A Framework for Managing Separation of Duty Policies

Sebastian Groll
University of Regensburg
Regensburg, Bavaria, Germany

Sascha Kern
Nexis GmbH
Regensburg, Bavaria, Germany

Ludwig Fuchs
Nexis GmbH
Regensburg, Bavaria, Germany

Günther Pernul
University of Regensburg
Regensburg, Bavaria, Germany

ABSTRACT

Separation of Duty (SoD) is a fundamental principle in information security. Especially large and highly regulated companies have to manage a huge number of SoD policies. These policies need to be maintained in an ongoing effort in order to remain accurate and compliant with regulatory requirements. In this work we develop a framework for managing SoD policies that pays particular attention to policy comprehensibility. We conducted seven semi-structured interviews with SoD practitioners from large organizations in order to understand the requirements for managing and maintaining SoD policies. Drawing from the obtained insights, we developed a framework, which includes the relevant stakeholders and tasks, as well as a policy structure that aims to simplify policy maintenance. We anchor the proposed policy structure in a generic IAM data model to ensure compatibility and flexibility with other IAM models. We then show exemplary how our approach can be enforced within Role-Based Access Control. Finally, we evaluate the proposed framework with a real-world IAM data set provided by a large finance company.

CCS CONCEPTS

• **Security and privacy → Access control**; *Usability in security and privacy*; *Formal security models*;

KEYWORDS

Separation of Duty, Identity and Access Management, Role-Based Access Control

ACM Reference Format:
Sebastian Groll, Sascha Kern, Ludwig Fuchs, and Günther Pernul. 2024. A Framework for Managing Separation of Duty Policies. In *The 19th International Conference on Availability, Reliability and Security (ARES 2024), July 30–August 02, 2024, Vienna, Austria.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3664476.3670912

1 INTRODUCTION

The concept of Separation of Duty (SoD) is widely considered a fundamental principle in information security within organizations. SoD aims to mitigate fraud and potential conflicts of interest by dividing responsibilities and tasks among different persons. For example, a banking employee should not be able to both issue and approve a loan without at least one more person being involved in the process. Regulatory requirements such as SOX [25], HIPAA [34], BAIT [12], VAIT [6] or Basel III [4] force organisations to specify, document, and enforce SoD rules. Therefore, SoD is addressed in the academic literature concerning various contexts and research areas.

Existing research has devoted considerable attention to the specification [2, 20, 31] and enforcement [7, 9, 33] of SoD rules. However, the creation and the maintenance of SoD rules has hardly been addressed by research so far. Existing works typically assume that rules already exist or are specified by some "administrator", who knows which rules are needed. However, big organisations usually deal with thousands of employees with different functions and responsibilities leading to millions of permissions and roles in numerous departments and applications. Thus, the necessary set of SoD rules is large and complex, and the knowledge required to specify and maintain correct rules spreads throughout the organization across domain experts, managers, departmental heads, risk managers, IT specialists, etc. [18]. Once SoD rules have been initially created, they must be maintained in an ongoing effort. Similar to any set of data, SoD rules can become outdated due to changes in both external compliance requirements and the organization's internal structure (e.g. permissions or roles). Consider, for example, a new version of HIPAA, or the introduction of a new HR system in the organization. Both will lead to new requirements for the existing SoD rules. Active SoD rules must therefore be regularly reviewed, updated, and eventually de-provisioned when their validity expires.

In order to be maintainable and manageable, the meaning and purpose of each SoD rule must be clear: Where does the rule originate from? Which legal text or guideline is responsible for its existence? What happens if the legal text changes or a new one is added? Who ensures that the rule is technically implemented correctly? etc. The knowledge required to answer these questions extends throughout the entire organization. Hence, SoD rules must be created and maintained within a structured processes with clear responsibilities and knowledge holders. They need to be enriched with descriptive, human-understandable information and the knowledge spread throughout the organization must be consolidated. To the best of our knowledge, the current research does neither provide an approach for deriving and maintaining SoD rules nor does it provide an approach to enrich SoD rules with descriptive information.

This work contributes to this research gap by proposing a framework for SoD rules that are easy to maintain. Our contributions are as follows: (i) We conduct seven expert interviews with SoD managers from large, compliance-driven companies. They serve to broaden the understanding of SoD management and to identify best practices for SoD maintenance. (ii) On this basis, we propose a framework for the creation and maintenance of SoD rules, that pays particular attention to descriptive information and comprehensibility. It defines the key stakeholders for SoD maintenance, and three types of SoD policies which are anchored in the integrated IAM data model by Kunz et al. [21]. (iii) After a conceptual definition, we formalize our easy to maintain SoD rules as an exemplary proof of concept in Role-Based Access Control (RBAC) [28] and show how they can be enforced. We then evaluate the framework by applying our approach to a real-world IAM data set provided by a large finance company. Our evaluation shows that the complexity of managing and maintaining SoD with the proposed framework is significantly lower than managing traditional SoD rules.

2 FOUNDATION

SoD rules can roughly be divided into rules with static and dynamic SoD properties [14, 31]. Static rules are generally applicable, e.g. "A user who is allowed to carry out Task A must not be allowed to carry out Task B". Dynamic SoD rules are only applicable in a specific context, e.g. at the same time, with the same object or operation. An example rule would be "A user must not carry out Task A and Task B for the same customer; however, the user can carry out Task A for one customer and Task B for another". For the sake of simplicity, we'll rely on static SoD rules throughout the remainder of the work. Whenever we refer to SoD rules, we mean static SoD rules. However, we discuss how our framework can be adopted to dynamic SoD rules in section 8.

SoD is particularly important in the context of Identity and Access Management (IAM). IAM is a domain of IT management that comprises a range of processes, technologies and policies (including SoD policies), which deal with the administration of digital identities and the provisioning of secure user access to digital resources. It commonly relies on the *principle of least privilege*, which determines that no user should have more authorizations than he or she requires to perform their duties. The structure of user authorizations is defined in IAM by Access Control Models (ACMs). While various ACMs have been developed for different scenarios, some have proven to be highly adaptive in theory and practice (especially Role-Based Access Control (RBAC) [28], Attribute-Based Access Control (ABAC) [16], and Discretionary Access Control (DAC) [27]). Regardless of the utilized ACM, the defined authorizations can be expressed as a set of User Permission Assignments (UPAs) (e.g. in the form of an access matrix [29]): A permission represents an operation and a resource (e.g. "read" and "file x"). If a permission is assigned to a user, the user is authorized to perform the access specified by it. Independent of an ACM, SoD rules can be defined as sets $sod(\{p_1, ..., p_n\}, k)$ where p_i are permissions and k, n are positive integers such that $1 < k \leq n$ [22]. The rule states that there must not exist any set of users smaller than k that possess the permissions $\{p_1, ..., p_n\}$ together.

While this is a very fine-grained way to specify rules, the enforcement is proven to be computationally expensive (NP-complete) [22]. SoD is extensively studied in the context of the RBAC model, in which roles are used as intermediaries between users and permissions. A more efficient way to specify SoD rules in RBAC are Mutual Exclusive Role Policies (MER). A MER defines an amount of roles that form a toxic combination and therefore a single user is not allowed to possess together. For example the roles "Compliance Officer" and "Cashier" could form such a toxic combination, because the cashier has to carry out financial transactions and the compliance officer needs to approve them. If both roles would be assigned to the same person, this person could manipulate transactions or make false bookings. Formally, MERs are defined as sets $mer(\{r_1, ..., r_n\}, t)$ with each $\mathrm{r_i}$ as a role and n, t as positive integers, such that $1 < t \leq n$ [22]. In contrast to the permission-based specification, a MER policy does not allow a user to have t or more roles from the set $\{r_1, ..., r_n\}$.

MER policies are more coarse-grained than permission-based SoD policies but faster to enforce. It is also possible to translate permission-based SoD policies into MER policies. However, this will result in stricter rules that are less flexible. Other models that are addressed in SoD research include workflows (e.g. [10]), modelling languages (e.g. [30, 36]) or petri nets (e.g. [19]). Nevertheless, most approaches use an underlying RBAC model for SoD specification and enforcement. Note that the terms SoD policy and SoD rule are mostly used interchangeably, while the term constraint often describes SoD rules in the context of an ACM (e.g. "MER constraint"). We maintain this terminology in the course of this work.

3 RELATED WORK

Existing SoD literature covers a wide range of topics: The specification of SoD rules is studied in ACMs like RBAC [2, 20, 31] or ABAC [3, 5], in workflows [10, 23] or in petri nets [11, 40]. Other works enhance languages like UML [26, 32] or BPMN [37, 38] with SoD support. Another related research domain deals with the enforcement of SoD rules [7, 9, 33]. Despite broad coverage of the SoD concept in general, few works are dedicated to the actual creation of maintenance of SoD rules in the context of an organization. Some algorithmic approaches exists for the generation and transformation of SoD rules: Li et al. [8, 22] study the translation between permission-based SoD rules and MERs. As mentioned previously, the objective is to create MER policies that are minimally restrictive while still enforcing the underlying permission-based SoD policies. However, these works focus only on the algorithmic transformation of policies and not on the derivation of policies within organizations.

Kijsanayothin et al. [15] propose an approach to derive MER policies from a workflow. They assume an SoD violation when the same person is allowed to perform consecutive create and update actions on the same object. We argue that SoD rules for organizations need to be more fine-grained and flexible, for example also read-permissions might be relevant for SoD. Additionally, for some real-world applications it might be hard to define which permissions exactly grant update or create privileges on objects. Wolf and Gehrke [35] propose a 6-step method to derive SoD rules in an organization. Their method starts with an early analysis phase, e.g. interviewing employees in order to discover and analyze relevant processes. It continues with the specification and translation of SoD rule sets and finally describes the extraction of live data from the applications and the enforcement of the specified SoD rules. While this work provides valuable insights into organizational aspects of SoD creation, its research scope is limited to Enterprise Resource Planning (ERP) systems. It provides neither a generic approach for deriving SoD rules nor semantics or a generic data structure for these rules. Furthermore, they do not map SoD rules to a standardized specification used in scientific literature.

The closely related concept of Access Control Policies (ACPs) desribes machine-processible rules which define positive or negative authorizations to regulate which access a user is allowed to make [29]. ACPs can express SoD rules, but are not limited to them. Several works in this research realm propose to align rules with human-understandable semantics to improve their maintainability. They share the assumption that the most important factor for low management complexity of rules is that they have can be associated with a real-world concept. Such a real-world concept (e.g. "all department heads may access the following resources") can be annotated to rules via attributes which represent that concept (e.g. "function = department head").

Fuchs et al. [13] highlight the high importance of annotated business semantics for effective management or maintenance of role-based ACPs. Molloy et al. [24] propose approaches to mine role-based ACPs based either on semantically meaningful user attributes or on formal concept lattices. Similarly, Jin et al. propose to assign roles to users based on user attributes [17]. Xu defines an interpretability metric which attempts to approximate the semantic meaningfulness of role-based or attribute-based ACPs by measuring their accordance with semantically meaningful user attributes [39]. To the best of our knowledge, no scientific work exists that provides a framework for the management of SoD policies in the context of an organization, or describes a generic approach to improve their human comprehensibility.

4 SEMI-STRUCTURED EXPERT INTERVIEWS

We conducted semi-structured expert interviews to gain a better understanding of how SoD rules are modeled and maintained in large organizations. We interviewed seven experts from six different organizations in accordance with the methodology proposed by Adams [1]. We prepared a questionnaire through which we went with the interviewees in a natural conversation. If further relevant points were mentioned or ambiguities arose, we deviated from the prepared catalogue in order to investigate these in more detail. The structure of the interviews can be divided into 3 blocks, which are briefly described below:

- **B1: Introduction and General Questions**
  Goal: Initiate interview. Ensure relevance of the interviewed organization.
  - Short introduction.
  - What is the participant's job position and and the participant's connection to SoD within the organization?
  - What is the organization's size? How many employees work there and how many digital identities are managed?
  - What is the main motivation for the organization to manage SoD?
- **B2: Structure and human understandability**
  Goal: Discover different SoD types and their structure. Determine how meaningful information is stored.
  - How is the organization's data model for SoD rules structured?
  - How do they make SoD understandable to people? Do they use descriptive names, or employ other attributes?
  - Which methods or tools exist in the organization to display SoDs in a human understandable way?
- **B3: Lifecycle and processes**
  Goal: Clarify stakeholders and responsible parties for SoD rule creation and maintenance. Determine how meaningful information is derived.
  - Who in the organization is responsible for SoD rule creation and maintenance?
  - How does the organization derive SoD rules?
  - How does the organization maintain SoD rules and ensure a sufficient data quality?

After conducting the interviews we structured and summarized the received answers. Naturally, there were variations in both terminology and content; for instance, some refer to roles as 'business roles' or 'organizational roles,' describing the same underlying concept. Despite these different perspectives, we could abstract several organizational structures for SoD management that are established in many or sometimes all of the considered organizations. Below we summarize significant results of the three interview blocks.

B1: The interviewed experts work on SoD projects for large organizations in different industry branches.

Organization O1 is an organization with about 17,000 employees in the engineering sector. Compared to more heavily regulated sectors, such as banking or insurance, O1 is subject to less stringent requirements. SoD rules are primarily applied to an ERP system for which a standardized set of SoD rules has been purchased. In this organization we interviewed an IAM consultant.

O2 is a large telecommunications company with about 18,000 employees. Due to both internal and external compliance requirements, SoD rules must be applied to many different application systems. They use purchased rule sets as well as self-created SoD rules. The self-created rules are derived using so-called SoD Classes, a concept we will explain further in Block B2. The interviewed expert is a risk manager.

O3 is a company with 33,000 employees. Being part of the highly regulated banking sector, it adheres to numerous legal regulations. SoD rules are created and managed using SoD Classes. Additionally, domain experts can also define pairwise mutual exclusions for roles and permissions directly. We interviewed one person working in IAM business support and one person working as a compliance manager.

O4 is an organization in the insurance sector with 7,000 employees. In the insurance sector, it is also common that compliance requirements require strong regulation through SoD. SoD rules are created and maintained solely using SoD Classes. The interviewee emphasized that the creation of SoD rules is a complex and politically sensitive task, as it relies heavily on the support of the domain experts. We interviewed a member of the IAM team.

Table 1: Overview of the seven interviewed SoD experts and their employing organizations.

Org.	Participants	Employees	Sector	Pairwise exclusive Permissions	Pairwise exclusive Roles	SoD Classes
O1	1	~17,000	Engineering	X		
O2	1	~18,000	Telecommunications	X		X
O3	2	~33,000	Finance	X	X	X
O4	1	~7,000	Insurance			X
O5	1	~300	Finance	X		X
O6	1	~4,500	Finance			X

O5 is an organization in the financial sector with about 300 employees. It is notable that even smaller companies in the banking sector are subject to strict regulations and therefore have a high administrative burden through SoD. The organization utilizes SoD Classes and pairwise mutual exclusions for permissions, that are created and maintained by the domain experts. The SoD rules have linked legal texts and descriptions explaining why the corresponding exclusion have to exist. There also exist different risk-levels for the exclusions which are useful when SoD violations are mitigated. According to O5 not all SoD violations have to be resolved: Sometimes it is possible to grant exemptions and accept the risk. The interviewed expert is an IAM consultant.

O6 is a banking company with strict regulations like the other organizations from the finance industry. SoD rules serve to avoid conflicts of interest as well as over-privileged users, especially IT administrators. They derive SoD rules using SoD Classes. The interviewed expert is a risk management officer. Table 1 summarizes the interviewed experts and their organizations. All employee numbers were rounded to ensure the anonymity of the companies.

All of the organizations use a form of RBAC with some sort of schema definition for role hierarchies. They also manage "direct" user-permission assignments which can be granted independently from the role model, e.g. through a user self service portal. Most of the organizations operate some form of automation logic which assigns roles or permissions to users based on user attributes or proprietary rules. However, for this, none of them uses formal ABAC or the eXtensible Access Control Markup Language (XACML). The amount of managed permissions ranges between 10,000 and 1,000,000. All interviewed experts named regulatory compliance as the primary driver of their SoD management.

B2: The interviewed experts described three prototypical structures for easily understandable and maintainable SoD rules: Pairwise mutually exclusive permissions, pairwise mutually exclusive roles, and the use of SoD Classes (cf. Table 1). The simplest way is the definition of pairwise mutual exclusions between two permissions or two roles. In this case, a user is not allowed to posses both permissions or both roles at once. In contrast to the MERs defined in literature (cf. Chapter 2), which allow the definition of larger sets and a minimum number of allowed roles, the mutual exclusions in the interviewed organizations were always limited to two roles or two permissions. Such pairwise mutual exclusions are easier to understand and manage. The interviewees also stated that it is important that every mutual exclusion requires a proper and up-to-date name and description annotation to remain human-understandable. Despite their simplicity, the management of mutual exclusions has some shortcomings: They quickly amount to large data sets, causing unnecessary complexity.

Another way the experts mentioned to define SoD rules is the use of so-called SoD Classes. SoD Classes represent groupings of tasks, functions or responsibilities in an organization that are designed to conflict with each other. The SoD Classes have a name with a semantic meaning and may also have a description for further information. They are assigned to roles and permissions and a user is not allowed to posses roles or permissions with conflicting SoD Classes. According to the interviewed experts, the main benefit of SoD Classes lies in their simple management and human understandability: Typically, large organizations manage hundreds of thousands of roles and permissions, while managing only 15 to 50 SoD Classes. Another advantage is that SoD Classes are well-suited for addressing the challenge of knowledge distribution among various stakeholders regarding SoD: Legal experts can define the SoD Classes and conflicts, while technical or Domain Experts can determine the matching SoD Class for a permission or role. This becomes evident in the next section.

B3: The experts named various stakeholders which are responsible for the SoD rule creation and maintenance: Normally, the responsibility for overall SoD management lies within an IAM team. However, Domain Experts, like owners of application systems, permissions, or roles, may also be responsible, especially in large systems. For example, the IAM Team may be responsible for the entirety of SoD rules and SoD Classes, but creating mutual exclusive permissions or roles or assigning SoD Classes to permissions may be the responsibility of the permission and roles owners. Also there usually is some kind of IAM governance department, that is responsible for risk assessment and the interpretation of legal texts and compliance requirements. The creation of SoD rules typically follows one of three prototypical methods: (i) *Buying complete rule sets*, (ii) *Defining SoD classes* and (iii) *individual fine-grained rule definition.*

(i) Buying a complete rule set is a common approach to ensure compliance with regulatory requirements. Such rule sets can for example be obtained from auditing companies and cover the permissions ranged in a specific kind of target system. Since an organization does not acquire the know-how necessary to maintain these rules when purchasing, a dependency on the supplier can arise: Without regular follow-up purchases, the purchased rule sets can out-date and lose their validity over time. From a purely compliance-driven point of view, however, they are the "safe bet": First, buying standardized rules sets that are confirmed to be compliant (at the time of purchase) gives an organization some degree of security that they will meet regulatory requirements for the covered application systems. Second, such standardized rule sets do not require to be human understandable since they are not maintained by the organization itself, but merely enforced. Therefore, those rule sets usually do not contain lots of human-understandable information. The rule sets are typically limited to one application and do not support organization-wide SoD policies. Additionally, they can also be expensive and are not available for every application, e.g. applications that are not used widely or in-house developments.

(ii) The use of SoD Classes involves multiple stakeholders: The IAM governance that defines the SoD Classes and various Domain Experts that are owners of permissions or applications. The SoD Classes are derived by the IAM governance by evaluating compliance regulations and using industry-standards. They do not usually change much over time and hence require little maintenance. The assignment of the SoD Classes to the permissions is carried out by a permission or application owners, depending on the respective application and how the application is managed. In order to keep the rules up to date, regular reviews can be carried out.

(iii) Finally, all organizations use some sort of fine-grained SoD management. This can include pairwise mutual exclusive roles and permissions, but also proprietary formats, e.g. logic-based rules or custom evaluation scripts. Managing these fine-grained SoD rules requires a deep understanding of the underlying authorization structures and has to be done by role, permission or application owners or administrators. These rules also need to be reviewed periodically to maintain their accuracy and timeliness.

5 DERIVING A FRAMEWORK FOR MANAGING AND MAINTAINING SOD

Based on the insights we gathered from the conducted expert interviews, as well as on existing SoD and IAM literature, we derived a framework for managing and maintaining SoD policies. The framework consists of three components: (i) The relevant stakeholders and their responsibilities, (ii) the three proposed SoD rule types along with the respective processes for rule creation and maintenance and (iii) the integration of the proposed SoD rule types into existing ACMs. The developed framework is presented in the remainder of this chapter. Figure 1 depicts the described tasks and responsibilities of the different stakeholders in our framework.

5.1 Stakeholders and responsibilities

In order for SoD policies to be well-understandable, they need to hold a real-world meaning. Such semantic meaningfulness does not arise from the structure of policies alone, but must be incorporated into them during their creation and maintenance. Not everyone involved in the management of SoD rules has all the necessary information available. Therefore, we first define three types of stakeholders that were described during the interviews:

- **IAM Governance / Risk Management** *(Called IAM Governance in the following)*: Responsible for complying with laws and regulatory requirements. Ensures that audit requirements are met. Manages the IAM at a high level, specifies processes and carries out risk assessments. Knows the laws and legal requirements, but does not necessarily have a deeper understanding of individual roles, permissions and IT applications.
- **IAM Team** Conducts IAM projects, is responsible for the implementation and enforcement of SoD policies and rules. Understands permissions and roles on a conceptual and technical level but cannot always assess their business implications. The IAM team size can vary from few persons up to a whole IAM Department.
- **Domain Experts** Are responsible for one or more IT application systems, processes or organizational units. Typical positions are department heads, business owners or technical owners like system admins. Domain experts know the tasks of users and the effects of permissions and roles in their responsibility in great detail. However, they do not have an overarching view of the IAM structure and may not necessarily be aware of possible interactions with other domains.

Since the necessary knowledge for creating and maintaining SoD rules is distributed across these domains, collaboration between the involved stakeholders is necessary. The IAM team serves as an intermediary between the IAM governance and the domain experts.

5.2 The SoD Matrix: A tool to reduce complexity

The SoD Matrix is a central control tool of our framework for SoD management. It acts as an intermediary between the IAM team, IAM governance and the domain experts. Preliminary for its creation is the definition of SoD classes, which are represented in the SoD matrix as rows and columns. A SoD class is an entity that describes a real-world concept relevant to SoD. An example for SoD classes could be "Payment Transactions" or "Internal Audit".

The SoD Matrix defines a pairwise exclusion whenever two SoD classes are incompatible with one another. Due to its Matrix form, the SoD Matrix is intuitively understandable and suitable for visualization. Figure 2 displays a (reduced) SoD Matrix as managed by a real-world finance company. The definition and maintenance of the SoD Matrix (and SoD classes) is carried out by the IAM governance through reading and interpreting legal texts and security standards. The IAM Governance has the required expertise and knowledge for this task. A deeper understanding of the applications or their permissions and roles is not required. For example, a member of IAM Governance could read in a compliance standard that functions related to payment transactions must not be combined with functions dealing with compliance and approval. The IAM governance employee would create the SoD classes "Payment Transactions" and "Internal Audit" as well as an exclusions between the two SoD classes in the SoD matrix.

In order to enforce the SoD matrix on the application level, the SoD classes need to be assigned to the permissions of the respective applications. This task is carried out by the domain experts, as it requires in-depth knowledge of the permissions, particularly regarding which actions can be performed with which permissions. At this point, it is crucial that the SoD classes have meaningful names and descriptions to be understood by the domain experts. For example a permission that allows a user to edit a financial transaction on an ERP system would be assigned to a SoD class "Payment Transactions" by a domain expert. Note that many permissions are

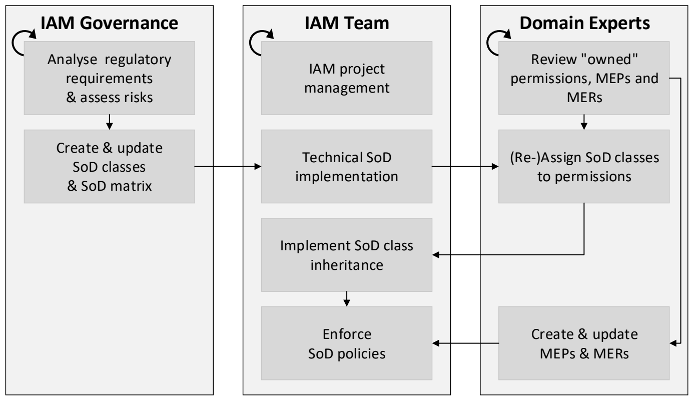
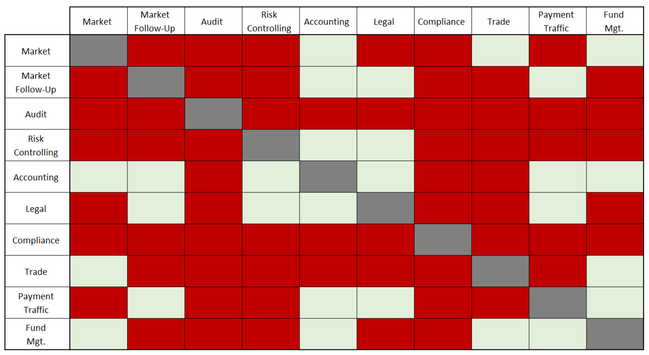

IAM Governance
Analyse regulatory requirements & assess risks
Create & update SoD classes & SoD matrix
IAM Team
IAM project management
Technical SoD implementation
Implement SoD class inheritance
Enforce SoD policies
Domain Experts
Review "owned" permissions, MEPs and MERs
(Re-)Assign SoD classes to permissions
Create & update MEPs & MERs

Figure 1: Responsibilities and activities of the proposed framework.

not relevant for SoD (e.g. "WiFi access"). All permissions without a matching SoD class are assigned a neutral SoD class which can never cause a SoD conflict with another SoD class.

Applying the SoD Matrix to all permissions results in a list of pairwise mutual permission exclusions. The IAM team must bring the SoD Matrix and the permissions with SoD classes together and enforce the resulting exclusions. These can be implemented according to any established SoD specification (cmp. section 2). Maintaining the resulting (technical) SoD rules no longer requires involving governance or domain experts. In return, governance and domain experts can keep the SoD components in their responsibility up-to-date on their own, without having to rely on the knowledge of others. In addition to the IAM project organization required for the creation of a SoD Matrix, the IAM team must provide the technical infrastructure and organize regular reviews. The described responsibilities and tasks can be seen in Figure 1.

5.3 Fine-grained exclusions with pairwise MERs and MEPs



	Market	Market Follow-Up	Audit	Risk Controlling	Accounting	Legal	Compliance	Trade	Payment Traffic	Fund Mgt.
Market	⬛	🟥	🟥	🟥	🟩	🟥	🟥	🟩	🟥	🟩
Market Follow-Up	🟥	⬛	🟥	🟥	🟩	🟩	🟥	🟥	🟩	🟥
Audit	🟥	🟥	⬛	🟥	🟥	🟥	🟥	🟥	🟥	🟥
Risk Controlling	🟥	🟥	🟥	⬛	🟩	🟩	🟥	🟥	🟥	🟥
Accounting	🟩	🟩	🟥	🟩	⬛	🟩	🟥	🟥	🟩	🟩
Legal	🟥	🟩	🟥	🟩	🟩	⬛	🟥	🟥	🟩	🟥
Compliance	🟥	🟥	🟥	🟥	🟥	🟥	⬛	🟥	🟥	🟥
Trade	🟩	🟥	🟥	🟥	🟥	🟥	🟥	⬛	🟥	🟩
Payment Traffic	🟥	🟩	🟥	🟥	🟩	🟩	🟥	🟥	⬛	🟩
Fund Mgt.	🟩	🟥	🟥	🟥	🟩	🟥	🟥	🟩	🟩	⬛

Figure 2: Matrix visualization of SoD classes and their pairwise mutual exclusions ("SoD Matrix").
Note: Red cells mark a pairwise exclusion, while green cells indicate that the two SoD classes are not conflicting. The depicted SoD Matrix defines 31 pairwise SoD class exclusions.

The SoD matrix enables well-maintainable and compliant, but coarse-grained SoD rules. In real-world applications, more precise SoD rules may be necessary. For example, two specific permissions (or roles) may form a toxic combination, but be included in compatible SoD classes. This can happen for technical reasons, especially since permissions function slightly differently in each application. Also there may not be a suitable SoD class for some roles.

The knowledge about these exclusions lies with the domain experts. With pairwise Mutually Exclusive Roles (MER) and pairwise Mutually Esclusive Permissions (MEP) they can create direct exclusions, in addition to the SoD matrix. In contrast to MERs specified in scientific literature, we propose using MERs and MEPs that require a mandatory description for each exclusion explaining its respective reason. Furthermore, they have a fixed cardinally of 2, which makes pairwise MERs an instance of the default RBAC MER: $mer_2\{r1, r2\} \equiv mer\langle\{r_1, r_2\}, 2\rangle$. The processes for creating and maintaining MERs and MEPs are performed by the domain experts and organized by the IAM team. Again the IAM team is responsible for providing a suitable infrastructure and organizing regular reviews.

5.4 Data model integration

Existing IAM literature defines numerous ACMs designed for different scenarios and requirements. Among the most commonly used ACMs are RBAC, ABAC and the somewhat older but still relevant DAC model. Alongside these theoretical data structures exist industry standards for identity data exchange, SSO and authorization delegation. Since a precise specification of the proposed SoD policies in a significant amount of ACMs and standards exceeds the scope of this work, we chose to anchor it in the conceptual IAM data model proposed by Kunz et al [21]. It provides a generic view over the central entities defined in the ACMs, e.g. RBAC and ABAC, and in the standards LDAP, SAML, SPML, OAuth, SCIM and XACML. To the best of our knowledge, it is the most generic IAM data model proposed in scientific literature. A more detailed formalization of the proposed SoD policies for the RBAC model is provided in section 6.

The conceptual data model defines a digital identity as a representation of a human user. An account is a representation of a user

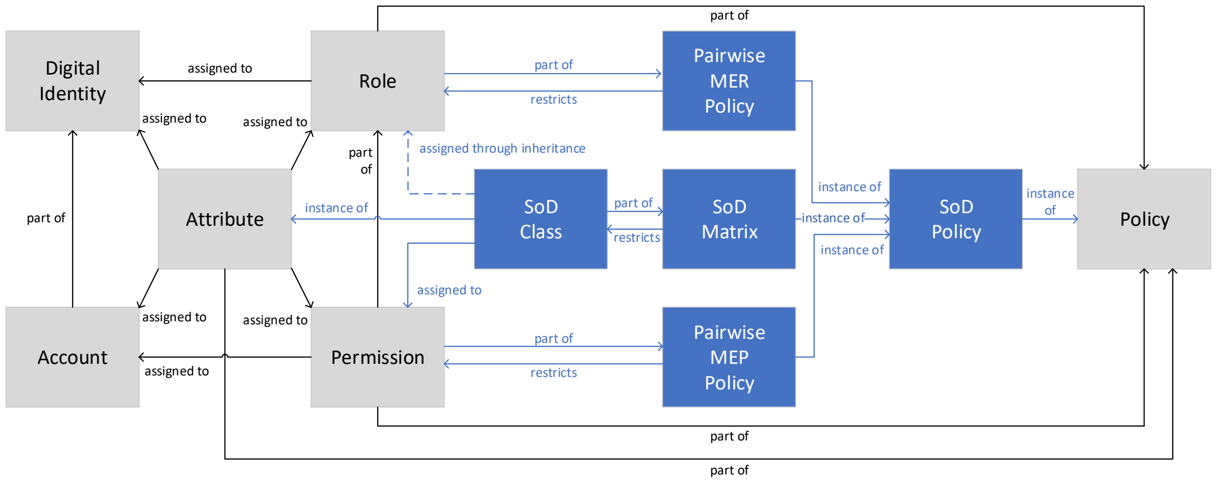


Figure 3: Anchoring of proposed SoD policies in the conceptual IAM data model by Kunz et al. [21].
Note: Grey entities were adopted from the original conceptual IAM model, while blue entities are introduced in the proposed SoD framework.

within an application system and can hence be part of a digital identity. Permissions enable users to execute certain actions, commonly expressed as a combination of a resource to be accessed and an access operation (e.g. modify files in a certain directory). Roles are semantic entities that bundle a set of permissions. They are valid organization-wide and can hence be assigned directly to a digital identity. Policies represent sets of rules that define authorizations, such as ABAC, SPML or XACML. Attributes represent meta-data that is assigned to any of the described entities. An example of a user attribute would be their job title or the department they work in. The original data model also defines a context entity, which we omit because it is not relevant for the scope of this work.

We extend the conceptual model by Kunz et al.[21] to cover the proposed SoD policies. SoD policy is a specific type of policy that constrains the authorization structure by defining mutually exclusive entitlements. A violation of this constraint represents a conflict that must be resolved. We define three kinds of SoD polices: Pairwise MERs, MEPs and the SoD matrix (along with its SoD classes). Pairwise MER and MEP policies, as specified before, define simple pairs of mutually exclusive roles or permissions. They are hence linked to exactly two permissions (MEP) or roles (MER) each and restrict their assignment to a digital identity: Any pair of mutually exclusive permissions or roles that is inherited by a single digital identity constitutes a violation of the defining policy which must be resolved to restore compliance.

The SoD matrix defines mutually exclusive SoD classes. The SoD classes are assigned to permissions as an attribute. Every permission has exactly one SoD class assigned, which may also be the neutral SoD class that indicates no SoD relevance. Roles inherit SoD classes transitively from their permissions. We constrain the model by demanding that roles are *homogeneous*, i.e. must not inherit permissions with more than one SoD class other than the neutral one. Each pair of mutually exclusive SoD classes in the SoD matrix indicates that no digital identity may inherit permissions from both of these classes. Figure 3 summarizes the SoD policies from the proposed framework, anchored in the conceptual IAM model.

6 POLICY FORMALIZATION AND ENFORCEABILITY IN RBAC

We extended the conceptual IAM model by Kunz et al. to anchor the SoD policies in a generic data model. In the course of this chapter, we will provide a more detailed formalization in the RBAC model to demonstrate its feasibility. We then show that the proposed SoD policies can be translated into classic MER constraints. We chose RBAC because it is a well-known ACM in SoD literature as well as in the organizations of the interviewed SoD experts. A practical evaluation is provided subsequently in section 7.

6.1 Model formalization

We adopt the preliminaries and notations from the RBAC SoD formalizations by Li et al. [22] for sakes of consistency.

Preliminaries. Let U be the set of all users, R the set of all roles and P the set of all permissions. An RBAC state γ is a 3-tuple $\langle UA, PA, RH \rangle$, where $UA \subset U \times R$ is the set of all user-to-role assignments, $PA \subset R \times P$ is the set of all role-to-permission assignments, and $RH \subset R \times R$ is the set of all role-to-role assignments. RH^* is the reflexive, transitive closure of RH and a partial order among roles in R. The functions $auth\_roles_\gamma[u]$ and $auth\_perms_\gamma[u]$ determine the roles and permissions that a user effectively inherits and are defined as:

$$auth\_roles_\gamma[u] = \{r \in R | \exists r_1 \in R[(u, r_1) \in UA \wedge (r_1, r) \in RH^*]\}$$

$$auth\_perms_\gamma[u] = \{p \in P | \exists r_1, r_2 \in R[(u, r_1) \in UA \wedge (r_1, r_2) \in RH^* \wedge (r_2, p) \in PA]\}$$

Definition 1. A pairwise MER policy mer_2 is a specific type of MER constraint with fixed value of $n = t = 2$. It defines a pair of mutually exclusive roles

$$mer_2\{r_1, r_2\}$$

with $r_1, r_2 \in R$ and $r_1 \neq r_2$. A pairwise MER policy is satisfied by an RBAC state γ if no single user inherits both roles defined by it as mutually exclusive:

$$sat_{mer_2}[\gamma] \triangleq \forall u \in U(|auth\_roles_\gamma[u] \cap \{r_1, r_2\}| < 2)$$

Definition 2. A pairwise MEP policy mep_2 defines a pair of mutually exclusive permissions

$$mep_2\{p_1, p_2\}$$

with $p_1, p_2 \in P$ and $p_1 \neq p_2$. It is satisfied by an RBAC state γ if no single user inherits both permissions defined by it as mutually exclusive:

$$sat_{mep_2}[\gamma] \triangleq \forall u \in U(|auth\_perms_\gamma[u] \cap \{p_1, p_2\}| < 2)$$

Definition 3. An organization manages a finite set of $I + 1$ SoD classes with $\emptyset$ being the *neutral SoD class*:

$$SOD\_CLASSES = \{\emptyset, class_1, ..., class_I\}$$

Every permission p is assigned a SoD class $assigned\_class_p[p] \in SOD\_CLASSES$. By default, any permission has $assigned\_class_p[p] = \emptyset$, unless it was assigned a non-neutral $class \in \{class_1, ..., class_I\}$. The set of all permissions which share a SoD class $class$ is determined by the function

$$perms[class] = \{p \in P | assigned\_class_p[p] = class\}$$

A role inherits SoD classes transitively from its child roles and permissions:

$$\begin{aligned} trans\_class[r] = \{class \in SOD\_CLASSES | \exists r_1 \in R[(r, r_1) \in RH^* \\ \wedge (r_1, p) \in PA \wedge assigned\_class_p[p] = class]\} \end{aligned}$$

The effectively assigned SoD classes are consequently:

$$assi\_class_r[r] = \begin{cases} \{\emptyset\} & \text{if } trans\_class[r] = \{\emptyset\} \\ & \text{or } trans\_class[r] = \{\} \\ trans\_class[r] \setminus \{\emptyset\} & \text{otherwise} \end{cases}$$

We call an RBAC state γ *homogeneous* if no role inherits more than one non-neutral SoD class:

$$homogeneous[\gamma] \triangleq |assi\_class_r[r]| = 1 \forall r \in R$$

The set of all roles which share a SoD class $class$ is determined by the function

$$roles[class] = \{r \in R | class \in assi\_class_r[r]\}$$

Definition 4. A user inherits all SoD classes of her assigned roles, except the neutral one:

$$\begin{aligned} assigned\_class_u[u] = \{class \in SOD\_CLASSES \setminus \{\emptyset\} | \\ \exists r_1 \in R[r_1 \in auth\_roles_\gamma[u] \wedge assi\_class_r[r_1] = class]\} \end{aligned}$$

Definition 5. A SoD matrix is a set of pairwise mutual SoD class exclusions. A pairwise mutual SoD class exclusion mec_2 defines a pair of mutually exclusive SoD classes

$$mec_2\{class_1, class_2\}$$

with $class_1, class_2 \in SOD\_CLASSES \setminus \emptyset$ and $class_1 \neq class_2$. It is satisfied by a homogeneous RBAC state γ if no single user inherits roles which are assigned both SoD classes defined by it as mutually exclusive:

$$sat_{mec_2}[\gamma] \triangleq \forall u \in U(|assigned\_class_u[u]| < 2)$$

Note that we intentionally limit the use of SoD policies to homogeneous RBAC states to improve role semantics. While the model formalization would be equally applicable for non-homogeneous RBAC states, we argue that an RBAC state can be made homogeneous relatively easy, which improves the overall policy understandability and maintainability (cmp. sections 5 and 8).

6.2 Policy Enforcement

In the following we show that the proposed SoD policies are compatible with established SoD specifications for RBAC. In RBAC, SoD rules are commonly specified as MER constraints $mer\langle\{r_1, ..., r_n\}, t\rangle$, with each r_i as a role and n, t as positive integers, such that $1 < t \leq n$. If t or more roles are assigned to a user, the rule is violated. Formally, an RBAC state satisfying an MER constraint is denoted as follows [22]:

$$sat_{mer}[\gamma] \triangleq \forall u \in U(|auth\_roles_\gamma[u] \cap \{r_1, ...r_n\}| < t)$$

Thus, to show how the proposed SoD policies can be enforced in RBAC, we have to show how the three types of SoD policies (pairwise MEP, MER and mutually exclusive SoD classes) can be translated to MER constraints, as defined above. This is trivial for the pairwise MER policy since it can be defined as MER with $n = t = 2$:

$$mer_2\{r_1, r_2\} \equiv mer\langle\{r_1, r_2\}, 2\rangle$$

A pairwise MEP policy is equivalent to the set of pairwise MER policies that defines pairwise exclusions for all roles which inherit the mutually exclusive permissions transitively:

$$\begin{aligned} mep_2\{p_1, p_2\} \equiv \{mer_2\{r_1, r_2\} | \\ \forall r_1 \in R[\exists r \in R | (r_1, r) \in RH^* \wedge (r, p_1) \in PA], \\ r_2 \in R[\exists r \in R | (r_2, r) \in RH^* \wedge (r, p_2) \in PA]\} \end{aligned}$$

A pairwise mutual SoD class exclusion is equivalent to the set of pairwise MER policies that defines pairwise exclusions for all roles with the specified SoD classes:

$$\begin{aligned} mec_2\{class_1, class_2\} \equiv \{mer_2\{r_1, r_2\} | \\ \forall r_1 \in \text{roles}[class_1], r_2 \in \text{roles}[class_2]\} \end{aligned}$$

Both pairwise MEPs and mutual SoD class exclusions can be translated into a set of pairwise MER policies. Since any pairwise MER policy can be expressed as a classic MER constraint, any SoD matrix can be translated into established MER constraints and enforced as such.

7 EVALUATION

We cooperated with a large financial services provider to evaluate the proposed framework. An IAM expert from the company also took part in the expert interviews (see organization O6 in table 1). The conceptual overview of stakeholders and responsibilities was generally agreed upon as it was synthesized from the expert interviews. We hence defined two evaluation criteria: (i) *Technical applicability*: Can we confirm that the proposed SoD matrix and mutual pairwise exclusions are compatible with existing SoD specifications and can be enforced as such? (ii) *Simplified policy management*: Can we confirm that the framework provides a complexity reduction compared to established MER policies?

We were allowed to analyze anonymized, productively used SoD entitlement data of the organization. O6 employs around 4,500 people and manages around 10,000 digital identities. Similar to the other companies in the expert interviews, the banking group organizes SoD policies through SoD classes and a SoD matrix. The permissions in the data set are organized in the form of an RBAC model with a flat role hierarchy: The roles are called "business roles". Business roles are valid in the context of the entire company and bundle users based on a functional assignment, e.g. “customer support” or “liquidity forecast”. The permissions are organized in two layers: "system entitlements" and "technical permissions". Technical permissions are fine-grained permissions that are only assigned to system entitlements. SoD classes are only assigned to system entitlements, which makes the technical permissions irrelevant for our use case. Hierarchy relations between two business roles or two system entitlements are not allowed. We analyzed a total of 14 SoD classes and about 8,000 system entitlements.

We implemented two algorithms that allow us to generate MERs based on the SoD matrix, SoD classes and system entitlements. While we could not include them in this work due to space constraints, we provide them on a GitHub repository along with sample data to execute them[1]. The first algorithm collects SoD classes from permissions and transitively writes them to the inheriting roles. It guarantees the roles’ SoD class homogeneity and marks roles that would otherwise inherit more than one SoD class. If a role would inherit more than one SoD class we assign $\times$, which represents an invalid SoD class state. These violations can be addressed by splitting the permissions of the role into multiple roles. The second algorithm transforms a SoD Matrix and a set of homogeneous roles into equivalent MER constraints.

By applying the algorithms on the data, we assigned at least one SoD class to 209 roles. Out of these, 5 had more than one SoD class. Based on these assignments, we generated a total of 12,295 MERs. Table 2 summarizes the results. We validated the results with the IAM experts from O6. The experts confirmed the correctness of our results and the usefulness of our approach. They pointed out that the effort and error rate in managing SoD classes is significantly lower compared to managing MERs. MERs, however, can be enforced more easily and are immediately compatible with common IAM systems. On this basis we conclude that the evaluation criteria were met:

(i) *Technical applicability*: We were able transform a SoD matrix and permissions assigned with SoD classes of a real-world banking group into classic MERs. This means that a SoD matrix can be enforced as such. The enforcement of pairwise MERs and MEPs is trivial and the correctness of our translation was confirmed by the IAM experts of O6.

(ii) *Simplified policy management*: The quantifiable data complexity required to manage a SoD matrix is significantly lower than the data complexity for equivalent MERs. The real-world data set contained 14 SoD classes, 32 pairwise SoD class exclusions, 274 permissions with a non-neutral SoD class and 209 roles with at least one non-neutral SoD-class, amounting to a total of 529 entities that need to be managed. In contrast, an equivalent set of pairwise MERs amounts to 12,295 entities. In theory, the MERs could be algorithmically optimized to cover the same exclusions with fewer MERs that have a higher cardinality. However, these MERs would be optimized for low data volume only and would not represent a human-understandable real-world concept. Note also that individual stakeholders have to manage fewer than 529 entities: Governance employees only need to maintain the SoD matrix with 14 SoD classes and 32 pairwise exclusions, and domain experts only manage permissions, of which only 274 have a non-neutral SoD class.

[1]https://github.com/sgroll/semantic_sod

Table 2: Output of algorithm execution on real world data

Basic data	
Roles	2,494
Permissions	7,972
Role-permission assignments	18,692
SoD class data	
SoD classes	14
Pairwise SoD class exclusions (SoD Matrix)	32
Permissions with non-neutral SoD class	274
Roles with non-neutral SoD class	209
Algorithm results	
Resulting MERs	12,295
Homogeneity violations	5

8 DISCUSSION & CONCLUSION

This work proposed a framework for the creation and maintenance of semantically meaningful SoD policies. The work is grounded on seven expert interviews with SoD practitioners from the industry. In these interviews we attempted to gain an understanding and summarize best practices for SoD management in large and compliance-driven organizations. The developed framework aims to provide structures for semantically meaningful SoD which remain aligned with established practices.

The framework starts with a definition of SoD stakeholders and their responsibilities. It then defines thee types of SoD policies which are designed to be well-maintainable and easily understandable. Finally, the defined SoD policies are anchored in an integrated IAM data model to generalize their scope. After presenting the proposed framework, this work provides a formalization of the three defined SoD policy types in the well-established RBAC model. We show that the policies can be translated into classic MER constraints and evaluate this with a real-world data set, thus proving that they can be evaluated and enforced efficiently.

The proposed SoD policies are easily maintainable and understandable because of three characteristics: (i) All managed policies have a semantic meaning derived from a real-world requirement. (ii) The SoD matrix decouples the creation of SoD exclusions from the management of permissions, which eliminates the need for collaboration between stakeholders from different knowledge domains. (iii) The quantifiable data complexity of SoD exclusions managed via the SoD matrix is significantly lower than the complexity of equivalent MERs. Our work currently only applies to rules with static SoD properties. This can be addressed by developing dynamic SoD classes: Similar to dynamic SoD rules, a user may posses permissions of conflicting SoD classes. When permissions of one of the two classes are used within a specified context (e.g. time, operation, customer etc.), the permissions of the other class cannot be used in this context. We leave a detailed specification and analyses of dynamic SoD classes for future work.

ACKNOWLEDGMENTS

The research leading to these results was supported by the German Federal Ministry of Education and Research as part of the DEVISE project (https://devise.ur.de/).

REFERENCES

[1] William C Adams. 2015. Conducting semi-structured interviews. *Handbook of practical program evaluation* (2015), 492–505.

[2] Gail-Joon Ahn and Ravi Sandhu. 2000. Role-Based Authorization Constraints Specification. 3, 4 (nov 2000), 207–226. https://doi.org/10.1145/382912.382913

[3] Nuray Baltaci and James Joshi. 2020. A Constraint and Risk-aware Approach to Attribute-based Access Control for Cyber-Physical Systems. *Computers & Security* 96 (05 2020), 101802. https://doi.org/10.1016/j.cose.2020.101802

[4] Basel Committee on Banking Supervision. 2011. *Basel III - A Global Regulatory Framework for More Resilient Banks and Banking Systems.* Bank for International Settlements.

[5] Khalid Bijon, R. Krishman, and R. Sandhu. 2013. Constraints specication in attribute based access control. *SCIENCE* 2 (01 2013), 131–144.

[6] Bundesanstalt für Finanzdienstleistungsaufsicht. 2024. *Versicherungsaufsichtliche Anforderungen an die IT (VAIT).*

[7] David W Chadwick, Wensheng Xu, Sassa Otenko, Romain Laborde, and Bassem Nasser. 2007. Multi-session Separation of Duties (MSoD) for RBAC. In *2007 IEEE 23rd International Conference on Data Engineering Workshop.* 744–753. https://doi.org/10.1109/ICDEW.2007.4401062

[8] Hong Chen and Ninghui Li. 2006. Constraint Generation for Separation of Duty *(SACMAT '06).* Association for Computing Machinery, New York, NY, USA, 130–138. https://doi.org/10.1145/1133058.1133077

[9] Weihe Chen, Zhu Tang, and Shiguang Ju. 2008. Enforcement of Spatial Separation of Duty Constraint. In *2008 The 9th International Conference for Young Computer Scientists.* 2108–2114. https://doi.org/10.1109/ICYCS.2008.223

[10] Jason Crampton. 2005. A Reference Monitor for Workflow Systems with Constrained Task Execution *(SACMAT '05).* Association for Computing Machinery, New York, NY, USA, 38–47. https://doi.org/10.1145/1063979.1063986

[11] Marcelo Antonio de Carvalho and Paulo Bandiera-Paiva. 2017. Evaluating ISO 14441 privacy requirements on role based access control (RBAC) restrict mode via Colored Petri Nets (CPN) modeling. In *2017 International Carnahan Conference on Security Technology (ICCST).* 1–8. https://doi.org/10.1109/CCST.2017.8167833

[12] Deutsche Bundesbank. 2017. Bankaufsichtliche Anforderungen an die IT (BAIT). Available at https://www.bundesbank.de/de/aufgaben/bankenaufsicht/einzelaspekte/risikomanagement/bait/bankaufsichtliche-anforderungen-an-die-it-598580.

[13] Ludwig Fuchs, Michael Kunz, and Günther Pernul. 2014. Role model optimization for secure role-based identity management. (2014).

[14] Virgil D Gligor, Serban I Gavrila, and David Ferraiolo. 1998. On the formal definition of separation-of-duty policies and their composition. In *Proceedings. 1998 IEEE Symposium on Security and Privacy (Cat. No. 98CB36186).* IEEE, 172–183.

[15] Rattikorn Hewett, Phongphun Kijsanayothin, and Aashay Thipse. 2008. Security analysis of role-based separation of duty with workflows. In *2008 Third International Conference on Availability, Reliability and Security.* IEEE, 765–770.

[16] Vincent Hu, David Ferraiolo, D. Kuhn, A. Schnitzer, Knox Sandlin, R. Miller, and Karen Scarfone. 2014. Guide to attribute based access control (ABAC) definition and considerations. *National Institute of Standards and Technology Special Publication* (01 2014), 162–800.

[17] Xin Jin, Ram Krishnan, and Ravi Sandhu. 2012. A role-based administration model for attributes. In *Proceedings of the first international workshop on secure and resilient architectures and systems.* 7–12.

[18] Sascha Kern, Thomas Baumer, Ludwig Fuchs, and Günther Pernul. 2023. Maintain High-Quality Access Control Policies: An Academic and Practice-Driven Approach. In *IFIP Annual Conference on Data and Applications Security and Privacy.* Springer, 223–242.

[19] Konstantin Knorr and Harald Weidner. 2001. Analyzing Separation of Duties in Petri Net Workflows. In *Information Assurance in Computer Networks*, Vladimir I. Gorodetski, Victor A. Skormin, and Leonard J. Popyack (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 102–114.

[20] D. Richard Kuhn. 1997. Mutual Exclusion of Roles as a Means of Implementing Separation of Duty in Role-Based Access Control Systems. In *Proceedings of the Second ACM Workshop on Role-Based Access Control* (Fairfax, Virginia, USA) *(RBAC '97).* Association for Computing Machinery, New York, NY, USA, 23–30. https://doi.org/10.1145/266741.266749

[21] Michael Kunz, Alexander Puchta, Sebastian Groll, Ludwig Fuchs, and Günther Pernul. 2019. Attribute quality management for dynamic identity and access management. *Journal of Information Security and Applications* 44 (2019), 64–79. https://doi.org/10.1016/j.jisa.2018.11.004

[22] Ninghui Li, Mahesh V. Tripunitara, and Ziad Bizri. 2007. On mutually exclusive roles and separation-of-duty. *ACM Trans. Inf. Syst. Secur.* 10, 2 (may 2007), 5–es. https://doi.org/10.1145/1237500.1237501

[23] Jan Mendling, Karsten Ploesser, and Mark Strembeck. 2008. Specifying Separation of Duty Constraints in BPEL4People Processes. In *Business Information Systems*, Witold Abramowicz and Dieter Fensel (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 273–284.

[24] Ian Molloy, Hong Chen, Tiancheng Li, Qihua Wang, Ninghui Li, Elisa Bertino, Seraphin Calo, and Jorge Lobo. 2008. Mining roles with semantic meanings. In *Proceedings of the 13th ACM symposium on Access control models and technologies.* 21–30.

[25] One Hundred Seventh Congress of the United States of America. 2002. Sarbanes-Oxley Act of 2002. Pub. L. No. 107-204, 116 Stat. 745.

[26] Indrakshi Ray, Na Li, Robert France, and Dae-Kyoo Kim. 2004. Using Uml to Visualize Role-Based Access Control Constraints *(SACMAT '04).* Association for Computing Machinery, New York, NY, USA, 115–124. https://doi.org/10.1145/990036.990054

[27] Pierangela Samarati and Sabrina Capitani de Vimercati. 2000. Access control: Policies, models, and mechanisms. In *International school on foundations of security analysis and design.* Springer, 137–196.

[28] Ravi S Sandhu. 1998. Role-based access control. In *Advances in computers.* Vol. 46. Elsevier, 237–286.

[29] Ravi S Sandhu and Pierangela Samarati. 1994. Access control: principle and practice. *IEEE communications magazine* 32, 9 (1994), 40–48.

[30] M. E. Shin and G. Ahn. 2001. Role-Based Authorization Constraints Specification Using Object Constraint Language. In *2012 IEEE 21st International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises.* IEEE Computer Society, Los Alamitos, CA, USA, 157. https://doi.org/10.1109/ENABL.2001.953406

[31] R.T. Simon and M.E. Zurko. 1997. Separation of duty in role-based environments. In *Proceedings 10th Computer Security Foundations Workshop.* 183–194. https://doi.org/10.1109/CSFW.1997.596811

[32] Karsten Sohr, Gail-Joon Ahn, Martin Gogolla, and Lars Migge. 2005. Specification and Validation of Authorisation Constraints Using UML and OCL. In *Computer Security – ESORICS 2005*, Sabrina de Capitani di Vimercati, Paul Syverson, and Dieter Gollmann (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 64–79.

[33] Karsten Sohr, Tanveer Mustafa, Xinyu Bao, and Gail-Joon Ahn. 2008. Enforcing Role-Based Access Control Policies in Web Services with UML and OCL. In *2008 Annual Computer Security Applications Conference (ACSAC).* 257–266. https://doi.org/10.1109/ACSAC.2008.35

[34] U.S. Department of Health & Human Services. n.d.. American Health Insurance Portability and Accountability Act. https://www.hhs.gov/hipaa/index.html. Accessed: 2023-06-10.

[35] Patrick Wolf and Nick Gehrke. 2009. Continuous compliance monitoring in ERP systems-A method for identifying segregation of duties conflicts. *Wirtschaftsinformatik Proceedings 2009* 39 (2009).

[36] Christian Wolter and Andreas Schaad. 2007. Modeling of Task-Based Authorization Constraints in BPMN. In *Business Process Management*, Gustavo Alonso, Peter Dadam, and Michael Rosemann (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 64–79.

[37] Christian Wolter and Andreas Schaad. 2007. Modeling of Task-Based Authorization Constraints in BPMN. In *Business Process Management*, Gustavo Alonso, Peter Dadam, and Michael Rosemann (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 64–79.

[38] Christian Wolter, Andreas Schaad, and Christoph Meinel. 2007. Deriving XACML Policies from Business Process Models. In *Web Information Systems Engineering – WISE 2007 Workshops*, Mathias Weske, Mohand-Saïd Hacid, and Claude Godart (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 142–153.

[39] Zhongyuan Xu. 2014. *Mining meaningful role-based and attribute-based access control policies.* Ph. D. Dissertation. State University of New York at Stony Brook.

[40] Benyuan Yang and Hesuan Hu. 2023. Analysis of Authorization Constraints via Integer Linear Programming. *IEEE Transactions on Knowledge and Data Engineering* 35, 3 (2023), 2258–2271. https://doi.org/10.1109/TKDE.2021.3124271