A Class of Probabilistic Models for Role Engineering

Mario Frank
Department of Computer Science,
ETH Zurich, Switzerland
mario.frank@inf.ethz.ch

David Basin
Department of Computer Science,
ETH Zurich, Switzerland
basin@inf.ethz.ch

Joachim M. Buhmann
Department of Computer Science,
ETH Zurich, Switzerland
jbuhmann@inf.ethz.ch

ABSTRACT

Role Engineering is a security-critical task for systems using role-based access control (RBAC). Different role-mining approaches have been proposed that attempt to automatically infer appropriate roles from existing user-permission assignments. However, these approaches are mainly combinatorial and lack an underlying probabilistic model of the domain.

We present the first probabilistic model for RBAC. Our model defines a general framework for expressing user permission assignments and can be specialized to different domains by limiting its degrees of freedom with appropriate constraints. For one practically important instance of this framework, we show how roles can be inferred from data using a state-of-the-art machine-learning algorithm. Experiments on both randomly generated and real-world data provide evidence that our approach not only creates meaningful roles but also identifies erroneous user-permission assignments in given data.

Categories and Subject Descriptors: K.6 [Management of Computing and Information Systems]: Security and Protection

General Terms: Security, Management, Algorithms

Keywords: RBAC, Role Mining, Machine Learning, Clustering

1. INTRODUCTION

Role-based access control [7] is a popular, widely used approach to administer permissions in security-sensitive environments. A role equips a user with a set of permissions and ideally represents a function that the user has within an enterprise. An access control system based on roles is easier to maintain than one working directly with individual assignments of permissions to users. However, this requires roles to be defined and users and permissions to be assigned to them. This process, known as role engineering, is a non-trivial challenge due to the high dimensionality of the solution space. It is also an essential and critical task for using RBAC in real-world domains [4]. In most enterprises, this process is carried out manually, despite the high costs this entails and the security risk due to erroneous assignments to roles. Automated approaches are therefore desirable since they have the potential to dramatically reduce both the costs and security risks.

One can distinguish two kinds of approaches to role engineering: *top-down* and *bottom-up*. Top-down engineering uses process descriptions, the organizational structure of the domain, or features of the employees as given by the Human Resources Department of an enterprise, to create roles. Currently, there are only manual top-down approaches. In [5], for instance, a work flow is proposed to manually engineer roles by analyzing business processes. In [14], a scenario-driven approach is presented where a scenario's requirements are analyzed according to its associated tasks. Permissions are granted that enable the task to be completed.

Bottom-up role engineering identifies roles by analyzing the existing assignments of users to permissions. Roles, and assignments from roles to users and from roles to permissions, must be found that approximate the existing user-permission assignments as best possible while limiting the number of roles that must be maintained. A number of automated, bottom-up approaches exist and, in this context, role engineering is often referred to as *role-mining*.

Existing bottom-up methods aim to approximate the user-permission assignments as *best possible* by finding a minimal set of roles, user-role assignments, and role-permission assignments, using mainly combinatorial methods. Such approaches have two major drawbacks. First, the existing assignments may contain errors. If the approach does not allow one to predicate how *likely* it is to observe a particular assignment, these errors cannot be identified as such and therefore are migrated to the RBAC system. Second, role engineering is not just a data compression problem. The roles should be as meaningful as possible with respect to the users assigned to them. They should ideally represent the particular job functions that groups of users have in a domain. Combinatorial methods that aim to minimize differences with the original assignments often result in synthetic roles that are difficult to understand and generalize poorly to new users.

These problems result from the lack of an underlying statistical model for role mining. In this paper, we propose a class of probabilistic models that subsumes different bottom-up role engineering scenarios for RBAC. We show how particular instances of our model class can be defined according

CCS'08, October 27–31, 2008, Alexandria, Virginia, USA.

to the given domain requirements. The advantage of a probabilistic approach is the ability to generalize from observations about existing assignments. This allows us to identify wrong or missing assignments and avoids the migration of these errors. Moreover, generalization facilitates the addition of new users with minimal information about them.

A statistical approach has another major advantage compared to pure combinatorial algorithms: A large number of users and permissions will improve the result since observing many existing user-permission assignments supports improved predictions of the model parameters. In contrast, combinatorial models suffer from increased computational costs and the tendency to perturb succinct role definitions by exceptions and erroneous user-permission assignments.

We also address algorithmic aspects in this paper and show, for one of our models, how to adopt standard inference algorithms to solve the role mining problem. We design experiments on randomly generated and real-world data in order to assess the quality of the resulting roles. Our results support the thesis that using a sound probabilistic model is advantageous for role engineering.

Overall, we see our contributions as follows. First, by introducing a model class that expresses the role mining problem in a probabilistic way, we provide an approach that addresses the two problems mentioned above: our approach prevents the migration of erroneous assignments to an RBAC system by identifying errors and infers meaningful roles instead of synthetic ones. Second, our model's structure supports extensions to hybrid data mining approaches combining information in both a top-down and bottom-up way. Such approaches are desirable since, for example, they simplify adding new employees to the system. Finally, we introduce two new quantitative measures on the quality of the role engineering results.

The remainder of this paper is organized as follows. After surveying related work in Section 2, we propose a new class of probabilistic models for role mining in Section 3 and introduce several instances of this model class. In Section 4, we present an inference algorithm for one of these instances and then we report on experiments with both generated and real data. As part of our experiments, we compare our probabilistic approach to the combinatorial method proposed in [12]. In Section 5, we summarize our results and indicate directions for future work.

2. RELATED WORK

Formal Representation.

The role-mining problem can be formulated as follows [16]. The user-permission assignments are given by an $M \times N$ Boolean assignment matrix $\mathbf{x}$. Rows represent users and columns represent permissions. If $x_{ij} = 1$, then user i has permission j, where $i \in \{1, .., M\}$ and $j \in \{1, .., N\}$. An $M \times K$ user-role assignment matrix $\mathbf{z}$ and a $K \times N$ role-permission assignment matrix $\mathbf{u}$ must be found that express the original matrix as the Boolean matrix product $\mathbf{x} = \mathbf{z} \otimes \mathbf{u}$. This is product is defined by

$$x_{ij} = \bigvee_k \left[z_{ik} \wedge u_{kj} \right] , \qquad (1)$$

where $k \in \{1, .., K\}$ is the index of a role. Most role-mining approaches use this model structure.

Role Interpretations and Algorithms.

In [16], roles are treated as sets of permissions: Each row in $\mathbf{u}$ is a role, where membership of permissions in the role is indicated by ones. Equivalently, a user i is characterized by the set of permissions that he owns. In the above model, these sets are determined by the row i in $\mathbf{x}$. As we explain in Section 3.1, this model is equivalent to one of the instances of our model class if no underlying probability distribution is considered. In the algorithm proposed in [17], all existing users are initially considered as candidate roles. Thus, each candidate role consists of all permissions that are assigned to a particular user. Afterwards, candidate roles are picked in a greedy manner to determine the final set of roles.

In [18], the roles are also represented as sets of permissions. Candidate roles are generated and then merged, split, or placed in a role hierarchy, as determined by a small set of given rules. A similar procedure is proposed in [15]. But there, roles and permissions are represented as sets of users. Thus, using the notation above, a role k is a column k in $\mathbf{z}$. The initial roles are constructed from existing permissions. Namely, an initial role is the set of users that are assigned to a given permission (a column in $\mathbf{x}$). Again, the initial roles are iteratively merged, split, or placed in a role hierarchy according to the cardinality of intersections of the roles.

In [10], roles are computed by randomly merging sets of permissions. The merging process is iterated until there is a complete tree structure of role proposals. From this tree, the final roles are selected by analyzing the number of associated users.

Equivalence with other Problems.

Three different variants of the bottom-up role-mining problem (RMP) are formally modeled in [16].

1. Find a minimal set of roles that express all existing user-permission assignments (Basic RMP).

2. Approximate the existing assignments with a minimal number of roles with a precision up to a given error δ (δ-RMP).

3. Approximate the existing assignments with minimal errors for a given number of roles (Min-Noise RMP).

The complexity of these problems is explored by analyzing their associated decision problems. For example, Basic RMP corresponds to the Set Basis Problem, defined in [8]: Given a collection of sets $S = \{S_1, S_2, \ldots, S_N\}$, find a basis $B = \{B_1, B_2, \ldots, B_K\}$ with least cardinality K such that for each S_i there exists a representation as a union of a subset of B. The solution proposed in [8] is a greedy algorithm that chooses from the basis candidate sets B_k, constructed from the pairwise intersections $S_i \cap S'_i$. In the context of role mining, the basis sets B_k are the roles and the original sets are the users S_i represented as sets of their permissions.

Without focusing on role mining, [12] considers a variant of the Set Basis Problem, called the Discrete Basis Problem (DBP). This problem requires *approximating* a collection of sets $S = \{S_1, S_2, .., S_N\}$ by a set of basis sets $B = \{B_1, B_2, .., B_K\}$ with a *given* cardinality K. A greedy algorithm is proposed in [12] that searches for such a basis set. We will explain this algorithm in the appendix.

In [11], DBP is identified as a variant of RMP. There, DBP, RMP, and SBP are formally modeled as binary integer programming problems. To approximately solve these

problems quickly, the authors present an extension to the greedy algorithm of [17].

3. A PROBABILISTIC MODEL

In this section, we present a new model of the role mining problem and derive its underlying probabilistic representation. As seen in the last section, in existing approaches, the role mining problem is usually modeled by decomposing the user-permission assignment matrix $\mathbf{x}$ into the product of two Boolean matrices $\mathbf{z} \otimes \mathbf{u}$. This matrix representation exhibits an inherent symmetry: Roles can either be regarded as groups of permissions assigned to the same users (the rows in $\mathbf{u}$) or as groups of users sharing the same permissions (the columns in $\mathbf{z}$). This symmetric representation has a major drawback: If users are only grouped according to their permissions and permissions only according to their associated users, then it is difficult to later add additional features for users or permissions. However, this is a necessary step for the extension of the model to a hybrid bottom-up/top-down approach. Our model breaks this symmetry by separately assigning users to user-groups and permissions to permission-groups and afterwards assigns user-groups to permission-groups. In this manner, users and permissions can be grouped separately according to prior information. In the following, we describe this decomposition in further detail.

We assume that there exists a decomposition of the set of users into groups that are not necessarily disjoint: Users are assigned to one or more groups by a Boolean assignment matrix $\mathbf{z}$. Each row i represents a user and the columns k represent user-groups. In practice, such a decomposition may be performed by the Human Resources Department of an enterprise, for example, by assigning users to divisions in the enterprise according to defined similarities of the employees. If such data is lacking, then the decomposition may just be given by the differences in the assigned permissions for each user.

Moreover, we assume that there is a decomposition of the permissions such that every permission belongs to one or more permission-groups. These memberships are expressed by the Boolean assignment matrix $\mathbf{y}$. Here the lth row of $\mathbf{y}$ represents the permission-group l and the jth column is the permission j. The assignment of permissions to permission-groups can be motivated, for instance, by technical similarities of the resources that the permissions grant access to. For example, in an object-oriented setting, permissions might be grouped that execute methods in the same class. Alternatively, permissions could be categorized based on the risk that is associated with granting someone a particular permission. Of course, permissions can also be grouped according to the users who own them.

For the moment, we do not consider the structure of permissions and users but we simply assume that there exists a prior structure. We denote user-groups by *business roles* whereas permission-groups are referred to as *technical roles*. Business roles are assigned to technical roles. We represent these assignments in a matrix $\mathbf{u}$. The above-mentioned Boolean assignment matrices have the following types:

- Users i to permissions j: $x_{ij} \in \{0,1\}$, where $i \in \{1,..,M\}$ and $j \in \{1,..,N\}$.
- Users i to business roles k: $z_{ik} \in \{0,1\}$, where $k \in \{1,..,K\}$.
- Technical roles l to permissions j: $y_{lj} \in \{0,1\}$, where $l \in \{1,..,L\}$.
- Business roles k to technical roles l: $u_{kl} \in \{0,1\}$.

Throughout this paper, the indices i, j, k, and l have the above scope and are used to index the above objects. Using this notation, the final $M \times N$ user-permission assignment matrix $\mathbf{x}$ is determined by the Boolean matrix product

$$\mathbf{x} = \mathbf{z} \otimes \mathbf{u} \otimes \mathbf{y} \quad \text{with} \quad x_{ij} = \bigvee_k \left[z_{ik} \wedge \left(\bigvee_l u_{kl} \wedge y_{lj} \right) \right] . \quad (2)$$

With these conventions, Equation 2 expresses when a user i is assigned to a permission j. This is graphically illustrated in Figure 1(a) in Section 4: A user is assigned to a permission if there is at least one path in the graph connecting them. As motivated in the introduction, we are interested in what the probability of such an assignment is. Starting from the logical expression, we derive below how likely it is to observe an assignment of a user i to a permission j. Being able to compute this probability, one can infer the unknown assignments $\mathbf{z}$, $\mathbf{u}$, and $\mathbf{y}$ such that the direct assignments $\mathbf{x}$ become most likely. In Section 4 we will give such an inference algorithm.

Since a permission may belong to multiple technical roles and a user may belong to multiple business roles, which in turn may be assigned to multiple technical roles, a user can be assigned to a permission in more than one way (cf. Figure 1(a): there may be multiple connecting paths). Therefore, it is easier to express how a user may *not* be assigned to a permission (we denote this by $\neg x := \overline{x}$) rather than computing the union over all possible assignment paths.

$$\begin{aligned}
p\left(\overline{x_{ij}}\right) &= p\left(\overline{\bigvee_k \left[z_{ik} \wedge \left(\bigvee_l u_{kl} \wedge y_{lj} \right) \right]} \right), \quad \begin{matrix} 1 \le k \le K \\ 1 \le l \le L \end{matrix} \\
&= \prod_k p\left(\overline{z_{ik} \wedge \underbrace{\left(\bigvee_l u_{kl} \wedge y_{lj} \right)}_{=:b_k}} \right) \\
&= \prod_k \left[p\left(z_{ik} \wedge \overline{b_k}\right) + \underbrace{p\left(\overline{z_{ik}} \wedge b_k\right) + p\left(\overline{z_{ik}} \wedge \overline{b_k}\right)}_{=p(\overline{z_{ik}})} \right] \\
&= \prod_k \left[p(\overline{z_{ik}}) + p(\overline{b_k}) p(z_{ik}) \right] \quad (3)
\end{aligned}$$

Note that in the step from the second to the third line, the correct probability is only obtained when summing over the probabilities of exclusive events (in particular: $\overline{a \wedge b} = \overline{a} \vee \overline{b}$ but $p\left(\overline{a \wedge b}\right) \neq p\left(\overline{a}\right) + p\left(\overline{b}\right)$). Given the definition of b_k, we have that

$$\begin{aligned}
p(\overline{b_k}) &= \prod_l p\left(\overline{u_{kl} \wedge y_{lj}}\right) \\
&= \prod_l \left[p(\overline{y_{lj}}) + p(\overline{u_{kl}}) p(y_{lj}) \right] . \quad (4)
\end{aligned}$$

Therefore, substituting this into $p(\overline{x_{ij}})$ yields

$$p(\overline{x_{ij}}) = \prod_k \left[p(\overline{z_{ik}}) + p(z_{ik}) \prod_l \left(p(\overline{u_{kl}}) p(y_{lj}) + p(\overline{y_{lj}}) \right) \right] . \quad (5)$$

This expression can be conditioned on the binary entries of $\mathbf{y}$ and $\mathbf{z}$.

$$p(\overline{x_{ij}} \mid z_i, y_j) = \prod_k 1^{1-z_{ik}} \cdot \left(\prod_l p(\overline{u_{kl}})^{y_{lj}} \cdot 1^{1-y_{lj}} \right)^{z_{ik}} = \prod_{k,l} p(\overline{u_{kl}})^{y_{lj} z_{ik}} \quad (6)$$

Using this, we can express the complete likelihood of the user-permission assignment matrix given the business roles and technical roles.

$$\begin{aligned} & p(\mathbf{x} \mid \mathbf{z}, \mathbf{y}) \\ = & \prod_{i,j} [1 - p(\overline{x_{ij}} \mid z_i, y_j)]^{x_{ij}} [p(\overline{x_{ij}} \mid z_i, y_j)]^{1-x_{ij}} \\ = & \prod_{i,j} \left[1 - \prod_{k,l} p(\overline{u_{kl}})^{y_{lj} z_{ik}} \right]^{x_{ij}} \left[\prod_{k,l} p(\overline{u_{kl}})^{y_{lj} z_{ik}} \right]^{1-x_{ij}} \end{aligned} \quad (7)$$

The complete data likelihood is then given by

$$p(\mathbf{x}, \mathbf{z}, \mathbf{y}) = p(\mathbf{x} \mid \mathbf{z}, \mathbf{y})\, p(\mathbf{z})\, p(\mathbf{y}) \,. \quad (8)$$

3.1 Instantiation by Introducing Constraints

The above model of user-permission assignments in an RBAC environment defines a very general framework. In particular, we have avoided any assumptions about the probabilities of the entries of $\mathbf{u}$, $\mathbf{y}$, and $\mathbf{z}$. In the derivation, we have only exploited the fact that these variables are Booleans and, therefore, they only take the values 0 or 1. Also, we have avoided any assumptions about the processes that lead to a particular decomposition of the set of users and the set of permissions. Moreover, we have not specified any possible constraints on the user decomposition, the permission decomposition, or the assignments from user-groups to permission-groups.

It turns out that this general model (as we will subsequently refer to it) has more degrees of freedom than required to represent the access control information present in many domains that arise in practice. It can be seen as a template for an entire class of models. By introducing constraints, we can instantiate this template to specialized models that fit the requirements of particular RBAC environments. These instances of the model class are given by augmenting the general model with assumptions on the probability distributions of the binary variables and giving constraints on the variables themselves. In the following, we will present three different models, which provide evidence of the generality of this model class.

Trivial Decomposition of the Set of Permissions.

In this model, each permission is restricted to be a member of only one permission-group and each permission-group can contain only a single permission. Formally:

$$\left(\forall l : \textstyle\sum_j y_{lj} = 1\right) \wedge \left(\forall j : \textstyle\sum_l y_{lj} = 1\right) \,. \quad (9)$$

The conditioned likelihood then becomes

$$p(\mathbf{x} \mid \mathbf{z}) = \prod_{i,j} \left[1 - \prod_k p(\overline{u_{kj}})^{z_{ik}} \right]^{x_{ij}} \left[\prod_k p(\overline{u_{kj}})^{z_{ik}} \right]^{1-x_{ij}} . \quad (10)$$

This "collapsed" model has the same symmetry present in the original model, which was discussed in the introduction. Equivalently, the same constraints could instead be applied to the users, leading to a model with the same structure. Due to this symmetry, the model is not capable of representing a distinction between business roles and technical roles.

This model suffices, however, to approximate existing assignments in a pure bottom-up approach. If we constrain the total number of roles, then we derive a probabilistic version of the Role Mining Problem defined in [16] and the Discrete Basis Problem presented in [12]. A graphical representation of the structure of this instance is given in Figure 1(b).

Upper-bounded Assignments to the User-groups.

It is useful in some domains to place an upper bound k_{max} on the number of business roles that a user belongs to: $\forall i : \sum_k z_{ki} \leq k_{max}$. Suppose, for example, that company employees can be classified based on the different business areas that they work in or the different kinds of contracts they have. The business roles can be used to formalize these categories on the set of employees. Moreover, this structure would naturally limit the number of business roles, e.g., to the number of different contract categories. Such a setting would also support the addition of new users to the system by Human Resources.

Note that even with a limited number of business roles, a user may have multiple technical roles.

Disjoint Decomposition Model.

Our next model has even stronger constraints. Namely, $k_{max} = 1$ and the number of assigned permission-groups per permission is limited to $l_{max} = 1$. This formalizes that each user belongs to exactly one user-group and each permission belongs exactly to one permission-group. Hence, both users and permissions are partitioned into disjoint business roles and technical roles, respectively. Some enterprises favor a disjoint decomposition because it reduces the complexity of the system while still retaining a high degree of flexibility, since users of a given user-group may still be assigned to multiple permission-groups. The technical advantage of this model is that inference is much easier if the business roles and the technical roles are disjoint. This can be seen in the conditioned likelihood, which now takes the convenient form

$$p(\mathbf{x} \mid \mathbf{z}, \mathbf{y}) = \prod_{k,l} [1 - p(\overline{u_{kl}})]^{n_{kl}^{(1)}} [p(\overline{u_{kl}})]^{n_{kl}^{(0)}} \text{ with } \quad (11)$$

$$n_{kl}^{(1)} = \sum_{\substack{i:z_{ik}=1,\\ j:y_{lj}=1}} \delta(x_{ij} - 1), \quad n_{kl}^{(0)} = \sum_{\substack{i:z_{ik}=1,\\ j:y_{lj}=1}} \delta(x_{ij}) \,.$$

Thus, we essentially have to count the number of assignments $n_{kl}^{(1)}$ (missing assignments $n_{kl}^{(0)}$) for each (user-group, permission-group) pair (k, l). Since both the user and permission groups are disjoint, there is a simple, illustrative representation for data generated or inferred by this model. Namely, the original assignment matrix $\mathbf{x}$ can be drawn with an order on the rows and columns that is given by the assignments to the groups. All rows (users) of the same group are ordered adjacent to each other and all columns (permissions) of a permission group are also ordered adjacent to each other. See Figure 2(b) for an example of this representation. The model structure is illustrated in Figure 1(c).

In the next section, we will describe an existing algorithm for inferring model parameters and illustrate it for this in-

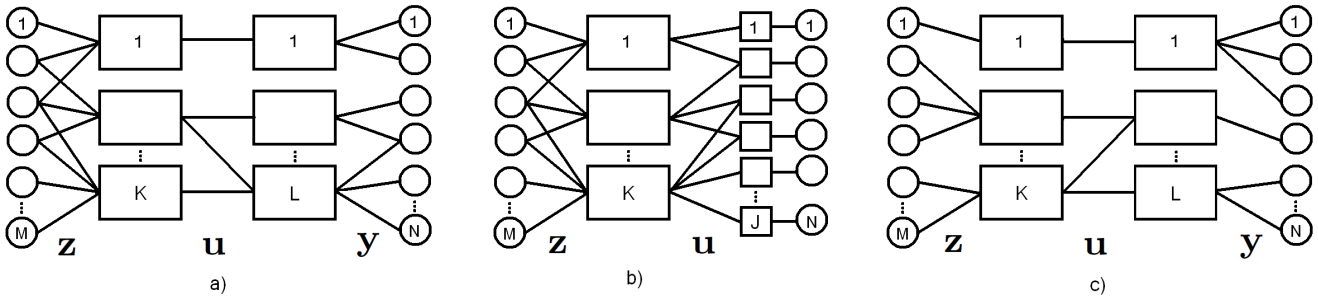

Figure 1: Illustration of the structure of three model instances. A user has a permission if there is at least one path connecting them. a) Full model. b) Model with trivial decomposition of the permissions. c) Disjoint Decomposition Model with only one business role per user and one technical role per permission.

stance of the model class. Afterwards we will present experimental results on both generated and real data.

4. INFERENCE FOR THE DISJOINT DECOMPOSITION MODEL

In this section, we present an inference algorithm well-suited for inferring model parameters for our last model, the Disjoint Decomposition Model (DDM).

The likelihood function $p(\mathbf{x} \mid \mathbf{z}, \mathbf{y})$ in Equation (11) expresses how probable it is to observe the actual user-permission assignments $\mathbf{x}$ for given values of the parameters $\mathbf{z}$ and $\mathbf{y}$. Given this function, the maximum likelihood principle provides a basis for selecting the parameters $\mathbf{z}$ and $\mathbf{y}$ that yield the highest probability of the data $\mathbf{x}$. The underlying probability distributions of the binary assignment variables $p(\overline{u_{kj}})$ is assumed to be Bernoulli, i.e., $p\{\overline{u_{kl}} = 0\} = \beta_{kl}$, for $0 \leq \beta_{kl} \leq 1$. Under this assumption, the conditioned data likelihood of DDM is

$$p(\mathbf{x} \mid \mathbf{z}, \mathbf{y}, \beta) = \prod_{k,l} [1 - \beta_{kl}]^{n_{kl}^{(1)}} [\beta_{kl}]^{n_{kl}^{(0)}} . \qquad (12)$$

The equations for the other instances from Section 3.1 can easily be produced by substituting the probabilities into the respective likelihood functions.

Maximizing the likelihood is a non-trivial optimization problem since the parameter space often has a high dimension. There exist many methods in the literature and surveying them would exceed the scope of this paper. Therefore, we limit ourselves to describing the main ideas behind an off-the-shelf algorithm that we have straightforwardly adapted to our problem.

The structure of the DDM is equivalent to the structure of the Infinite Relational Model (IRM), presented in [9]. The IRM is used there to cluster objects (here the users) to object-groups (user-groups) by their relations to features (the permissions). Reciprocally, features (permissions) are partitioned into feature-groups (permission-groups) according to their relation to objects (users). We essentially solve the same problem (infer the z_{ik} and y_{lj}) with the additional objective of deciding which user-groups are assigned to which permission-groups (infer the u_{kl}). Hence, we can use the same assumptions on the underlying probability distributions given in [9], which we summarize in the next section.

4.1 A Gibbs Sampling Algorithm

We now describe a sampling algorithm that can be used to infer the model parameters of the Disjoint Decomposition Model. The algorithm randomly samples assignments from a Dirichlet process mixture model by using Gibbs sampling as presented in [13] (Algorithm 3). Here, we shall summarize it using our notation.

The main idea behind the inference algorithm is to iteratively assign users to user-groups. At each iteration, each user is randomly assigned to a group according to the probability that the user belongs to that group (this can also be the group that the user already has been assigned to in the previous iteration). Alternatively, again with a probability that must be computed at each step, the user creates a new user-group with the user being the only member. In later iterations, other users may be assigned to that user-group as well (with a given probability, as explained below). User-groups that remain empty after an update step are deleted.

After each iteration over the users, the same procedure is performed on the permissions, while the user-role assignments are kept constant. The iterations cluster the original user-permission assignment matrix $\mathbf{x}$ by alternately moving whole rows and columns. Theoretically, Gibbs sampling always converges to the global optimum in the limit of infinite time. Our algorithm terminates either upon convergence or after reaching a predefined maximal number of iterations.

As described, the behavior of the Gibbs sampler is determined by the probability distributions over the assignments. In each step, the Gibbs sampler successively calculates the probabilities of the following $K+1$ exclusive events for each user i': assigning i' to one of the K existing user-groups k or creating a new role $k = K + 1$. To randomly assign a user to an existing user group, we need only compare the probabilities of different choices with respect to each other. Hence, we only have to compute the probabilities up to a constant factor.

$$p(z_{i'k} = 1 \mid \mathbf{x}, \mathbf{z}_{i \neq i',\cdot}, \mathbf{y}) \propto p(\mathbf{x} \mid \mathbf{z}, \mathbf{y})\, p(z_{i'k} = 1 \mid \mathbf{z}_{i \neq i',\cdot}) \quad (13)$$

When assigning the user i' to the user-group k, the current user-role assignments for all users other than i' are used to compute the probabilities.[1] Note that the formula (13) is written for sampling steps on the users. When the algorithm

[1] $\mathbf{z}_{p_r(i),p_c(k)}$ is the matrix that contains the rows of $\mathbf{z}$ where the predicate $p_r(i)$ holds and the columns where the predicate $p_c(k)$ holds. Here $\cdot$ denotes the always true predicate.

runs on permissions, then z must be replaced by y, k by l, and i by j, throughout the formula.

In the following, we show how the probability parameters are set in [9] for the terms of this equation. Probabilities of the user-permission assignments are

$$\begin{aligned} p\{u_{kl} = 0 \mid \mathbf{z}, \mathbf{y}, \beta\} &= \beta_{kl} \\ p\{u_{kl} = 1 \mid \mathbf{z}, \mathbf{y}, \beta\} &= 1 - \beta_{kl}\,. \end{aligned} \tag{14}$$

Thus, knowing the membership of users and permissions, the probability of a user being assigned to a member of the permission-group l is the same for all users of the role k. It is the probability of the assignment u_{kl} of the business role k to the technical role l. Each β_{kl} has a prior probability (the probability of β_{kl} to take a given value without knowing the data) given by the Beta distribution $P_b(\beta_{kl}; \gamma, \gamma)$ with the positive hyperparameter γ.

The term $p(z_{i'k} = 1 \mid \mathbf{z}_{i \neq i', k})$ in the right-hand side of (13) denotes the Dirichlet process [2, 6] that determines the a priori probabilities of the assignments of users to business roles without knowing the permissions for these users. This term penalizes assignments according to the number of roles and their cardinalities.

$$p(z_{i'k} = 1 \mid \mathbf{z}_{i \neq i', \cdot}) = \begin{cases} \frac{n_k}{N-1+\alpha} & n_k > 0 \\ \frac{\alpha}{N-1+\alpha} & n_k = 0 \end{cases} \tag{15}$$

It thereby decreases the probabilities of excessive creation of unnecessary roles. In this way, the Dirichlet process implements inference without knowing the number of roles in advance. It can, in theory, generate an infinite number of roles, but it penalizes unnecessary roles by weighting these user-group probabilities proportional to their cardinality n_k. N is the total number of users and α is a nonnegative hyperparameter that can be interpreted as the cardinality that each hypothetical role has in advance. To derive and explain the Dirichlet process in detail is outside the scope of this paper. In our context, just consider it as a weighting factor that avoids the excessive creation of new roles.

The first term on the right-hand side of (13) is called *evidence*. It denotes the probability $p(\mathbf{x} \mid \mathbf{z})$ of a particular user given the assignments to a user-group. The complete evidence for all users and permissions is computed from the conditioned likelihood as in (12) and the Beta distribution $P_b(\beta_{kl}; \gamma, \gamma)$ and is

$$p(\mathbf{x} \mid \mathbf{z}, \mathbf{y}) = \prod_{k,l} \frac{\mathrm{Beta}(n_{kl}^{(1)} + \gamma, n_{kl}^{(0)} + \gamma)}{\mathrm{Beta}(\gamma, \gamma)}\,. \tag{16}$$

Here Beta(.,.) is the Beta function (also known as "Euler integral of the first kind") that appears in the Beta distribution $P_b(.;.,.)$ as a normalization factor.

In summary, the algorithm assigns rows of the user-permission assignment matrix (users) to groups where the other rows (members of the same group) are similar to that row. For such groups, the assignment probability distributions have very sharp maxima compared to other suboptimal assignments. These random assignments are computed by alternating iterations over the rows and columns, as can be seen in the box below. Thus, the assignment matrix iteratively becomes more clustered.

The matrices in Figure 2 illustrate how this procedure looks in practice. Rows are users, columns are permissions, and dots indicate assignments between users and permissions. The assignments in (a) are randomly generated according to the general model (7). The roles are then inferred with the Disjoint Decomposition Model (11). Since all business roles and all technical roles are disjoint, the original matrix can be ordered according to the members of their entities, as depicted in (b). As described in Section 4.2, erroneous assignments and missing assignments can easily be identified in this arrangement. They could be reported or even automatically removed if desired, as shown in (c).

input : binary matrix $\mathbf{x}$ as defined in Section 3, α, γ, $iter_{\max}$ and $d_{\min}$
output: role assignment matrices $\mathbf{z}$ and $\mathbf{y}$ as defined in Section 3

$\mathbf{z} \leftarrow \mathbf{0}$; $z_{i1} \leftarrow 1$, for all i
$\mathbf{y} \leftarrow \mathbf{0}$; $y_{1j} \leftarrow 1$, for all j
for $iter \leftarrow 1$ **to** $iter_{max}$ **do**
  $\mathbf{z}^{\mathrm{old}} \leftarrow \mathbf{z}$; $\mathbf{y}^{\mathrm{old}} \leftarrow \mathbf{y}$
  for $i \leftarrow 1$ **to** M **do**
    $z_{ik} \leftarrow 0$, for all k
    $z_{ik} \leftarrow 1$, for one k sampled from Eq. 13
  end
  for $j \leftarrow 1$ **to** N **do**
    $y_{lj} \leftarrow 0$, for all l
    $y_{lj} \leftarrow 1$, for one l sampled from Eq. 13
  end
  if $\left(d_{min} < d\left(\mathbf{z}^{old}, \mathbf{z}\right)\right) \wedge \left(d_{min} < d\left(\mathbf{y}^{old}, \mathbf{y}\right)\right)$ **then**
    break
  end
end

As outlined in the box above, in every iteration of the main loop, the function $d(\cdot, \cdot)$ computes the fraction of entries of the current assignment matrices $\mathbf{y}$ and $\mathbf{z}$ that differ from their last state. The loop repeats until either the assignment matrices converge to a stable configuration or it breaks after a predefined maximal number of iterations.

4.2 Error Detection and Subpartitioning

The algorithm just described infers the assignments of users to user-groups and permission to permission-groups. These are the assignment matrices $\mathbf{z}$ and $\mathbf{y}$. We now explain how we determine the assignments between business roles and technical roles $\mathbf{u}$ given $\mathbf{z}$ and $\mathbf{y}$.

A pair consisting of a user-group and a permission-group (business role and technical role) is called a bicluster. Each bicluster has one empirical estimator $\hat{\beta}_{kl}$ of the probability that the entire user-group is assigned to the permission-group. All members of this bicluster share this assignment probability $\hat{\beta}_{kl} = (n_{kl}^{(1)} + \gamma)/(n_{kl}^{(1)} + n_{kl}^{(0)} + 2\gamma)$. Based on $\hat{\beta}_{kl}$, one must decide whether to set the corresponding u_{kl} to 1 or 0. To solve this, we employ the threshold ϵ,

$$u_{kl} = \begin{cases} 1 & \hat{\beta}_{kl} \geq 1 - \epsilon \\ 0 & \hat{\beta}_{kl} < 1 - \epsilon\,, \end{cases} \tag{17}$$

where ϵ is the noise ratio that we expect in the data. This is the estimated percentage of wrong assignments that do exist in the original user-permission assignment matrix $\mathbf{x}$. Given $\hat{\beta}_{kl}$, we can identify erroneous or missing assignments and automatically correct them or alert an administrator to check these inconsistencies. As an example, if we assume 5%

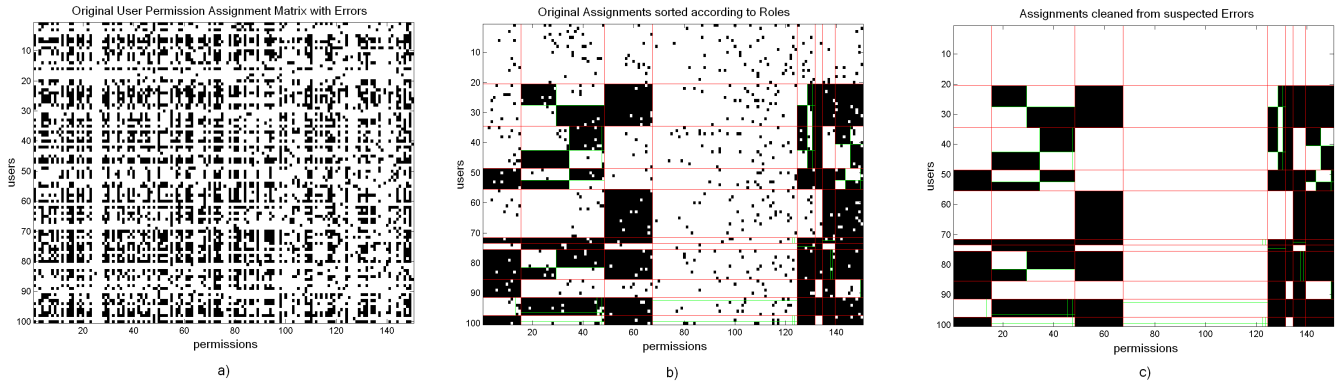

Figure 2: a) Illustration of a randomly generated user-permission assignment matrix. b) Users and permissions ordered according to the result of the Gibbs sampling algorithm for the Disjoint Decomposition Model (DDM). c) Possibly erroneous assignments detected.

noise in the data and find a bicluster with $\hat{\beta}_{kl} = 0.98$, then the 2% of missing assignments are very likely to be errors. Figure 2(c) provides an example of an assignment matrix after correcting such errors.

We denote a user-group to permission-group assignment with probability $\hat{\beta}_{kl}$, where $\epsilon < \hat{\beta}_{kl} < 1-\epsilon$, as an "indifferent bicluster". This means that, for the probability $\hat{\beta}_{kl}$, the deviation of this indifferent bicluster from a pure bicluster cannot be explained by the overall noise in the data. There are instead two alternative explanations:

1. There is actually no structure for the assignments within the bicluster. All assignments are exceptions. All missing assignments are intentionally missing.

2. The underlying structure within the bicluster was not discovered by the Gibbs sampler. The assignments of users to user-groups in a given sampling step are randomly drawn according to the probabilities of the user being assigned to that role. These probabilities factor into a product over the evidence (16) of all the permission-groups that the permissions of the user are assigned to. Members of indifferent biclusters (k', l') have very high factors in (16) for their parts lying within the neighboring biclusters $(k', l \neq l')$. Thus, the probability for them to be reassigned to their user-group k' is very high. As a result, in every iteration, these members are again reassigned, even if a small fraction of their permissions lies within a suboptimal bicluster (k', l').

A pragmatic way to check for both cases is to partition the given bicluster into smaller biclusters. This can be done by considering all user-permission assignments within the bicluster as a smaller independent assignment matrix $\mathbf{x}'$ and to run the algorithm again independently on $\mathbf{x}'$. If there is a structure that is hidden by the neighbors, this procedure will discover it. If not, the algorithm will return the identical result as before.

An alternative way to enforce splits would be to choose a very high hyperparameter α for the Dirichlet process. However, this parameter choice renders the rough structure of the data nearly invisible to the algorithm and results in a high number of very small roles. In contrast, independent subpartitioning preserves the large-scale structure and can

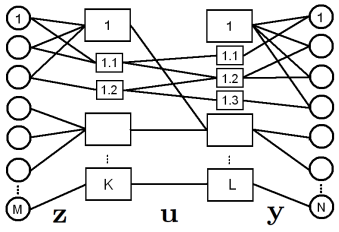

Figure 3: Extension of the Disjoint Decomposition Model to independent sub-roles. For each pair of business-role and technical-role, the corresponding members may also belong to subroles.

simultaneously find focused roles for more specialized users (or rarely used permissions).

Extending the model to the concept of subpartitions or focused roles leads to a model structure that differs slightly from the original Disjoint Decomposition Model. Originally, each user is constrained to only one business role. This assumption still holds for the dominant roles themselves, but now additionally for each user-group permission-group pair (for each bicluster) the members may be additionally assigned to (a single!) focused bicluster. Figure 3 illustrates this structure. The focused role assignments are inferred independently from the rest of the data successively after the main assignments are found. Therefore, both decompositions are independent of each other and the original constraints still hold within each level of the model. Also, there are no assignments between focused business roles and dominant technical roles and vice-versa.

4.3 Experiments

In this section, we assess the performance of the Gibbs sampler for the Disjoint Decomposition Model (DDM) in comparison with the combinatorial algorithm "Discrete Basis Problem solver" (DBPsolver) [12]. We perform experiments using both randomly-generated data and real enterprise data. Our experiments are with respect to two measures. For the random data, we measure how good the resulting roles represent the generated data under the influence of randomly changed assignments (noise). This is an

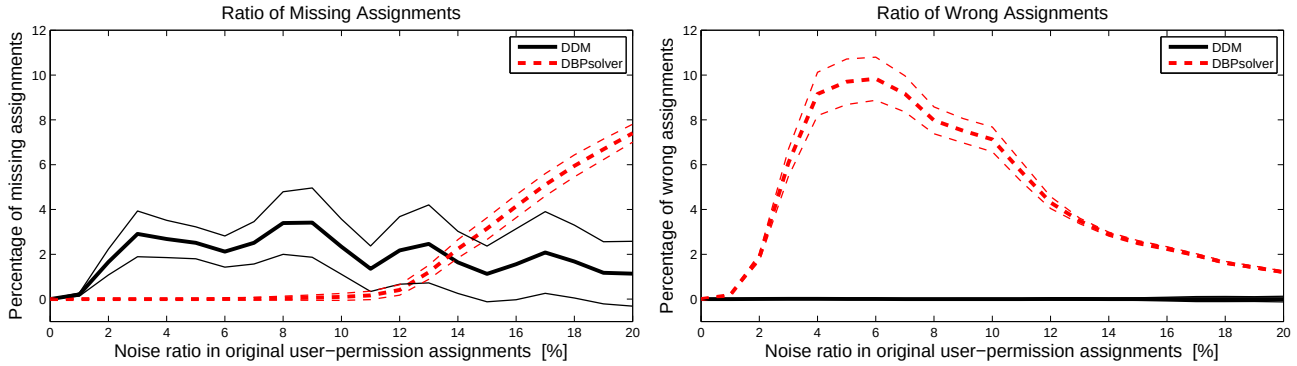

Figure 4: Differences to the correct user-permission assignment matrix. The dashed line represents the DBPsolver and the solid line the Gibbs sampler for the Disjoint Decomposition Model (DDM). The thin lines give the standard deviation over all runs. Both methods approximate an erroneous assignment matrix with a varying percentage of errors (noise). The left plot gives the ratio of missing assignments after the approximation and the right plot the ratio of wrong assignments.

appropriate measure as synthetic datasets enable us to test the algorithms by comparison with the ground truth. For actual enterprise data, this comparison is not possible and for these experiments we introduce a measure that we can evaluate based on the significance of the roles.

In the following experiments, we also present results using the "Discrete Basis Problem solver", presented in [12]. This allows us to compare our results with a state-of the art approach. This algorithm was designed to solve the Discrete Basis Problem, which is equivalent to the role mining problem as shown in [11]. We outline it in the appendix. Given its combinatorial nature, this algorithm is representative for most existing approaches (see Section 2). Most methods initially compute proposed roles with respect to co-occurrences of permissions and then pick from this set in a greedy manner.

Assessment on Synthetic Data.

We generate a user-permission assignment matrix $\mathbf{x}$ according to the general model described in Section 3.1. For a fixed number of 200 users, 200 permissions, 10 business-roles, and 5 technical roles, the assignments were generated as follows. For each user, the memberships in one or more business-roles are randomly drawn and indicated in the assignment matrix $\mathbf{z}$. Similarly, memberships in one or more technical roles are drawn for each permission (thereby creating $\mathbf{y}$). Finally, assignments between business roles and technical roles are randomly drawn ($\mathbf{u}$). The final user-permission assignment matrix $\mathbf{x}$ is then $\mathbf{x} = \mathbf{z} \otimes \mathbf{u} \otimes \mathbf{y}$. In order to simulate a real-world setting, we add errors to this matrix by randomly flipping a given fraction of the entries of $\mathbf{x}$ from one to zero or from zero to one. This results in both erroneous and missing assignments. In the following, we will refer to this random errors as noise. In Figure 2(a), one can see a user-permission assignment matrix $\mathbf{x}$ that has been generated this way (for illustrative purposes we have chosen a lower number of users and permissions for this figure). In principle, one could also generate structured noise, i.e. complete roles that are wrong or missing. However, since these are exactly the patterns the algorithms are searching for, they would have no chance to discriminate them from valid structures (if there is no side information from other sources) and this in turn would not allow for a comparison of the methods.

On the generated assignment matrix $\mathbf{x}$, we performed role-mining using the algorithm for the Disjoint Decomposition Model (DDM) and the Discrete Basis Problem solver (DBP-solver). Since the DBPsolver does not infer the number of roles K itself, we must set that number in advance. We do this by simply computing the results for K ranging from 1 to 100 and pick the best result according to DBPsolver's objective function. A maximum number of 100 roles is reasonable considering that the randomly generated data has at most 10 business roles and 5 technical roles. An average run with DBPsolver (for the best K only), implemented in C++, requires 1.3 seconds. DDM, with a much less efficient MATLAB implementation, needs 40 seconds. An example of a user-permission assignment matrix approximated by DDM is given in Figure 2(c).

From the two resulting approximations of the noisy user-permission assignments matrix, we compute the differences to the original noiseless matrix (the ground truth). The measures that we use to compare the two approaches are the relative number of assignments that have been added with respect to the ground truth (the ratio of wrong assignments) and the relative number of assignments that have been removed with respect to the ground truth (the ratio of missing assignments). These two quantities have a different security-relevance. Granting someone a permission that he *should not* have is usually more serious than not assigning a permission that he *should* have. The former may result in a security breach whereas the latter will typically only result in a call to the help desk, when the missing permission is actually needed. Since this relevance is domain dependent, we refrain from merging them into a single performance measure and collect them separately.

Discussion for the Synthetic Data.

In Figure 4, the described quality measures are plotted versus the noise level in the generated assignment matrix $\mathbf{x}$. This noise level runs from 0% to 20%. Both plots have the same scale. As indicated in the left plot, the DBPsolver can cover every existing assignment to roles up to a noise level of 12%. In doing so, it even covers the assignments

that are missing in the original data due to noise. However, this can only be done at the cost of granting wrong permissions, as can be seen by comparison with the right plot: At noise levels where coverage is good, there is a high percentage of wrong assignments, up to 9% of the total number of assignments. Due to this tradeoff, the two curves of the DBPsolver run contrary to each other. The problem is that the algorithm is unable to identify noise. It can only decide between having a very good coverage at the cost of wrong assignments or avoiding wrong assignments at the cost of poor coverage.

As we see, DDM's results vary only slightly over the entire noise range. One can conclude that it is very robust to erroneous data. As one can clearly see in the right plot, DDM produces a strictly conservative approximation of the original assignments. No additional permissions with respect to the ground truth are granted. The only added permissions are those that have been missing in the noisy data. In turn, all wrong permissions of the original data are discovered and set to zero. The ratio of true assignments that are missed by DDM are, on average, 2% over the whole scale. For high noise levels this is superior to DBPsolver. For DDM there is no tradeoff between coverage and conservativeness. It only adds assignments if there is high evidence that an assignment is missing due to an error.

In summary, the results of DDM are more robust to noise than those of DBPsolver. This is desirable, since, in a real domain, one usually does not know how many errors are in the existing user-permission assignments. Even more important, DDM finds and corrects for wrong assignments while DBPsolver migrates them. For small noise levels, the DBPsolver achieves higher coverage rates while for high noise levels DDM is superior. Over the entire noise range, DDM is better at avoiding wrong assignments.

Assessment on Real-World Data.

The data set that we use for our experiments is provided by a large enterprise. It consists of 5000 users and 1323 permissions and it is a randomly picked subset of all users within one division of the enterprise. The Gibbs sampling algorithm on the Disjoint Decomposition Model needs roughly 5 hours to infer roles, whereas DBPsolver needs approximately one hour for a run with a given number of roles. Assuming 5% noise in the data, the Gibbs sampler for the Disjoint Decomposition model finds 65 dominant business roles and 303 technical roles and covers 87% of the assignments. In addition to the assignments, the model also obtains evidence on where there should be definitely *no* assignments. By using both kinds of information, the assignment matrix can be covered with tiles of ones and zeros. This result allows us to measure a coverage rate for the full assignment matrix. For the evaluated data, this rate is 95%. The large difference to the coverage of only the assignments results from two facts: (1) The given assignment matrix is very sparse and therefore large tiles of zeros can be found. (2) The inference of the assignments between business roles and technical roles is very conservative. In other words, for indifferent biclusters without substructure, the assignment u_{kl} between the user-group k and the permission-group l is 0. The assignments within such biclusters must be checked manually for errors or they can be assigned individually.

Since the DBPsolver is unable to infer the number of roles by itself, one must provide this number in advance. Thus, the coverage can be as high as the experimenter desires. In order to be able to compare the coverage rates, we ran an example for $K = 300$. With this number of roles, 98% coverage is achieved, which is a very good approximation of the existing assignments. This provides evidence that approximating given assignments without considering errors is the strength of the DBPsolver.

However, in contrast to the experiments with generated data, we do not know about the errors in the existing assignments here. Therefore, comparing algorithms just with respect to their coverage may lead to incorrect conclusions. There is no benefit in representing the data with 100% coverage if a substantial fraction of the data is erroneous. The evaluation with the synthetic data has shown that, in these cases, a conservative estimator may lead to better results by identifying errors and correcting for them.

Therefore, we instead compare the algorithms with respect to another quantity: the ability to infer meaningful roles. To do this, we must first define a measure of "meaningfulness". What features render a role mining result interesting? An elementary assumption in bottom-up role engineering is that the existing user-permission assignments in a domain mirror, in some way, the actual tasks that the users in the domain have. In turn, the roles that are constructed should be representative for a particular subset of business tasks. This is a prerequisite if the roles are to be conveniently maintained and to facilitate the addition of new users. The measure of interestingness that we introduce here is based on these considerations.

If there is side information (features) $\mathbf{f}$ available for the users that gives insight in the task structure of the domain, then the inferred roles should be as informative as possible for these features. In order to maximize this mutual information[3] $I(\mathbf{z}, \mathbf{f}) = H(\mathbf{z}) - H(\mathbf{z}\,|\mathbf{f})$ between roles and features, the overall role entropy given the features $H(\mathbf{z}\,|\mathbf{f})$ has to be minimal.

$$\begin{aligned} H(\mathbf{z}\,|\mathbf{f}) &= -\sum_{\nu} p\,(f_\nu) \sum_{k} p\,(z_{\cdot,k} \mid f_\nu) \log\left(p\,(z_{\cdot,k} \mid f_\nu)\right) \\ &= \sum_{\nu} p\,(f_\nu)\, H_\nu^{\mathbf{z}} = E_f[H_\nu^{\mathbf{z}}] \end{aligned} \tag{18}$$

We call $H_\nu^{\mathbf{z}}$ the *role entropy* of a feature f_ν for user-role assignments $\mathbf{z}$. It measures how homogeneous the users having the feature f_ν are distributed over all inferred roles. $E_f[H_\nu^{\mathbf{z}}]$ denotes the expectation value of $H_\nu^{\mathbf{z}}$. A good role mining result should decompose the features by roles as best possible leading to low role entropies. With this objective, such side information could, in principle, be used for a hybrid bottom-up/top-down role mining approach using one of the described model instances. However, this will be subject of future work. For the moment, we just use this measure to assess pure bottom-up role mining results. Any feature providing information about the task structure of the enterprise, can be used in this manner to evaluate user-groups or business roles.

Results for Real-World Data.

In addition to the user-permission assignments, the data set that we received from our industry partner also contains a job code for each of the users provided by Human Resources. A job code is a number jc_i assigned to a user according to his contract type (e.g., secretary, division manager, CEO). Each user has a single job code. Therefore, by

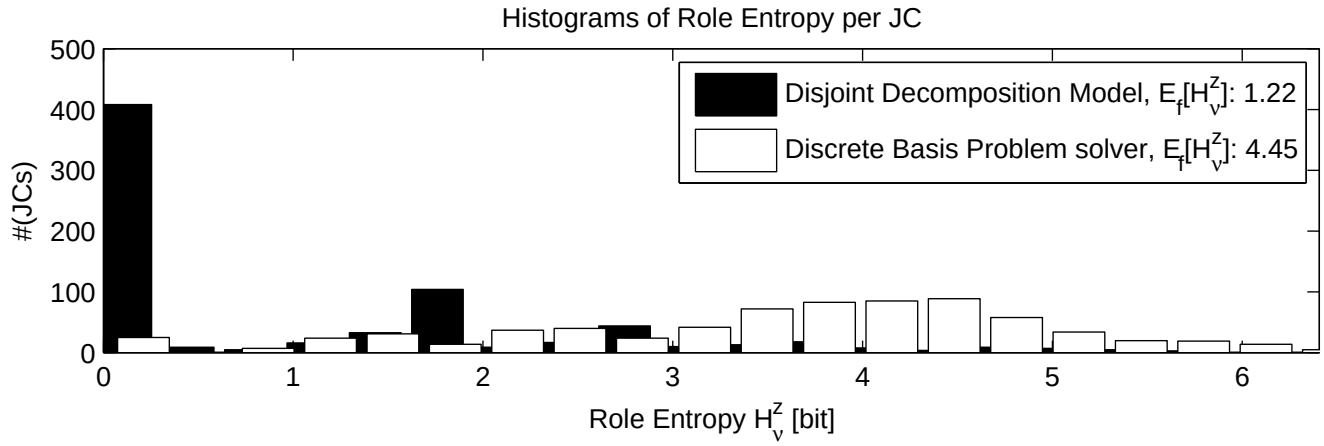

Figure 5: Analysis of the job code distribution over inferred roles. The two histograms show the role entropy $H_\nu^{\mathbf{z}}$ of contract types (the job codes) that the users have. Job codes with lower entropy are better represented by the inferred roles. The black histogram gives the role entropies of a role system found by statistical analysis with the Disjoint Decomposition Model. The white histogram displays the entropies for DBPsolver.

writing jc_i we intentionally use the same index as for the users. In total, there are 724 job codes.

Assuming that the contract type provides information on what the tasks of a user are, we can use the job codes as features, i.e., the side information discussed above. A good role system captures this job code structure of the users via its inferred user-groups. Ideally, most user-groups can be identified as sets of users with the same job code. In turn, the job codes are assigned to only a few user-groups, leading to a small role entropy $H_\nu^{\mathbf{z}}$.

The histograms of the role entropy in Figure 5 illustrate how often job codes are distributed over the roles, both by the DDM and the DBPsolver. The entropies of roles found by DDM are clearly lower on average than the entropies of roles found by DBPsolver. With the DDM, there are many job codes whose users end up in only a single role ($H_\nu^{\mathbf{z}} = 0$), which is highly desirable. In these cases, one could imagine adding access control information for new employees to a system based just on their contract type. The distribution of the DBPsolver roles is very flat and is centered at a high entropy. This indicates that very specific permissions (e.g., for specialized tasks) have been combined with common permissions (e.g., to use standard office software) into synthetic roles that do not capture the heterogeneous permission structure of the domain.

In summary, DDM finds more meaningful roles than DBPsolver with respect to the role entropy. This provides evidence that the variety of roles found by our probabilistic approach better represents the inherent business structure of the enterprise. Such roles are less synthetic and are easier to interpret as representatives of particular tasks within the enterprise.

5. CONCLUSION

We have derived a class of probabilistic models from the logical structure of RBAC, which are capable of representing different domain requirements. Algorithmically, we have shown how to estimate model parameters for one particular, practically important, model in this class, using an off-the-shelf machine-learning algorithm. We have designed experiments using both synthetic and real-world data that allow us to measure our approach's performance and compare it to a state-of-the-art alternative. The experimental results provide strong evidence that probabilistic role engineering outperforms existing combinatorial approaches, as represented by the DBPsolver in this study. In particular, our approach can identify potentially wrong or missing assignments in existing data and automatically correct them. Moreover, the inferred roles are closely correlated with the business functions in an enterprise, which substantially facilitates RBAC administration.

Several directions of research emerge from our study of probabilistic role mining. First, we will develop learning algorithms for inferring roles for the other models in our model class. Second, our model can be extended to a hybrid bottom-up/top-down approach. We will investigate ways to use existing business-relevant data, for instance enterprise hierarchies or employee job codes, for such an extension. Finally, with security administration in mind, we will investigate ways to facilitate the addition of new users and permissions to an existing RBAC system. In principle, a model that supports generalization should be able to endow a new user with the required permissions from limited information about that user, e.g., a small representative subset of privileges.

Acknowledgments.

This work was partially supported by the Zurich Information Security Center. It represents the views of the authors.

6. REFERENCES

[1] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. *SIGMOD Rec.*, 22(2):207–216, 1993.

[2] C. E. Antoniak. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *The Annals of Statistics*, 2(6):1152–1174, November 1974.

[3] T. M. Cover and J. A. Thomas. *Elements of information theory*. Wiley-Interscience, New York, NY, USA, 1991.

[4] E. J. Coyne. Role engineering. In *RBAC '95: Proceedings of the first ACM Workshop on Role-based access control*, page 4, New York, NY, USA, 1996. ACM.

[5] P. Epstein and R. Sandhu. Engineering of role/permission assignments. In *ACSAC '01: Proceedings of the 17th Annual Computer Security Applications Conference*, page 127, Washington, DC, USA, 2001. IEEE Computer Society.

[6] T. S. Ferguson. A Bayesian analysis of some nonparametric problems. *Annals of Statistics*, 1(2):209–230, 1973.

[7] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli. Proposed NIST standard for role-based access control. *ACM Trans. Inf. Syst. Secur.*, 4(3):224–274, 2001.

[8] J. F. Gimpel. The minimization of spatially-multiplexed character sets. *Commun. ACM*, 17(6):315–318, 1974.

[9] C. Kemp, J. B. Tenenbaum, T. L. Griffths, T. Yamada, and N. Ueda. Learning systems of concepts with an infinite relational model. In *Proceedings of the 21st National Conference on Artificial Intelligence*, 2006.

[10] M. Kuhlmann, D. Shohat, and G. Schimpf. Role mining - revealing business roles for security administration using data mining technology. In *SACMAT '03: Proceedings of the eighth ACM symposium on Access control models and technologies*, pages 179–186, New York, NY, USA, 2003. ACM.

[11] H. Lu, J. Vaidya, and V. Atluri. Optimal Boolean matrix decomposition: Application to role engineering. In *Proceedings of the 24th International Conference on Data Engineering (ICDE)*, pages –, 2008.

[12] P. Miettinen, T. Mielikäinen, A. Gionis, G. Das, and H. Mannila. The Discrete Basis Problem. In *Lecture Notes in Artificial Intelligence*, pages 335–346, Berlin, Germany, 2006. Springer.

[13] R. M. Neal. Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9(2):249–265, 2000.

[14] G. Neumann and M. Strembeck. A scenario-driven role engineering process for functional RBAC roles. In *SACMAT '02: Proceedings of the seventh ACM symposium on Access control models and technologies*, pages 33–42, New York, NY, USA, 2002. ACM.

[15] J. Schlegelmilch and U. Steffens. Role mining with ORCA. In *SACMAT '05: Proceedings of the tenth ACM symposium on Access control models and technologies*, pages 168–176, New York, NY, USA, 2005. ACM.

[16] J. Vaidya, V. Atluri, and Q. Guo. The Role Mining Problem: Finding a minimal descriptive set of roles. In *The Twelth ACM Symposium on Access Control Models and Technologies*, pages 175–184, Sophia Antipolis, France, June20-22 2007.

[17] 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*, New York, NY, USA, 2006. ACM Press.

[18] D. Zhang, K. Ramamohanarao, and T. Ebringer. Role engineering using graph optimisation. In *SACMAT '07: Proceedings of the 12th ACM symposium on Access control models and technologies*, pages 139–144, New York, NY, USA, 2007. ACM.

APPENDIX

The Discrete Basis Problem Solver.

A $M \times N$ matrix $\mathbf{x}$ must be approximated by the $M \times K$ matrix $\mathbf{z}$ and the $K \times N$ matrix $\mathbf{u}$ via the Boolean matrix product $\mathbf{x} = \mathbf{z} \otimes \mathbf{u}$, for a given K. The rows of $\mathbf{x}$ are interpreted as sets of items (here, users characterized by their sets of permissions), the rows of $\mathbf{u}$ are the basis sets (the roles), and the rows of $\mathbf{z}$ indicate which basis sets are used to approximate a row of $\mathbf{x}$ (the roles a user is assigned to).

In an initial step, the algorithm computes a set of candidate roles using association rule mining [1]: An $N \times N$ association matrix A is computed whose entries $A_{j_1 j_2}$ are the pairwise associations between permissions j_1 and j_2: $A_{ij} := \langle x_{.j_1}, x_{.j_2} \rangle \,/\, \langle x_{.j_1}, x_{.j_1} \rangle$ with the inner product $\langle ., . \rangle$. In our notation, $A_{j_1 j_2}$ is the empirical probability for a user to have permission j_2 given that he already has permission j_1. Before starting with a greedy algorithm, all entries of $A_{j_1 j_2}$ higher than a threshold τ are set to 1 and the others are set to 0.

Starting from this point, the rows of the role-matrix $\mathbf{u}$ are iteratively filled by the rows of A and the user-role assignments in $\mathbf{z}$ are set. In the kth step, both a row from A is picked as $u_{k.}$ and the entries of $z_{.k}$ are set such that the objective function

$$R\left(\mathbf{x}, \mathbf{z}, \mathbf{u}, w^+, w^-\right) = w^+ \left| \left\{ (i,j) : x_{ij} = 1 \wedge (\mathbf{z} \otimes \mathbf{u})_{ij} = 1 \right\} \right| - w^- \left| \left\{ (i,j) : x_{ij} = 0 \wedge (\mathbf{z} \otimes \mathbf{u})_{ij} = 1 \right\} \right|$$

is maximized. This objective function aims at best covering the existing assignments while avoiding additional assignments. The parameters w^+ and w^- penalize missing assignments and additional assignments respectively.