

The RBAC96 family of models is undoubtedly the most well known formulation of RBAC [1], and provides the basis for the ANSI RBAC standard [2]. RBAC_0, the simplest RBAC96 model, defines a set of users U, a set of sessions S, a set of roles R, a set of permissions P, a user-role assignment relation $UA \subseteq U \times R$ and a permission-role assignment relation $PA \subseteq P \times R$. A user u is authorized for role r if $(u, r) \in UA$; a role r is authorized for permission p if $(p, r) \in PA$; and u is authorized for p if there exists a role r such that $(u, r) \in UA$ and $(p, r) \in PA$. We represent RBAC_0 *state* as a pair (UA, PA).

RBAC_1 introduces the concept of a *role hierarchy*, which is modeled as a partial order on the set of roles $(R, \leqslant)$. In other words, the role hierarchy is a binary relation $RH \subseteq R \times R$ that is reflexive, anti-symmetric and transitive. The role hierarchy semantics provide an economical way of representing RBAC state. In particular: if $(u, r) \in UA$ and $r \geqslant r'$, then u is (implicitly) authorized for r'; and if $(p, r) \in PA$ and $r \leqslant r'$ then r' is (implicitly) authorized for p.

In this paper we will assume that the role hierarchy has been "flattened" by encoding all authorized relationships in the user-role and permission-role relations, so that RBAC state is simply represented by the RBAC_0 state (UA, PA). Any RBAC_1 state can be transformed into an equivalent RBAC_0 state (in the sense that precisely the same set of requests are authorized) in polynomial time, using an algorithm based on some appropriate graph traversal algorithm.

We write $\mathsf{Prms}(r, PA)$ to denote the set of permissions for which role $r \in R$ is authorized, and, for $S \subseteq R$, we write $\mathsf{Prms}(S, PA)$ to denote the set of permissions for which the roles in S are collectively authorized. That is,

$$\mathsf{Prms}(r, PA) = \{p \in P : (p, r) \in PA\} \quad \text{and} \quad \mathsf{Prms}(S, PA) = \bigcup_{s \in S} \mathsf{Prms}(s, PA).$$

2.2 The Set Cover Problem

Let X be a finite set and let $\mathcal{C}$ be a collection of subsets of X such that $X = \bigcup_{C \in \mathcal{C}} C$, and let $\mathcal{D} \subseteq \mathcal{C}$. Then we write $U_{\mathcal{D}}$ to denote $\bigcup_{D \in \mathcal{D}} D$. (By definition, $U_{\mathcal{D}} \subseteq X$; in particular, $U_{\mathcal{C}} = X$).

Definition 1. *Let X be a finite set and let $\mathcal{C}$ be a collection of subsets of X such that $U_{\mathcal{C}} = X$. Let $V \subseteq X$. We say $\mathcal{D} \subseteq \mathcal{C}$ is a* cover *of V if $U_{\mathcal{D}} \supseteq V$; $\mathcal{D}$ is a* perfect cover *of V if $U_{\mathcal{D}} = V$.*

The definition above is more general than the usual definition associated with the set cover problem. In particular, our notion of a "perfect cover" is what usually corresponds to a "cover" in the literature. However, in Sect. 3 we will need to be able to distinguish between covers and perfect covers, hence the more general definition.

Clearly, there exists at least one perfect cover of X (namely $\mathcal{C}$). Note that any cover of X is necessarily perfect, since $U_{\mathcal{C}} = X$. There are two natural questions we might ask given X and $\mathcal{C}$:

Problem 1 (The set cover decision problem). For a given integer k, does there exist a perfect cover $\mathcal{D}$ of X such that $|\mathcal{D}| \leqslant k$?

Problem 2 (The set cover optimization problem). What is the smallest integer m for which there exists a perfect cover of X of cardinality m?

The set cover decision problem is NP-complete [8] with respect to the parameter $|\mathcal{C}|$. The set cover optimization problem is NP-hard, because there exists a (trivial) polynomial time Turing reduction from the set cover decision problem to the set cover optimization problem.[1]

3 Variations on the Set Cover Problem

Throughout this section, we assume we are given a universe X and $\mathcal{C}$, a collection of subsets of X. We define an equivalence relation on the powerset of $\mathcal{C}$: $\mathcal{D} \sim \mathcal{D}'$ if and only if $U_{\mathcal{D}} = U_{\mathcal{D}'}$. The equivalence classes defined by $\sim$ give rise to a partition of the powerset of $\mathcal{C}$: the elements of an equivalence class are all subsets of $\mathcal{C}$, and all elements in an equivalence class are perfect covers of the same subset of X. If there exists a perfect cover of $V \subseteq X$ – that is, there exists $\mathcal{D} \subseteq \mathcal{C}$ such that $U_{\mathcal{D}} = V$ – then we write $[V] \subseteq \mathcal{C}$ to denote the equivalence class in which each element of $[V]$ is a perfect cover of V. That is, $[V] = \{\mathcal{D} \subseteq \mathcal{C} : U_{\mathcal{D}} = V\}$.

We write $\mathsf{PCov}(X, \mathcal{C})$ to denote the set of subsets of X for which perfect covers exist in $\mathcal{C}$. Clearly, $(\mathsf{PCov}(X, \mathcal{C}), \subseteq)$ is a partially ordered set. When X and $\mathcal{C}$ are obvious from context, we will simply write PCov for $\mathsf{PCov}(X, \mathcal{C})$.

[1] If we have an oracle that can solve the optimization problem, we can solve the decision problem by checking whether the solution of the associated optimization problem has cardinality less than or equal to k.

Example 1. Let $X = \{1, 2, 3, 4\}$ and let $\mathcal{C} = \{C_1, C_2, C_3, C_4\}$, where $C_1 = \{1\}$, $C_2 = \{2, 4\}$, $C_3 = \{3, 4\}$ and $C_4 = \{1, 2, 4\}$. Then

$$\mathsf{PCov} = \{\{1\}, \{2, 4\}, \{3, 4\}, \{1, 2, 4\}, \{1, 3, 4\}, \{2, 3, 4\}, \{1, 2, 3, 4\}\}$$

and, for example,

$$[\{1, 2, 4\}] = \{\{C_4\}, \{C_1, C_4\}, \{C_2, C_4\}, \{C_1, C_2\}, \{C_1, C_2, C_4\}\}$$
$$[\{1, 3, 4\}] = \{\{C_1, C_3\}\}$$

3.1 The Kernel and Shell

We now introduce the notion of the kernel and shell of V (given X and $\mathcal{C}$). Informally, the kernel of V represents the largest perfect cover contained in V. We shall see that the kernel of V can be computed in polynomial time, a result that has a number of useful applications. The shell identifies those sets that could contribute to a cover of V.

Definition 2. *Let $V \subseteq X$. Define $\mathcal{K}(V) = \{C \in \mathcal{C} : C \subseteq V\}$. Then we call $U_{\mathcal{K}(V)} \subseteq X$ the* kernel *of V (with respect to $\mathcal{C}$).*

For brevity, we write $\mathsf{ker}(V)$, rather than $U_{\mathcal{K}(V)}$, to denote the kernel of V. Note that $\mathsf{ker}(V) \in \mathsf{PCov}$ and $\mathsf{ker}(V) \subseteq V$, by definition. We now state and prove two elementary results.

Proposition 1. *Let $Z \in \mathsf{PCov}$ such that $Z \subseteq V$. Then $Z \subseteq \mathsf{ker}(V)$.*

Proof. Since $Z \in \mathsf{PCov}$, there exists $\mathcal{D} \subseteq \mathcal{C}$ such that $Z = U_{\mathcal{D}}$. For any $C \in \mathcal{D}$, we have $C \subseteq V$ (otherwise, $Z \not\subseteq V$). Hence, $C \in \mathcal{K}(V)$ by definition and hence $\mathcal{D} \subseteq \mathcal{K}(V)$. Therefore $Z = U_{\mathcal{D}} \subseteq U_{\mathcal{K}(V)} = \mathsf{ker}(V)$. □

Proposition 2. *$V \in \mathsf{PCov}$ if and only if $V = \mathsf{ker}(V)$.*

Proof. The result follows immediately if $V = \mathsf{ker}(V)$ since $\mathsf{ker}(V) \in \mathsf{PCov}$. Assume now that $V \in \mathsf{PCov}$. Since $V \subseteq V$, we may apply Proposition 1 to deduce that $V \subseteq \mathsf{ker}(V)$. Hence, we have $V = \mathsf{ker}(V)$, since $\mathsf{ker}(V) \subseteq V$, by definition. □

Corollary 1. *Let $V \subseteq X$. Determining whether $V \in \mathsf{PCov}$ is in P.*

Proof. By Proposition 2, $V \in \mathsf{PCov}$ if and only if $V = \mathsf{ker}(V)$. Clearly, we can check in polynomial time whether $V = \mathsf{ker}(V)$. □

Definition 3. *Let $V \subseteq X$. Define $\mathcal{S}(V) = \{C \in \mathcal{C} : C \cap V \neq \emptyset\}$. Then we call $U_{\mathcal{S}(V)} \subseteq X$ the* shell *of V (with respect to $\mathcal{C}$).*

Similarly, we write $\mathsf{shell}(V)$ to denote the shell of V. Note that $\mathsf{shell}(V) \in \mathsf{PCov}$ and $\mathsf{shell}(V) \supseteq V$, by definition.

3.2 Minimality, Optimality and Irreducibility

Let us assume that $V \notin \mathsf{PCov}$ and consider the problem of finding an "approximation" of V among the members of PCov. (We will formalize the notion of approximation shortly.) The results above suggest that the best "under-approximation" of V is $\mathsf{ker}(V)$. It seems natural to consider "over-approximation" in terms of those elements of PCov that contain V and have minimal cardinality. More formally, we have the following definitions.

Definition 4. *Given X, $\mathcal{C}$ and $V \subseteq X$ such that $V \notin \mathsf{PCov}$, we say*

- $T \in \mathsf{PCov}$ *is a* container *of V if $T \supset V$.*
- $T \in \mathsf{PCov}$ *is a* minimal container *of V if T is a container of V and for any other container T' of V, $|T| \leqslant |T'|$.*[2]

In other words, T is a minimal container of V if it is perfectly covered by some subset of $\mathcal{C}$, contains V, but contains as few elements outside V as possible for a set that is perfectly covered.[3]

Definition 5. *Given X, $\mathcal{C}$ and $V \subseteq X$ such that $V \notin \mathsf{PCov}$, we say*

- $\mathcal{D} \subseteq \mathcal{C}$ *is* irreducible *if for all $\mathcal{D}' \subset \mathcal{D}$, $U_{\mathcal{D}'} \subset U_{\mathcal{D}}$.*
- $\mathcal{D} \subseteq \mathcal{C}$ *is a* minimal cover *of V if $\mathcal{D} \in [T]$ for some minimal container T of V.*
- $\mathcal{D} \in [T]$ *is an* optimal cover *of V if T is a minimal container of V and $\mathcal{D}$ is irreducible.*

Informally, $\mathcal{D}$ is irreducible if there is no redundancy in $\mathcal{D}$: we cannot remove any element of $\mathcal{D}$ without changing $U_{\mathcal{D}}$. Each $T \in \mathsf{PCov}$ is associated with the equivalence class $[T]$, which is a collection of subsets of $\mathcal{C}$. Every member of $[T]$ is a perfect cover of T. If T is a minimal container of V, then every element of $[T]$ is a minimal cover of V. Each such equivalence class contains at least one irreducible element.

Example 2. Using our running example, let $V = \{1, 2, 3\}$. Then a minimal container of $\{1, 2, 3\}$ is $\{1, 2, 3, 4\}$. The irreducible covers in $[\{1, 2, 3, 4\}]$ (and hence optimal covers of $\{1, 2, 3\}$) are $\{C_3, C_4\}$ and $\{C_1, C_2, C_3\}$.

Proposition 3. *Given $\mathcal{D} \subseteq \mathcal{C}$, we can compute $\mathcal{E} \subseteq \mathcal{D}$ such that $\mathcal{E}$ is irreducible and $U_{\mathcal{E}} = U_{\mathcal{D}}$ in polynomial time.*

Proof. Figure 1 illustrates an algorithm called IRR-Gen: on input $\mathcal{D} \subseteq \mathcal{C}$, IRR-Gen returns an irreducible set $\mathcal{E} \subseteq \mathcal{D}$ such that $U_{\mathcal{E}} = U_{\mathcal{D}}$. At the ith iteration,

[2] Equivalently, there does not exist $T' \in \mathsf{PCov}$ such that $T' \supseteq V$ and $|T'| < |T|$.

[3] This is important in the context of RBAC because we want to minimize the number of additional permissions for which a set of roles is authorized for outside some specified set of permissions.

```
Input: 𝒟 ⊆ 𝒞;  Output: ℰ
let ℰ = ∅
while 𝒟 ≠ ∅ {
  choose C ∈ 𝒟
  𝒟 = 𝒟 \ {C}
  if C ⊈ U_{𝒟∪ℰ} then ℰ = ℰ ∪ {C} }
return ℰ
```

Fig. 1. The IRR-Gen algorithm

the algorithm arbitrarily chooses an element C from $\mathcal{D}$, and checks whether the removal of C from $\mathcal{D}$ would affect the set of elements originally covered by $\mathcal{D}$. If it does, C must be included in $\mathcal{E}$, otherwise C can be ignored. The overall time complexity of the IRR-Gen algorithm is polynomial in $|\mathcal{D}|$ and $|X|$. □

Note that IRR-Gen is non-deterministic ("choose $C \in \mathcal{D}$") and $[T]$ may contain more than one irreducible set, so different runs of the algorithm on input $\mathcal{D}$ might return different irreducible sets $\mathcal{E}$ depending on the order in which the elements of $\mathcal{D}$ are processed. Consider $\mathcal{D} = \{C_1, C_2, C_3, C_4\} \in [\{1, 2, 3, 4\}]$, then processing $\mathcal{D}$ in the order C_1, C_2, C_3, C_4, for example, yields $\mathcal{E} = \{C_3, C_4\}$, whereas processing $\mathcal{D}$ in the order C_4, C_3, C_2, C_1 yields $\mathcal{E} = \{C_1, C_2, C_3\}$.

Corollary 2. *Given X, $\mathcal{C}$ and $T \in \mathsf{PCov}$, we can compute an irreducible element of $[T]$ in polynomial time.*

Proof. Since $T \in \mathsf{PCov}$, $T = \mathsf{ker}(T)$ by Proposition 2. Hence, we can compute $\mathcal{K}(T)$ in polynomial time and $\mathcal{K}(T) \in [T]$. Hence, we need to find an irreducible set $\mathcal{D} \subseteq \mathcal{K}(T)$ such that $U_{\mathcal{D}} = T$. This can be done in polynomial time using the IRR-Gen algorithm with input $\mathcal{K}(T)$. □

3.3 The Minimal Cover Problem

The minimal cover problem is fundamental to solving the IDRM problem. We first state an elementary result that enables us to make a useful simplifying assumption.

Proposition 4. *Given X, V and $\mathcal{C}$, define: $X' = X \setminus \mathsf{ker}(V)$; $V' = V \setminus \mathsf{ker}(V)$; and $\mathcal{C}' = \{C \setminus \mathsf{ker}(V) : C \in \mathcal{C}, C \not\subseteq V\}$. Then:*

1. *$U_{\mathcal{C}'} = X'$;*
2. *for all $C' \in \mathcal{C}'$, $C' \not\subseteq V'$;*
3. *if $\mathcal{D}$ is a minimal cover of V', then $D \cup \mathcal{K}(V)$ is a minimal cover of V.*

Proof.

1. Since $\mathcal{C}' = \{C \cap X' : C \in \mathcal{C}, C \not\subseteq V\}$ and $X' = X \setminus \mathsf{ker}(V)$, $U_{\mathcal{C}'} = U_{\mathcal{C}} \cap X' = X \cap X' = X'$.
2. If $C' \in \mathcal{C}'$, then $C' = C \setminus \mathsf{ker}(V)$ for some $C \in \mathcal{C}$ such that $C \not\subseteq V$. Therefore, $C' = C \setminus \mathsf{ker}(V) \not\subseteq V \setminus \mathsf{ker}(V) = V'$.

3. Suppose, in order to obtain a contradiction, that $\mathcal{D} \cup \mathcal{K}(V)$ is not a minimal cover of V. Then, since $\mathcal{K}(V)$ only adds elements from V, $\mathcal{D}$ cannot be a minimal cover of V', which is the desired contradiction. □

Let $V \notin \mathsf{PCov}$ and suppose we are interested in finding a minimal cover of V. Then we may construct (in polynomial time) a new instance of the problem, by replacing X and $\mathcal{C}$ with X' and $\mathcal{C}'$, where $|X| \geqslant |X'|$ and $|\mathcal{C}| \geqslant |\mathcal{C}'|$. In particular, we omit any $C \in \mathcal{C}$ such that

- $C \cap V = \emptyset$ (since any such C cannot contribute to a cover of V);
- $C \subseteq V$ (since, by Proposition 4, we can compute a minimal cover $\mathcal{D}$ of $V \setminus \mathsf{ker}(V)$ to obtain a minimal cover $\mathcal{D} \cup \mathcal{K}(V)$ of V).

Henceforth, we assume that our problem instance is in this "canonical form": that is, $C \cap V \neq \emptyset$ and $C \not\subseteq V$ for all $C \in \mathcal{C}$. We now define a number of problems associated with container, minimal cover and optimal cover.

Problem 3 (The container decision problem). Given $X, \mathcal{C}, V \subseteq X$ and an integer k, does there exist a container T of V such that $|T| \leqslant |V| + k$?

Problem 4 (The container optimization problem). Given $X, \mathcal{C}$ and $V \subseteq X$, find a minimal container of V.

Problem 5 (The minimal cover problem). Given $X, \mathcal{C}$ and $V \subseteq X$, find a minimal cover of V.

Problem 6 (The optimal cover problem). Given $X, \mathcal{C}$ and $V \subseteq X$, find an optimal cover of V.

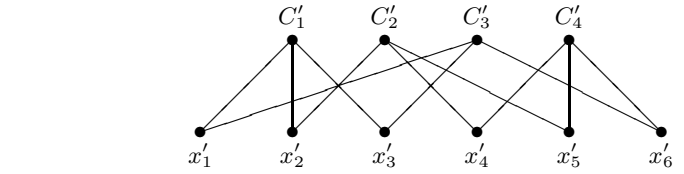
Theorem 1. *The container decision problem is NP-complete.*

Proof. It is easy to see that the container decision problem is in NP, because a nondeterministic algorithm need only guess a subset T of X and check in polynomial time whether $T \supset V$, $\mathsf{ker}(T) = T$ (that is, $T \in \mathsf{PCov}$) and $|T| \leqslant |V| + k$.

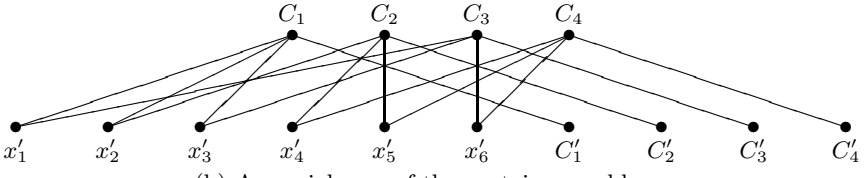
We now show a polynomial time transformation from the set cover decision problem to a special case of the container decision problem. Let $(X', \mathcal{C}', k)$ be an instance of the set cover decision problem. We transform it into an instance $(X, \mathcal{C}, V)$ of a special case of the container decision problem in the following way:

- Let $X = X' \cup \mathcal{C}'$ and $V = X'$;
- Define a collection $\mathcal{C} = \{C' \cup \{C'\} : C' \in \mathcal{C}'\}$.

This transformation is illustrated in Fig. 2. It can be seen that each C contains a single element (namely C') that does not belong to V. Moreover, each C contains at least one element of X', since $C' \in \mathcal{C}'$ can be assumed to be non-empty. In other words, the resulting instance is a special case of the container decision problem (in which each element of $\mathcal{C}$ contains precisely one distinct element that is not in V).



(a) A set cover problem



(b) A special case of the container problem

Fig. 2. Correspondence between the set cover and container problems

We now show that there exists a set cover $\mathcal{D}'$ of size k if and only if there exists a container T of V such that $|T| = |V| + k$. First, suppose there exists a set cover $\mathcal{D}'$ with size k, then $U_{\mathcal{D}'} = X' = V$. By construction, there exists $\mathcal{D}$ with $|\mathcal{D}| = k$, and $U_{\mathcal{D}} = V \cup \mathcal{D}' = T$. Hence $T \supset V$, $T \in \mathsf{PCov}$ and $|T| = |V| + k$.

Conversely, suppose there exists a container T of V with size $|T| = |V| + k$. Since $T \in \mathsf{PCov}$, there exists $\mathcal{D}$ such that $U_{\mathcal{D}} = T \supset V$. Note that $|\mathcal{D}| = k$, since (by construction) each element of $\mathcal{C}$ contains precisely one element not in V. Moreover $U_{\mathcal{D}} \supset V$. Hence the corresponding set $\mathcal{D}' \subseteq \mathcal{C}'$ is a cover of X' and has cardinality k. □

Corollary 3. *The container optimization problem is NP-hard.*

Proof. The result follows from the fact that the associated decision problem is NP-complete (or we can use the construction illustrated in Fig. 2 to solve the set cover optimization problem using the container optimization problem). □

Corollary 4. *The minimal cover problem is NP-hard.*

Proof. We exhibit a polynomial time Turing reduction from the container optimization problem to the minimal cover problem. Suppose there exists an oracle for the minimal cover problem. Then given an instance $(X, \mathcal{C}, V)$ of the container optimization problem, we query the oracle for the minimal cover problem on instance $(X, \mathcal{C}, V)$, to obtain a minimal cover $\mathcal{D} \subseteq \mathcal{C}$ of V. Then we simply compute $U_{\mathcal{D}} \subseteq X$, which is, by definition, a minimal container of V. □

Corollary 5. *The optimal cover problem is NP-hard.*

Proof. We show that the minimal cover problem is polynomial time Turing equivalent to the optimal cover problem. Clearly, any solution for the optimal cover problem is a solution for the minimal cover problem. We now show a polynomial time Turing reduction from the optimal cover problem to the minimal cover problem. Given any instance $(X, \mathcal{C}, V)$ of the optimal cover problem, we

query an oracle to obtain a solution $\mathcal{D}$ for the minimal cover problem. We can then compute $\mathcal{D}' = \text{IRR-Gen}(\mathcal{D})$ in polynomial time, which is a solution to the optimal cover problem. □

3.4 The Irreducible Cover Problem

In this section, we will not be concerned with containers of V. Instead we will be concerned with all covers of X that are irreducible. We say $\mathcal{D}$ is an *irreducible cover* of X if $\mathcal{D}$ is irreducible and $U_{\mathcal{D}} = X$.

Problem 7 (The irreducible cover decision problem). Given X, $\mathcal{C}$ and a positive integer k, does there exist $\mathcal{D} \subseteq \mathcal{C}$ such that $\mathcal{D}$ is an irreducible cover of X and $|\mathcal{D}| \leqslant k$?

Problem 8 (The irreducible cover optimization problem). Given X and $\mathcal{C}$, find $\mathcal{D} \subseteq \mathcal{C}$ such that $\mathcal{D}$ is an irreducible cover of X and $|\mathcal{D}|$ is minimized.

Problem 9 (The irreducible cover enumeration problem). Given X and $\mathcal{C}$, find all $\mathcal{D} \subseteq \mathcal{C}$ such that $\mathcal{D}$ is an irreducible cover of X.

Theorem 2. *The irreducible cover decision problem is NP-complete. The irreducible cover optimization and enumeration problems are NP-hard.*

Proof. It is easy to see that the irreducible cover decision problem is in NP, because a nondeterministic algorithm need only guess a subset $\mathcal{D}$ of $\mathcal{C}$ and check whether $\mathcal{D}$ is an irreducible cover of X and $|\mathcal{D}| \leqslant k$. Checking whether $\mathcal{D}$ is an irreducible cover of X can be done in polynomial time by checking whether $U_{\mathcal{D}} = X$ and checking whether $\mathcal{D}$ is irreducible can be done in polynomial time by confirming whether $\mathcal{D} = \text{IRR-Gen}(\mathcal{D})$.

Clearly, we can use an algorithm that solves the irreducible cover problem to solve the set cover problem. It is obvious that there is an irreducible cover of cardinality less than or equal to k if and only if there is some cover of cardinality less than or equal to k.

There are trivial polynomial time Turing reductions from the irreducible cover decision problem to both the irreducible cover optimization and irreducible cover enumeration problems. In the first case, we query an oracle and return "yes" for the decision problem if the cardinality of the cover returned by the oracle is less than or equal to k. In the second case, let us assume that the oracle returns a list of irreducible covers in order of increasing cardinality. Then to solve the decision problem, we simply need to determine whether the cardinality of the first element in the list is less than or equal to k. □

4 Covering Problems in RBAC

The results of the previous section, particularly those on problems associated with minimal containers, may be of independent mathematical interest, but

in this section we apply these results to a number of problems in the RBAC literature.

Given an instance (R, P, PA) of the RBAC_0 model and an instance $(X, \mathcal{C})$ of the set cover problem, P is synonymous with X and $\{\mathsf{Prms}(r, PA) : r \in R\}$ is synonymous with $\mathcal{C}$. (This assumes that each role is assigned to at least one permission in P, and each permission is assigned to at least one role in R.) Henceforth, when PA is obvious from context, we will simply write $\mathsf{Prms}(r)$ and $\mathsf{Prms}(S)$ rather than $\mathsf{Prms}(r, PA)$ and $\mathsf{Prms}(S, PA)$, respectively.

Given $Q \subseteq P$, $\mathcal{K}(Q)$ comprises those sets of permissions that are contained within Q. In other words, $\mathcal{K}(Q)$ is synonymous with those roles that are only authorized for permissions in Q. Similarly, $\mathcal{S}(Q)$ is synonymous with those roles that are authorized for at least one permission in Q.

4.1 The Inter-domain Role Mapping Problem

Du and Joshi studied the *inter-domain role mapping* (IDRM) problem, defined below [3].

Problem 10 (The IDRM problem). Given R, P, $PA \subseteq P \times R$ and $Q \subseteq P$, find $S \subseteq R$ such that $\mathsf{Prms}(S) = Q$ and $|S|$ is minimized.

It is worth noting that many instances of the IDRM problem, as defined above, may not have a solution, since there may not exist $S \subseteq R$ such that $\mathsf{Prms}(S) = Q$. Hence, we define a preliminary question.

Problem 11 (The preliminary IDRM problem). Given R, P, PA and $Q \subseteq P$, does there exist $R_Q \subseteq R$ such that $\mathsf{Prms}(R_Q) = Q$?

We first note that Problem 11 can be decided in polynomial time, since it can be answered by determining whether $Q = \mathsf{ker}(Q)$. If so, then $R_Q = \mathcal{K}(Q)$. Having answered the preliminary IDRM problem, we may then pose the following problems.

Problem 12 (The exact IDRM decision problem). Given R, P, PA, $Q \subseteq P$, $R_Q \subseteq R$ such that $\mathsf{Prms}(R_Q) = Q$, and an integer k, does there exist $S \subseteq R_Q$ such that $\mathsf{Prms}(S) = Q$ and $|S| \leqslant k$.

Problem 13 (The exact IDRM optimization problem). Given R, P, PA, $Q \subseteq P$, and $R_Q \subseteq R$ such that $\mathsf{Prms}(R_Q) = Q$, find $S \subseteq R_Q$ such that $\mathsf{Prms}(S) = Q$ and $|S|$ is minimized.

Clearly, the set cover decision problem is identical to the exact IDRM decision problem. Given any instance $(X, \mathcal{C}, k)$ of the set cover decision problem, we simply set $X = Q$ and $\mathcal{C} = \{\mathsf{Prms}(r) : r \in R_Q\}$. Then k members of R_Q cover Q if and only if k members of $\mathcal{C}$ cover X. In other words, the exact IDRM decision problem is NP-complete, and the exact IDRM optimization problem is NP-hard.

It is also worth observing that there appears to be no good reason to minimize $|S|$: it is not clear why S is preferable to S' if $\mathsf{Prms}(S) = \mathsf{Prms}(S')$ and $|S| < |S'|$.

Moreover, if there is no solution to the IDRM problem (that is, there does not exist $S \subseteq R$ such that $\mathsf{Prms}(S) = Q$) then it is the permissions for which an approximate solution S is authorized that should be of interest, rather than $|S|$. In order to address concerns about the appropriateness of the IDRM problem, we previously proposed two problems derived from the IDRM problem [4].

Problem 14 (The IDRM-safety problem). Given P, R, PA and $Q \subseteq P$, find $S \subseteq R$ such that $\mathsf{Prms}(S) \subseteq Q$ and $|\mathsf{Prms}(S)|$ is maximized.

Problem 15 (The IDRM-availability problem). Given P, R, PA and $Q \subseteq P$, find $S \subseteq R$ such that $\mathsf{Prms}(S) \supseteq Q$ and $|\mathsf{Prms}(S)|$ is minimized.

The IDRM-safety problem is concerned with ensuring that no permission outside Q is authorized for any role in S, while authorizing S for as many permissions as possible in Q. The availability approach to IDRM ensures that all permissions in Q are authorized for at least one role in S, but seeks to minimize the number of additional permissions for which S is authorized. We noted that exhaustive search could be used to compute an exact solution to these problems and presented algorithms to produce approximate solutions to those problems, but did not establish the computational complexity of these problems.

Although there is an obvious correspondence between the exact IDRM problem and the set cover problem (as we illustrated above), there is no obvious way of transforming the IDRM-availability problem to the set cover problem, since we are simultaneously concerned with covering Q while minimizing what is covered outside Q. Clearly, however, the IDRM-availability problem does map very easily to, and is no harder than, the minimal cover problem discussed in Sect. 3.

Theorem 3. *The IDRM-safety problem is in P; the IDRM-availability problem is NP-hard.*

Proof. The largest subset of Q for which a perfect cover exists is, by Proposition 1, $\mathsf{ker}(Q)$ which can be computed in polynomial time. Hence, the IDRM-safety problem is in P.

Clearly the IDRM-availability problem is in NP. We now exhibit a polynomial time Turing reduction from the minimal cover problem to the IDRM-availability problem. Given any instance $(X, \mathcal{C}, V)$ of the minimal cover problem, we can transform it into an instance (P, Q, R, PA) of the IDRM-availability problem in polynomial time. In particular, we let $X = P$, $V = Q$, and for each $C \in \mathcal{C}$, define $r_C \in R$ and $\mathsf{Prms}(r_C) = C \subseteq X = P$. Clearly, a solution $S \subseteq R$ to this instance of the IDRM-availability problem provides a solution to the given instance of the minimal cover problem. □

4.2 The User Authorization Query Problem

Zhang and Joshi recently defined the *user authorization query* (UAQ) problem in a hybrid role hierarchy [5]. Wickramaarachchi *et al* [6] provided the following, more general, definition of UAQ.

Problem 16 (The UAQ problem). Given P, R, PA and (P_l, P_u, obj), where $P_l, P_u \subseteq P$ and $obj \in \{\max, \min\}$, find $S \subseteq R$ such that the following conditions hold:

- $P_l \subseteq \mathsf{Prms}(S) \subseteq P_u$ and $|\mathsf{Prms}(S)|$ is maximized if $obj = \max$;
- $P_l \subseteq \mathsf{Prms}(S) \subseteq P_u$ and $|\mathsf{Prms}(S)|$ is minimized if $obj = \min$.[4]

Let us rephrase the question so that we are concerned with finding $Q \subseteq P$ such that Q is perfectly covered and $P_l \subseteq Q \subseteq P_u$. Then we can find $S \subseteq R$ that solves the UAQ problem in polynomial time by computing $S = \mathcal{K}(Q)$.

Now we can compute $\mathsf{ker}(P_u)$ in polynomial time. Note also that for any solution Q, we must have $Q \subseteq \mathsf{ker}(P_u)$, by Proposition 1, since Q is perfectly covered and $Q \subseteq P_u$. Then three cases must be considered:

1. $P_l \subseteq \mathsf{ker}(P_u)$ and $obj = \max$;
2. $P_l \subseteq \mathsf{ker}(P_u)$ and $obj = \min$;
3. $P_l \not\subseteq \mathsf{ker}(P_u)$.

Case (3) means that no such Q can be found, since $Q \subseteq \mathsf{ker}(P_u)$. For case (1), we can simply take $Q = \mathsf{ker}(P_u)$, by Proposition 1. In other words, the UAQ problem posed by Wickramaarachchi *et al* only has a solution if $P_l \subseteq \mathsf{ker}(P_u)$. Moreover, the only form of the problem that cannot be answered in polynomial time is $(P_l, P_u, \min)$. Henceforth, we restrict our attention to UAQ problem of this form.

Theorem 4. *The UAQ problem and the container optimization problem are polynomial time Turing equivalent.*

Proof. We first show that there is a polynomial time Turing reduction from UAQ to container optimzation. We have to find the smallest Q such that Q is perfectly covered and $P_l \subseteq Q \subseteq \mathsf{ker}(P_u)$. We define $R_{\text{new}} = \{r \in R : \mathsf{Prms}(r) \subseteq P_u\}$ and $P_{\text{new}} = \mathsf{ker}(P_u)$. Then to answer the UAQ instance, we need only answer the container optimization instance for $X = P_{\text{new}}$, $V = P_l$ and $\mathcal{C} = \{\mathsf{Prms}(r) : r \in R_{\text{new}}\}$.

To complete the proof, we show that there is a polynomial time Turing reduction from container optimization to UAQ. The obvious transformation, previously used in the proof of Theorem 3, suffices. □

4.3 Separation of Duty

Li *et al* recently studied a number of interesting questions regarding the enforcement of *static separation of duty* (SSoD) constraints in the context of RBAC [7,9].

[4] In the original paper [6], given a set of constraints C and a user u, they require that u can activate the set of roles S without violating any constraint in C. There is also an additional condition on the cardinality of the solution set S (which essentially requires the computation of either a maximal or minimal element in the appropriate equivalence class). We omit these considerations, which do not affect the complexity of the problem, for clarity and simplicity.

Informally, an SSoD constraint (Q, k) is satisfied if no set of $k-1$ users is collectively authorized for Q. Note that an SSoD constraint cannot be satisfied if $k-1$ roles are collectively authorized for Q (assuming every role is assigned to at least one user). Li *et al* were concerned with re-writing an SSoD constraint in terms of *static mutually exclusive role* (SMER) constraints, in such a way that the satisfaction of the SMER constraints implied the satisfaction of the SSoD constraint. Hence, it is of interest to know whether the SSoD constraint is enforceable. We now describe three problems associated with separation of duty.

Problem 17 (The SSoD enforceability decision problem). Given P, R, PA, $Q \subseteq P$ and an integer k, does there exist $S \subseteq R$ such that $\mathsf{Prms}(S) \supseteq Q$ and $|S| \leqslant k$?

Problem 18 (The SSoD enforceability optimization problem). Given P, R, PA and $Q \subseteq P$, find $S \subseteq R$ such that $\mathsf{Prms}(S) \supseteq Q$ and $|S|$ is minimized.

Problem 19 (The RSSoD generation problem). Given P, R, PA and $Q \subseteq P$, find all $S \subseteq R$ such that $\mathsf{Prms}(S) \supseteq Q$ and for any $S' \subset S$, $\mathsf{Prms}(S') \not\supseteq Q$.

Note that these questions are only concerned with the existence of covers of Q (and not with any additional permissions that might be authorized for any given cover). Hence, we may simply set $X = Q$ and $\mathcal{C} = \{\mathsf{Prms}(r) \cap Q : r \in \mathcal{S}(Q)\}$. The SSoD enforceability decision problem is, therefore, identical to the set cover decision problem (and hence is NP-complete).[5]

The SSoD enforceability optimization problem is of interest for two reasons. First, given Q, we may wish to know the smallest number of users that are collectively authorized for Q in order to assess whether this presents some potential violation of enterprise security policies or statutory requirements. Second, this problem has been studied by Zhang and Joshi, although they study the problem in a rather different context and give it a different name [5]. Zhang and Joshi provided algorithms to compute an approximate solution for the problem but did not study its computational complexity. Clearly, the SSoD enforceability optimization problem is identical to the set cover optimization problem, which is NP-hard.

When seeking to enforce an SSoD constraint using SMER constraints, it is necessary to compute the set of RSSoD constraints [7]. Li *et al* define an RSSoD constraint (essentially as described in Problem 19 above), but provide no analysis of the complexity of computing the set of all such constraints. Note that an RSSoD constraint is a set of roles that cover Q and contains no redundancy. In other words, the RSSoD generation problem is identical to the irreducible cover enumeration problem and is, therefore, NP-hard (Theorem 2). The above results are summarized in the following theorem.

Theorem 5. *The SSoD enforceability decision problem is NP-complete; the SSoD enforceability optimization problem and the RSSoD generation problem are NP-hard.*

[5] Li *et al* showed that the SSoD enforceability decision problem is NP-complete by showing that a particular subcase is NP-complete [7].

Table 1. A summary of problems in RBAC and their computational complexities

Problem Name	**Equivalent Set Cover Problem**	**Complexity Class**
Preliminary IDRM	$V \in \mathsf{PCov}$? (that is, $V = \mathsf{ker}(V)$?)	P
Exact IDRM decision	Set cover decision	NP-complete
Exact IDRM optimization	Set cover optimization	NP-hard
IDRM-safety	Compute $\mathsf{ker}(V)$	P
IDRM-availability	Minimal cover	NP-hard
SSoD enforceability decision	Set cover decision	NP-complete
SSoD enforceability optimization	Set cover optimization	NP-hard
UAQ	Container optimization	NP-hard
RSSoD generation	Irreducible cover enumeration	NP-hard

5 Concluding Remarks

In this paper, we study some variations on the set cover problem. We define the notions of container, minimal container, minimal cover, irreducible cover and optimal cover, and establish complexity results for a number of problems associated with these notions.

Our results establish the computational complexity of a number of fundamental problems in RBAC: in particular, the IDRM-safety and availability problems, the UAQ problem and the RSSoD generation problem. We summarize our results in Table 1.

The minimal cover problem is NP-hard. In other words, it is unlikely that there exists an algorithm that computes an exact solution to the problem in polynomial time. Clearly, we can devise a naïve algorithm that considers every possible subset of $\mathcal{C}$ to compute an exact solution to the minimal cover problem.

There is a well known "greedy" algorithm for computing a good approximate solution to the set cover optimization problem in polynomial time [10]. This iterative algorithm sequentially selects elements from $\mathcal{C}$. At the ith iteration it selects $C_i \in \mathcal{C}$ such that $|C_i \cap V_{i-1}|$ is maximized, where V_{i-1} is the members of V that remain uncovered after the $(i-1)$th iteration. Here $|C_i \cap V_{i-1}|$ is a measure of the "benefit" of selecting C_i. An extension of this approach can be used to compute an approximate solution to the weighted set cover problem [11]. These algorithms are designed to minimize the number of sets used (as required by the set cover problem). When computing a minimal cover, however, we are not concerned with the number of elements in the cover. Instead, we are concerned with satisfying two different objectives simultaneously: to compute a cover of V and to minimize the number of elements outside V that are covered.

We previously proposed an approximate algorithm for computing solutions to the IDRM-availability problem based on the greedy algorithm for the weighted set cover problem [4]. Given X, $\mathcal{C}$ and $V \subseteq X$, we defined a "cost" function $\gamma : \mathcal{C} \to \mathbb{R}^+$ and a "benefit" function $\beta : \mathcal{C} \to \mathbb{R}^+$, where

$$\gamma(C) = |C| \cdot |C \setminus V| + \frac{1}{|V|} \quad \text{and} \quad \beta(C) = |V_{i-1} \cap C| \,.$$

We then defined an iterative algorithm that chooses $C_i \in \mathcal{C}$ at the ith iteration such that $\gamma(C_i)/\beta(C_i)$ is minimized. Informally, the algorithm chooses C_i because C_i contains relatively few elements outside V and relatively many elements of V that remain uncovered after the $(i-1)$th iteration. However, we did not provide a theoretical justification or conduct any experimental work to establish how good the approximate solutions generated by this algorithm were.

A natural extension to this algorithm is to re-compute the cost function γ_i at each iteration. Specifically, we define $\gamma_i(C, T_{i-1}) = |C| \cdot |C \setminus T_{i-1}|$, and initialize the "target" T_0 to V. At the ith iteration the algorithm

1. selects $C_i \in \mathcal{C}$ such that $\gamma_i(C_i, T_{i-1})/\beta_i(C_i)$ is minimized, and
2. target T_i is expanded to include those new elements of C_i; that is, $T_i = C_i \cup T_{i-1}$.

The advantage of this approach is that in choosing C, we expand V to $V \cup C$, and it may be that we can choose C' to cover other elements of V without including any elements outside $V \cup C$. Consider, for example, $X = \{1, 2, 3, 4\}$, $\mathcal{C} = \{C_1, C_2, C_3, C_4\}$, where $C_1 = \{1, 3\}$, $C_2 = \{2, 3\}$, $C_3 = \{1, 4\}$, $C_4 = \{2, 4\}$, and $V = \{1, 2\}$. If C_1 is chosen at the first step of the algorithm, then, at the second iteration, we choose C_2 (since V has been expanded to include 3 from C_1) to obtain the container $\{1, 2, 3\}$. In contrast, our earlier algorithm can choose between C_2 and C_4 at the second step, the latter choice ultimately resulting in the container $\{1, 2, 3, 4\}$.

We plan to develop different algorithms using different cost functions described above, and conduct some experimental work to test the quality of the approximate solutions generated by these algorithms. We would then like to establish an *approximation ratio* for the best algorithm we obtained in the experimental work. More specifically, let $\mathcal{D} \subseteq \mathcal{C}$ be a cover of V returned by an approximate algorithm. Then we define the *quality* of $\mathcal{D}$ to be $|U_{\mathcal{D}}|$. The approximation ratio of the algorithm indicates that the ratio between the quality of approximate solution returned by the algorithm and the quality of the exact solution is bounded by some function of $|\mathcal{C}|$ and $|V|$ (see the work of Johnson [10], Chvatal [11] and Feige [12] on the set cover problem, for example).

Acknowledgements. We would like to thank the anonymous referees for their careful reading and cogent analysis of the shortcomings of a preliminary version of this paper. The final version has been much improved as a result of the referees' insightful feedback.

References

1. Sandhu, R., Coyne, E.J., Feinstein, H., Youman, C.E.: Role-based access control models. IEEE Computer 29(2), 38–47 (1996)
2. American National Standards Institute: ANSI INCITS 359-2004 for Role Based Access Control (2004)

3. Du, S., Joshi, J.B.D.: Supporting authorization query and inter-domain role mapping in presence of hybrid role hierarchy. In: Proceedings of the 11th ACM Symposium on Access Control Models and Technologies, pp. 228–236 (2006)
4. Chen, L., Crampton, J.: Inter-domain role mapping and least privilege. In: Proceedings of the 12th ACM Symposium on Access Control Models and Technologies, pp. 157–162 (2007)
5. Zhang, Y., Joshi, J.B.D.: UAQ: A framework for user authorization query processing in RBAC extended with hybrid hierarchy and constraints. In: Proceedings of the 13th ACM Symposium on Access Control Models and Technologies, pp. 83–92 (2008)
6. Wickramaarachchi, G.T., Qardaji, W.H., Li, N.: An efficient framework for user authorization queries in RBAC systems. In: Proceedings of the 14th ACM Symposium on Access Control Models and Technologies, pp. 23–32 (2009)
7. Li, N., Tripunitara, M.V., Bizri, Z.: On mutually exclusive roles and separation-of-duty. ACM Transactions on Information and System Security 10(2) (2007)
8. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Company, New York (1979)
9. Chen, H., Li, N.: Constraint generation for separation of duty. In: Proceedings of the Eleventh ACM Symposium on Access Control Models and Technologies, pp. 130–138 (2006)
10. Johnson, D.S.: Approximation algorithms for combinatorial problems. Journal of Computer and System Sciences 9(3), 256–278 (1974)
11. Chvatal, V.: A greedy heuristic for the set-covering problem. Mathematics of operations research 4(3), 233–235 (1979)
12. Feige, U.: A threshold of $\ln n$ for approximating set cover. Journal of the ACM 45(4), 634–652 (1998)