The Role Mining Problem: Finding a Minimal Descriptive Set of Roles*

Jaideep Vaidya
MSIS Department and CIMIC
Rutgers University
180 University Ave, Newark,
NJ 07102
jsvaidya@rbs.rutgers.edu

Vijayalakshmi Atluri
MSIS Department and CIMIC
Rutgers University
180 University Ave, Newark,
NJ 07102
atluri@rutgers.edu

Qi Guo
MSIS Department and CIMIC
Rutgers University
180 University Ave, Newark,
NJ 07102
qiguo@rutgers.edu

ABSTRACT

Devising a complete and correct set of roles has been recognized as one of the most important and challenging tasks in implementing role based access control. A key problem related to this is the notion of goodness/interestingness – when is a role good/interesting? In this paper, we define the *role mining problem* (RMP) as the problem of discovering an optimal set of roles from existing user permissions. The main contribution of this paper is to formally define RMP, and analyze its theoretical bounds. In addition to the above basic RMP, we introduce two different variations of the RMP, called the *δ-approx RMP* and the *Minimal Noise RMP* that have pragmatic implications. We reduce the known "set basis problem" to RMP to show that RMP is an NP-complete problem. An important contribution of this paper is also to show the relation of the role mining problem to several problems already identified in the data mining and data analysis literature. By showing that the RMP is in essence reducible to these known problems, we can directly borrow the existing implementation solutions and guide further research in this direction.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection—*Access controls*; H.2.8 [**Database Management**]: Database Appliations—*Data Mining*

General Terms

Security

Keywords

RBAC, role engineering, role mining

1. INTRODUCTION

Role-based access control (RBAC) has been adopted successfully by a variety of commercial systems. As a result, RBAC has become the norm in many of today's organizations for enforcing security. Basically, a role is nothing but a set of permissions. Roles represent organizational agents that perform certain job functions within the organization. Users, in turn, are assigned appropriate roles based on their qualifications [18, 3].

However, one of the major challenges in implementing RBAC is to define a complete and correct set of roles. This process, known as *role engineering* [2], has been identified as one of the costliest components in realizing RBAC [4]. Essentially, role engineering is the process of defining roles and assigning permissions to them.

There are two basic approaches towards role engineering: *top-down* and *bottom-up*. Under the top-down approach, roles are defined by carefully analyzing and decomposing business processes into smaller units in a functionally independent manner. These functional units are then associated with permissions on information systems. In other words, this approach begins with defining a particular job function and then creating a role for this job function by associating needed permissions. Often, this is a cooperative process where various authorities from different disciplines understand the semantics of business processes of one another and then incorporate them in the form of roles. Since there are often dozens of business processes, tens of thousands of users and millions of authorizations, this is rather a difficult task. Therefore, relying solely on a top-down approach in most cases is not viable, although some case studies [19] indicate that it has been done successfully by some organizations (though at a high cost).

In contrast, since organizations do not exist in a vacuum, the bottom-up approach utilizes the existing permission assignments to formulate roles. Starting from the existing permissions before RBAC is implemented, the bottom-up approach aggregates these into roles. It may also be advantageous to use a mixture of the top-down and the bottom-up approaches to conduct role engineering. While the top-down model is likely to ignore the existing permissions, a bottom-up model may not consider business functions of an organization [9]. However, the bottom-up approach excels in the fact that much of the role engineering process can be automated. Role mining can be used as a tool, in conjunction with a top-down approach, to identify potential or candidate

roles which can then be examined to determine if they are appropriate given existing functions and business processes.

There have been several attempts to propose good bottom-up techniques to finding roles. Kuhlmann et al. [10] present a clustering technique similar to the well known k-means clustering, which requires pre-defining the number of clusters. In [20], Schlegelmilch and Steffens propose an agglomerative clustering based approach to role mining (called ORCA), which discovers roles by merging permissions appropriately. However, in ORCA, the order in which permissions are merged determines the outcome of roles. Moreover, it does not allow overlapping roles (i.e., a user cannot play multiple roles), which is a significant drawback. More recently, Vaidya et al. [21] propose an approach based on subset enumeration, called RoleMiner, which eliminates the above limitations.

An inherent problem with all of the above approaches is that there is no formal notion of *goodness/interestingness* of a role. All of the algorithms above present heuristic ways to find a set of candidate roles. While offering justifications for the identified roles, there is no integrative view of the entire set of roles. For insightful bottom-up analysis, we need to define interestingness metrics for roles. [21] takes a first step towards this by ordering candidate roles on the basis of their support (i.e., roles that are more prevalent are ordered higher). However, this metric still is quite ad-hoc and preliminary. Also, while one may come up with interestingness metrics for a role by itself, this does not directly lead to the notion of a good collection of roles. Indeed, there is no formal definition of what is a good *collection of roles*. Defining this is critical for the security administrator to gain confidence and be able to fully utilize the output of any role mining algorithm beyond a piece-meal fashion.

The main contribution of this paper is to formally define the role mining problem, and analyze its theoretical bounds. Assuming that we can represent the user permissions as a binary matrix, informally, we define the *basic* role mining problem as follows: Given a $m \times n$ binary matrix A representing the user-permissions, decompose A into two matrices B and C, where B is a $m \times k$ matrix representing the user-role assignment and C is $k \times n$ matrix representing the role-permission association, such that k is minimal.

It is important to note that it is quite easy to come up with *some* decomposition of matrix A. For example, two extreme cases are 1) where each user is placed in a role by itself (i.e., $k = m$, $B = I$, the identity matrix, and $C = A$), and 2) where each permission is placed in a role by itself (i.e., $k = n$, $B = A$, and $C = I$, the identity matrix). Both of these decompositions are accurate, but are not necessarily minimal. An alternative decomposition, is to place all users in a single role (i.e., $k = 1$). However, this decomposition is likely to be very inaccurate unless all of the users indeed have the same set of permissions. In none of these cases are the roles likely to be accurate (i.e., close to reality). What we are really interested in, is a fairly accurate set of roles. In this regard, we consider the minimal set of roles as the accurate set. Minimality is a good notion, since it allows us to formally define the problem. Without semantics (i.e., human expert knowledge), minimality serves as a best approximation for realizing good *descriptive* roles.

One may note that, in a specific implementation of RBAC, the most useful set of roles may be different from the minimal set. It is analogous to employing the best normalization in designing a database schema. However, from the practical point of view, one may denormalize the database for improving the query response. Similar to our analogous example, the minimal set of roles gives us a good set of roles to begin with. At least, it shows the bare minimum required to accurately describe the current state of the organization. We argue that this is likely to be of immense help to the security administrator. In this paper, we formally define the basic RMP problem and show that the decision version is NP-complete by reducing the known NP-complete *set basis problem* to this.

We also consider several interesting variations of the basic RMP, including the *δ-approx Role Mining Problem* (δ-approx RMP) and the *Minimal Noise Role Mining Problem* (MinNoise RMP). These are of practical importance. Both the δ-approx RMP and the MinNoise RMP are likely to result in a lower number of roles than the basic RMP – and might more accurately model the dynamic state of the organization. We describe these individually below: While solving the basic RMP, the goal is to identify the minimal set of roles such that the original user-permission assignment matrix is decomposed. However, if we allow a slight inaccuracy in the decomposition such that when multiplied it does not generate the original matrix, it may still be acceptable. It is this variation of the basic RMP that we recognize as the δ-approx RMP. Moreover, when discovering roles, one may state the number of roles to be identified. Given the specified set of roles, one may come up with a decomposition of the user-permission assignment. Note that, in this process, since we rigidly set the order of the matrices to be decomposed, they may not generate the original matrix when multiplied. This discrepancy denoted as *noise* should be at a minimum. We recognize this problem as the MinNoise RMP. We show that the complexity of both the δ-approx RMP and the MinNoise RMP is NP-complete.

We have discovered that our basic role mining problem is identical to the problem of *database tiling* recently proposed by Geerts et al. [6]. We show how our basic RMP can be mapped to the database tiling and present an algorithm to use tiling to discover roles. Similarly, the recently proposed *discrete basis problem* [14] is identical to the MinNoise RMP, and we show the mapping between our MinNoise RMP to the discrete basis problem.

This paper is organized as follows. In section 2 we review the RBAC model and some preliminary definitions employed in the paper. In section 3, we define our basic role mining problem as well as its variations and prove results about their complexity. In sections 4 and 5 we show the mappings of our RMP to the database tiling and the discrete basis problem, respectively. Finally, section 6 provides some insight into our ongoing and future research.

2. PRELIMINARIES

We adopt the NIST standard of the Role Based Access Control (RBAC) model [3]. For the sake of simplicity, we do not consider sessions, role hierarchies or separation of duties constraints in this paper. In other words, we restrict ourselves to RBAC_0 without considering sessions.

DEFINITION 1 (RBAC).

- $U, ROLES, OPS,$ and OBJ are the set of users, roles, operations, and objects.

- $UA \subseteq U \times ROLES$, a many-to-many mapping user-to-role assignment relation.

- PRMS (the set of permissions) $\subseteq \{(op, obj)|op \in OPS \bigwedge obj \in OBJ\}$

- $PA \subseteq ROLES \times PRMS$, a many-to-many mapping of role-to-permission assignments.[1]

- $UPA \subseteq U \times PRMS$, a many-to-many mapping of user-to-permission assignments.

- $assigned\_users(R) = \{u \in U|(u, R) \in UA\}$, the mapping of role R onto a set of users.

- $assigned\_permissions(R) = \{p \in PRMS|(p, R) \in PA\}$, the mapping of role R onto a set of permissions.

We now need some additional definitions from [13] pertaining to boolean matrix multiplication:

DEFINITION 2 (BOOLEAN MATRIX MULTIPLICATION). *A Boolean matrix multiplication between Boolean matrices $A \in \{0,1\}^{m \times k}$ and $B \in \{0,1\}^{k \times n}$ is $A \otimes B = C$ where C is in space $\{0,1\}^{m \times n}$ and*

$$c_{ij} = \bigvee_{l=1}^{k} (a_{il} \wedge b_{lj}).$$

DEFINITION 3 (L_1 NORM). *The L_1 norm of a d-dimensional vector $v \in X^d$, for some set X, is*

$$\| v \|_1 = \sum_{i=1}^{d} |v_i|.$$

The L_1-norm also defines a distance metric between vectors, referred to as L_1-metric and defined as

$$\| v - w \|_1 = \sum_{i=1}^{d} |v_i - w_i|.$$

Finally, the L_1-metric between vectors is expanded to matrices in a natural way, i.e., if A and B are matrices in $X^{n \times m}$, for some set X, then

$$\| A - B \|_1 = \sum_{i=1}^{n} \| a_i - b_i \|_1 = \sum_{i=1}^{n} \sum_{j=1}^{m} |a_{ij} - b_{ij}|.$$

The L_1-metric allows us to count the difference between two matrices – i.e., to figure out how good an approximation one is of the other. When the L_1-metric is 0, the two matrices are identical. Other metrics (and distances) can also be used – [13] discusses some alternatives and their implications.

3. THE ROLE MINING PROBLEM

Given m users, n permissions and k roles (i.e., $|U| = m$, $|PRMS| = n$, $|ROLES| = k$), the user-to-role mapping can be represented as an $m \times k$ boolean matrix where a 1 in cell $\{ij\}$ indicates the assignment of role j to user i. Similarly, the role-to-permission mapping can be represented as an $k \times n$ boolean matrix where a 1 in cell $\{ij\}$ indicates the

[1]Note that in the original NIST standard [3], PA was defined as $PA \subseteq PRMS \times ROLES$, a many-to-many mapping of permission-to-role assignments.

assignment of permission j to role i. Finally, the user-to-permission mapping can be represented as an $m \times n$ boolean matrix where a 1 in cell $\{ij\}$ indicates the assignment of permission j to user i.

We now introduce the notion of δ-consistency between UA, PA, and UPA which is critical to the notion of accuracy of the roles. The L_1 norm defined above is useful in defining this.

DEFINITION 4 (δ-CONSISTENCY). *A given user-to-role assignment UA, role-to-permission assignment PA and user-to-permission assignment UPA are δ-consistent if and only if*

$$\| M(UA) \otimes M(PA) - M(UPA) \|_1 \leq \delta$$

where $M(UA), M(PA)$, and $M(UPA)$ denote the matrix representation of UA, PA and UPA respectively.

Essentially, the notion of δ-consistency allows us to bound the degree of difference between the user-to-role assignment UA, role-to-permission assignment PA and user-to-permission assignment UPA. For UA, PA, and UPA to be δ-consistent, the user-permission matrix generated from UA and PA should be within δ of UPA.

3.1 RMP and its variants

In this section, we present the basic RMP and two of its variants, δ-approx RMP and the MinNoise RMP.

DEFINITION 5 (ROLE MINING PROBLEM (RMP)). *Given a set of users U, a set of permissions $PRMS$, and a user-permission assignment UPA, find a set of roles, $ROLES$, a user-to-role assignment UA, and a role-to-permission assignment PA 0-consistent with UPA and minimizing the number of roles, k.*

Given the user-permission matrix, the basic Role Mining problem asks us to find a user-to-role assignment UA and a role-to-permission assignment PA such that UA and PA *exactly* describe UPA while minimizing the number of roles. Put another way, it asks us what is the minimum number of roles necessary to fully describe the given data (and what are those roles, and the corresponding user assignments)?

While exact match is a good thing to have, at times we may be satisfied with an approximate match. For example, consider a case where we have 1000 users and 100 permissions. The size of UPA is 5000 (i.e., 5000 user-permission assignments are allowed out of the possible $100,000$). Now, suppose 100 roles are required to *exactly* match the given user-permission data. However, if we allow approximate matching – i.e., if it is good enough to match 99% of the matrix (4950 of the user-permission assignments), assume that the minimum number of roles required is only 60. As long as we do not add any spurious permissions (i.e., no extra 1s are added), the second case is clearly better than the first, since we significantly reduce the number of roles. This significantly reduces the burden of maintenance on the security administrator while leaving only a few user-permission assignments uncovered. Also, any given user-permission assignment is only a snapshot of the current state of the organizations. Permissions and (to a lesser extent, Roles) are dynamic. Thus while exact match may be the best descriptor in the static case, it is probably not good for the dynamic case. Approximate match might be a prudent choice for dynamic data. The notion of δ-consistency is useful, since it

	p_1	p_2	p_3	p_4	p_5
u_1	0	1	0	0	1
u_2	1	1	1	0	1
u_3	1	1	0	1	1
u_4	1	1	1	0	0

Table 1: User-privilege assignment

helps to bound the degree of approximation. Therefore, we now define the approximate Role Mining Problem using δ-consistency.

DEFINITION 6 (δ-APPROX RMP). *Given a set of users U, a set of permissions PRMS, a user-permission assignment UPA, and a threshold δ, find a set of roles, ROLES, a user-to-role assignment UA, and a role-to-permission assignment PA, δ-consistent with UPA and minimizing the number of roles, k.*

It should be clear that the basic Role Mining Problem defined earlier is simply a special case of the δ-approx RMP (with δ set to 0). Instead of bounding the approximation, and minimizing the number of roles, it might be interesting to do the reverse – bound the number of roles, and minimize the approximation. We call this the Minimal Noise Role Mining Problem (MinNoise RMP). Thus, we fix the number of roles that we would like to find, but now we want to find those roles that incur minimal difference with respect to the original user-permission matrix (UPA). The security administrator might want to do this when he is looking for the top-k roles that describe the problem space well enough, and are still (in some sense) robust to noise.

DEFINITION 7 (MINIMAL NOISE RMP). *Given a set of users U, a set of permissions PRMS, a user-permission assignment UPA, and the number of roles k, find a set of k roles, ROLES, a user-to-role assignment UA, and a role-to-permission assignment PA, minimizing*

$$\| M(UA) \otimes M(PA) - M(UPA) \|_1$$

where $M(UA), M(PA)$, and $M(UPA)$ denote the matrix representation of UA, PA and UPA respectively.

We can clarify these problems further by means of an example. Table 1 shows a sample user-privilege assignment (UPA), for 4 users and 5 privileges. Tables 2(a) and 2(b) depict a user-role assignment (UA) and role-privilege assignment (PA) that completely describe the given user-privilege assignment (i.e., $M(UA) \otimes M(PA) = M(UPA)$). Indeed, the given UA, PA, and $ROLES$ are optimal. It is not possible to completely describe the given UPA with less than 3 roles. Tables 3(a) and 3(b) depict the optimal user-role assignment (UA) and role-privilege assignment (PA) 2-consistent, 3-consistent, as well as 4-consistent with UPA. Tables 3(c) and 3(d) show the optimal user-role assignment (UA) and role-privilege assignment (PA) 5-consistent with UPA. Similarly, if we set $k = 2$, Tables 3(a) and 3(b) depict one possible optimal minimal noise UA and PA. Tables 4(a) and 4(b) depict another optimal UA and PA for the MinNoise RMP. Both represent correct solutions to the MinNoise RMP, though the second one does not incorrectly cover any 0s with 1s.

	r_1	r_2	r_3
u_1	0	0	1
u_2	1	0	1
u_3	0	1	1
u_4	1	0	0

(a) User-role assignment

	p_1	p_2	p_3	p_4	p_5
r_1	1	1	1	0	0
r_2	1	1	0	1	0
r_3	0	1	0	0	1

(b) Role-permission assignment

Table 2: Basic Role Mining Problem

	r_1	r_2
u_1	0	1
u_2	1	1
u_3	1	1
u_4	1	0

(a) User-role assignment

	p_1	p_2	p_3	p_4	p_5
r_1	1	1	1	0	0
r_2	0	1	0	0	1

(b) Role-permission assignment

	r_1
u_1	0
u_2	1
u_3	1
u_4	1

(c) User-role assignment

	p_1	p_2	p_3	p_4	p_5
r_1	1	1	1	0	1

(d) Role-permission assignment

Table 3: δ-approx RMP

	r_1	r_2
u_1	0	1
u_2	1	1
u_3	0	1
u_4	1	0

(a) User-role assignment

	p_1	p_2	p_3	p_4	p_5
r_1	1	1	1	0	0
r_2	0	1	0	0	1

(b) Role-permission assignment

Table 4: MinNoise RMP

3.2 Complexity

Before proceeding any further, we would like to establish some results on the complexity of these problems. The Role Mining Problem, the δ-approx RMP, and the MinNoise RMP Problem are all optimization problems. The theory of NP-completeness applies to decision problems. Therefore, in order to consider the complexity of the problems, we now frame the decision version of these problems.

DEFINITION 8 (DECISION RMP). *Given a set of users U, a set of permissions $PRMS$, a user-permission assignment UPA, and $k \geq 0$, are there a set of roles, $ROLES$, a user-to-role assignment UA, and a role-to-permission assignment PA 0-consistent with UPA such that $|ROLES| \leq k$?*

DEFINITION 9 (DECISION δ-APPROX RMP). *Given a set of users U, a set of permissions $PRMS$, a user-permission assignment UPA, a threshold $\delta \geq 0$, and $k \geq 0$, are there a set of roles, $ROLES$, a user-to-role assignment UA, and a role-to- permission assignment PA, δ-consistent with UPA such that $|ROLES| \leq k$?*

DEFINITION 10 (DECISION MINNOISE RMP). *Given a set of users U, a set of permissions $PRMS$, a user-permission assignment UPA, the number of roles k, and a noise threshold θ, are there a set of k roles, $ROLES$, a user-to-role assignment UA, and a role-to-permission assignment PA, such that*

$$\| M(UA) \otimes M(PA) - M(UPA) \|_1 \leq \theta$$

where $M(UA), M(PA)$, and $M(UPA)$ denote the matrix representation of UA, PA and UPA respectively?

We can now prove that decision RMP, decision δ-approx RMP, and decision MinNoise RMP are all NP-complete (Indeed, some of these results have already been obtained in the literature[13, 6]). Proving that a problem π is NP-Complete consists of four main steps [5]:

1. showing that π is in NP

2. selecting a known NP-complete problem π'

3. constructing a transformation f from π' to π, and

4. proving that f is a (polynomial) transformation

The problem π' used to reduce from is the "set basis problem" defined below:

DEFINITION 11 (SET BASIS PROBLEM). *Given a collection C of subsets of a finite set S, and a positive integer $K \leq |C|$, is there a collection B of subsets of S with $|B| = K$ such that, for each $c \in C$, there is a sub-collection of B whose union is exactly c?*

THEOREM 1. *The decision RMP is NP-complete.*

PROOF. ● The decision Role Mining Problem is in NP. The set of roles $ROLES$, the user-to-role assignment UA, and the role-to-permission assignment PA together form the polynomial certificate/witness.

● We select the set basis problem as π'

● The transformation is quite simple. Given an instance of the set basis problem, here is how we transform it to an instance of the decision Role Mining Problem: S denotes the permissions $PRMS$. C denotes UPA. Thus, every set $c \in C$ stands for one user u. Now, the answer to the decision role mining problem directly provides the answer to the set basis problem.

● The transformation is clearly polynomial (since it is a direct one-to-one mapping).

$\square$

THEOREM 2. *The decision δ-approx RMP is NP-complete.*

PROOF. ● The decision δ-approx RMP is in NP. The set of roles $ROLES$, the user-to-role assignment UA, and the role-to-permission assignment PA together form the polynomial certificate/witness. It only takes polynomial time to compute

$$\| M(UPA) - (M(UA) \otimes M(PA)) \|_1$$

and ensure that it is less than or equal to δ, and that $|ROLES| \leq k$.

● We select the set basis problem as π'

● The transformation is quite simple. Given an instance of the set basis problem, here is how we transform it to an instance of the decision Role Mining Problem: S denotes the permissions $PRMS$. C denotes UPA. Thus, every set $c \in C$ stands for one user u. δ is set to 0. Now, the answer to the decision approx role mining problem directly provides the answer to the set basis problem.

● The transformation is clearly polynomial.

$\square$

THEOREM 3. *The decision MinNoise RMP is NP-complete.*

PROOF. ● The decision MinNoise RMP is in NP. The set of roles $ROLES$, the user-to-role assignment UA, and the role-to-permission assignment PA together form the polynomial certificate/witness. It only takes polynomial time to compute

$$\| M(UPA) - (M(UA) \otimes M(PA)) \|_1$$

and ensure that it is less than or equal to θ, and $|ROLES| = k$.

● We select the set basis problem as π'

● The transformation is quite simple. Given an instance of the set basis problem, here is how we transform it to an instance of the decision Role Mining Problem: S denotes the permissions $PRMS$. C denotes UPA. Thus, every set $c \in C$ stands for one user u. Set $\theta = 0$. Now, the answer to the decision MinNoise RMP directly provides the answer to the set basis problem.

● The transformation is clearly polynomial.

$\square$

	p1	p2	p3	p4	p5	p6	p7
u1	1	1	0	0	1	1	1
u2	0	0	0	1	1	1	1
u3	1	1	0	1	1	0	0
u4	1	1	0	0	0	0	0

(a) A 4 × 7 user-to-permission assignment (UPA)

	p1	p2	p3	p4	p5	p6	p7
u1	1	1	0	0	1	1	1
u2	0	0		1	1	1	1
u3	1	1	0	1	1	0	0
u4	1	1	0	0	0	0	0

(b) Shaded areas indicate tiles, the 3 identified roles

	R1	R2	R3
u1	1	0	1
u2	0	1	1
u3	1	1	0
u4	1	0	0

(c) user-to-role assignment (UA)

	p1	p2	p3	p4	p5	p6	p7
R1	1	1	0	0	0	0	0
R2	0	0	0	1	1	0	0
R3	0	0	0	0	1	1	1

(d) permission-to-role assignment (PA)

Figure 1: An example of mapping basic RMP to Minimum Tiling Problem

Instead of asking for the user-role assignment, UA, as well as the role-permission assignment PA, we could consider the problem of obtaining each individually. For exact cover, [14, 13] shows that given the set of roles, and the role-permission assignment, one can determine the user-role assignment in polynomial time. However, when an approximate answer is required, such as in the MinNoise RMP, determining the user-role assignment requires $O(2^k mn)$ time – this is known as *fixed parameter tractable* since the solution is exponential only in terms of a fixed parameter. Unfortunately, k refers to the number of roles which is likely to be quite large in practice, making this quite infeasible. It remains to be seen if finding the user-role assignment in the case of the δ-approx RMP is any easier.

In the following sections, we show that the RMP along with several variants can be mapped to other problems already studied in the data mining and data analysis literature. We discuss the complexity for each variant along with suggested methods for solving the problem.

4. MAPPING THE RMP TO THE TILING PROBLEM

In this section, we demonstrate the equivalence of the Role Mining Problem with the Tiling Databases problem. This mapping allows us to directly borrow existing implementation solutions to RMP. In fact, the original Database Tiling paper by Geerts et al. [6] looked at a set of five problems, one of which exactly matches the role mining problem. We now describe the relevant problems studied and then discuss their implications.

4.1 Tiling Databases

Consider a binary matrix of size $m \times n$ where the number of rows, m, can be viewed as the number of objects and the number of columns, n, can be viewed as the number of attributes. A 1 in cell $\{ij\}$ denotes that object i has/owns attribute j (i.e., some relationship exists between object i and attribute j). Now, let an itemset I denote a collection (subset) of the attributes. Then a *tile* t corresponding to an itemset I consists of the columns in itemset I as well as all the rows that have 1s in all the columns in I. The area of a tile is defined as the cardinality of the tile (i.e., the number of 1s in the tile).

Informally, a tile consists of a block of ones in a boolean database as shown in Figure 1(b). A collection of (possibly overlapping) tiles constitutes a tiling. Among the collection of 5 related problems defined in [6], the *Minimum Tiling* problem is of the most interest to us, which is defined below.

DEFINITION 12   (MINIMUM TILING). *Given a boolean matrix, find a tiling of the matrix with area equal to the total number of 1s in the matrix and consisting of the least possible number of tiles.*

4.2 Mapping Basic RMP to Minimum Tiling

To see that the Minimum Tiling problem corresponds exactly to the basic RMP, one must first see how a tile corresponds to a role. As defined above, a tile is just a block of 1s – i.e., a collection of rows and columns that all have 1s. Remember that without semantics, a role is simply a collection of permissions. Thus, inherently, in any tile, the collection of the columns provides the role-to-permission assignment (PA) for that role. At the same time, the collection of rows denotes those users/entities that have that role – thus the collection of rows corresponds to the user-to-role assignment (UA) for that role. As such, any tiling corresponds to a set of roles and their role/permission and user/role assignments. If the tiling completely covers the entire matrix – then all 1s have been covered, meaning that all user/permission assignments have been covered. Since each tile corresponds to a role, if the tiling is minimal and covers the entire matrix, this means that we have found a set of minimal roles such that they completely describe the given user-permission assignment.

The following example clearly demonstrates this mapping. In the context of tiling databases, Figure 1(a) shows the boolean matrix representing a transactional database consisting of 4 transactions and 7 items. Rows denote the trans-

actions and columns denote the items. We may order transactions from top to bottom sequentially as $1 - 4$ and items from left to right as $1 - 7$. A 1 in cell $\{ij\}$ represents that transaction i contains item j. Figure 1(b) shows a tiling of the matrix consisting of 3 tiles. The shaded region represents a tile. Thus, Tile 1=$\{(1,1), (1,2), (3,1), (3,2), (4,1), (4,2)\}$. Tile 2=$\{(2,4), (2,5), (3,4), (3,5)\}$ and Tile 3=$\{(1,5), (1,6), (1,7), (2,5), (2,6), (2,7)\}$. As one can see, Tiles 2 and 3 overlap on cell $(2,5)$. Figure 1(b) also gives the minimum tiling of the matrix. It is not possible to find a tiling the covers the entire matrix with less than 3 tiles. We can view the same problem from the role mining perspective. As described before, each tile corresponds to a role. Figure 1(c) and 1(d) show an optimal UA and PA, such that $M(UA) \otimes M(PA) = M(UPA)$. Again, the decomposition is optimal in the sense that it is impossible to find only two roles such that UA and PA will be 0-consistent with UPA.

Formally, we can reduce the Minimum Tiling problem to the basic RMP as follows.

THEOREM 4. *The Minimum Tiling problem is identical to the basic Role Mining Problem.*

PROOF. To show that the two problems are identical we show that their inputs and outputs exactly match. Thus, for every input instance, the output of both problems have a direct one-to-one mapping.

- The input to both problems is a $m \times n$ boolean matrix.

- For any problem instance, the Minimum Tiling problem returns a set of tiles that completely cover the input while minimizing the number of tiles. Each tile corresponds to a role, R. For each tile, we extract the set of columns C, in the tile. For each column $c \in C$, add the assignment $\{c, R\}$ to PA. Similarly, for each row i, belonging to the tile, add the assignment $\{i, R\}$ to UA. Add R to $ROLES$.

- The resulting set of roles ($ROLES$), user-role assignment (UA), and permission-role assignment (PA) are guaranteed to be a solution to the basic RMP. (i.e., UA and PA are 0-consistent with the corresponding UPA, and the number of roles is minimal). To prove the 0-consistency, it is sufficient to note that $UA \otimes PA$ gives us the original tiling of the input matrix which is equivalent to the original UPA. We can prove the minimality by contradiction. Assume that a different solution to the RMP exists – consisting of $ROLES'$, UA' and PA' where $|ROLES'| < |ROLES|$. In this case, we can transform this solution into a corresponding solution for tiling. For each role $r \in ROLES'$, create the corresponding tile t_R consisting of the permissions given by PA' and the users given by UA'. The union of all tiles $\bigcup_R T_R$ gives a tiling of the matrix. This tiling covers the entire matrix since UA' and PA' are 0-consistent with UPA. However, the number of tiles is the same as $|ROLES'|$ which is less than $|ROLES|$. But that means that the earlier solution is not minimal – and we have a contradiction. Therefore, the solution to the tiling databases problem directly maps to a solution for the role mining problem.

☐

Thus, the Minimum Tiling problem exactly corresponds to the basic RMP.

4.3 Algorithm to Discover Minimal Roles

Since the Minimum Tiling problem is equivalent to the basic RMP, the algorithms developed for Minimum Tiling now directly apply. [6] proposes a greedy approximation algorithm to find the minimum tiling of any given database. This algorithm depends on finding all maximal tiles having an area over a given threshold. A depth first search strategy is used to find all large tiles. [6] prove that the Minimum Tiling problem can be approximated within the factor $O(\log mn)$, given an oracle that finds for any database D and tiling T, the tile t such that the $area(T \cup t)$ is the maximum (i.e., the oracle returns the tile which covers as much of the remaining uncovered part of the database). Such an oracle can be implemented reasonably efficiently by adapting the maximal tile algorithm. [6] provides more detail on this. We now briefly present the adapted algorithm for the basic RMP.

Algorithm 1 presents the basic RMP algorithm. It consists of two phases. In the first phase, we find a minimum tiling for the given UPA. In the second phase, we convert the tiling into $ROLES$, UA, and PA. As described earlier, phase 1 uses a simple greedy strategy of adding the largest uncovered tile to the current tiling, until UPA is completely covered (i.e., the largest uncovered tile remaining is empty). Algorithm 2 describes the procedure for finding the largest uncovered tile from UPA.
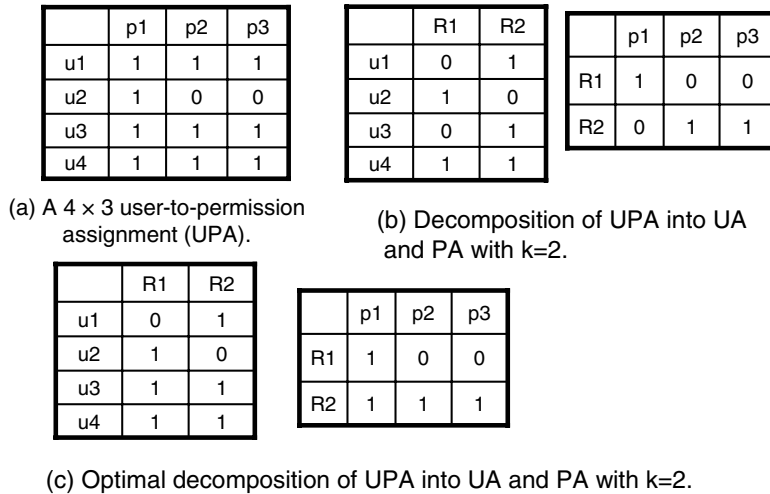

Algorithm 1 RMP(UPA)

1: {Find the minimum tiling for UPA}
2: $T \leftarrow \{\}$
3: **while** $(T' \leftarrow \text{LUTM}(UPA,T)) \neq \{\}$ **do**
4:     $T \leftarrow T \cup T'$
5: **end while**
6: {Convert the minimum tiling into UA and PA}
7: $ROLES \leftarrow \{\}, UA \leftarrow \{\}, PA \leftarrow \{\}$
8: **for** each tile $t \in T$ **do**
9:     Create a new role R and add it to $ROLES$
10:     Extract the set of permissions P in the tile
11:     For each permission $p \in P$, add the assignment $\{p, R\}$ to PA
12:     Extract the set of users U in the tile
13:     For each user $u \in U$, add the assignment $\{u, R\}$ to UA
14: **end for**

The LUTM algorithm (Algorithm 2) is a depth-first recursive algorithm that finds the largest uncovered tile. In order to do a depth-first search, we simply assume some canonical order over the permissions. The key idea behind the algorithm is that all large tiles containing a permission $i \in PRMS$, but not containing any permission lower than i (according to the canonical order) can be found in the so-called i-conditional database [7]. In our context, the P-conditional database UPA^P consists of all user-to-permission assignments that contain P, but from which all permissions before the last permission in P and that last permission itself would have been removed. Now, any large tile that is found in this conditional database, at once implies a corresponding large tile including P. Therefore, whenever we want to compute an area associated with a set of permissions P' in UPA^P, we simply need to add $|P|$ to the width of the area ($|P'|$) and multiply this with $|U(P')|$ [6]. We

	p1	p2	p3
u1	1	1	1
u2	1	0	0
u3	1	1	1
u4	1	1	1

(a) A 4 × 3 user-to-permission assignment (UPA).

	R1	R2
u1	0	1
u2	1	0
u3	0	1
u4	1	1

	p1	p2	p3
R1	1	0	0
R2	0	1	1

(b) Decomposition of UPA into UA and PA with k=2.

	R1	R2
u1	0	1
u2	1	0
u3	1	1
u4	1	1

	p1	p2	p3
R1	1	0	0
R2	1	1	1

(c) Optimal decomposition of UPA into UA and PA with k=2.

Figure 2: An example of mapping MinNoise RMP to DBP

modify the original LTM algorithm [6] to return the largest uncovered tile. For this, we keep track of the current largest uncovered tile, LT, and its uncovered area, LTarea. The main steps of the algorithm are as follows:

Step 1: Originally, LT and LTarea are initialized to the empty set and 0, respectively. The current set of permissions being considered, P is also initialized to the empty set. Lines 1 and 2 perform this initialization.

Step 2: Line 3 starts the main loop of the algorithm, and iterates over each permission separately. On lines 4-7, if the uncovered area of the current tile being considered is larger than the current known best, the best is updated to this. i.e., LT and LTarea always refer to the largest uncovered tile seen so far. Over here, we need to clarify what we mean by uncovered area. For any tile, the uncovered area is the number of 1s that the tile covers that are not already covered in the existing tiling – i.e., the uncovered area refers to that part of the tile that is new and not seen before.

Referring back to Figure 1(b), assume that the current tiling consists of Tile 1 and Tile 2. Now, the covered area is simply the distinct number of 1s included in the Tiling. In our case, since the tiles do not overlap, the overall covered area is equal to 10 (6 for Tile 1 and 4 for Tile 2).

Now, suppose we are considering Tile 3. The uncovered area of Tile 3 is 5 (since the total number of 1s in Tile 3 is 6, and one out of those 1s, at position {u2,p5} is already covered in the current tiling). Thus, given a database and an existing tiling, whenever a new tile is considered, it is easy to compute the uncovered area by simply removing the already covered area from the area of the tile.

Step 3: Lines 8-12 creates the conditional database UPA^P.

Step 4: Finally, line 13 invokes the algorithm recursively to calculate the largest uncovered area in the smaller conditional database. Since the conditional database progressively shrinks, the algorithm is guaranteed to finish after all the permissions have been considered. The

algorithm shown here is quite simple. However, its efficiency can be significantly improved by using several pruning techniques – more details can be found in [6].

Algorithm 2 LUTM(UPA,T)

1: $P \leftarrow \{\}$
2: $LT \leftarrow \{\}$, $AreaLT \leftarrow 0$
3: **for** $\forall p \in PRMS$ **do**
4:    **if** uncovered area of $t(P \cup \{p\}) > $ AreaLT **then**
5:       $LT \leftarrow t(P \cup \{p\})$
6:       Update $AreaLT$ to have uncovered area of $t(P \cup \{p\})$
7:    **end if**
8:    {Create the conditional database for recursion}
9:    $UPA^{(P \cup \{p\})} \leftarrow \{\}$
10:    **for** $(\forall q | (q \in PRMS) \cap (q > p))$ **do**
11:       Add $(q, U(\{p\}) \cap U(\{q\}))$ to $UPA^{(P \cup \{p\})}$
12:    **end for**
13:    Compute $T((P \cup \{p\}), UPA^{(P \cup \{p\})})$ recursively
14: **end for**

---

5. MAPPING THE MINNOISE RMP TO THE DISCRETE BASIS PROBLEM

In this section, we demonstrate the direct equivalence of the MinNoise RMP to the Discrete Basis problem. This mapping again allows us to directly borrow existing implementation solutions. Miettinen, in his thesis [13], studies a set of three related problems and shows that these are NP-complete. We now describe the relevant problems studied and then discuss their implications.

The Discrete Basis problem [14] studies the problem of finding a basis from given data. Similar to Principal Component Analysis (PCA), the discrete basis problem is a technique for simplifying a dataset, by reducing multidimensional datasets to lower dimensions for summarization, analysis, and/or compression. Unlike PCA, the discrete basis problem only considers boolean data, and finds boolean bases.

We have already introduced some of the notation used for defining the discrete basis problem from [14]. Formally, the discrete basis problem is defined as follows:

DEFINITION 13   (DISCRETE BASIS PROBLEM). *Given a matrix $C \in \{0,1\}^{n \times d}$ and a positive integer $k \leq min\{n,d\}$, find a matrix $B \in \{0,1\}^{k \times d}$ minimizing*

$$l_\otimes(C,B) = min_{S \in \{0,1\}^{n \times k}} \parallel C - S \otimes B \parallel_1$$

The Discrete Basis Problem only asks for a discrete basis. A related problem is the Basis Usage problem:

DEFINITION 14   (BASIS USAGE PROBLEM). *Given a matrix $C \in \{0,1\}^{n \times d}$ and a matrix $B \in \{0,1\}^{k \times d}$, find a matrix $S \in \{0,1\}^{n \times k}$ minimizing*

$$\parallel C - S \otimes B \parallel_1$$

Together, the Discrete Basis Problem and the Basis Usage Problem correspond to the MinNoise RMP. C represents the user-privilege assignment, UPA. B represents the role-permission assignment, PA. S represents the user-role assignment UA. The following example clearly demonstrates this equivalence.

In the context of the discrete basis problem, the input is a boolean matrix, where the rows and columns might stand for anything – users and permissions, or documents and words. For now, we assume that these show the user-permission assignment, UPA. Thus, Figure 2(a) is a $n \times m$ input binary matrix where $n = 4, m = 3$. Given the positive integer $k = 2$ ($k < min\{m,n\}$), Figure 2(b) shows one possible decomposition into a usage matrix S and basis vector matrix B. As we can see, in this case $|C - S \otimes B|$ is 2.[2] Figure 2(c) shows a better decomposition since $|C - S \otimes B| = 0$. Indeed this is the best (optimal) decomposition possible for the given input matrix. Note that the discrete basis problem only asks for the optimal basis B (i.e., role-permission assignment PA). Given B, the basis usage problem asks for the optimal usage matrix S (i.e., user-role assignment UA). In our case, the MinNoise RMP asks for both PA and UA together. The difference is semantic – in either case, the problem (as stated) is NP-complete [13].

However, splitting the problem into two parts (i.e., finding optimal PA, and then finding optimal UA given PA) does help in the case of the basic RMP. For the basic RMP, we wish to *exactly* match the given UPA. In this case, while the discrete basis problem (finding optimal PA) remains NP-hard, the basis usage problem (finding UA given PA) becomes polynomial. A simple algorithm for the basis usage problem in this case is as follows: For each user and for each role, if the set of permissions of the role is a subset of the permissions of the user, then assign that role to that user. Since we only assign a role to a user as long as all of its permissions are owned by the user, there are no mistakes (and we have an exact match). Obviously, this assumes that the provided basis is complete (i.e., that each user can be exactly described using some subset of the roles), and thus all of the required roles are assigned to the user. Thus, after going through the entire set of users and permissions, we automatically come up with the optimal UA. The running

time of this algorithm is clearly polynomial in the size of the input [13].

Miettinen [13] also shows that the discrete basis problem cannot be approximated to in polynomial time within any constant factor unless $P = NP$. This essentially shuts the door on any attempt to find an approximation algorithm for the problem. However, heuristic solutions based on association rule mining are proposed and seem to give fairly good results on simulated data. Again, [13] provides further details on this. Other heuristics can also be used. One possibility is to extend the RoleMiner algorithm [21] to find the best candidates to describe the dataset. As part of future work, we intend to comprehensively test a set of heuristics (including the one in [13]) to determine what really works well in our domain.

6. CONCLUSIONS AND FUTURE RESEARCH

In this paper, we have formally defined the *role mining problem* (RMP) for conducting a bottom-up role engineering. In addition to the basic RMP, we also define the *δ-approx RMP* and the *MinNoise RMP* that are useful when performing role mining in real world settings. We have analyzed the theoretical bounds of the basic RMP as well as its variants and have shown that all of them are NP-complete problems. We have mapped these problems to the recently proposed problems in the area of data mining and data analysis – the database tiling and the discrete basis. As a result, we could borrow the implementation solutions proposed for these problem and directly apply them to solve the basic RMP and MinNoise RMP. We are currently working towards a solution to the δ-approx RMP variant.

Also, in mathematics, the problem of finding boolean rank / schein rank of a matrix is exactly the same as the basic RMP. It has been earlier proven that finding the Schein rank is NP-complete [11]. This matches our results. Other properties of the boolean rank have also been studied [1]. It would be interesting to investigate what other results are directly applicable to our problem and see if they offer new insight into our domain. Bipartite graphs and bicliques are another way of defining the RMP and its variants. Several papers have looked at different variants of this (e.g., [8] and [17]) – though most concentrate on finding one biclique from a bipartite or general graph. Conjunctive clustering [15] generalizes this to finding multiple bicliques, which is more relevant to our problem. We also need to see which solutions among this work can be utilized for our problem.

Since the RMP and its variants are NP-complete, it is important to come up with heuristic strategies for achieving implementations with reasonable complexity. In fact, the recently proposed RoleMiner solution [21] could also serve as a heuristic strategy for the basic RMP. We intend to investigate this and other possibilities in real settings to create a set of tools for the security administrator. Moreover, most of the role mining approaches employ clustering techniques or its variants to discover roles. We are currently investigating other data mining techniques including association rule mining (specifically closed itemset mining [16, 12]) for role discovery.

---

[2]We keep the notations of matrix product and L_1 norm as what they originally are in DBP paper [14], even if they are slightly different with those used in RMP.

Acknowledgments

7. REFERENCES

[1] C. Damm, K. H. Kim, and F. Roush. On covering and rank problems for boolean matrices and their applications. In *Computing and Combinatorics: 5th Annual International Conference, COCOON'99*, volume 1627 of *Lecture Notes in Computer Science*, pages 123 – 133. Springer-Verlag, 1999.

[2] E.J.Coyne. Role-engineering. In *1st ACM Workshop on Role-Based Access Control*, 1995.

[3] D. Ferraiolo, R. Sandhu, S. Gavrila, D. Kuhn, and R. Chandramouli. Proposed nist standard for role-based access control. *TISSEC*, 2001.

[4] M. P. Gallagher, A. O'Connor, and B. Kropp. The economic impact of role-based access control. *Planning report 02-1, National Institute of Standards and Technology*, March 2002.

[5] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, chapter 3. W. H. Freeman, 1979.

[6] F. Geerts, B. Goethals, and T. Mielikainen. Tiling databases. In *Discovery Science*, Lecture Notes in Computer Science, pages 278 – 289. Springer-Verlag, 2004.

[7] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In W. Chen, J. Naughton, and P. A. Bernstein, editors, *2000 ACM SIGMOD Intl. Conference on Management of Data*, pages 1–12. ACM Press, 05 2000.

[8] D. S. Hochbaum. Approximating clique and biclique problems. *J. Algorithms*, 29(1):174–200, 1998.

[9] A. Kern, M. Kuhlmann, A. Schaad, and J. Moffett. Observations on the role life-cycle in the context of enterprise security management. In *7th ACM Symposium on Access Control Models and Technologies*, June 2002.

[10] M. Kuhlmann, D. Shohat, and G. Schimpf. Role mining - revealing business roles for security administration using data mining technology. In *Symposium on Access Control Models and Technologies (SACMAT)*. ACM, June 2003.

[11] G. Markowsky. Ordering d-classes and computing schein rank is hard. *Semi-group Forum*, 44:373–375, 1992.

[12] T. Mielikäinen. Intersecting data to closed sets with constraints. In B. Goethals and M. J. Zaki, editors, *FIMI*, volume 90 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2003.

[13] P. Miettinen. The discrete basis problem, master's thesis. Master's thesis, University of Helsinki, 2006.

[14] P. Miettinen, T. Mielikainen, A. Gionis, G. Das, and H. Mannila. The discrete basis problem. In *Knowledge Discovery in Databases: PKDD 2006*, Lecture Notes in Artificial Intelligence, pages 335 – 346, 2006.

[15] N. Mishra, D. Ron, and R. Swaminathan. On finding large conjunctive clusters. In *Learning Theory and Kernel Machines: 16th Annual Conference on Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003*, volume 2777 of *Lecture Notes in Computer Science*, pages 448 – 462. Springer, 2003.

[16] F. Pan, G. Cong, A. K. H. Tung, J. Yang, and M. J. Zaki. Carpenter: finding closed patterns in long biological datasets. In *KDD*, pages 637–642, 2003.

[17] R. Peeters. The maximum edge biclique problem is np-complete. *Discrete Appl. Math.*, 131(3):651–654, 2003.

[18] R. S. Sandhu et al. Role-based Access Control Models. *IEEE Computer*, pages 38–47, February 1996.

[19] A. Schaad, J. Moffett, and J. Jacob. The role-based access control system of a european bank: A case study and discussion. *In Proceedings of ACM Symposium on Access Control Models and Technologies*, pages 3–9, May 2001.

[20] J. Schlegelmilch and U. Steffens. Role mining with orca. In *Symposium on Access Control Models and Technologies (SACMAT)*. ACM, June 2005.

[21] J. Vaidya, V. Atluri, and J. Warner. Roleminer: mining roles using subset enumeration. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, pages 144–153, 2006.