UiO : University of Oslo

Hamed Arshad

Toward Semantic Attribute-Based Access Control

Fine-grained protection of data in e-Health

Thesis submitted for the degree of Philosophiae Doctor

Department of Informatics
The Faculty of Mathematics and Natural Sciences

2022

To all the people who helped me during my PhD journey.

Abstract

Access control plays a crucial role in providing security and privacy in e-Health systems. Attribute-Based Access Control (ABAC) is a fine-grained, dynamic access control model that has several advantages over the traditional access control models. ABAC works based on the attributes of the subject, object, action, and environment and is supposed to be the best fit for distributed environments. However, as the mismatch between attributes in a distributed environment like the e-Health domain is inevitable, ABAC is augmented with semantic technologies, forming Semantic Attribute-Based Access Control (SABAC), to take into account the semantic relationships between attributes.

Hitherto, a considerable number of SABAC schemes have been proposed for different contexts. This dissertation performs a systematic literature review on the existing SABAC schemes to identify different strategies for developing SABAC as well as open problems towards an ideal SABAC.

One of the main problems of the access control systems is the fact that they rely on a trusted reference monitor that not only can be easily bypassed, e.g., by direct access to the data, but also restricts the scalability, as it should always be online. This issue is addressed by proposing a cryptographic counterpart for access control systems, e.g., Attribute-Based Encryption (ABE) for ABAC. However, there is a lack of a cryptographic counterpart for SABAC. Hence, this dissertation proposes the first Semantic Attribute-Based Encryption (SABE) framework through the combination of ABE schemes and semantic technologies. The proposed SABE framework is developed based on two different schemes namely, Semantically-Enriched Key (SEK) and Semantically-Enriched Access Structure (SEAS). Although SABE can be considered as a cryptographic SABAC, obligations as an important feature of ABAC are missing in ABE (and accordingly SABE) schemes. Obligations are meant to enforce extra constraints that cannot be addressed using normal access control policies. Hence, this dissertation extends ABE schemes with enforceable obligations based on the trusted hardware enclaves provided by Intel Software Guard Extensions (SGX) to form an Attribute-Based Encryption with enforceable OBligations (OB-ABE) scheme. The dissertation also proposes a formal language for the specification of enforceable obligations in OB-ABE. The proposed SABE-SEK, SABE-SEAS, and OB-ABE schemes are implemented to evaluate their performances compared to the conventional ABE schemes. The security of the proposed schemes is also formally verified using the widely accepted formal verification methods and tools.

Another issue with ABAC and SABAC is the lack of a formal mechanism for analyzing access control policies when authoring or integrating them. ABAC-based systems basically address the inconsistencies between access control policies at runtime (when making a decision) based on some combining algorithms

provided by the XACML standard. However, runtime conflict resolution negatively affects the performance of the access control system. There are also other properties for the access control policies that need to be analyzed. Therefore, this dissertation proposes an approach to analyze the access control policies using a process algebra, called mCRL2. The proposed approach analyzes the completeness, consistency, and safety of the access control policies before their deployment.

Preface

This thesis is submitted in partial fulfillment of the requirements for the degree of *Philosophiae Doctor* at the University of Oslo. The research presented here was conducted at the University of Oslo, under the supervision of Professor Christian Johansen and Professor Olaf Owe.

The thesis is a collection of four papers, presented in chronological order of writing, which have been adjusted to the format of the thesis. The papers are preceded by an introductory part that relates them to each other and provides background information and motivation for the work. I am the main author of all the papers.

Acknowledgments

I would like to express my profound gratitude to Christian Johansen, my main supervisor, for his excellent guidance and for giving me the opportunity to work together. I am very grateful for his permanent support, scientific guidance, constant optimism, and encouragement from the first day. I have learned a lot from Christian. I also appreciate his careful revision of all my work. Christian, you are amazing, and I am very lucky that I have had this chance to have such a nice supervisor. I would never forget your help and support. Thank you very much.

I would also like to thank Olaf Owe, my secondary supervisor, for his kind support and guidance. I am very happy that I have had the privilege to work with Olaf. He is an excellent supervisor and cares about all people who work with him. Olaf is like a father to his students. Thank you so much Olaf for everything.

I would like to thank Josef Noll for his amazing guidance, supervision, and support. Josef is a great supervisor, boss, and friend. I am proud that I have had the chance to work with Josef. Thank you very much Josef.

Martin Steffen deserves special thanks for his guidance, support, and positive discussions. He is a very knowledgeable person who has supported me whenever I needed his help. Thank you very much Martin.

During my PhD, I have had two research visits at the Charmers University of Technology in Sweden as well as at the Luxembourg University in Luxembourg. I would like to thank Gerardo Schneider and Pablo Picazo-Sanchez for their supervision and the nice collaboration that we had during my research visit at the Charmers University of Technology. I would also like to thank Ross Horne, Sjouke Mauw, and Sergiu Bursuc for their hospitality and support in Luxembourg. I have learned a lot from all these people, and I cannot say thank you enough.

Words cannot express my gratitude to Tim A.C. Willemse from the Eindhoven University of Technology for his important and valuable help and support during the last stage of my PhD. I have learned a lot from Tim in a short period. Tim is a super nice and positive person. Thank you very much Tim.

My thanks also go to the administration staff at the Department of Informatics, especially to Øystein Christiansen and Mozhdeh Sheibani Harat for their support during my PhD.

Hamed Arshad
Oslo, December 2022

List of Papers

Paper I

Hamed Arshad, Christian Johansen, and Olaf Owe, "Semantic Attribute-Based Access Control: A review on current status and future perspectives". *Journal of Systems Architecture.*Vol. 129, (2022), pp. 1–24. DOI: 10.1016/j.sysarc.2022.102625.

Paper II

Hamed Arshad, Christian Johansen, Olaf Owe, Pablo Picazo-Sanchez, and Gerardo Schneider, "Semantic Attribute-Based Encryption: A Framework for Combining ABE schemes with Semantic Technologies". *Information Sciences.* (2022). DOI: 10.1016/j.ins.2022.10.132.

Paper III

Hamed Arshad, Pablo Picazo-Sanchez, Christian Johansen, and Gerardo Schneider, "Attribute-Based Encryption with Enforceable Obligations". *Journal of Cryptographic Engineering.* DOI: 10.1007/s13389-023-00317-1.

Paper IV

Hamed Arshad, Ross Horne, Christian Johansen, Olaf Owe, and Tim A. C. Willemse "Process Algebra Can Save Lives: Static Analysis of XACML Access Control Policies using mCRL2". *In: Mousavi, M.R., Philippou, A. (eds) Formal Techniques for Distributed Objects, Components, and Systems. FORTE 2022. Lecture Notes in Computer Science, vol 13273. Springer, Cham.* DOI: 10.1007/978-3-031-08679-3_2.

Contents

List of Figures

List of Tables

Part I: Overview

Chapter 1

Introduction

Summary

This dissertation aims to address security and privacy challenges in e-Health systems by proposing a secure access control mechanism for protecting resources. This chapter first, in the motivation section, describes the necessity of e-Health systems followed by some security and privacy challenges that may arise in such systems. Since access control is a fundamental security mechanism and Attribute-Based Access Control (ABAC) is a popular choice in e-Health systems, the usefulness of ABAC and associated challenges when employed in e-Health systems are also discussed. The motivation section ends with open problems that are of interest to this dissertation. Accordingly, the research goals are formulated and the employed research methods are described. Finally, this chapter provides the outline of this dissertation.

1.1 Motivation

The growth of the aging population causes an increase in the rate of chronic diseases such as diabetes, cardiovascular diseases, and mental illnesses. Such diseases require long-term treatment with frequent hospital/clinic-based checkups, which in turn induces excessive costs and stress on the patients due to the repeated trips to the hospital. This causes significant adverse effects on the patient's quality of life [134]. Without regular monitoring and medical care, chronic diseases can cause critical conditions for the patients. Therefore, developing a system that can enable patients diagnosed with chronic diseases to receive remote treatment at home is useful for both the patients and the medical infrastructure (facilities, doctors, staff, etc.) [100]. Providing home-based long-term medical care services for chronic patients has the potential of enhancing the quality of their lives. Nowadays, information and communication technologies are increasingly used in the medical sector to improve and facilitate health care delivery services. For example, using e-Health services patients and doctors can access medical services and information at any time and anywhere via the Internet [10]. Patients without leaving home can obtain almost the same medical services as at the hospital. Specifically, patients in rural areas are no longer required to travel long distances to visit a doctor. The medical staff can remotely monitor the health condition of the patients and physicians can treat patients in a remote place at the right time and lower cost. Therefore, e-Health systems provide more convenience for patients and reduce the patients' expenses such as travel and hospitalization costs. Besides, the patients' medical records stored in databases of medical servers allow doctors to provide more accurate diagnoses and prescribe

better treatments [136].

Security and privacy are very important issues in e-Health systems that deal with the patients' personal (health) information. Adversaries may launch different security attacks aiming to breach security and privacy. For instance, an adversary may try to get access to the patients' medical information, which are stored in the databases of medical centers (or cloud databases), and (illegally) change them, which may lead to significant harm to the patients [10]. They may also misuse such personal information and breach the privacy of the patients as most patients do not want others (like business competitors or even relatives) to have access to their health information (they prefer a high level of confidentiality and anonymity). Therefore, the risk of using new services and applications should be considered and proper security measures should be taken into account. Access control plays a crucial role in providing security and privacy of information and users in e-Health systems. Access control, as a fundamental security mechanism, is part of the authorization process where the system checks the access requests against policies to ensure that only authorized (legal) users get access to resources in a system.

Since the seminal work of Lampson in the early 1970s [77], several access control models have been proposed. One of the most promising access control models is ABAC [62, 102], which provides fine-grained protection based on a set of attributes and access control policies. ABAC is a successor of Role-Based Access Control (RBAC) where the involved entities (e.g., subject or object) have associated attributes that are used to provide access control. ABAC has several advantages over traditional access control models such as Mandatory Access Control, Discretionary Access Control, and RBAC, and has reached the maturity of OASIS standards with the eXtensible Access Control Markup Language (XACML) [8] and the Security Assertion Markup Language (SAML) [28]. Under the ABAC model, there is no need to assign capabilities (access rights) to subjects (e.g., users, groups, and roles) in advance. Upon receiving an access request, the access decision would be made based on the attributes of the requested object (resource), the attributes of the requester (subject), the conditions of the environment (e.g., time of the day, authentication level, location), the attributes of the desired action, and the predefined access control policies.

ABAC is a popular choice in e-Health systems because of the flexibility given by the attributes and the way they cater for fine-grained access control in emergencies [2, 52, 68, 92, 95, 109, 110]. Nevertheless, only a modest amount of adoption can be seen in the Health & Home care IT solutions in Europe. Even if in other industries RBAC can be enough, for medical data and processes the ABAC and more granular extensions of it are desired due to the highly sensitive and private nature of the information being accessed and the collaborative nature of the work. ABAC can handle non-trivial access policies like for collaborative access control, needed in e-Hospitals, where multiple subjects should be involved, with varying attributes and roles. A classic example is when a Doctor needs to be present (logged in) in order for a Nurse to perform a procedure. Detailed auditing of health-care processes (like administering medicines, preparing operation rooms, home visits) can be done using the notion of obligations in which the decision-

point instructs the enforcement-point to first perform some audit/logging actions before granting or denying access. Moreover, with the increased digitization of health records and eRegistries (e.g., consider the new DNA banks), and smart homes producing additional environmental data, the health sector enters into the big data era. Hence, ABAC, which offers fine-grained protection, makes it possible to grant access only to those pieces of the data that are needed to provide the respective health services (instead of granting access to someone for the whole data set or for whole records).

ABAC has appeared to be useful in open and distributed systems. However, since such systems are heterogeneous, the attributes of the involved entities, i.e., the subject, object, action, and environment, may not necessarily match those specified in the policies defined for accessing services or objects. For example, an e-Health system may represent adult patients with an attribute *"age"*, while patients may want to demonstrate this by providing an attribute *"hasDrivingLicense"*. However, the access control engine cannot infer that having a driving license means that the requester is an adult person. In ABAC, this issue could be addressed when defining a policy by including all the possible synonyms (semantically) of each attribute, e.g., by specifying several policies for the same object or one general policy covering all the synonyms of attributes. However, when a change occurs in a policy, a large number of policies may need to be updated accordingly, which in turn makes the management of policies a complicated and error-prone task.

To address such problems, another type of access control called Semantic Attribute-Based Access Control (SABAC) has emerged as an extension of ABAC. The goal of SABAC is to augment ABAC with semantic technologies in order to take into consideration the semantic relationships between the involved entities when making a decision. Semantic technologies help the access control engine to infer implicit knowledge (e.g., semantic synonyms of attributes) from explicit knowledge, i.e., the attributes provided in the requests and predefined policies.

A concern in SABAC is inconsistencies in access control policies as they can cause availability (denying authorized requesters access) and safety (granting access to unauthorized requesters) problems. Most of the existing SABAC schemes either rely on the XACML combining algorithms, which is a runtime approach, for conflict resolution (those that extended the XACML standard) or did not address conflict resolution at all. However, the implicit knowledge provided by the semantic technologies in SABAC may incur extra inconsistencies between policies because every policy may cover more requests (compared to ABAC). Therefore, SABAC lacks a proper solution for resolving conflicts when authoring/integrating policies.

On the other hand, SABAC (and ABAC), like every access control mechanism, relies on a trusted reference monitor that checks all access requests against access control policies. The reference monitor should always be online (and trusted) to intercept the access requests, which restricts scalability. Besides, the reference monitor can be easily bypassed, e.g., by getting direct access to the data on a storage device. In response, cryptographic access control techniques are proposed to overcome such shortcomings [72]. For instance, Attribute-Based

Encryption (ABE) schemes are developed to protect resources in a fine-grained manner based on a set of attributes and access structures (i.e., access control policies) [17, 24, 63, 133]. In contrast to ABAC, ABE [14, 49] does not rely on a trusted engine (monitor), but uses cryptographic techniques to provide fine-grained data protection based on Access Structures (ASs) (i.e., access control policies) represented as boolean formulae over public attributes. Any user who holds a set of public attributes satisfying the AS can decrypt the ciphertext (more precisely this is the case in Ciphertext-Policy Attribute-Based Encryption (CP-ABE)). For instance, a user having the following attributes $S = \{doctor, hospitalA\}$ can decrypt ciphertexts encrypted under the following AS: $\mathcal{T} = ((doctor \wedge hospitalA) \vee (researcher \wedge instituteB))$ as the set S satisfies $\mathcal{T}$. ABE makes it possible to encrypt data not only for a single user (identified by a unique attribute) but also for a group of users (identified by a set of public attributes).

Until now, a considerable number of ABE schemes have been proposed and employed in several domains such as e-Health [85], online social networks [103], hardware security [48], fog computing [69], and storing sensitive data in public clouds [79]. Furthermore, real world companies like Zeutro[1] deploy security systems based on ABE. Moreover, standards like ETSI[2] have been defined (TS 103 458 and TS 103 532) presenting applications to industrial IoT and cloud.

ABE might be considered as a cryptographic replacement of ABAC, but it does not achieve the goals of SABAC. This is because the existing ABE schemes are not semantic-aware, i.e., they do not take into account the semantics of attributes. For instance, suppose the Electronic Health Records (EHR) of *PatientA* is encrypted based on the following AS: (*Medical Doctor* $\wedge$ *Employer* = *Emergency Hospital*). A surgeon working at the Emergency Hospital with attributes {*Surgeon*, *Employer* = *Emergency Hospital*} will not be able to access the EHR of *PatientA* because she does not hold the *Medical Doctor* attribute. Even though a surgeon is a medical doctor, ABE works syntactically and cannot infer such knowledge. Any basic medical ontology would have *Surgeon* as a subconcept of *Medical Doctor* and would allow an inference engine to infer this information, which can then be added as the extra attribute needed in such emergency cases.

Interoperability problems are notoriously common also in e-Health where medical staff from different healthcare institutions, which may use different terminologies for attributes, need to access data like EHR. When coupled with a growing trend of moving medical records into public clouds [13, 132][3], ABE gains even more relevance. For example, *Medical Doctor*, *Physician*, *Lege*[4], and *läkare*[5] attributes may be used in different hospitals (in different countries) to represent a medical doctor as they are semantically the same.

[1] https://bit.ly/3gvWRGE
[2] https://bit.ly/3xiLoQk
[3] https://ibm.co/2Tz2LxL
[4] In Norwegian "*Lege*" means "Physician"
[5] In Swedish "*läkare*" means "Physician"

On the other hand, ABE schemes do not offer an important feature of ABAC called *obligations*. Obligations, which are meant to enforce extra constraints that cannot be managed through normal policies, are greatly desired, e.g., in e-Health, because of the accountability and highly interactive style of work where many types of actions must be logged, various authorizations are needed from experts (i.e., confirmations, e.g., from the doctor on duty), or simply sending notifications to relevant parties (like to family/guardians) are required by law. For instance, in an e-Health system, one can define an obligation for writing a log for each action taken by the system (which is very useful to keep track of "who did what treatment and when"), or an obligation to send notifications and possibly also waiting for an acknowledgment before granting access.

1.2 Research Goals

Until now, a considerable number of research efforts have been conducted in developing semantic attribute-based access control schemes. However, as explained in the previous section, there still exists a number of open problems that need to be addressed in order to achieve an ideal SABAC. Hence, the main goal of this dissertation is

> *to develop a secure fine-grained, flexible, semantic-aware access control system.*

To achieve our main goal, we consider the following research questions:

RQ1: *How can Semantic Attribute-Based Access Control be realized?*

It is essential to demonstrate the strategies for developing semantic attribute-based access control, which is a fine-grained, flexible, semantic-aware access control model. Answering this question helps us to get an overview of the existing strategies/solutions and adopt the best suitable strategy when proposing/developing an SABAC.

RQ2: *How can a Cryptographic Semantic Attribute-Based Access Control scheme be developed?*

SABAC like every access control mechanism relies on a trusted reference monitor that can be bypassed and may restrict scalability. To improve security and scalability, the second research question focuses on developing a cryptographic counterpart for SABAC.

RQ3: *How can access control policies be verified?*

Access control policies play a crucial role in achieving a secure access control system. Hence, since it is important to verify that access control policies are complete, correct, and consistent, the third research question focuses on the formal verification of access control policies.

Table 1.1 demonstrates the connection between the above research questions and the included research papers.

Table 1.1: Connection between research questions and research papers

Research Questions	Contributions
RQ1	Paper I
RQ2	Paper II and Paper III
RQ3	Paper IV

1.3 Research Methods

This section summarizes the methods and tools that are used in this dissertation in order to address the research questions formulated in the previous section.

According to [39], there are four different research methods for cyber-security research namely *observational*, *theoretical*, *experimental*, and *applied*, where each includes subcategories as shown in Figure 1.1.

In this dissertation, we have favored employing *observational*, *theoretical*, *experimental*, and *applied* research methods. In the first step, in order to address the first research question, we applied *observational* research methods (more specifically, *exploratory study*) to collect, analyze, and interpret observations about existing SABAC schemes. To do so, we conducted a *Systematic Literature Review* of the existing research efforts on SABAC based on the guidelines provided in [73, 74]. In order to conduct the review, we defined a set of research questions, accompanied by their objectives, along with a search strategy, which determines how to find relevant studies. We searched *Google Scholar*, *Springer*, *Elsevier Scopus*, *Web of Science*, *Science Direct*, *IEEE Xplore Digital Library*, *Citeseer library*, and *ACM digital library* based on a number of keywords and strings to find the relevant research efforts. We defined some inclusion and exclusion criteria for selecting the set of papers for the review. Then, we used the *forward snowballing process* described in [131] to identify more relevant papers by checking the studies that cited the papers in the initial set. Finally, we summarized, compared, and discussed the selected papers in an *objective* manner.

We applied *theoretical*, *experimental*, and *applied* research methods when addressing the second research question. Based on *theoretical* research methods, we first combined ABE schemes with semantic technologies to make ABE schemes semantic-aware and to enable cross-domain interoperability. Then, we augmented ABE schemes with enforceable obligations through the use of trusted hardware enclaves. According to *formal theoretical* research methods, we formally verified the security of the proposed schemes by reusing standard techniques used for the underlying ABE scheme (in Paper II), and the ProVerif formal verification tool [1, 15] (in Paper III). In other words, we provided a number of lemmas, theorems, and their proofs when analyzing the security of the proposed schemes. We also employed *applied* research methods because they help to figure out the effectiveness of a proposal (in addressing a problem) compared to previous solutions. Since *applied* research methods consist of design, implementation, and test, we implemented and tested the schemes proposed in Paper II and Paper III. In both Paper II and Paper III, we performed some experiments through some

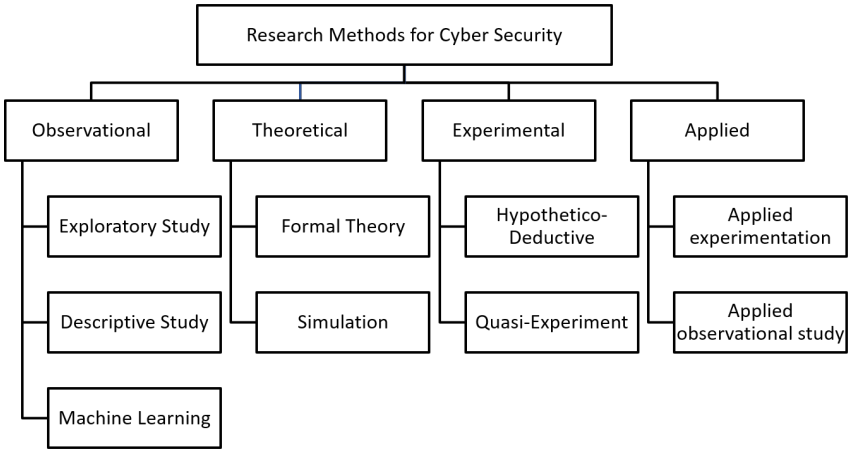


Figure 1.1: Research methods for cyber security.

examples to demonstrate that our proposed schemes achieve their goals. However, we applied the *quasi-experiment* research method for these experiments as the ontology changes and also real obligations were not considered. Based on *applied experimentation* research methods, the performance of the proposed schemes is evaluated through the execution of different tests.

The third research question is addressed by proposing an approach to formally verify access control policies using process algebra. Based on *formal theoretical* research methods, we proposed an approach that translates access control policies into a process algebra. We also formally specified desired properties of access control policies. Then, in order to show that the proposed approach works as it should, based on the *hypothetico-deductive* method, we analyzed the specifications using a model checker. A prototype of the proposed approach is also designed, implemented, and tested according to *applied* research methods.

The list of research methods applied in the included research papers is provided in Table 1.2.

1.4 Structure of the Dissertation

The structure of the rest of this dissertation is as follows:

- Chapter 2 provides the basic information on the XACML standard, which provides a policy language and a reference architecture for ABAC, semantic technologies, ABE, and Intel Software Guard Extensions (SGX). Chapter 2 also presents the state of the art on SABAC and related work on ABE, SGX-based schemes, and obligation specification.

Table 1.2: Research methods applied in the research papers.

		Paper I	Paper II	Paper III	Paper IV
Observational	Exploratory Study Descriptive Study Machine Learning	✓			
Theoretical	Formal Theory Simulation		✓	✓	✓
Experimental	Hypothetico-Deductive Quasi-Experiment		✓	✓	✓
Applied	Applied experimentation Applied observational study		✓	✓	✓

- Chapter 3 provides a summary of the research papers included in this dissertation.
- Chapter 4 concludes the first part of this dissertation by providing a summary of contributions, answering the research questions, and discussing the limitations and future work.
- Part II provides the full text of the papers that are part of this dissertation.

Chapter 2

Background

This chapter has two parts, where the first part provides the preliminaries on the XACML standard, semantic technologies, attribute-based encryption, and Intel SGX as these were used in the included research papers. The second part of this chapter first presents the state of the art on semantic attribute-based access control and then provides the related work on attribute-based encryption, SGX-based schemes, obligation specification languages, and approaches for analysis of access control policies based on the research questions and included papers.

2.1 Preliminaries

2.1.1 XACML Standard

The eXtensible Access Control Markup Language is a standard managed by OASIS (Organization for the Advancement of Structured Information Standards)[2], providing both a policy language, as well as a reference architecture for ABAC. The standard also specifies the process by which the requests are evaluated based on the predefined policies.

Three main benefits of XACML are:

- XACML is *policy-based*, which makes it possible to describe different authorization scenarios easily. Besides, it is easy to audit the authorization artifacts and to create complex policies depending on the authorization scenarios.
- XACML is *attribute-based* or *multi-factor*. The XACML policies are based on attributes of subjects, objects, actions, and environment. For example, subjects (users) can be described in terms of age, date of birth, citizenship, and clearance level. Similarly, objects (resources) may have attributes such as: location, classification, and owner.
- XACML is *technology-neutral*. It means that the XACML can be developed using different languages, e.g., C#, Python, and Java, and also can be used in different applications such as legacy applications or mainframe applications.

2.1.1.1 Policy specification language

The XACML policy language provides, as shown in Fig. 2.1, elements for defining policies, rules, algorithms for combining rules and policies, obligations, and attributes of involved entities.

[2] https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml

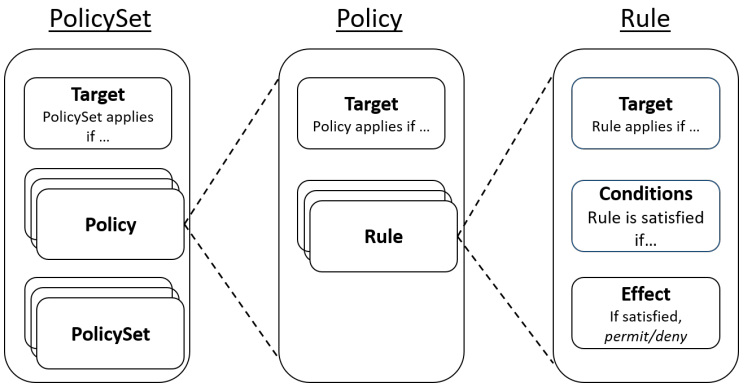


Figure 2.1: The structure of the policy language

The main structural elements of the XACML policy language are PolicySet, Policy, and Rule.

A PolicySet contains multiple policies that have the same general purpose. The *target* is used to determine to which requests is this PolicySet applicable. A PolicySet may also contain references to other PolicySets, i.e., policies in other PolicySets can also be included in a PolicySet. A Policy is composed of one or more rules. The *target* of a Policy is more fine-grained than that of a PolicySet. A Rule includes a set of *conditions* as its core part and an *effect* element, which can be either a Deny or Permit. A Rule also has a *target* specifying a more fine-grained set of access requests for which the Rule is applicable.

Targets are statements based on subject, object, action, and environment attributes that help the access control engine to find the right (applicable) policies (as well as policysets and rules) when receiving an access request. In other words, targets expedite the decision-making process as the access control engine (decision making engine) does not evaluate an access request against all the existing policies. However, targets are not powerful enough because they are limited to a AND/OR/AND structure. Besides, attributes in a target can only be matched to single constants. Hence, *conditions* are added to rules to overcome these limitations. In a *condition*, attributes can be matched against not only constant values but also other attributes. Besides, it is also possible to use non-Boolean functions. If the attributes existing in a request match those in the target of a rule, the conditions part will be checked and if the conditions evaluate to true, then the effect of the rule will be returned to the policy containing the rule. However, if the conditions are not applicable to the access request, or if an error happens, NotApplicable or Indeterminate, respectively, will be returned as the effect of the rule.

Since a Policy may contain more than one Rule, contradictory effects may be returned for the same access request. For example, suppose Alice, who is a programmer, wants to access ObjectA at 4:00 AM and the applicable Policy includes the two following rules:

Rule 1: (Job-title = Programmer) AND (Action-id = Read) AND (Object-id = ObjectA) → (Effect = Permit).

Rule 2: (18:00 < Current-Time < 08:00) → (Effect = Deny)

Since both rules are applicable to Alice's request, the parent policy receives both Permit and Deny effects, which are in contradiction to each other. In the XACML standard, such issues are addressed by means of combining algorithms, which combine the results of conflicting rules (*rule-combining algorithms*) and conflicting policies (*policy-combining algorithms*). For example, the "Permit Overrides" combining algorithm states that in the case of receiving a Permit and a Not Applicable, an Indeterminate or a Deny, the final decision is Permit, i.e., Permit overrides other effects.

2.1.1.2 Architecture

The reference architecture of the XACML standard, shown in Fig. 2.2, includes the (possibly distributed) components described below.

A Policy Enforcement Point (PEP) is responsible for protecting objects, which are to be understood rather generally, including applications, services, and files. The PEP receives incoming requests and forwards them to the second component of the XACML architecture, i.e., the Context Handler, which converts them into XACML requests and forwards the converted ones to a Policy Decision Point (PDP). The PDP is the core of the architecture which evaluates incoming access requests against access control policies and makes decisions which it returns to the PEP (through the Context Handler) to be enforced. Two components in the XACML architecture namely the Policy Information Point (PIP) and the

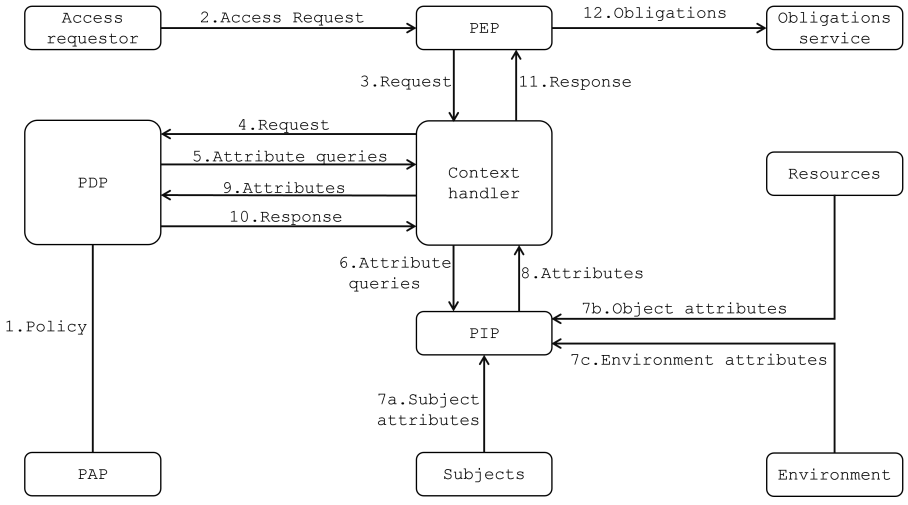


Figure 2.2: XACML's reference architecture

Policy Administration Point (PAP) support the decision function, where the latter contains the access control policies and allows administrators to create and manage policies. If the PEP does not provide enough information to the PDP to reach a decision (i.e., there exist attributes that are referenced in the applicable access control policies but not provided in the access request), the PDP can retrieve the missing information from the PIP, which stores attribute values. PIPs could be databases containing product or document information, user directories, LDAP, or an Active Directory.

For example, suppose Alice sends a request to access ObjectA. The PEP intercepts Alice's request and forwards it to the Context Handler, which converts the request into a request in the XACML format and sends the converted request, i.e., {(`Subject-id = Alice`), (`Action-id = Read`), (`Object-id = ObjectA`)}, to the PDP. The PDP loads the applicable access control policy (policies), based on the target of policies and the information provided in the request, from the PAP. Assume that the applicable policy is as follows:

(`Job-title = Programmer`) AND (`Action-id = Read`) AND (`Object-id = ObjectA`) AND (`Subject.Clearance-level` > `ObjectA.Classification`) → (`Effect = Permit`).

There are attributes referenced in the applicable policy that are not provided in the request. Hence, the PDP sends a request to the PIP to obtain all missing attributes, which are: 1) the job title of Alice, i.e., `Job-title = Programmer`, 2) the clearance level of Alice, i.e., `Subject.Clearance-level`, and 3) the classification of ObjectA, i.e., `ObjectA.Classification`. The PIP response is that Alice is a programmer with the clearance level *secret* and ObjectA's classification is *confidential*, i.e., {(`Job-title = Programmer`), (`Subject.Clearance-level = 2`), (`ObjectA.Classification` = 1)}. Based on this information, the PDP reaches a decision Permit and sends it to the PEP to be enforced.

A policy may include obligation expressions to enforce extra constraints that cannot be managed by normal policies (e.g., writing logs, sending emails, or showing warnings). Obligations can be used in several scenarios such as healthcare environments, governmental settings, or public-sector services. For example, as shown in Fig. 2.3, an obligation can be sending an email to the administrator of the system (administrator@example.com) for every unsuccessful access attempt. The PDP asks the PEP (by including the obligation into the response) to enforce the obligation (in addition to the access decision) if the decision matches the value of the "FulfillOn" attribute (of the obligation).

One of the key aspects of the XACML architecture is the fact that it is a loosely coupled architecture as the management function (provided by the PAP and PIP), the decision function (provided by the PDP), and the enforcement function (provided by the PEP) are cleanly decoupled. For a single point of management, several PDPs may exist and in turn, a single PDP may serve several PEPs. Besides, the XACML architecture can be used on different applications and technologies as the PEP and the Context Handler will adapt the requests

```
<Obligation FulfillOn="Deny " ObligationId="send-email">
<AttributeAssignment DataType="http://www.w3.org/2001/XML
Schema#string" AttributeId="email">administrator@example.com
</AttributeAssignment>
</Obligation>
```

Figure 2.3: An XACML obligation

and responses to the specific target application (and specific technology) and still talk back to the same PDP.

2.1.2 Semantic Technologies

Semantic technologies are a collection of methods, techniques, and tools that enhance data interoperability and the power of data by bringing into account the meaning rather than the structure of the data.

The basic format for representing semantic data is Resource Description Framework (RDF)[1] [88], which represents data (or knowledge) in a "subject, predicate, object" pattern, e.g., "Programmer isA Job-title". A set of such triples can be represented as a directed graph, i.e., an RDF graph. The set of names (subjects, predicates, and objects) in a graph is called the vocabulary of the graph. In other words, concepts in a certain domain and relationships between them can be described and represented by means of vocabularies. Vocabularies are useful not only for organizing knowledge, but also for resolving ambiguities when integrating different datasets. RDF Schema (RDFS)[2] [21] was proposed as a language for defining simple vocabularies, based on the concepts of class and property, allowing to perform simple inferences about these. More complex vocabularies are called ontologies, and are created using languages such as Web Ontology Language (OWL), which provides richer semantics than RDFS.

Despite the widespread use of RDFS and OWL, there are useful semantic relationships that cannot be expressed using these, e.g., it is difficult to specify that a wholesale customer is a person who purchases large lots of items. Such more complex relationships can be handled using Semantic Web Rule Language (SWRL) [58]. SWRL is a popular and standard rule markup language that combines Horn logic and OWL ontologies, making it possible to specify complex inference rules in addition to the basic ones reflecting inheritance. Inference rules make it also possible to infer new knowledge (i.e., inferring implicit relationships from explicit ones). For example, suppose a dataset has two triples `(Alice isA Person)` and `(Alice hasAge 19)`. An ontology may state that "every person of age above 18 is an adult", which is written in SWRL as the rule $\forall$`x,y.isA(?x, Person)` $\wedge$ `hasAge(?x, ?y)` $\wedge$ `swrlb:greaterThanOrEqual(?y, 18)` $\rightarrow$ `isA(?x, Adult)`. Hence, a new

[1] https://www.w3.org/TR/rdf11-mt/
[2] https://www.w3.org/TR/rdf-schema/

relationship `(Alice isA Adult)` can be added to the dataset as a result of the inference process, done by a reasoner (also known as the rule engine, reasoning engine, or semantic reasoner) based on a set of facts and axioms.

In order to retrieve information from ontologies, a query language is required. SPARQL [114] was developed as a query language just like SQL for relational databases or XQuery for XML texts. For example, assume a triple `(Alice isA Doctor)` exists in a dataset. A SPARQL request may be issued as `(Alice isA ?medicalstaff)`, where `?medicalstaff` is a variable. The query engine finds `Doctor` as a possible value for `?medicalstaff` and returns it as a possible answer.

There exist multiple tools for creating and managing ontologies, and performing the reasoning process such as: *Jena* [90], *Protégé* [47], *Jess* [44], *Racer* [51], and *Pellet* [120], where all except *Jess* are free and open source. *Jess* is not open source, but it is free for educational purposes (a license is required for commercial purposes). Further details about these tools are provided in Section II.2.2.

2.1.3 Attribute-Based Encryption

In conventional public-key cryptography, data are encrypted for a particular receiver using the receiver's public key. Hence, if the same data should be encrypted for several receivers, all the public keys of the receivers are needed. This is even more problematic for data that need to be stored encrypted for sharing with future, yet unknown, users. In response to this, Goyal et al. [49] proposed the first ABE scheme by which the encryptor can encrypt a message under a set of public attributes (instead of just an identity as in Identity-Based Encryption (IBE) schemes [18]). Therefore, data can be encrypted for a group of recipients holding the same public attributes.

More concretely, ABE is a kind of public-key cryptography in which the private key of a user and the ciphertext are dependent upon attributes. There exist several variations of ABE, e.g., CP-ABE and Key-Policy Attribute-Based Encryption (KP-ABE). In a KP-ABE scheme, ciphertexts are associated with a set of attributes and private keys are generated based on access structures. Hence, a ciphertext can be decrypted if the access structure of a private key satisfies the attributes required by a ciphertext. Contrary, in a CP-ABE scheme, private keys are associated with a set of attributes and the ciphertexts are encrypted based on access structures. Hence, a user can decrypt a ciphertext if her private key (her attributes) satisfies the access structure of the ciphertext. For example, if a doctor has the following public attributes $S = \{doctor, hospitalA\}$, then she can decrypt ciphertexts encrypted under the following AS: $\mathcal{T} = ((doctor \vee \text{caregiver}_{\text{id}} = 31415) \vee (researcher \wedge Norway))$. On the contrary, a researcher who works in Sweden cannot access the data (decrypt the ciphertext) because the attribute set S of the researcher does not satisfy $\mathcal{T}$.

When a user joins the system, she claims to have a set of public attributes and a Trusted Authority (TA) is in charge to validate them. If deemed appropriate, the TA provides her with a private key associated with the set of attributes she

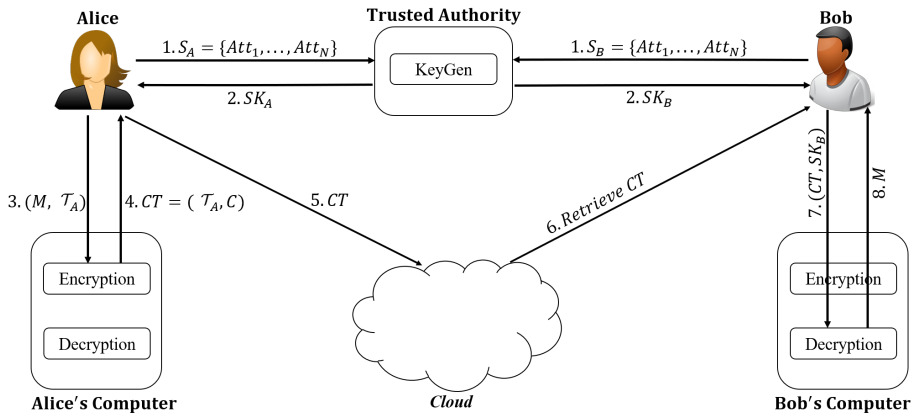


Figure 2.4: Ciphertext-Policy Attribute-Based Encryption

holds. This authentication process is usually out of the scope of ABE schemes since it is always assumed that the TA has the knowledge—or the corresponding mechanisms—to prove that users really have the attributes they claim to have. Hence, it is assumed that users cannot cheat TA and they are provided with the public attributes they actually have.

In this dissertation, we are more interested in CP-ABE because it is considered as the cryptographic counterpart of ABAC. Hence, we briefly describe the main algorithms that a CP-ABE [14] is made up of: Setup, KeyGen, Encryption, and Decryption. While the first two algorithms are run by the TA, the other two (i.e., Encryption and Decryption) are executed by the users of the system. In more detail:

Setup(1^λ) This algorithm takes a security parameter as input and generates a master secret key, MK, and a set of public parameters, PP.

KeyGen(MK, S, PP) The key generation algorithm produces a private key, SK, for a provided set of attributes, $S = \{Att_1, ..., Att_N\}$, using the master secret key and the public parameters.

Encryption(M, $\mathcal{T}$, PP) encrypts a message M based on the provided access structure, $\mathcal{T}$, and returns a ciphertext $CT = (\mathcal{T}, C)$, where C is the encrypted version of M.

Decryption(CT, SK, PP) decrypts a ciphertext using a provided private key, SK, which is related to a set of attributes satisfying the access structure included in CT, and public parameters.

Figure 2.4 shows an example where two users, Alice and Bob, join the system. First, they provide their public attributes (S_A and S_B) to the TA (1) and they receive the private keys (SK_A and SK_B) associated to their attributes (2). After

that, Alice runs the Encryption algorithm producing CT (3 and 4) and sends the ciphertext to the cloud where Bob can get it (5). When Bob retrieves CT from the cloud (6), he runs the Decryption algorithm (7) and finally, he obtains the plaintext (8). Figure 2.4 depicts all the steps in a CP-ABE scheme; however, it does not mean that all the steps are required for all kinds of operations. For instance, if Alice wants to encrypt a message, she only needs to run the `Encryption` algorithm and provide the message, the desired access structure, and the public parameters (i.e., for encryption, the data owner does not need to get a private key for her attributes).

2.1.4 Intel Software Guard Extensions

Intel Software Guard Extensions (SGX) [91] is a set of extensions for secure computation available on Intel's new generation of CPUs (6^{th} generation and later). The goal of SGX is to protect the confidentiality and integrity of the execution code against unauthorized accesses by privileged software such as the operating system, BIOS, or Hypervisor. In fact, it provides a kind of a Trusted Execution Environment (TEE) by means of transparent encryption and enforcing strict hardware access control. Such a trusted (isolated and protected) execution environment is called *enclave* in SGX terminology, where no one can see the computations and secrets inside it. Therefore, the SGX handles secrets and executes software in a trustworthy environment on an untrusted system (the operating system and memory).

The main functionalities of an enclave are *isolation*, *sealing*, and *attestation*. An enclave provides an isolated environment such that the data and code inside an enclave cannot be accessed by other processes. It has a hardware-resident key for sealing (encrypting and authenticating) data passed to the host environment. The code, data, and metadata of an enclave, and the results of computations performed inside the enclave, can be signed and attested by means of local as well as remote attestation. As represented in Figure 2.5, an SGX-enabled application includes two parts: trusted and non-trusted, where the trusted part resides encrypted in the memory (i.e., in an enclave).

Isolation: An enclave resides in the Enclave Page Cache (EPC), which is a part of the memory guarded by the hardware as shown in Figure 2.5. The maximum size of EPC is 120 MB out of which only 90 MB can be used by an application and 4 KB is used for page chunks. The data and code of an enclave will be copied into the EPC (pages inside it) when loading an enclave program and the EPC pages can be accessed only when the processor is running in *enclave mode*. The hash measurement of the page contents, which is called MRENCLAVE, also will be stored along with the data and code inside the EPC. Each enclave will be assigned an identity by which the hardware can control access to the contents of the enclave. The hardware can ensure that pages of an enclave can be accessed only by executable code pages of the same enclave (with the same enclave identity).

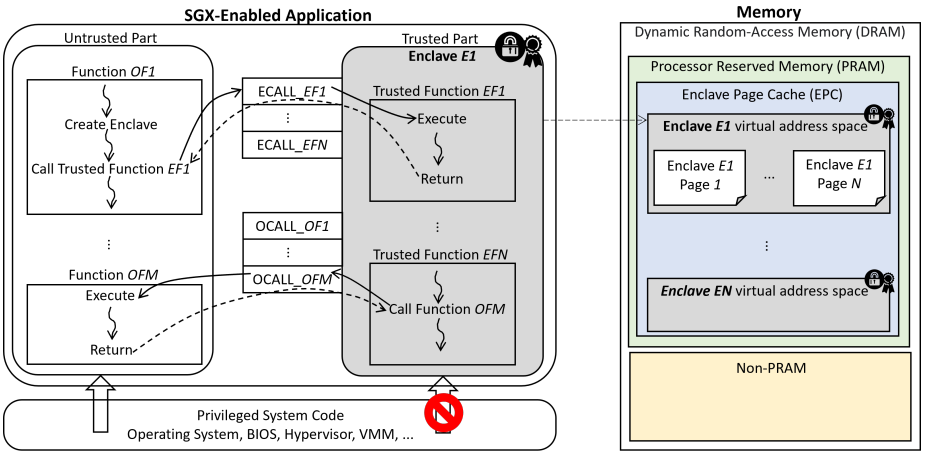


Figure 2.5: Structure of an SGX-enabled application and the memory layout

Sealing: There is a secret key inside every Intel SGX CPU called the *root seal key* that can be used by enclaves to generate another key for sealing (encrypting and authenticating) their data and code. An enclave generates the sealing key, which is called the *seal key*, using EGETKEY instructions, then it encrypts/authenticates both its data and code with the seal key and stores them in the untrusted memory. The seal key is unique for each enclave, which means that other enclaves (even the ones on the same platform) cannot generate/get the same *seal key*. Therefore, the sealed data can only be accessed/retrieved by the enclave that owns them.

Attestation: The Intel SGX provides local and remote attestations by which different enclaves (on the same or different platforms) can attest each other. By means of the attestation, an enclave can be assured that it is dealing with the right code and data. Secret materials (e.g., secret keys) can then be provided to an enclave (remotely) in a secure manner by means of the attestation process.

The local attestation can be used between two enclaves on the same platform (the same CPU on the same machine). As mentioned before, every Intel SGX processor has a unique root seal key that is the same for all enclaves on the same platform. Different enclaves on the same platform can use that key for the authentication process. An enclave can generate a Message Authentication Code (MAC)—which is called a *report*—for the hash value of its contents (MRENCLAVE) and its metadata with a *Report Key* which is a common key that all SGXs can generate.

The remote attestation provides MAC reports to be verified by enclaves on other platforms (e.g., remote machines). As represented in Figure 2.6, when an SGX-enabled application on a user's machine wants to receive some services (for example, secret keys) from a service provider (a remote server) or when a remote server wants to verify that an untampered enclave is running on a legitimate

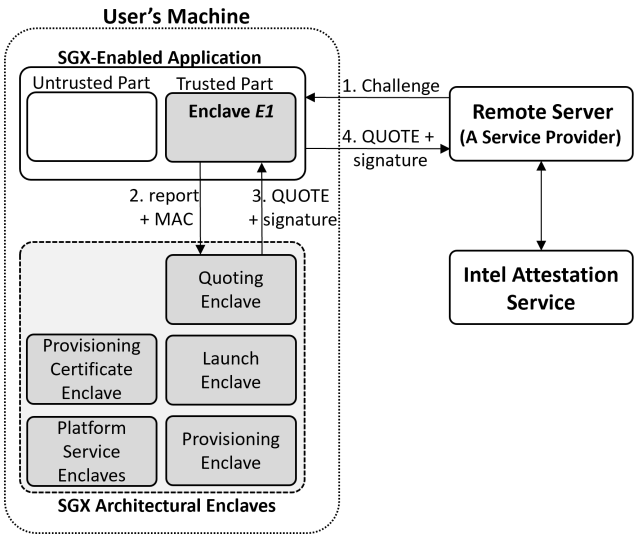


Figure 2.6: SGX Remote Attestation. Architectural Enclaves are Intel provided enclaves.

CPU (or to provide, for instance, secret keys to an SGX-enabled application), a challenge message including a nonce will be sent to the SGX-enabled application through a established secure communication (message 1 in Figure 2.6). The SGX-enabled application's enclave (in this case, Enclave *E1*) generates a local report, which includes the challenge's response and a temporary key that can be used for key exchange and securing subsequent communication, and a MAC for the report. Then, the SGX-enabled application's enclave provides the generated report and MAC to a special enclave, called Quoting Enclave, for verification and signing (2). The Quoting Enclave, which is one of the Intel provided SGX architectural enclaves and exists on the CPU, verifies the provided report, creates a QUOTE structure for the verified report, signs the QUOTE with a private key for an anonymous group signature scheme called Intel Enhanced Privacy ID (EPID), and then sends the QUOTE and signature to the SGX-enabled application's enclave (3). The SGX-enabled application forwards the received QUOTE and signature to the remote service provider, i.e., the challenger (4). The remote service provider can verify the received information and the response to the initial challenge using the relevant EPID public key certificate and revocation information obtained through the Intel Attestation Service. The remote service provider can use the temporary key provided by the SGX-enabled application for key exchange.

Since the SGX encrypts both the memory and the plaintext of the secrets available inside the CPU, it is assumed that no one can open the CPU content. SGX has been used in many different contexts to achieve a higher level of security [20, 42, 93, 116, 126].

Interactions: The code executed inside an SGX enclave is considered as the trusted part of an application and can interact with the untrusted part of the application through Enclave Calls (ECALLs) and Outside Calls (OCALLs) as illustrated in Figure 2.5. The trusted functions can be invoked using ECALLs. Functions inside an enclave can call untrusted functions (those that are outside SGX enclaves) through OCALLs. In other words, we enter an enclave using an ECALL and inside the enclave we can use an OCALL to do, for example, I/O operations.

We refer those who are concerned about the security of SGX to Section III.A, which surveys the existing possible attacks against SGX and corresponding countermeasures.

2.2 Literature review

2.2.1 Semantic Attribute-Based Access Control

This subsection provides a summary of the state of the art on Semantic Attribute-Based Access Control based on the literature review done in Paper I.

SABAC has recently emerged as an extension of ABAC. The goal of SABAC is to augment ABAC with semantic technologies in order to take into account the semantic relationships between the involved entities when making a decision. Semantic technologies help the access control engine to infer implicit knowledge (e.g., semantic synonyms of attributes) from explicit knowledge, i.e., the attributes provided in the requests and predefined policies. Based on the literature review conducted in Paper I, the existing SABAC schemes can be categorized as *Extensions of the XACML Standard*, *New Policy Languages*, and *Hybrid Models*.

The first work that augmented ABAC with semantic technologies was by Damiani et al. [34], which extended the XACML policy language. Particularly, they extended the XACML policy language to include ontology-based metadata associated with subjects and objects (through the use of RDF assertions) in the access control policies. Another line of SABAC frameworks was started by Priebe et al. [105], who, instead, extended the XACML architecture. As shown in Fig. 2.7, they added two new components, i.e., an inference engine and an ontology administration point (OAP), to the architecture of XACML. The inference engine helps to find other attributes that are semantically relevant to those provided in an access request with the help of a domain ontology provided by OAP.

Muppavarapu and Chung [97] and Shen [118] also extended the XACML architecture in the same way as Priebe et al. [105]. However, Shen's scheme, as opposed to the schemes of Priebe et al. and Muppavarapu and Chung, performs the semantic reasoning at the system initialization and not at the time of making a decision (as the semantic component is connected to the PIP and not context handler), which in turn makes the decision-making process faster.

Durbeck et al. [38] also proposed an SABAC scheme which is the same as that of Shen in the sense that the semantic component is connected to the PIP. However, in Durbeck et al.'s scheme, an access requester has to check the

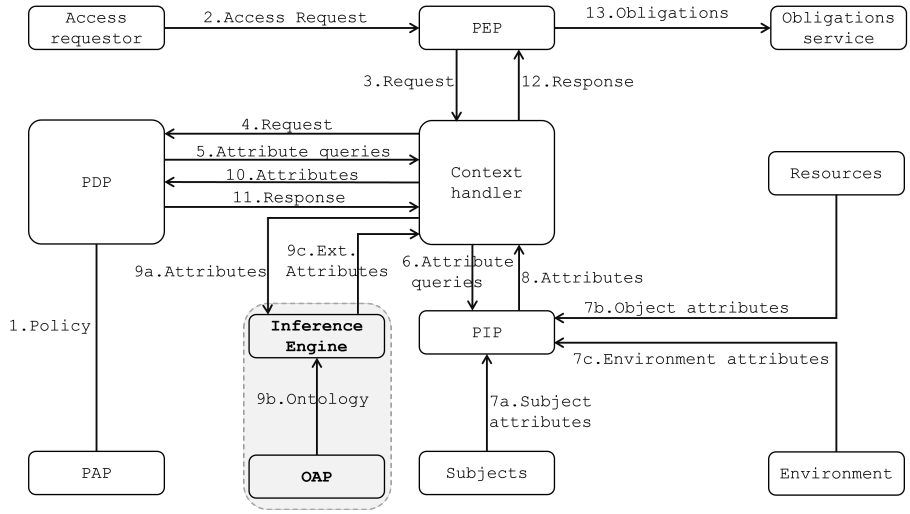


Figure 2.7: An example of extending the XACML architecture

applicable access control policies before issuing an access request (to determine all the required attributes), which might have a negative effect on the performance of the system as extra work needs to be done for every single request. Furthermore, retrieving a policy, which may contain information about the subject, object, action, or environment, by the access requester may cause leakage of information.

Dersingh et al. [36] extended both XACML policy language and architecture.

Shen and Cheng [119] updated the scheme proposed in [118] by using SWRL as the policy specification language (to enhance the expressiveness of the policy specification) and replacing the ordinary PDP with a Jess inference engine. Calvillo et al. [27] also proposed a similar SABAC scheme. However, in Calvillo et al.'s scheme, the PIP is decomposed into three different PIPs for the subject, object, and environment, and are connected to three related knowledge bases.

Drozdowicz et al. [37] extended the XACML architecture by replacing the PIP with another component called SemanticPIP, which is an integration of the PIP and the HL7 security and privacy ontology [16].

Ciuciu et al. [32], Zhao and Wang [138], Zhang et al. [137], Hsu [59], Calvillo-Arbizu et al. [25], Liu and Wang [86], and Hilia et al. [54] also extended the XACML standard towards SABAC. These schemes are described with details in Section I.4.1.

Another line of research on SABAC proposes new policy specification languages that incorporate semantic technologies. Jin and Fang-Chun [70] proposed an ontology-based ABAC by means of description logics to decrease the complexity of the policy specification and improve the interoperability. They used the restricted $ALL(D)$ [12] description logic language as the basis for expressing access control policies and attributes. Amini and Jalili [7] proposed an SABAC scheme based on the $MA(DL)^2$ logic [6], which is a combination

of deontic and description logics and supports the specification and inference of access control policies. Trivellato et al. [124] proposed an ontology-based context-aware policy specification language called POLIPO inspired from Datalog with constraints [81]. Calvillo-Arbizu [26], Iqbal and Noll [66], Lu et al. [87], and Verginadis et al. [127] also proposed SABAC schemes which fall into this category and are further detailed in Section I.4.2.

There exist a number of SABAC schemes [5, 64] that are based on the combination of two different access control models, i.e., an ABAC and a Semantic-Based Access Control (SBAC), like the one represented in Figure 2.8. SBAC is an access control model for restricting access based on semantic relationships between involved entities as defined in an ontology and SWRL rules using semantic reasoners. The idea behind hybrid SABAC schemes is to take the advantages of both ABAC and SBAC models. Section I.4.3 provides more details about the hybrid SABAC schemes.

2.2.2 Attribute-Based Encryption

We augment the ABE schemes with semantic technologies and enforceable obligations, respectively, in Paper II and Paper III. Hence, this subsection summarizes the related work on Attribute-Based Encryption.

In 2005, Sahai and Waters [112] introduced the concept of Attribute-Based Encryption. They proposed a new type of IBE through which one can encrypt a piece of data for a group of recipients enabling multicast encryption [121]. After a year, Goyal et al. [49] proposed a KP-ABE in which ciphertexts are associated with a set of attributes and private keys are generated based on access structures. Hence, a ciphertext can be decrypted if the access structure of a private key satisfies the attributes required by a ciphertext. Bethencourt et al. [14] proposed the first CP-ABE scheme in which private keys are associated with a set of

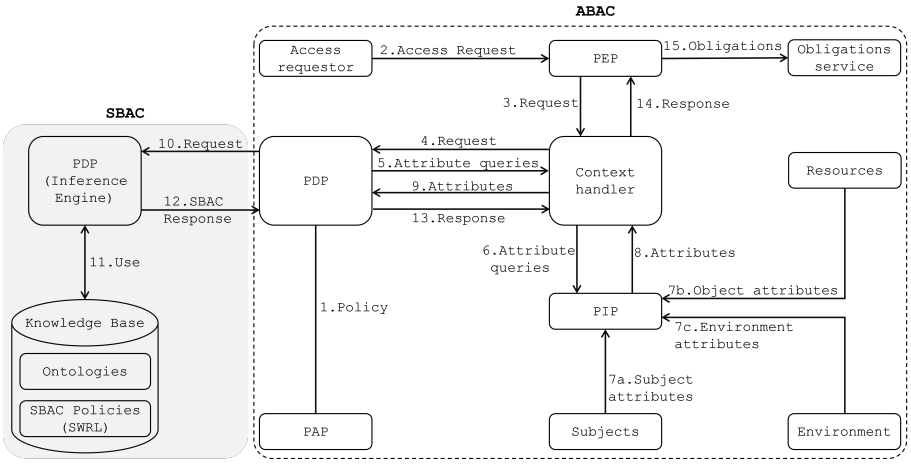


Figure 2.8: The hybrid SABAC scheme proposed in [5]

attributes and the ciphertexts are produced based on access structures. Till now, a considerable number of KP-ABE [78, 101] and CP-ABE [50, 130] schemes have been proposed.

A combination of KP-ABE and CP-ABE, to have both types of ABE at the same time, was proposed by Attrapadung and Imai [11]. Müller et al. [96] proposed a CP-ABE scheme for distributed environments, where several authorities manage attributes and generate private keys. Yu et al. [133] employed proxy re-encryption and lazy re-encryption techniques to improve the efficiency of KP-ABE.

ABE schemes rely on a trusted authority in generating private keys for attributes. The trusted authority, which has full power on private keys, may behave maliciously. Thus, ABE schemes suffer from the *key escrow* problem. There are a huge number of research studies in the literature [61, 135] addressing the key escrow problem in ABE schemes. For instance, in [29], the key escrow problem was addressed by incorporating several TAs cooperating to generate private keys. However, such a multi-authorities ABE scheme may be susceptible to the collusion of TAs. Hu et al. [61] proposed a multi-authorities CP-ABE scheme addressing the key escrow problem and collusion attacks (i.e., the collusion of the authorities). Zhang et al. [135] proposed a multi-authorities KP-ABE scheme addressing collusion attacks and user privacy.

Tang and Ji [122] added a verification property to both single-authority and multi-authorities versions of KP-ABE, by which users can verify the correctness of the received private keys as errors may occur during creation or transmission of the keys. Wang et al. [128, 129] combined a hierarchical IBE scheme and a CP-ABE scheme to address the revocation problem in ABE schemes (revoking access rights from users who are no longer legitimate).

2.2.3 SGX-Based Schemes

The OB-ABE scheme proposed in Paper III is based on the trusted hardware enclaves provided by Intel SGX. Hence, in this subsection, we briefly review other security schemes that are also based on Intel SGX.

Since SGX provides a trusted execution environment, researchers have tried to use it in different contexts to achieve a higher level of security [20, 33, 42, 82, 93, 115, 116, 126].

The SGX is also used to provide an accountable and trustworthy function as a service (FaaS) [4]. The trusted execution environment provided by the SGX ensures both the integrity of the outsourced computations, and the correctness of the measured usage information (for correct billing). It also minimizes the leakage of the information to the service provider. In [106], Qiang et al. enhance the security (confidentiality and integrity) of serverless computing services with the help of Intel SGX. The remote attestation capability of the SGX is used to provide integrity (verifying the integrity of the function modules) and the SGX enclave is used to provide confidentiality (protecting the core modules of the API gateway). Deployment of cloud micro services suitable for dealing with sensitive

data is very important and difficult. In [19], Intel SGX is integrated into micro services to address the security and privacy concerns related to sensitive data.

PRESAGE [30] is a framework for genomic data outsourcing (i.e., genetic testing) that uses Intel SGX to provide acceptable levels of security and privacy. The Intel SGX is used not only for sealing the genomic data (to be stored on public clouds) but also for providing secure genetic query matching by means of the remote attestation. In [76], a privacy preserving approach is proposed for DNA processing. The Intel SGX plays a key role in the proposed approach as it makes the read alignment (finding the position of DNA sequences) secure. It is demonstrated that the proposal is much faster than other related approaches thanks to Intel SGX. Shaon et al. [117] also used SGX to propose a secure framework for genetic data analytics in untrusted cloud setups.

Tramer and Boneh [123] proposed a framework for efficient and secure execution of deep neural networks using the Intel SGX, where DNN computations are divided between untrusted and trusted entities. The DNN inference is outsourced, performing the computations on a faster (untrusted) co-located processor, which are then verified in a trusted execution environment. Cheng et al. [31] improved the security of blockchains by means of hardware-based trusted execution environments, with a solution based on the Intel SGX.

Pires et al. [104] claimed that querying search engines endangers the privacy of web users as well as existing privacy-preserving solutions are not secure against user re-identification attacks, and used Intel SGX to preserve the privacy of search engines' users. In [40], a secure and privacy-preserving architecture (called Fidelius) for web browsers is proposed using Intel SGX. Fidelius protects user secrets even if the underlying browser and operating system are fully under control of an adversary. PubSub-SGX [9] is a content-based publish/subscribe system in which SGX is incorporated to guarantee the integrity and confidentiality of data and preserve the privacy of users.

In [53], SGX is used to process data streams in the Internet of Things environments securely or to develop secure in-memory database engines managing confidential data in rack-scale environments [113].

2.2.4 Obligation Specification

One of the contributions of Paper III is proposing a formal language for the specification of obligations. Hence, this subsection provides the related work on obligation specification languages.

There are several access control policy specification languages that support also the specification of obligations, e.g., XACML [102], Rei [71], and Ponder [35]. However, they only offer syntactic elements for obligation specification, and do not cover different types of obligations, thus none provide a concrete model for specifying obligations. For instance, it is not possible to specify conditional obligations, pre-obligations, or repetitive ones using Rei and Ponder.

XACML is an OASIS standard providing an XML-based policy specification language, with no formal foundation, and a reference architecture. The policy language allows specifying obligations as part of policies. An obligation has two

elements `ObligationID` and `FulfillOn`, where the second determines when the obligation is mandatory, with value being either `Permit` or `Deny`. XACML does not support different types of obligations, and obligations are black boxes. XACML has introduced *advices* as optional obligations, which can be ignored if it is not possible to fulfil them.

Rei [71] is a general-purpose policy language based on deontic logic [46], which supports security, management, and conversation policies. Obligations are not the first-class entities in Rei. However, Rei has mechanisms to detect and resolve conflicts between prohibition policies and non-complex obligations.

Ponder [35] is an object-oriented language for the specification of security and management (of network and distributed systems) policies. In Ponder, obligations are considered as condition-action rules that can be triggered by events. It is claimed that complex obligations can also be specified by means of concurrency operators of Ponder. Ponder addresses the conflicts between obligations (actions of obligations) and policies by halting the target application, which may not be an acceptable solution.

SPL [111] is a policy specification language that defines obligations as events that should be performed in the future and after performing the current event. If obligations cannot be performed (e.g., because of conflicts between obligations and policies), then everything will be rolled back to the state before performing the event triggering the obligation. The expressiveness of the SPL is limited as it does not support different types of obligations.

OSL [56] is an obligation specification language based on linear temporal logic (LTL) for distributed usage control. Obligations in OSL are either "usage restrictions", prohibiting given usages under certain conditions, or "action requirements", mandating the execution of certain actions conditionally (based on time, purpose, event-defined, environment, and cardinality) or unconditionally. The syntax and semantics of OSL are formalized in the formal language Z. OSL obligations can be converted to XrML and ODRL [65], which are digital right management (DRM) specification languages, and then be enforced by existing DRM mechanisms.

Irwin et al. [67] presented a formal metamodel to model and analyze a system from the obligations point of view. They provided formal definitions of secure states (the states without unfulfilled obligations) and accountable states for obligation management based on their metamodel. Accountable states refer to the states that identify those who did not fulfill obligations. Irwin et al.'s approach is rather restricted and does not cover different types of obligations such as conditional, reoccurring, and pre- obligations. Irwin et al.'s approach is more about the analysis of the obligation enforcement rather than the specification of different types of obligations.

Li et al. [80] extended the XACML standard by designing a language for the specification of obligations. They modeled obligations as state machines communicating with the policy enforcement point (a component of the XACML architecture that is in charge of enforcing access decisions and obligations) through events. Based on Li et al.'s proposal, an obligation can mandate the

occurrence of a series of events (and not only one event), which may be dependent on each other.

PoCo [41] is an enforcement system and a language based on the simply-typed lambda-calculus for the composition of policies containing obligations. PoCo makes it possible to compose complex atomic obligations. It prevents insecure situations, which may happen due to incomplete execution of obligations or execution of obligations violating other policies, by allowing policies to check the obligations associated with other policies before their execution.

Ni et al. [98] proposed an obligation model for Privacy-aware Role Based Access Control [99]. The obligation model addresses the undesired issues that may happen due to conflicts between obligations and policies, e.g., an obligation mandates performing an action that is prohibited by another policy. They also provided solutions to determine the relationships between obligations, for instance, an obligation may cover another one. Ni et al. claim that their proposal needs to be improved by addressing unfulfilled obligations, providing a solution for the accountability problem, and providing a mechanism for the optimization of the execution order of obligations.

Hilty et al. [55] provided a formal model for the specification of policies and obligations based on distributed temporal logics. In their model, obligations can be expressed as formulas without past time temporal operators. Hilty et al.'s model supports post-obligations and addresses the observability problem, i.e., helping the reference monitor to be able to check if a post-obligation is fulfilled.

2.2.5 Analysis of access control policies

Until now, a number of researches have been conducted to analyze and verify access control policies and systems using formal methods.

Bryans [22] used the Communicating Sequential Processes (CSP) [57] and the FDR model-checking tool for analyzing the equivalence of RBAC access control policies. Bryans did not consider the condition part of rules, obligations, and advice when specifying XACML policies in CSP.

Kolovski et al. [75] formalized XACML policies using the $\mathcal{SHOIN}$ description logic (DL). They used Pellet DL reasoner to analyze the formalized policies and find equivalent, redundant, and incompatible policies. However, Kolovski et al. also did not take into account rule conditions, obligations, and advice.

Ahn et al. [3] proposed an approach for translating XACML-based RBAC policies into Answer Set Programming (ASP) [83, 89] and then analyzing them (verifying if they control access as intended) using ASP solvers. However, Ahn et al.'s approach does not handle obligations, advice, and complex conditions and attribute functions. Besides, the proposed approach is only a translator from XACML to ASP and there is no way to specify access control policies in ASP.

Fisler et al. [43] proposed a method for analyzing XACML-based RBAC policies by representing the policies using Multi-Terminal Binary Decision Diagrams (MTBDD) [45]. However, the proposed method does not verify all the desired properties of policies and does not take into account all elements of XACML policies.

Rao et al. [108] proposed an algebra, called Fine-grained Integration Algebra (FIA), for integration of XACML policies. FIA also uses MTBDDs for representing XACML policies. Policies can be integrated by mapping operations on the policies onto operations on the MTBDDs, which represent policies. After mapping operations, the resulted MRBDD can be traversed to generate an XACML policy that is the result of the integration of two or more policies.

Hu et al. [60] proposed a policy-based segmentation method to detect and resolve policy anomalies (conflicts and redundancies). Hu et al.'s method first represents (parses) policies using the Binary Decision Diagram (BDD) [23], then it transfers rules into Boolean expressions. Next, it replaces Boolean expressions with Boolean variables. After that, it identifies anomalies using two proprietary algorithms.

Morisset et al. [94] also employed BDDs to address the problem of missing information in ABAC. They proposed a framework for efficient extended evaluation of XACML policies, which checks all the possible outcomes of the evaluation of a given request by considering all possible values for the hidden attributes (i.e., by extending the initial request).

Lin et al. [84] proposed a policy analyzer through the combination of MTBDDs (to represent/parse policies) and a SAT solver (to check if two representations are similar). The main goal was to find the similarities between XACML policies.

Turkmen et al. [125] proposed a framework based on satisfiability modulo theories (SMT) for the verification of XACML policies. The goal was to convert policies into SMT formulas and then verifying the desired properties using SMT solvers.

Another relevant work is the formalization of XACML in terms of multi-valued logics presented in [107]. Ramli et al. [107] provided an abstract syntax for XACML and formalized combining algorithms as operators on a partially ordered set of decisions.

Chapter 3

Overview of the Research Papers and Contributions

This chapter provides an overview of the research papers included in this dissertation along with their contributions. The second part of the dissertation provides the full text of the included papers.

3.1 Paper I: Semantic Attribute-Based Access Control: A review on current status and future perspectives

3.1.1 Summary

Attribute-based access control uses the attributes of the involved entities (i.e., subject, object, action, and environment) to provide access control. Despite various advantages offered by ABAC, it is not the best fit for distributed and heterogeneous environments where the attributes of an entity may not necessarily match (syntactically) those used in the access control policies. Therefore, Semantic Attribute-Based Access Control has emerged as an extension of ABAC that takes into consideration the semantics of attributes by combining ABAC with semantic technologies. SABAC not only facilitates interoperability, but also enhances the expressiveness of access control policies. Over the last decade, a number of research efforts have been conducted in developing semantic attribute-based access control schemes. However, there exists no survey paper on SABAC schemes giving an overview of the existing approaches. Hence, Paper I comprehensively reviews the conducted research efforts for developing SABAC. The main goal of Paper I is to provide a comprehensive summary of the conducted research studies that is useful for researchers who want to enter and make contributions to this field. Furthermore, the paper identifies open problems and possible research entry points by demonstrating the advantages and disadvantages of the previous studies.

3.1.2 Contributions

Main contributions of Paper I can be summarized as follows:

- To the best of our knowledge, this is the first survey paper on SABAC.
- Reviewing the conducted research efforts on SABAC systematically and providing a comprehensive summary of them.
- Representing different SABAC architectures in an unified manner based on the reference architecture of the XACML standard.

- Classifying the existing SABAC schemes based on different criteria.
- Providing an in-depth comparison and discussion about the existing SABAC schemes.
- Describing the properties of, what we call, *ideal SABAC*.
- Identifying open problems and research directions towards the ideal SABAC.

3.2 Paper II: Semantic Attribute-Based Encryption: A Framework for Combining ABE schemes with Semantic Technologies

3.2.1 Summary

Attribute-Based Encryption is a cryptographic solution to protect resources in a fine-grained manner based on a set of public attributes. This is similar to attribute-based access control in the sense that both rely on public attributes and access control policies to grant access to resources. In other words, ABE can be considered as as the cryptographic counterpart of ABAC. However, ABE schemes do not consider the semantics of attributes provided by users or required by access structures. Such semantics not only improve the functionality by making proper access decisions but also enable cross-domain interoperability by making users from one domain able to access and use resources of other

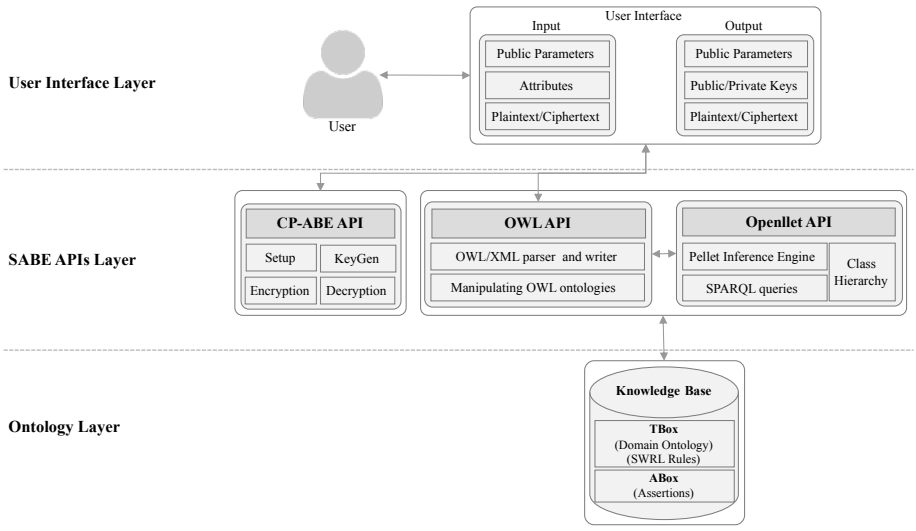


Figure 3.1: The architecture of the SABE framework.

domains. Therefore, ABE schemes cannot be considered as a cryptographic replacement for SABAC. In order to achieve a cryptographic counterpart of SABAC, Paper II proposes a Semantic ABE (SABE) framework by augmenting a classical CP-ABE scheme with semantic technologies using a generic procedure by which any CP-ABE scheme can be extended to a SABE. The proposed SABE framework is implemented in Java and the source code is publicly available. The experiment results confirm that the performance of the proposed framework is promising.

3.2.2 Contributions

Major contributions of Paper II are as follows:

- To the best of our knowledge, this is the first research effort towards Semantic Attribute-Based Encryption.
- Proposing an SABE framework, which is modular, generic, scalable, and extensible, by augmenting a classical CP-ABE scheme with semantic technologies using a generic procedure by which any CP-ABE scheme can be extended to an SABE. The architecture of the proposed framework is represented in Figure 3.1.
- Proposing two different approaches namely *Semantically-Enriched Key* and *Semantically-Enriched Access Structure*, which are represented, respectively, in Figure 3.2 and Figure 3.3, for the SABE framework.
- Formal verification of the security of the proposed schemes by reusing standard techniques done for the underlying CP-ABE scheme.
- Implementing the proposed SABE framework based on the two proposed approaches in Java and releasing the source code publicly.

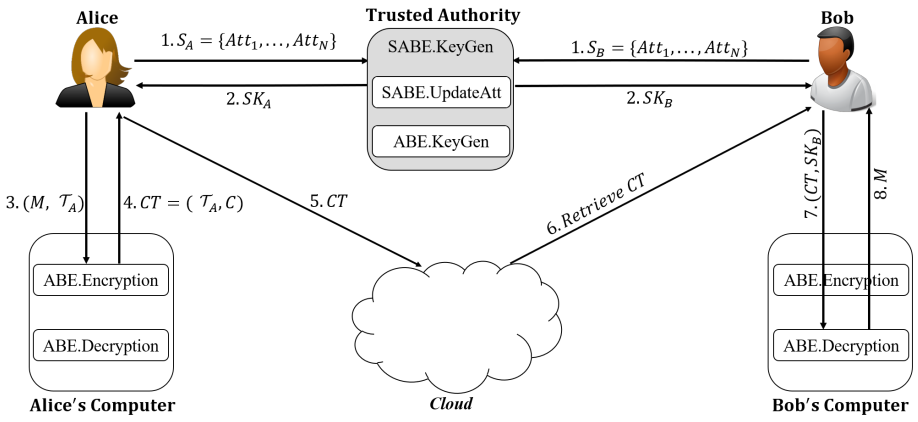


Figure 3.2: The proposed Semantically-Enriched Key approach towards SABE.

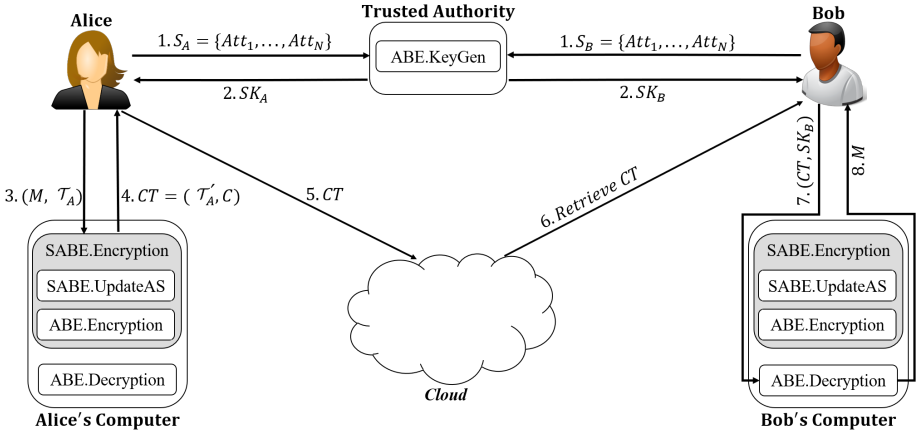


Figure 3.3: The proposed Semantically-Enriched Access Structure approach towards SABE.

- Making ABE schemes semantic-aware and enabling cross-domain interoperability with negligible performance overhead according to the results of performing different experiments as listed in Table 3.1.

Table 3.1: The comparison between the execution time (in milliseconds) of SABE (SEK and SEAS) and CP-ABE

Algorithm	Input Size	CP-ABE [14]	SABE	
			SEK	SEAS
Key Generation	-	279.97	303.03	279.97
Encryption	1 MB	94.12	94.12	140.49
	100 MB	28.76	228.76	275.13
	1 GB	1595.18	1595.18	1641.55
Decryption	1 MB	33.72	33.83	33.76
	100 MB	381.85	383.42	382.27
	1 GB	3224.38	3266.48	3241.57

3.3 Paper III: Attribute-Based Encryption with Enforceable Obligations

3.3.1 Summary

Attribute-Based Encryption schemes lack the notion of *obligations*, which is common in Attribute-Based Access Control systems (e.g., XACML and UCON), to define and enforce extra constraints that happen before approving or denying an access request. Obligations are greatly desired, e.g., in e-Health, because of the accountability and highly interactive style of work where many types of actions must be logged, various authorizations are needed from experts (i.e., confirmations, e.g., from the doctor on duty), or simply sending notifications to relevant parties (like to family/guardians) are required by law. Paper III proposes Attribute-Based Encryption with enforceable OBligations (OB-ABE), a system for extending any classical ABE with enforceable obligations. Paper III defines a system architecture having Intel SGX (to provide trusted hardware enclaves) as the core component used for enforcing obligations. It also provides a formal language for the specification of obligations that must be enforced by a trusted hardware enclave before releasing a plaintext. ProVerif (security protocol/system verifier) is used to formally verify the main property of OB-ABE, called "enforceable obligations", i.e., if a message is encrypted along with an obligation, then the decryption algorithm may return the message only after enforcing the attached obligation. OB-ABE has two more properties: (i) OB-ABE is a "conservative extension" of the underlying ABE scheme, preserving its security properties; (ii) OB-ABE is "backward compatible" in the following sense: any ciphertext produced by an ABE scheme can be decrypted by its extended OB-ABE version, and moreover a ciphertext produced by an OB-ABE scheme can be decrypted by its underlying ABE scheme provided that the ciphertext does not have obligations attached. A prototype of the proposed OB-ABE is also implemented in C using Intel SGX[1].

3.3.2 Contributions

The following lists the main contributions of Paper III:

- To the best of our knowledge, this is the first research effort augmenting Attribute-Based Encryption with obligations, which is an advanced feature of modern access control systems.

- Proposing Attribute-Based Encryption with enforceable OBligations (OB-ABE) as a general extension with obligations of any ABE scheme based on a real-world e-Health case study taken from the SCOTT project[2]. The architecture of the proposed OB-ABE scheme is shown in Figure 3.4.

[1] The source code is available at https://github.com/haamedarshad/OB-ABE
[2] https://scottproject.eu/

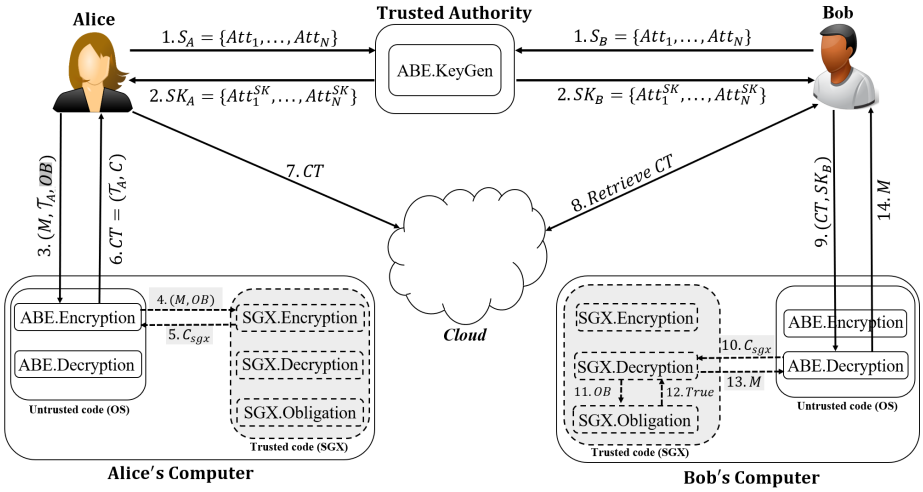


Figure 3.4: OB-ABE's architecture

- Proposing a formal language for the specification of complex obligations that must be enforced by a trusted hardware enclave before releasing a plaintext.
- Formal verification of three key properties of OB-ABE, e.g., enforceable obligations, backward compatibility, and conservative extension.
- Providing a prototype implementation of OB-ABE based on Intel SGX and making the source code publicly available.
- Bringing an important feature of access control systems in ABE schemes with negligible performance overhead (based on the results of performing different experiments as listed in Table 3.2).
- Surveying possible security attacks against Intel SGX and corresponding countermeasures.
- Providing a comprehensive list of security schemes that are based on Intel SGX.

3.4 Paper IV: Process Algebra Can Save Lives: Static Analysis of XACML Access Control Policies using mCRL2

3.4.1 Summary

The eXtensible Access Control Markup Language (XACML) is an OASIS standard for access control systems that is much used in health care due to its

Table 3.2: A comparison between the execution time of OB-ABE and CP-ABE.

Algorithm	Input size	Scheme		
		OB-ABE		CP-ABE
		With Obligations	Without Obligations	
Encryption	1 MB	22.88 ms	22.17 ms	22.09 ms
	100 MB	720.08 ms	691.59 ms	691.33 ms
	1 GB	7252.90 ms	6929.21 ms	6926.07 ms
Decryption	1 MB	9.56 ms	9.45 ms	9.40 ms
	100 MB	934.60 ms	928.85 ms	927.30 ms
	1 GB	9699.58 ms	9648.73 ms	9644.74 ms

fine-grained, attribute-based policy definitions, useful in dynamic environments such as the emergency ward. A notorious problem in XACML is detection of conflicts, which arise especially when combining policies, such as when health institutions merge. This paper proposes an approach to formally verify XACML policies using the process algebra mCRL2. Our formal translation of XACML policies into mCRL2, using our automated tool XACML2mCRL2, enables us to verify the above property, called *consistency*, as well as other policy properties such as *completeness* and *obligation enforcement*. Verifying policy properties statically allows us to resolve inconsistencies in advance, thus avoiding situations where an access request is denied in a critical situation (e.g., in an ambulance, when lives may be put in danger) just because of incomplete or inconsistent policies. The mCRL2 toolset is especially useful for modeling behaviors of interactive systems, where XACML would be only one part. Therefore, we verify an access control system together with an intended health care system that it is supposed to protect. For this, we exemplify how to verify safety and liveness properties of an assisted living and community care system.

3.4.2 Contributions

The major contributions of Paper IV are listed below:

- Presenting a methodology for formal verification of XACML access control policies, which is built on top of mCRL2 as shown in Figure 3.5.

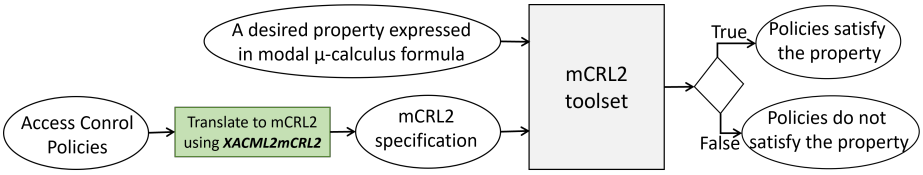


Figure 3.5: Analyzing access control policies using mCRL2

- Providing a tool, XACML2mCRL2, that automatically translates XACML policies into mCRL2 specifications.
- XACML2mCRL2 was awarded the FORTE 2022 Best Software Artefact.
- Formulating important properties of access control policies, e.g., completeness, consistency, and obligation enforcement, using the first-order modal μ-calculus.
- Presenting an approach for analyzing the XACML policies in a given context, i.e., together with the system that these policies are supposed to protect.
- Validating the proposed approach using a real-world use case in e-Health by verifying important properties such as liveness and safety.

Chapter 4

Conclusion

This chapter first presents a summary of our research contributions, then returns to the research questions defined in Chapter 1 and discusses how they have been answered by the research papers. Finally, this chapter presents the limitations of the dissertation and potential research directions.

4.1 Summary of Contributions

First, we performed a systematic literature review on Semantic Attribute-Based Access Control to identify strategies for developing SABAC as well as trends and gaps in the field. Based on the literature review, we identified several open problems towards an ideal SABAC and we tried to address a few of them.

One of the open problems was developing cryptographic solutions for SABAC, as access control systems rely on a trusted reference monitor that restricts the scalability and can be easily bypassed. Since Attribute-Based Encryption schemes are considered as a cryptographic solution for Attribute-Based Access Control, we decided to extend them in a way to be suitable for SABAC as well. Hence, we proposed a modular, generic framework combining ABE schemes with semantic technologies. The SABAC strategies identified by the systematic literature review on SABAC guided us on augmenting ABE schemes with semantic technologies. We proposed two different schemes for extending ABE schemes towards Semantic Attribute-Based Encryption. We not only formally verified the security of the proposed schemes but also implemented both schemes to evaluate their performance, which were promising.

Next, we added an important feature of ABAC called *obligations* to the ABE schemes. To do this, we proposed a generic framework based on Intel SGX by which any ABE schemes can be extended to an Attribute-Based Encryption with enforceable OBligations scheme. We also proposed a formal language for the specification of obligations. We formally verified the main properties of our proposal. In addition, we implemented a prototype of the proposed scheme and made the source code publicly available for further research.

We also proposed a methodology supported by a tool for formal verification of access control policies, which is built on top of mCRL2. Our XACML2mCRL2 tool implements the mapping from XACML policies into mCRL2 specifications. The mapping covers every element of XACML policies, i.e., policy set, policy, and rule, and allows us to formally verify the completeness and consistency of the XACML policies using the first-order modal μ-calculus. Moreover, in contrast to other related approaches, XACML2mCRL2 takes into account the obligation expressions. The model checker provided by the mCRL2 toolset automatically generates counterexamples useful for detecting and resolving incomplete and

inconsistent policies. We also modeled an e-Health use case to analyze the XACML policies in context, i.e., together with the system that these policies are supposed to protect. Our methodological approach to formal verification of access control policies can potentially be used to avoid critical problems in, for example, e-Health systems.

4.2 Answers to the research questions

4.2.1 RQ1: How can Semantic Attribute-Based Access Control be realized?

The first paper addresses this research question by performing a systematic literature review of Semantic Attribute-Based Access Control schemes. Paper I analyzes, discusses, and compares the existing SABAC schemes based on a set of research questions. Paper I addresses **RQ1** by demonstrating the strategies by which a semantic attribute-based access control can be achieved. Furthermore, Paper I points out several open problems towards an ideal SABAC, which three of them are the motivations for Paper II, Paper III, and Paper IV.

4.2.2 RQ2: How can a Cryptographic Semantic Attribute-Based Access Control Scheme be developed?

This research question is addressed by the second paper. Based on the literature review, Attribute-Based Encryption schemes, which are developed to protect resources in a fine-grained manner based on a set of attributes and access structures, are considered as a cryptographic replacement of ABAC. Hence, Paper II combines ABE schemes with semantic technologies in order to achieve semantic-aware ABE schemes by making ABE schemes able to use implicit knowledge from an ontology, while facilitating the interoperability between ABE schemes used in different domains. Paper II proposes two different schemes namely *Semantically-Enriched Key* and *Semantically-Enriched Access Structure*, where the first one utilizes the semantic technologies in the key generation process and the latter enriches the access structures (before encryption) utilizing semantic technologies. The experiment results confirm that the framework proposed in Paper II, which is a modular, extensible, scalable, and generic framework, can be considered as a cryptographic replacement for Semantic Attribute-Based Access Control.

ABE schemes lack the notion of obligations, which is common in Attribute-Based Access Control systems, to define and enforce additional constraints that happen before approving or denying an access request. Paper III provides a general way of augmenting ABE schemes with enforceable obligations, thus bringing one important feature from ABAC into ABE. Paper III defines a system architecture having Intel Software Guard Extensions (to provide trusted hardware enclaves) as the core component used for enforcing obligations added on top of any ABE scheme. Paper III also proposes a formal language for the specification of obligations.

4.2.3 RQ3: How can access control policies be verified?

The fourth paper addresses this research question by modeling and analyzing XACML access control policies using the mCRL2. Using the approach provided by Paper IV, different properties of the access control policies, e.g., completeness, consistency, and obligation enforcement, can be formally verified both independently and in context, i.e., together with the system that these policies are supposed to protect. Therefore, our methodological approach to formal verification of access control policies can potentially be used to avoid critical problems in, for example, e-Health systems.

4.3 Limitations

The OB-ABE scheme proposed in Paper III is based on Intel SGX's enclaves, which provide a trusted execution environment. However, there are still controversies about the security of SGX. Intel SGX has been under scrutiny since its early releases and several vulnerabilities have been discovered. However, these are often published along with corresponding countermeasures. Hence, the proposed OB-ABE scheme relies on the concept of trusted hardware enclaves. Furthermore, in Paper III, we assumed that there exists a library of different obligations, e.g., sending email, writing logs in Merkle trees, etc.; however, such a library yet needs to be developed.

In Paper II, we developed two mock ontologies for healthcare and social networks. For real-world scenarios, such ontologies need to be developed by domain experts who know the domain and have the required knowledge for developing the ontologies. Moreover, the quality, performance, and usefulness of the developed ontologies should be evaluated.

4.4 Future work

In Paper II, we proposed two different schemes namely SABE-SEK and SABE-SEAS towards Semantic Attribute-Based Encryption. In SABE-SEK, we used the semantic technologies in the key generation process to generate semantically-enriched private keys, i.e., we connected the *KeyGen* algorithm of a CP-ABE scheme to a semantic component. For the SABE-SEAS, we connected the semantic component to the *Encryption* algorithm to enrich and update the access structures. If the common ontology changes, which is rather infrequent since an ontology describes the relationships between attributes and not users, then the affected private keys (in SABE-SEK) may need to be revoked and regenerated according to the new ontology. Besides, ontology changes in SABE-SEAS may affect the existing ciphertexts as they may need to be re-encrypted. For future work, we plan to add the semantic component to the *Decryption* algorithm to address aspects regarding the *dynamicity* of the ontology. However, considering semantic relationships between attributes at the time of decryption may require private key update. To update a private key, the *Decryption* algorithm needs

not only to be executed in a trusted execution environment (as it deals with the private keys) but also to have secure communication with the *KeyGen* algorithm. These can be achieved using the trusted hardware enclaves provided by Intel SGX.

We also plan to make the decryption in ABE schemes accountable. With ABE, several users may share the same attributes; thus, they can decrypt the same ciphertexts. Hence, a user may misuse the private key to decrypt a ciphertext and no one can detect who has decrypted the ciphertext. For example, the medical recodes of patients (stored in the medical servers) can be decrypted by the patient (owner), her GP, and other medical staff (provided that there is an emergency). A malicious medical staff may try to decrypt the medical records of a specific patient and misuse them (while there is no emergency) and no one will know that a malicious staff has decrypted the medical records. However, for the sake of privacy, patients may like to know if someone has decrypted their medical records (a new requirement). Therefore, there is a need to make the entities who have the capability to decrypt the medical records accountable for their use of the decryption key (private key).

In Paper IV, we analyzed different properties of the access control policies according to the requirements of our case study. However, we believe more properties like *policy-redundancy* can be considered. Two policies are redundant (similar) if they cover the same set of access requests. Such a property helps to remove redundant policies, which in turn the performance of policy evaluation will be affected. Optimizing policies by avoiding redundancies (which results in overlap-free policies) improves the performance. For future work, we plan to consider more properties for access control policies.

Bibliography

[1] Abadi, M., Blanchet, B., and Comon-Lundh, H. "Models and Proofs of Protocol Security: A Progress Report". In: *Computer Aided Verification*. Vol. 5643. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2009, pp. 35–49.

[2] Afshar, M., Samet, S., and Hu, T. "An Attribute Based Access Control Framework for Healthcare System". In: *Journal of Physics: Conference Series* vol. 933 (2017), p. 012020.

[3] Ahn, G. et al. "Representing and Reasoning about Web Access Control Policies". In: *Proceedings of the 34th Annual IEEE International Computer Software and Applications Conference, COMPSAC 2010, Seoul, Korea, 19-23 July 2010*. IEEE Computer Society, 2010, pp. 137–146.

[4] Alder, F. et al. "S-FaaS: Trustworthy and Accountable Function-as-a-Service using Intel SGX". In: *Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop*. New York, NY, USA: ACM, 2019, pp. 185–199.

[5] Amini, M. and Arasteh, M. "A combination of semantic and attribute-based access control model for virtual organizations". In: *ISC International Journal of Information Security* vol. 7, no. 1 (2015), pp. 27–45.

[6] Amini, M. and Jalil, R. *MA (DL) 2 Logical Language Family for Policy Specification and Inference*. Tech. rep. Sharif University of Technology, 2010.

[7] Amini, M. and Jalili, R. "Multi-level authorisation model and framework for distributed semantic-aware environments". In: *IET Information Security* vol. 4, no. 4 (2010), pp. 301–321.

[8] Anderson, A. et al. "eXtensible Access Control Markup Language (XACML) Version 1.0". In: *OASIS Standard* (2003).

[9] Arnautov, S. et al. "PubSub-SGX: Exploiting Trusted Execution Environments for Privacy-Preserving Publish/Subscribe Systems". In: *37th Symposium on Reliable Distributed Systems (SRDS)*. Salvador, Brazil: IEEE Computer Society, 2018, pp. 123–132.

[10] Arshad, H. and Nikooghadam, M. "Three-Factor Anonymous Authentication and Key Agreement Scheme for Telecare Medicine Information Systems". In: *Journal of Medical Systems* vol. 38, no. 12 (2014), p. 136.

[11] Attrapadung, N. and Imai, H. "Dual-Policy Attribute Based Encryption". In: *International Conference on Applied Cryptography and Network Security*. Vol. 5536. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2009, pp. 168–185.

[12] Baader, F. and Hanschke, P. "A Scheme for Integrating Concrete Domains into Concept Languages". In: *Proceedings of the 12th International Joint Conference on Artificial Intelligence. Sydney, Australia, August 24-30, 1991*. Ed. by Mylopoulos, J. and Reiter, R. Morgan Kaufmann, 1991, pp. 452–457.

[13] Barua, M., Lu, R., and Shen, X. "SPS: Secure personal health information sharing with patient-centric access control in cloud computing". In: *IEEE global communications conference (GLOBECOM)*. Atlanta, GA, USA: IEEE, 2013, pp. 647–652.

[14] Bethencourt, J., Sahai, A., and Waters, B. "Ciphertext-Policy Attribute-Based Encryption". In: *IEEE symposium on security and privacy (SP'07)*. Berkeley, CA, USA: IEEE, 2007, pp. 321–334.

[15] Blanchet, B., Abadi, M., and Fournet, C. "Automated verification of selected equivalences for security protocols". In: *The Journal of Logic and Algebraic Programming* vol. 75, no. 1 (2008), pp. 3–51.

[16] Blobel, B. et al. "HL7 Version 3 Standard: Security and Privacy Ontology, Release 1". In: (2014).

[17] Bobba, R. et al. "Attribute-Based Messaging: Access Control and Confidentiality". In: *ACM Transactions on Information and System Security (TISSEC)* vol. 13, no. 4 (2010), 31:1–31:35.

[18] Boneh, D. and Franklin, M. "Identity-based encryption from the Weil pairing". In: *Annual international cryptology conference*. Berlin, Heidelberg: Springer, 2001, pp. 213–229.

[19] Brenner, S. et al. "Secure cloud micro services using Intel SGX". In: *IFIP International Conference on Distributed Applications and Interoperable Systems*. Cham: Springer, 2017, pp. 177–191.

[20] Brenner, S. et al. "SecureKeeper: Confidential ZooKeeper using Intel SGX". In: *Middleware Conference*. New York, NY, USA: ACM, 2016, p. 14.

[21] Brickley, D. "Resource Description Framework (RDF) Schema Specification". In: *http://www.w3.org/TR/rdf-schema* (2000).

[22] Bryans, J. W. "Reasoning about XACML policies using CSP". In: *Proceedings of the 2nd ACM Workshop On Secure Web Services, SWS 2005, Fairfax, VA, USA, November 11, 2005*. ACM, 2005, pp. 28–35.

[23] Bryant, R. E. "Graph-Based Algorithms for Boolean Function Manipulation". In: *IEEE Trans. Computers* vol. 35, no. 8 (1986), pp. 677–691.

[24] Buehrer, D. J. and Wang, C. "CA-ABAC: Class Algebra Attribute-Based Access Control". In: *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, Macau, China, December 4-7, 2012*. IEEE Computer Society, 2012, pp. 220–225.

[25] Calvillo-Arbizu, J., Roman-Martinez, I., and Roa-Romero, L. M. "Standardized access control mechanisms for protecting ISO 13606-based electronic health record systems". In: *Proceedings of IEEE-EMBS International Conference on Biomedical and Health Informatics, BHI 2014, Valencia, Spain, June 1-4, 2014*. IEEE, 2014, pp. 539–542.

[26] Calvillo-Arbizu, J., Román, I., and Roa, L. M. "Empowering citizens with access control mechanisms to their personal health resources". In: *International journal of medical informatics* vol. 82, no. 1 (2013), pp. 58–72.

[27] Calvillo-Arbizu, J. et al. "Privilege Management Infrastructure for Virtual Organizations in Healthcare Grids". In: *IEEE Transactions on Information Technology in Biomedicine* vol. 15, no. 2 (2011), pp. 316–323.

[28] Cantor, S. et al. *Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0*. 2005.

[29] Chase, M. "Multi-authority Attribute Based Encryption". In: *Theory of cryptography conference*. Berlin, Heidelberg: Springer, 2007, pp. 515–534.

[30] Chen, F. et al. "PRESAGE: privacy-preserving genetic testing via software guard extension". In: *BMC medical genomics* vol. 10, no. 2 (2017), p. 48.

[31] Cheng, R. et al. "Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contracts". In: *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. Stockholm, Sweden: IEEE, 2019, pp. 185–200.

[32] Ciuciu, I. et al. "Ontology Based Interoperation for Securely Shared Services: Security Concept Matching for Authorization Policy Interoperability". In: *4th IFIP International Conference on New Technologies, Mobility and Security, NTMS 2011, Paris, France, February 7-10, 2011*. IEEE, 2011, pp. 1–5.

[33] Coppolino, L. et al. "A comparative analysis of emerging approaches for securing java software with Intel SGX". In: *Future Generation Computer Systems* vol. 97 (2019), pp. 620–633.

[34] Damiani, E. et al. "Extending Policy Languages to the Semantic Web". In: *Web Engineering - 4th International Conference, ICWE 2004, Munich, Germany, July 26-30, 2004, Proceedings*. Ed. by Koch, N., Fraternali, P., and Wirsing, M. Vol. 3140. Lecture Notes in Computer Science. Springer, 2004, pp. 330–343.

[35] Damianou, N. et al. "The Ponder Policy Specification Language". In: *POLICY*. Vol. 1995. Lecture Notes in Computer Science. Baltimore, Maryland, USA: Springer, 2001, pp. 18–38.

[36] Dersingh, A. et al. "Utilizing Semantic Knowledge for Access Control in Pervasive and Ubiquitous Systems". In: *Mob. Networks Appl.* Vol. 15, no. 2 (2010), pp. 267–282.

[37] Drozdowicz, M., Ganzha, M., and Paprzycki, M. "Semantically Enriched Data Access Policies in eHealth". In: *Journal of Medical Systems* vol. 40, no. 11 (2016), 238:1–238:8.

[38] Dürbeck, S. et al. "A Semantic Security Architecture for Web Services The Access-eGov Solution". In: *2010 International Conference on Availability, Reliability and Security*. IEEE. 2010, pp. 222–227.

[39] Edgar, T. and Manz, D. *Research Methods for Cyber Security*. Syngress, 2017.

[40] Eskandarian, S. et al. "Fidelius: Protecting user secrets from compromised browsers". In: *2019 IEEE Symposium on Security and Privacy (SP)*. San Francisco, CA, USA: IEEE, 2019, pp. 264–280.

[41] Ferguson, D. et al. "PoCo: A Language for Specifying Obligation-Based Policy Compositions". In: *Proceedings of the 2020 9th International Conference on Software and Computer Applications*. ICSCA 2020. Langkawi, Malaysia: Association for Computing Machinery, 2020, pp. 331–338.

[42] Fisch, B. et al. "Iron: functional encryption using Intel SGX". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. CCS '17. Dallas, Texas, USA: ACM, 2017, pp. 765–782.

[43] Fisler, K. et al. "Verification and change-impact analysis of access-control policies". In: *27th International Conference on Software Engineering (ICSE 2005), 15-21 May 2005, St. Louis, Missouri, USA*. ACM, 2005, pp. 196–205.

[44] Friedman-Hill, E. J. *Jess, the java expert system shell*. Tech. rep. Sandia Labs., Livermore, CA (United States), 1997.

[45] Fujita, M., McGeer, P. C., and Yang, J.-Y. "Multi-Terminal Binary Decision Diagrams: An Efficient Data Structure for Matrix Representation". In: *Formal Methods in System Design* vol. 10, no. 2 (1997), pp. 149–169.

[46] Gabbay, D. et al. *Handbook of deontic logic and normative systems*. Milton Keynes, UK: College Publication, 2013.

[47] Gennari, J. H. et al. "The evolution of Protégé: an environment for knowledge-based systems development". In: *International Journal of Human-Computer Studies* vol. 58, no. 1 (2003), pp. 89–123.

[48] Gorbunov, S., Vaikuntanathan, V., and Wee, H. "Attribute-based encryption for circuits". In: *Journal of the ACM (JACM)* vol. 62, no. 6 (2015), pp. 1–33.

[49] Goyal, V. et al. "Attribute-based Encryption for Fine-grained Access Control of Encrypted Data". In: *Proceedings of the 13th ACM Conference on Computer and Communications Security*. CCS '06. Alexandria, Virginia, USA: ACM, 2006, pp. 89–98.

[50] Goyal, V. et al. “Bounded ciphertext policy attribute based encryption”. In: *International Colloquium on Automata, Languages, and Programming*. Berlin, Heidelberg: Springer, 2008, pp. 579–591.

[51] Haarslev, V. and Möller, R. “RACER System Description”. In: *Automated Reasoning, First International Joint Conference, IJCAR 2001, Siena, Italy, June 18-23, 2001, Proceedings*. Ed. by Goré, R., Leitsch, A., and Nipkow, T. Vol. 2083. Lecture Notes in Computer Science. Springer, 2001, pp. 701–706.

[52] Hathaliya, J. J. and Tanwar, S. “An exhaustive survey on security and privacy issues in Healthcare 4.0”. In: *Computer Communications* vol. 153 (2020), pp. 311–335.

[53] Havet, A. et al. “Securestreams: A reactive middleware framework for secure data stream processing”. In: *Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems*. DEBS ’17. Barcelona, Spain: ACM, 2017, pp. 124–133.

[54] Hilia, M. et al. “Semantic Based Authorization Framework For Multi-Domain Collaborative Cloud Environments”. In: *The 8th International Conference on Ambient Systems, Networks and Technologies (ANT 2017) / The 7th International Conference on Sustainable Energy Information Technology (SEIT 2017), 16-19 May 2017, Madeira, Portugal*. Ed. by Shakshuki, E. M. Vol. 109. Procedia Computer Science. Elsevier, 2017, pp. 718–724.

[55] Hilty, M., Basin, D. A., and Pretschner, A. “On Obligations”. In: *ESORICS*. Vol. 3679. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2005, pp. 98–117.

[56] Hilty, M. et al. “A Policy Language for Distributed Usage Control”. In: *ESORICS*. Vol. 4734. Lecture Notes in Computer Science. Dresden, Germany: Springer, 2007, pp. 531–546.

[57] Hoare, C. A. R. “Communicating Sequential Processes”. In: *Commun. ACM* vol. 21, no. 8 (1978), pp. 666–677.

[58] Horrocks, I. et al. “SWRL: A Semantic Web Rule Language Combining OWL and RuleML”. In: *W3C Member submission* vol. 21 (2004), p. 79.

[59] Hsu, I. “Extensible access control markup language integrated with Semantic Web technologies”. In: *Information Sciences* vol. 238 (2013), pp. 33–51.

[60] Hu, H., Ahn, G., and Kulkarni, K. “Anomaly discovery and resolution in web access control policies”. In: *16th ACM Symposium on Access Control Models and Technologies, SACMAT 2011, Innsbruck, Austria, June 15-17, 2011, Proceedings*. ACM, 2011, pp. 165–174.

[61] Hu, S., Li, J., and Zhang, Y. “Improving Security and Privacy-Preserving in Multi-Authorities Ciphertext-Policy Attribute-Based Encryption”. In: *KSII Transactions on Internet & Information Systems* vol. 12, no. 10 (2018), pp. 5100–5119.

[62] Hu, V. C. et al. “Guide to Attribute Based Access Control (ABAC) Definition and Considerations”. In: *NIST Special Publication (SP)* vol. 800, no. 162 (2014), pp. 1–47.

[63] Hur, J. and Noh, D. K. “Attribute-Based Access Control with Efficient Revocation in Data Outsourcing Systems”. In: *IEEE Transactions on Parallel and Distributed Systems* vol. 22, no. 7 (2011), pp. 1214–1221.

[64] Husain, M. F. et al. “Ontology based policy interoperability in geo-spatial domain”. In: *Computer Standards & Interfaces* vol. 33, no. 3 (2011), pp. 214–219.

[65] Iannella, R. “The Open Digital Rights Language: XML for Digital Rights Management”. In: *Information Security Technical Report* vol. 9, no. 3 (2004), pp. 47–55.

[66] Iqbal, Z. and Noll, J. “Towards Semantic-Enhanced Attribute-Based Access Control for Cloud Services”. In: *11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2012, Liverpool, United Kingdom, June 25-27, 2012*. Ed. by Min, G. et al. IEEE Computer Society, 2012, pp. 1223–1230.

[67] Irwin, K., Yu, T., and Winsborough, W. H. “On the modeling and analysis of obligations”. In: *CCS*. CCS ’06. Alexandria, Virginia, USA: ACM, 2006, pp. 134–143.

[68] Al-Issa, Y., Ottom, M. A., and Tamrawi, A. “eHealth Cloud Security Challenges: A Survey”. In: *Journal of Healthcare Engineering* vol. 2019 (2019), pp. 1–15.

[69] Jiang, Y. et al. “Ciphertext-policy attribute-based encryption against key-delegation abuse in fog computing”. In: *Future Generation Computer Systems* vol. 78 (2018), pp. 720–729.

[70] Jin, P. and Fang-Chun, Y. “Description Logic Modeling of Temporal Attribute-Based Access Control”. In: *2006 First International Conference on Communications and Electronics (ICCE)*. IEEE. 2006, pp. 414–418.

[71] Kagal, L., Finin, T. W., and Joshi, A. “A Policy Language for a Pervasive Computing Environment”. In: *POLICY*. Lake Como, Italy: IEEE Computer Society, 2003, p. 63.

[72] Kayem, A. V. D. M., Akl, S. G., and Martin, P. *Adaptive Cryptographic Access Control*. Vol. 48. Advances in Information Security. Springer, 2010.

[73] Kitchenham, B. “Procedures for Performing Systematic Reviews”. In: *Keele, UK, Keele University* vol. 33, no. 2004 (2004), pp. 1–26.

[74] Kitchenham, B. et al. “Systematic literature reviews in software engineering–A systematic literature review”. In: *Information and Software Technology* vol. 51, no. 1 (2009), pp. 7–15.

[75] Kolovski, V., Hendler, J. A., and Parsia, B. “Analyzing web access control policies”. In: *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*. ACM, 2007, pp. 677–686.

[76] Lambert, C. et al. “MaskAl: Privacy Preserving Masked Reads Alignment using Intel SGX”. In: *37th Symposium on Reliable Distributed Systems (SRDS)*. Salvador, Brazil: IEEE, 2018, pp. 113–122.

[77] Lampson, B. W. “Protection”. In: *ACM SIGOPS Operating Systems Review* vol. 8, no. 1 (Jan. 1974), pp. 18–24.

[78] Lewko, A., Sahai, A., and Waters, B. “Revocation systems with very small private keys”. In: *IEEE Symposium on Security and Privacy*. Oakland, CA, USA: IEEE, 2010, pp. 273–285.

[79] Li, J. et al. “Secure attribute-based data sharing for resource-limited users in cloud computing”. In: *Computers & Security* vol. 72 (2018), pp. 1–12.

[80] Li, N., Chen, H., and Bertino, E. “On practical specification and enforcement of obligations”. In: *CODASPY*. San Antonio Texas, USA: ACM, 2012, pp. 71–82.

[81] Li, N. and Mitchell, J. C. “DATALOG with Constraints: A Foundation for Trust Management Languages”. In: *Practical Aspects of Declarative Languages, 5th International Symposium, PADL 2003, New Orleans, LA, USA, January 13-14, 2003, Proceedings*. Ed. by Dahl, V. and Wadler, P. Vol. 2562. Lecture Notes in Computer Science. Springer, 2003, pp. 58–73.

[82] Li, X. et al. “A survey on the security of blockchain systems”. In: *Future Generation Computer Systems* vol. 107 (2020), pp. 841–853.

[83] Lifschitz, V. “What Is Answer Set Programming?” In: *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*. AAAI Press, 2008, pp. 1594–1597.

[84] Lin, D. et al. “EXAM: a comprehensive environment for the analysis of access control policies”. In: *Int. J. Inf. Sec.* Vol. 9, no. 4 (2010), pp. 253–273.

[85] Liu, J., Huang, X., and Liu, J. K. “Secure sharing of Personal Health Records in cloud computing: Ciphertext-Policy Attribute-Based Signcryption”. In: *Future Generation Computer Systems* vol. 52 (2015). Special Section: Cloud Computing: Security, Privacy and Practice, pp. 67–76.

[86] Liu, Z. and Wang, J. “A fine-grained context-aware access control model for health care and life science linked data”. In: *Multimedia Tools and Applications* vol. 75, no. 22 (2016), pp. 14263–14280.

[87] Lu, Y. and Sinnott, R. O. “Semantic privacy-preserving framework for electronic health record linkage”. In: *Telematics Informatics* vol. 35, no. 4 (2018), pp. 737–752.

[88] Manola, F., Miller, E., McBride, B., et al. “RDF primer”. In: *W3C recommendation* vol. 10, no. 1-107 (2004), p. 6.

[89] Marek, V. W. and Truszczynski, M. “Stable Models and an Alternative Logic Programming Paradigm”. In: *The Logic Programming Paradigm - A 25-Year Perspective*. Artificial Intelligence. Springer, 1999, pp. 375–398.

[90] McBride, B. “Jena: A Semantic Web Toolkit”. In: *IEEE Internet Computing* vol. 6, no. 6 (2002), pp. 55–59.

[91] McKeen, F. et al. “Innovative Instructions and Software Model for Isolated Execution”. In: *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy*. HASP ’13. Tel-Aviv, Israel: ACM, 2013.

[92] Meddah, N., Jebrane, A., and Toumanari, A. “Scalable lightweight ABAC scheme for secure sharing PHR in cloud computing”. In: *International Conference on Advanced Information Technology, Services and Systems*. Tangier, Morocco: Springer, 2017, pp. 333–346.

[93] Mokhtar, S. B. et al. “X-search: revisiting private web search using Intel SGX”. In: *Middleware Conference*. Middleware ’17. Las Vegas, Nevada: ACM, 2017, pp. 198–208.

[94] Morisset, C., Willemse, T. A. C., and Zannone, N. “A framework for the extended evaluation of ABAC policies”. In: *Cybersecur.* Vol. 2, no. 1 (2019), p. 6.

[95] Mukherjee, S. et al. “Attribute based access control for healthcare resources”. In: *Proceedings of the 2nd ACM Workshop on Attribute-Based Access Control*. ABAC ’17. Scottsdale, Arizona, USA: ACM, 2017, pp. 29–40.

[96] Müller, S., Katzenbeisser, S., and Eckert, C. “Distributed attribute-based encryption”. In: *International Conference on Information Security and Cryptology*. Seoul, Korea: Springer, 2008, pp. 20–36.

[97] Muppavarapu, V. and Chung, S. M. “Semantic-Based Access Control for Grid Data Resources in Open Grid Services Architecture - Data Access and Integration (OGSA-DAI)”. In: *20th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2008), November 3-5, 2008, Dayton, Ohio, USA, Volume 2*. IEEE Computer Society, 2008, pp. 315–322.

[98] Ni, Q., Bertino, E., and Lobo, J. “An obligation model bridging access control policies and privacy policies”. In: *SACMAT*. SACMAT ’08. Estes Park, CO, USA: ACM, 2008, pp. 133–142.

[99] Ni, Q. et al. “Privacy-Aware Role-Based Access Control”. In: *IEEE Secur. Priv.* Vol. 7, no. 4 (2009), pp. 35–43.

[100] Nikooghadam, M. and Zakerolhosseini, A. “Secure Communication of Medical Information Using Mobile Agents”. In: *Journal of Medical Systems* vol. 36, no. 6 (2012), pp. 3839–3850.

[101] Ostrovsky, R., Sahai, A., and Waters, B. “Attribute-based encryption with non-monotonic access structures”. In: *Proceedings of the 14th ACM conference on Computer and communications security*. CCS ’07. Alexandria, Virginia, USA: ACM, 2007, pp. 195–203.

[102] Parducci, B., Lockhart, H., and Rissanen, E. “Extensible access control markup language (XACML) version 3.0”. In: *OASIS Standard* vol. 2013, no. 1 (2013), pp. 1–154.

[103] Picazo-Sanchez, P., Pardo, R., and Schneider, G. “Secure photo sharing in social networks”. In: *IFIP International Conference on ICT Systems Security and Privacy Protection*. Rome, Italy: Springer, 2017, pp. 79–92.

[104] Pires, R. et al. “CYCLOSA: Decentralizing Private Web Search Through SGX-Based Browser Extensions”. In: *38th International Conference on Distributed Computing Systems (ICDCS)*. Vienna, Austria: IEEE, 2018, pp. 467–477.

[105] Priebe, T., Dobmeier, W., and Kamprath, N. “Supporting Attribute-based Access Control with Ontologies”. In: *Proceedings of the The First International Conference on Availability, Reliability and Security, ARES 2006, The International Dependability Conference - Bridging Theory and Practice, April 20-22 2006, Vienna University of Technology, Austria*. IEEE Computer Society, 2006, pp. 465–472.

[106] Qiang, W., Dong, Z., and Jin, H. “Se-Lambda: Securing Privacy-Sensitive Serverless Applications Using SGX Enclave”. In: *International Conference on Security and Privacy in Communication Systems*. Singapore, Singapore: Springer, 2018, pp. 451–470.

[107] Ramli, C. D. P. K., Nielson, H. R., and Nielson, F. “The logic of XACML”. In: *Sci. Comput. Program.* Vol. 83 (2014), pp. 80–105.

[108] Rao, P. et al. “An algebra for fine-grained integration of XACML policies”. In: *14th ACM Symposium on Access Control Models and Technologies, SACMAT 2009, Stresa, Italy, June 3-5, 2009, Proceedings*. ACM, 2009, pp. 63–72.

[109] Ray, I. et al. “Applying attribute based access control for privacy preserving health data disclosure”. In: *IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*. Las Vegas, NV, USA: IEEE, 2016, pp. 1–4.

[110] Ray, I. et al. “Using attribute-based access control for remote healthcare monitoring”. In: *International Conference on Software Defined Systems (SDS)*. Valencia, Spain: IEEE, 2017, pp. 137–142.

[111] Ribeiro, C. et al. “SPL: An Access Control Language for Security Policies and Complex Constraints”. In: *NDSS*. San Diego, California: The Internet Society, 2001, pp. 1–19.

[112] Sahai, A. and Waters, B. “Fuzzy identity-based encryption”. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Aarhus, Denmark: Springer, 2005, pp. 457–473.

[113] Sartakov, V. et al. “STANlite–a database engine for secure data processing at rack-scale level”. In: *International Conference on Cloud Engineering (IC2E)*. Orlando, FL, USA: IEEE, 2018, pp. 23–33.

[114] Seaborne, A. and Prud’hommeaux, E. “SPARQL Query Language for RDF”. In: *W3C recommendation (January 2008)* (2006).

[115] Severinsen, K. M. “Secure Programming with Intel SGX and Novel Applications”. MA thesis. Universitetet i Oslo, 2017.

[116] Sfyrakis, I. and Gross, T. “UniGuard: Protecting Unikernels Using Intel SGX”. In: *IEEE International Conference on Cloud Engineering (IC2E)*. Orlando, FL, USA: IEEE, 2018, pp. 99–105.

[117] Shaon, F. et al. “SGX-BigMatrix: A practical encrypted data analytic framework with trusted processors”. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’17. Dallas, Texas, USA: ACM, 2017, pp. 1211–1228.

[118] Shen, H.-B. “A Semantic- and Attribute-Based Framework for Web Services Access Control”. In: *2010 2nd International Workshop on Intelligent Systems and Applications*. IEEE. 2010, pp. 1–4.

[119] Shen, H. and Cheng, Y. “A Context-Aware Semantic-Based Access Control Model for Mobile Web Services”. In: *Advanced Research on Computer Science and Information Engineering*. Ed. by Shen, G. and Huang, X. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 132–139.

[120] Sirin, E. and Parsia, B. “Pellet: An OWL DL Reasoner”. In: *Proceedings of the 2004 International Workshop on Description Logics (DL2004), Whistler, British Columbia, Canada, June 6-8, 2004*. Ed. by Haarslev, V. and Möller, R. Vol. 104. CEUR Workshop Proceedings. CEUR-WS.org, 2004.

[121] Sookhak, M. et al. “Attribute-based data access control in mobile cloud computing: Taxonomy and open issues”. In: *Future Generation Computer Systems* vol. 72 (2017), pp. 273–287.

[122] Tang, Q. and Ji, D. “Verifiable Attribute Based Encryption”. In: *IJ Network Security* vol. 10, no. 2 (2010), pp. 114–120.

[123] Tramèr, F. and Boneh, D. “Slalom: Fast, Verifiable and Private Execution of Neural Networks in Trusted Hardware”. In: *ICLR*. New Orleans, Louisiana, United States: OpenReview.net, 2019, pp. 1–19.

[124] Trivellato, D. et al. “A Semantic Security Framework for Systems of Systems”. In: *Int. J. Cooperative Inf. Syst.* Vol. 22, no. 1 (2013).

[125] Turkmen, F. et al. “Formal analysis of XACML policies using SMT”. In: *Comput. Secur.* Vol. 66 (2017), pp. 185–203.

[126] Tychalas, D., Tsoutsos, N. G., and Maniatakos, M. “SGXCrypter: IP protection for portable executables using Intel’s SGX technology”. In: *22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*. Chiba, Japan: IEEE, 2017, pp. 354–359.

[127] Verginadis, Y. et al. “Context-aware Policy Enforcement for PaaS-enabled Access Control”. In: *IEEE Transactions on Cloud Computing* (2019).

[128] Wang, G., Liu, Q., and Wu, J. “Hierarchical attribute-based encryption for fine-grained access control in cloud storage services”. In: *Proceedings of the 17th ACM conference on Computer and communications security*. CCS ’10. Chicago, Illinois, USA: ACM, 2010, pp. 735–737.

[129] Wang, G. et al. “Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers”. In: *Computers & Security* vol. 30, no. 5 (2011), pp. 320–331.

[130] Waters, B. “Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization”. In: *International Workshop on Public Key Cryptography*. Taormina, Italy: Springer, 2011, pp. 53–70.

[131] Wohlin, C. “Guidelines for snowballing in systematic literature studies and a replication in software engineering”. In: *18th International Conference on Evaluation and Assessment in Software Engineering, EASE ’14, London, England, United Kingdom, May 13-14, 2014*. Ed. by Shepperd, M. J., Hall, T., and Myrtveit, I. ACM, 2014, 38:1–38:10.

[132] Yeh, L. et al. “Cloud-Based Fine-Grained Health Information Access Control Framework for Lightweight IoT Devices with Dynamic Auditing and Attribute Revocation”. In: *IEEE Transactions on Cloud Computing* vol. 6, no. 2 (2018), pp. 532–544.

[133] Yu, S. et al. “Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing”. In: *INFOCOM 2010. 29th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 15-19 March 2010, San Diego, CA, USA*. IEEE, 2010, pp. 534–542.

[134] Zhang, K. et al. “Security and privacy for mobile healthcare networks: from a quality of protection perspective”. In: *IEEE Wireless Communications* vol. 22, no. 4 (2015), pp. 104–112.

[135] Zhang, L., Liang, P., and Mu, Y. “Improving privacy-preserving and security for decentralized key-policy attributed-based encryption”. In: *IEEE Access* vol. 6 (2018), pp. 12736–12745.

[136] Zhang, L., Zhu, S., and Tang, S. “Privacy Protection for Telecare Medicine Information Systems Using a Chaotic Map-Based Three-Factor Authenticated Key Agreement Scheme”. In: *IEEE Journal of Biomedical and Health Informatics* vol. 21, no. 2 (2017), pp. 465–475.

[137] Zhang, S., Yang, H., and Wang, B. “Realization Distributed Access Control Based on Ontology and Attribute with OWL”. In: *Advances in Electronic Engineering, Communication and Management Vol. 1*. Springer, 2012, pp. 583–588.

[138] Zhao, Y. and Wang, X. “Semantic Similarity-Based Web Services Access Control”. In: *Autonomous Systems: Developments and Trends*. Ed. by Unger, H., Kyamakya, K., and Kacprzyk, J. Vol. 391. Studies in Computational Intelligence. Springer, 2012, pp. 339–349.

Part II: Papers

Paper I

Semantic Attribute-Based Access Control: A review on current status and future perspectives

Hamed Arshad, Christian Johansen, Olaf Owe

Journal of Systems Architecture. Vol. 129, (2022), pp. 1–24. DOI: 10.1016/j.sysarc.2022.102625.

Abstract

Attribute-based access control (ABAC) uses the attributes of the involved entities (i.e., subject, object, action, and environment) to provide access control. Despite various advantages offered by ABAC, it is not the best fit for distributed and heterogeneous environments where the attributes of an entity may not necessarily match (syntactically) those used in the access control policies. Therefore, another type of access control called Semantic Attribute-Based Access Control (SABAC) has emerged that takes into account the semantics of attributes by combining ABAC with semantic technologies. SABAC not only facilitates interoperability but also enhances the expressiveness of access control policies. Over the last decade, a number of research efforts have been conducted in developing semantic attribute-based access control schemes. However, there exists no survey paper on SABAC schemes, giving an overview of the existing approaches. Hence, this paper comprehensively reviews the conducted research efforts for developing SABAC. The main goal of this paper is to provide a comprehensive summary of the conducted research studies that is useful for researchers who want to enter and make contributions to this field. Furthermore, the paper identifies open problems and possible research entry points by demonstrating the advantages and disadvantages of the previous studies.

I.1 Introduction

Attribute-Based Access Control (ABAC) is a successor of Role-Based Access Control (RBAC) where the involved entities (e.g., subject or object) have associated attributes that are used to provide access control. Under the ABAC model, there is no need to assign capabilities (access rights) to subjects (e.g.,

users, groups, and roles) in advance. Upon receiving an access request, the access decision would be made based on the attributes of the requested object (resource), attributes of the requester (subject), conditions of the environment (e.g., time of the day, authentication level, location), attributes of the desired action, and predefined access control policies. ABAC has several advantages over traditional access control models such as Mandatory Access Control, Discretionary Access Control, or Role-Based Access Control (RBAC), and has reached the maturity of OASIS standards with the eXtensible Access Control Markup Language (XACML) [4] and the Security Assertion Markup Language (SAML) [23].

ABAC is appeared to be useful in open and distributed systems. However, since such systems are heterogeneous, the attributes of the involved entities may not necessarily match those specified in the policies defined for accessing services or objects. For example, an e-healthcare system may represent adult patients with an attribute *"age"*, while patients may want to demonstrate this by providing an attribute *"hasDrivingLicense"*. However, the access control engine cannot infer that having a driving license means that the requester is an adult person. In ABAC, this issue could be addressed when defining a policy by including all the possible synonyms (semantically) of each attribute, e.g., by specifying several policies for the same object or one general policy covering all the synonyms of attributes. However, when a change occurs in a policy, a large number of policies may need to be updated accordingly, which in turn makes the management of policies a complicated and error-prone task.

In order to address such problems, another type of access control called the Semantic Attribute-Based Access Control (SABAC) has emerged as an extension of ABAC. The goal of SABAC is to augment ABAC with semantic technologies in order to take into account the semantic relationships between the involved entities when making a decision. Semantic technologies help the access control engine to infer implicit knowledge (e.g., semantic synonyms of attributes) from explicit knowledge, i.e., the attributes provided in the requests and predefined policies.

A considerable number of semantic attribute-based access control schemes have been proposed, yet there exists no survey paper on this topic. Hence, this paper systematically reviews the existing techniques for SABAC and their (dis)advantages up to May 2020. We identify several areas for improvement in Section I.6, aiming for what we defined in Section I.5.1 as an ideal SABAC. To motivate well these improvements and areas for further investigation, in Section I.4 we perform an in-depth analysis and comparison of the existing SABAC schemes. For this purpose, we also present the architectures of these schemes in a uniform manner, mapping to the standard XACML architecture.

Contributions:

- Reviewing the conducted research efforts on SABAC systematically and providing a comprehensive summary of them (Section I.4).

- Representing different SABAC architectures in a unified manner based on the reference architecture of the XACML standard (throughout Section I.4).
- Classifying the existing SABAC schemes based on different criteria (Section I.4, Fig. I.5, and Fig. I.6).
- Providing an in-depth comparison and discussion of the existing SABAC schemes (Section I.5 and Table I.2).
- Describing the properties of an ideal SABAC (Section I.5.1).
- Identifying open problems and research directions towards the ideal SABAC (Section I.6).

Outline:

Section III.3 provides basic information on the XACML standard, which provides a policy language and a reference architecture for ABAC, and semantic technologies. Section I.3 details the review methodology that we employed in this paper. In Section I.4, we review the existing SABAC schemes by demonstrating the advantages, disadvantages, key features, and considerations in the design of each scheme. Based on the comparison and discussion provided in Section I.5, we describe an ideal SABAC scheme (Section I.5.1) and identify open problems and potential research directions (Section I.6). Related work is provided in Section I.7 and conclusions appear in Section IV.6.

I.2 Preliminaries

This section provides general information on the XACML standard and semantic technologies.

I.2.1 XACML Standard

The eXtensible Access Control Markup Language is a standard managed by OASIS (Organization for the Advancement of Structured Information Standards)[1], providing both a policy language, as well as a reference architecture for ABAC. The standard also specifies the process by which the requests are evaluated based on the predefined policies.

Three main benefits of XACML are:

- XACML is *policy-based*, which makes it possible to describe different authorization scenarios easily. Besides, it is easy to audit the authorization artifacts and create complex policies depending on the authorization scenarios.

[1] https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml

- XACML is *attribute-based* or *multi-factor*. The XACML policies are based on attributes of subjects, objects, actions, and environment. For example, subjects (users) can be described in terms of age, date of birth, citizenship, and clearance level. Similarly, objects (resources) may have attributes such as location, classification, and owner.
- XACML is *technology-neutral*. It means that the XACML can be developed using different languages, e.g., C#, Python, and Java, and also can be used in different applications such as legacy applications or mainframe applications.

I.2.1.1 Policy specification language

The XACML policy language provides, as shown in Fig. I.1, elements for defining policies, rules, algorithms for combining rules and policies, obligations, and attributes of involved entities.

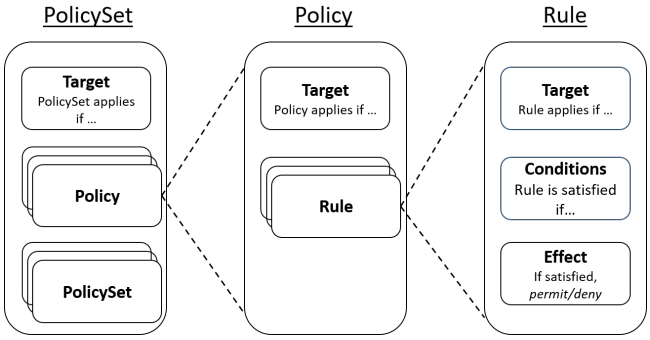


Figure I.1: The structure of the XACML policy language.

The main structural elements of the XACML policy language are PolicySet, Policy, and Rule.

A PolicySet contains multiple policies that have the same general purpose. The *target* is used to determine to which requests is this PolicySet applicable. A PolicySet may also contain references to other PolicySets, i.e., policies in other PolicySets can also be included in a PolicySet. A Policy is composed of one or more rules. The *target* of the Policy is more fine-grained than that of the PolicySet. A Rule includes a set of *conditions* as its core part and an *effect* element, which can be either a Deny or Permit. A Rule also has a *target* specifying a more fine-grained set of access requests for which the Rule is applicable.

Targets are statements based on subject, object, action, and environment attributes that help the access control engine to find the right (applicable) policies (as well as policy sets and rules) when receiving an access request. In other words, targets expedite the decision-making process as the access control engine (decision-making engine) does not evaluate an access request against all the existing policies. However, targets are not powerful enough because they

are limited to a AND/OR/AND structure. Besides, attributes in a target can only be matched to single constants. Hence, *conditions* are added to rules to overcome these limitations. In a *condition*, attributes can be matched against not only constant values but also other attributes. Besides, it is also possible to use non-Boolean functions. If the attributes existing in a request match those in the target of a rule, the conditions part will be checked and if the conditions evaluate to true, then the effect of the rule will be returned to the policy containing the rule. However, if the conditions are not applicable to the access request, or if an error happens, NotApplicable or Indeterminate, respectively, will be returned as the effect of the rule.

Since a Policy may contain more than one Rule, contradictory effects may be returned for the same access request. For example, suppose Alice, who is a programmer, wants to access ObjectA at 4:00 AM and the applicable Policy includes the two following rules:

Rule 1: (Job-title = Programmer) AND (Action-id = Read) AND (Object-id = ObjectA) → (Effect = Permit).

Rule 2: (18:00 < Current-Time < 08:00) → (Effect = Deny)

Since both rules are applicable to Alice's request, the parent policy receives both Permit and Deny effects, which are in contradiction to each other. In the XACML, such issues are addressed by means of combining algorithms, which combine the results of conflicting rules (*rule-combining algorithms*) and conflicting policies (*policy-combining algorithms*). For example, the "Permit Overrides" combining algorithm states that in the case of receiving a Permit and a Not Applicable, an Indeterminate, or a Deny, the final decision is Permit, i.e., Permit overrides other effects.

I.2.1.2 Architecture

The reference architecture of the XACML standard, shown in Fig. I.2, includes the (possibly distributed) components described below.

A Policy Enforcement Point (PEP) is responsible for protecting objects, which are to be understood rather generally, including applications, services, and files. The PEP receives incoming requests and forwards them to the second component of the XACML architecture, i.e., the Context Handler, which converts them into XACML requests and forwards the converted ones to a Policy Decision Point (PDP). The PDP is the core of the architecture which evaluates incoming access requests against access control policies and makes decisions which it returns to the PEP (through the Context Handler) to be enforced. Two components in the XACML architecture namely the Policy Information Point (PIP) and the Policy Administration Point (PAP) support the decision function, where the latter contains the access control policies and allows administrators to create and manage policies. If the PEP does not provide enough information to the PDP to reach a decision (i.e., there exist attributes that are referenced in the applicable access control policies but not provided in the access request), the

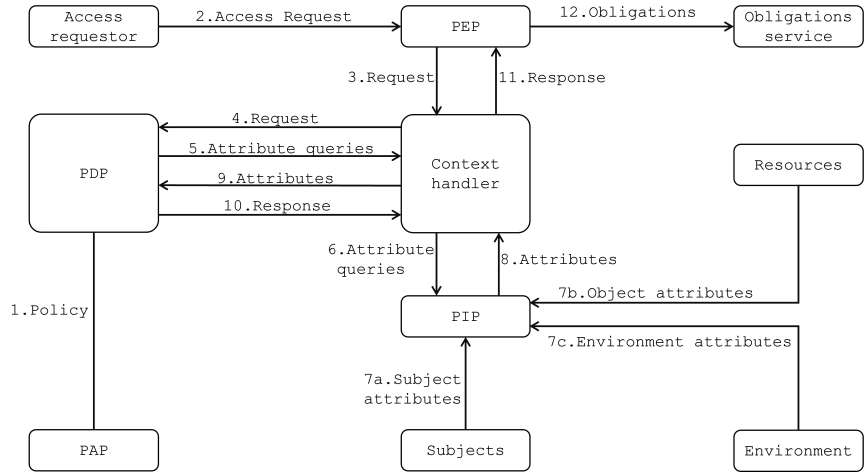


Figure I.2: The reference architecture of the XACML [taken from the standard].

PDP can retrieve the missing information from the PIP, which stores attribute values. PIPs could be databases containing product or document information, user directories, LDAP, or an Active Directory.

For example, suppose Alice sends a request to access ObjectA. The PEP intercepts Alice's request and forwards it to the Context Handler, which converts the request into a request in the XACML format and sends the converted request, i.e., {(`Subject-id = Alice`), (`Action-id = Read`), (`Object-id = ObjectA`)}, to the PDP. The PDP loads the applicable access control policy (policies), based on the target of policies and the information provided in the request, from the PAP. Assume that the applicable policy is as follows:

(`Job-title = Programmer`) AND (`Action-id = Read`) AND (`Object-id = ObjectA`) AND (`Subject.Clearance-level` $>$ `ObjectA.Classification`) $\rightarrow$ (`Effect = Permit`).

There are attributes referenced in the applicable policy that are not provided in the request. Hence, the PDP sends a request to the PIP to obtain all missing attributes, which are: 1) the job title of Alice, i.e., `Job-title = Programmer`, 2) the clearance level of Alice, i.e., `Subject.Clearance-level`, and 3) the classification of ObjectA, i.e., `ObjectA.Classification`. The PIP response is that Alice is a programmer with the clearance level secret and ObjectA's classification is confidential, i.e., {(`Job-title = Programmer`), (`Subject.Clearance-level = 2`), (`ObjectA.Classification` $= 1$)}. Based on this information, the PDP reaches a decision Permit and sends it to the PEP to be enforced.

A policy may include obligation expressions to enforce extra constraints that cannot be managed by normal policies (e.g., writing logs, sending emails, or showing warnings). Obligations can be used in several scenarios such as

```
<Obligation FulfillOn="Deny " ObligationId="send-email">
<AttributeAssignment DataType="http://www.w3.org/2001/XML
Schema#string" AttributeId="email">administrator@example.com
</AttributeAssignment>
</Obligation>
```

Figure I.3: An obligation specified in XACML

healthcare environments, governmental settings, or public-sector services. For example, as shown in Fig. I.3, an obligation can be sending an email to the administrator of the system (administrator@example.com) for every unsuccessful access attempt. The PDP asks the PEP (by including the obligation into the response) to enforce the obligation (in addition to the access decision) if the decision matches the value of the "FulfillOn" attribute (of the obligation).

One of the key aspects of the XACML architecture is the fact that it is a loosely coupled architecture as the management function (provided by the PAP and PIP), the decision function (provided by the PDP), and the enforcement function (provided by the PEP) are cleanly decoupled. For a single point of management, several PDPs may exist and in turn, a single PDP may serve several PEPs. Besides, the XACML architecture can be used on different applications and technologies as the PEP and the Context Handler will adapt the requests and responses to the specific target application (and specific technology) and still talk back to the same PDP.

I.2.2 Semantic Technologies

Semantic technologies are a collection of methods, techniques, and tools that enhance data interoperability and the power of data by bringing into account the meaning rather than the structure of the data.

The basic format for representing semantic data is Resource Description Framework (RDF) [1] [70], which represents data (or knowledge) in a "subject, predicate, object" pattern, e.g., "Programmer isA Job-title". A set of such triples can be represented as a directed graph, i.e., an RDF graph. The set of names (subjects, predicates, and objects) in a graph is called the vocabulary of the graph. In other words, concepts in a certain domain and relationships between them can be described and represented by means of vocabularies. Vocabularies are useful not only for organizing knowledge but also for resolving ambiguities when integrating different datasets. RDF Schema (RDFS) [2] [15] was proposed as a language for defining simple vocabularies, based on the concepts of class and property, allowing to perform simple inferences about these. More complex vocabularies are called ontologies, and are created using languages such as Web Ontology Language (OWL), which provides richer semantics than RDFS.

[1] https://www.w3.org/TR/rdf11-mt/
[2] https://www.w3.org/TR/rdf-schema/

Despite the widespread use of RDFS and OWL, there are useful semantic relationships that cannot be expressed using these, e.g., it is difficult to specify that a wholesale customer is a person who purchases large lots of items. Such more complex relationships can be handled using Semantic Web Rule Language (SWRL) [50]. SWRL is a popular and standard rule markup language that combines Horn logic and OWL ontologies, making it possible to specify complex inference rules in addition to the basic ones reflecting inheritance. Inference rules make it also possible to infer new knowledge (i.e., inferring implicit relationships from explicit ones). For example, suppose a dataset has two triples `(Alice isA Person)` and `(Alice hasAge 19)`. An ontology may state that "every person of age above 18 is an adult", which is written in SWRL as the rule $\forall$`x,y.isA(?x, Person)` $\wedge$ `hasAge(?x, ?y)` $\wedge$ `swrlb:greaterThanOrEqual(?y, 18)` $\rightarrow$ `isA(?x, Adult)`. Hence, a new relationship `(Alice isA Adult)` can be added to the dataset as a result of the inference process, done by a reasoner (also known as the rule engine, reasoning engine, or semantic reasoner) based on a set of facts and axioms.

In order to retrieve information from ontologies, a query language is required. SPARQL [87] was developed as a query language just like SQL for relational databases or XQuery for XML texts. For example, assume a triple `(Alice isA Doctor)` exists in a dataset. A SPARQL request may be issued as `(Alice isA ?medicalstaff)`, where `?medicalstaff` is a variable. The query engine finds `Doctor` as a possible value for `?medicalstaff` and returns it as a possible answer.

There exist multiple tools for creating and managing ontologies, and performing the reasoning process; those used in the reviewed works are briefly surveyed below.

Jena [72] is a free and open-source Java-based framework for creating semantic-based applications. Jena supports RDF/RDFS, DAML+OIL, and OWL ontologies, comprises APIs for RDF and OWL, a SPARQL query engine, a generic rule engine, and in-memory and persistent storage for RDF triples. Different reasoners can be used as plug-ins into Jena [74].

Protégé [41] is a Java-based ontology editor and framework for creating semantic-based applications for various domains. It is an open-source tool that is extensible and freely available. It is possible to use all the reasoners in Protégé (as plug-ins). It supports RDF/RDFS, OIL, DAML+OIL, OWL, and SWRL [5, 10].

Jess [40] is a Java-based rule engine (i.e., reasoner). The architecture of Jess includes a rule base as the knowledge base, a working memory as the fact base, and an inference engine. It works by matching facts in the fact base to the rules in the knowledge base by means of the Rete algorithm [39], which can handle many-to-many matching problems in a very efficient manner. Jess, as one of the fastest and lightweight reasoners, has a LISP-like syntax that is easy to learn and employ [49]. Though it is not open-source, it is free for educational purposes (a license is required for commercial purposes).

Racer [42] is an OWL reasoner that is the predecessor of RacerPro [44], supporting RDF/RDFS, OIL, DAML+OIL, OWL, and WSML ontologies. It

features a conjunctive query language called nRQL, by which numeric constraints, negation as failure, and substring properties are supported. It works on all platforms and provides proper APIs for Java and Common Lisp. It is free and open-source and employs very good optimization techniques.

Pellet [91] is a Java-based reasoner supporting RDF/RDFS, OWL, DAML-S, OWL-S, and WSML ontologies. It is open-source and can be used with OWL API and Jena. It works on all platforms and is one of the most comprehensive and mature reasoners, supporting SWRL rule reasoning [46].

Racer and Pellet completely support T-Box (terminology box including concepts, relationships, and constraints) and A-Box (assertion box containing assertions on individuals) reasoning [43, 92]; whereas, Jena and Jess conditionally support them [25, 40]. T-Box and A-Box (which form a knowledge base), the mechanism for reasoning on them, and the set of constructs for defining concepts characterize a Description Logic (DL) system. Description Logics are languages for formal representation of the knowledge in a domain [51], e.g., OWL is based on DLs.

I.3 Review methodology

This paper conducts a literature review based on the guidelines provided in [62, 63] to analyze conducted research efforts on SABAC.

In order to conduct a systematic literature review, the research questions need to be defined and motivated first. Then, a search strategy determining how to find relevant studies need to be developed. Next, inclusion and exclusion criteria for selecting the set of studies for the review should be specified.

I.3.1 Research questions

This paper analyzes, discusses, and compares the existing SABAC schemes based on the following research questions, which are accompanied by their objectives. The first question (**RQ1**) is the main driving force behind the lengthy Section I.4, where we give details for the major existing SABAC schemes.

RQ1 How is SABAC realized?

- It is essential to demonstrate the strategies by which semantic attribute-based access control can be achieved. It helps researchers and developers to have an overview of the existing strategies/solutions and adopt the best suitable strategy when proposing/developing an SABAC system.

RQ2 What is the main goal of using semantic technologies in SABAC schemes?

- Since semantic technologies may be used for various reasons (e.g., finding synonyms of attributes and improving the expressiveness

of the policy specification), it is important to determine the main applications of semantic technologies within SABAC schemes.

RQ3 Do the reviewed schemes have a formal specification?

- The aim is to check if an SABAC scheme (or a part of it) is formally verified. In order to formally verify properties such as liveness, deadlock-freeness, safety, security, correctness, completeness, etc., an access control scheme needs to be specified formally.

RQ4 How are SABAC policies specified?

- We identify whether policy specification languages for SABAC are extensions of existing ABAC or semantic languages, or new policy languages, and list their specific features.

RQ5 What is the target application domain?

- Each domain may have different requirements that need to be addressed when proposing a solution for them. This question identifies the target domain of each proposal.

RQ6 Is the performance evaluated?

- The performance evaluation helps to see the effect of semantic technologies on the performance, e.g., by comparing SABAC with ABAC.

RQ7 Which semantic technologies are used?

- We identify which languages and technologies of the semantic web stack are used (e.g., data modeling languages, rule markup languages, query languages, and inference engines).

RQ8 What kind of ontology is used?

- We clarify whether a new ontology is developed or an existing ontology is used. This question helps in determining the quality of the ontology used in each SABAC scheme. If an existing ontology (which is developed and evaluated correctly) is used, it has a positive effect on the performance and correctness of the whole scheme as the ontology is validated in advance. If a new ontology is developed, then this question determines whether it is based on an existing one and how it is evaluated (in terms of quality, performance, and usefulness). The question also identifies the methodology that is applied in developing the ontology.

RQ9 Does a prototype of the proposed SABAC scheme exist?

- Implementation helps to examine if an SABAC scheme can be realized in the real world. This question determines if there exists any tool demonstrating the proposal. Besides, it identifies whether the source code of the tool is publicly available (for the non-commercial ones).

RQ10 Is the scheme supported by a case study?

- Case studies help to gain a better and deep understanding of how a scheme would work in real-world scenarios.

RQ11 Does the scheme address conflict resolution?

- The aim is to determine the best ways to fix conflicts between semantic access control policies.

I.3.2 Selection process

To find the relevant research efforts, we considered Google Scholar, Springer, Elsevier Scopus, Web of Science, Science Direct, IEEE Xplore Digital Library, Citeseer library, and ACM digital library. Moreover, we used the following keywords and strings to search through the above-mentioned databases until May 2020: XACML, Semantics, Semantic Technologies, Ontology, ABAC, SABAC, and Attribute-Based Access Control and ((XACML OR ABAC OR Attribute-Based Access Control) AND (Semantic Web Technologies OR Semantic Technologies OR Semantics OR Ontology)).

Those studies that proposed an SABAC scheme and were published in peer-reviewed journals, book chapters, and conferences are included. Other studies that augmented XACML with semantic technologies to achieve a goal different than proposing an SABAC scheme (e.g., supporting RBAC, privacy, etc.), duplicate studies, non-English studies, and short papers are excluded.

Searching the specified strings and keywords in the selected databases resulted in 94 research studies. However, four of them were excluded as duplicate studies. The number of papers decreased to 24 after reading (full read) the 90 remaining papers. These 24 papers are used as the initial set for the snowballing process, described in [98]. Forward snowballing, which helps to identify more relevant papers by checking the studies that cited the papers in the initial set, led to 11 more papers. However, most of them were duplicates apart from two papers. Therefore, the focus of this paper is on 26 papers, published as journal papers, conference papers, or book chapters.

The selected papers are summarized, compared, and discussed in the following sections, ending up with a comparison given in Table 1. We summarize each paper by representing the main idea and the way that the proposed SABAC scheme works. Then, the selected papers are compared to each other (and discussed) based on the research questions (see Sec. I.3.1). Fig. I.4 shows the number of studies on SABAC per year.

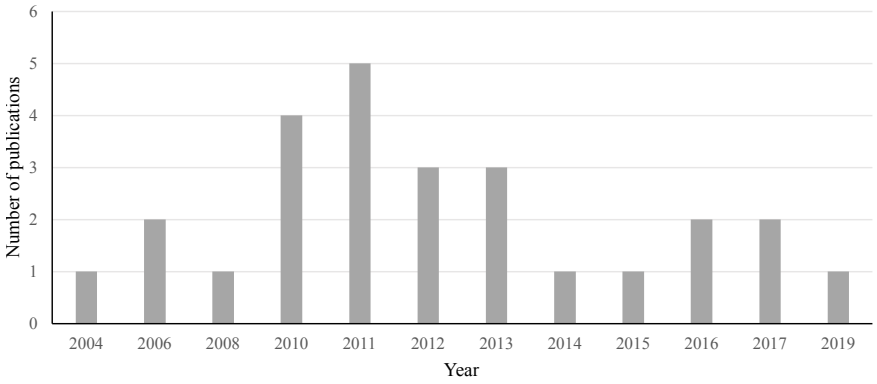


Figure I.4: The number of (included) publications per year.

I.4 Semantic Attribute-Based Access Control Schemes

This section reviews existing semantic attribute-based access control schemes, looking at their advantages and disadvantages, and describing their key features and designs. To allow for a comparative study, we present the different architectures in a unified manner, based on the reference architecture of the XACML standard from Fig. I.2.

Based on the literature review, the existing SABAC schemes can be categorized as *Extensions of XACML*, *New Policy Languages*, and *Hybrid Models*, as represented in Fig. I.5. The existing SABAC schemes can also be classified based on their decision-making engine and target domain as demonstrated in Fig. I.6.

In the following subsections, the SABAC schemes appear in chronological order, and a summary of their contributions is given in Table I.1.

I.4.1 Extensions of XACML

Several articles have proposed SABAC schemes as extensions of either the XACML *policy language* or the XACML *architecture*, e.g., by adding new elements to the language, adding new components to the architecture, or changing the functionality of the existing components.

The first work that augmented ABAC with semantic technologies was by Damiani et al. [29] in 2004, which extended the XACML policy language. Particularly, they extended the XACML *context* to include metadata associated with both subjects and objects. They modified the `AttributeValue` element (in the XACML policy language) to make it possible to use RDF assertions as a value for attributes. Accordingly, they modified the `MatchId` element, which specifies the matching function for attributes values (e.g., string-equal, string-regexp-match, rfc822Name-match, or anyURI-equal), by introducing a

new function, called metadataQuery. The goal of Damiani et al. was to enrich XACML policies by including ontology-based metadata associated with subjects and objects through the use of RDF assertions. Damiani et al.'s approach is based on an RDFS ontology for the healthcare domain; however, it is not specified how the ontology was developed and evaluated. Damiani et al.'s approach does not have any formal foundation and resolves conflicts at runtime (when evaluating an access request) and based on XACML combining algorithms. Damiani et al. did not provide a prototype of their proposed approach and accordingly, they did not evaluate its performance. Instead, they used a research example as a kind of case study to describe their proposal.

Another line of SABAC frameworks was started by Priebe et al. [81], who, instead, extended the XACML architecture. We have drawn in Fig. I.7 the two new components, i.e., an inference engine and an ontology administration point (OAP), that [81] added to the architecture of XACML. When the context handler receives a request from the PEP, it sends the attributes that exist in the request to a Jena inference engine. The inference engine combines the received attributes with an OWL ontology provided by the OAP and performs the reasoning process to find semantically relevant attributes. The context handler uses the inferred attributes (also called inferred implicit knowledge) obtained as answers to SPARQL queries to update the request. Next, the context handler sends the updated request to the PDP, which evaluates it and gives a response to the context handler and PEP. Priebe et al. provided a prototype of their proposal; however, they did not evaluate the performance.

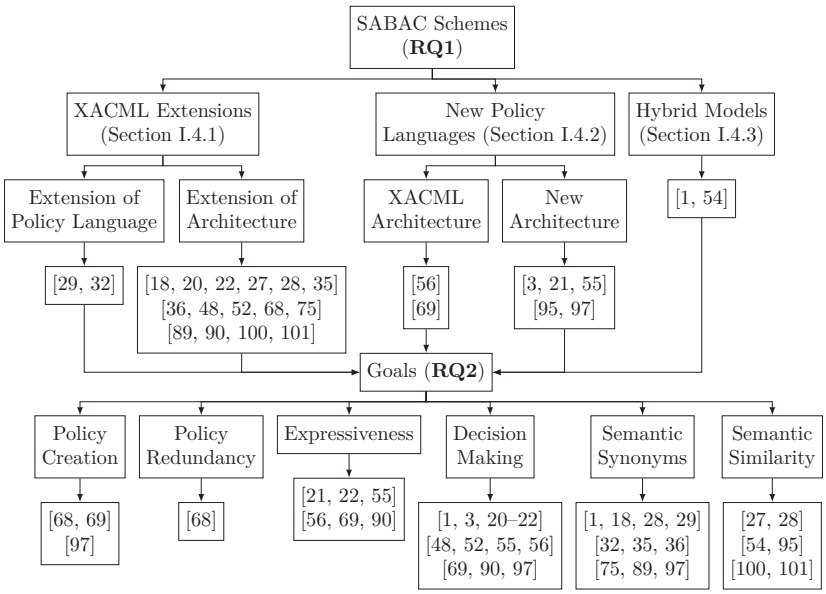


Figure I.5: A wide classification of SABAC schemes.

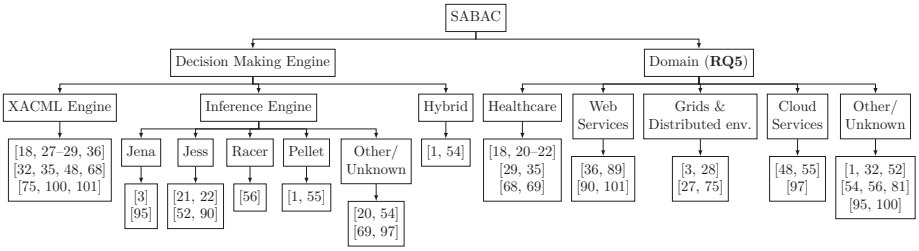


Figure I.6: The classification of SABAC schemes based on the target domain and decision making engine.

Priebe et al.'s scheme does not have any specific target domain. They employed a simple new ontology without mentioning the methodology used for developing it. Priebe et al.'s proposal uses standard XACML combining algorithms for conflict resolution.

In 2008, Muppavarapu and Chung [75] combined semantic technologies with the XACML standard for the good of improving the interoperability in Data Grids. They proposed an architecture having several components which are named differently than those of the XACML standard. However, the data flow, i.e., the mechanism for processing an access request, is the same as that of Priebe et al.'s scheme. Muppavarapu and Chung used a simple new OWL ontology along with SWRL rules (to specify more complex relationships between concepts in the ontology) to extend the list of attributes that users provide in access requests. The policy specification language was the XACML policy language and the conflict resolution strategy was the XACML standard's approach (i.e.,

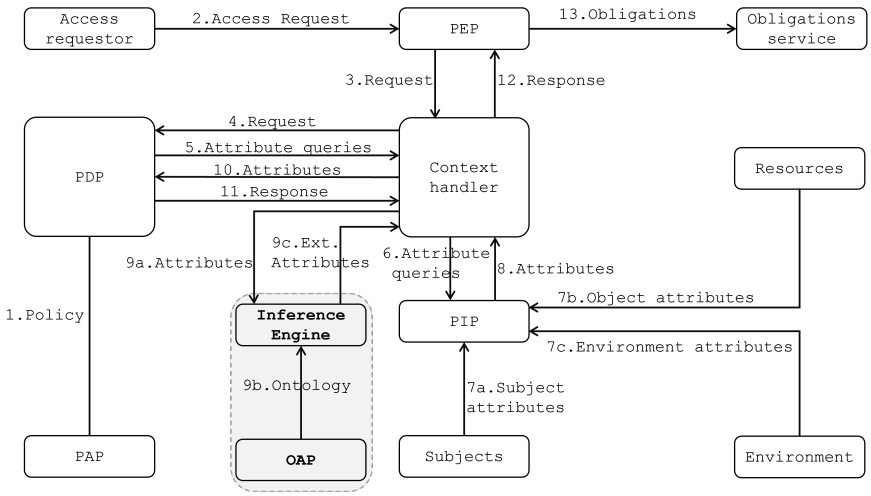


Figure I.7: The architecture of Priebe et al.'s scheme [81]

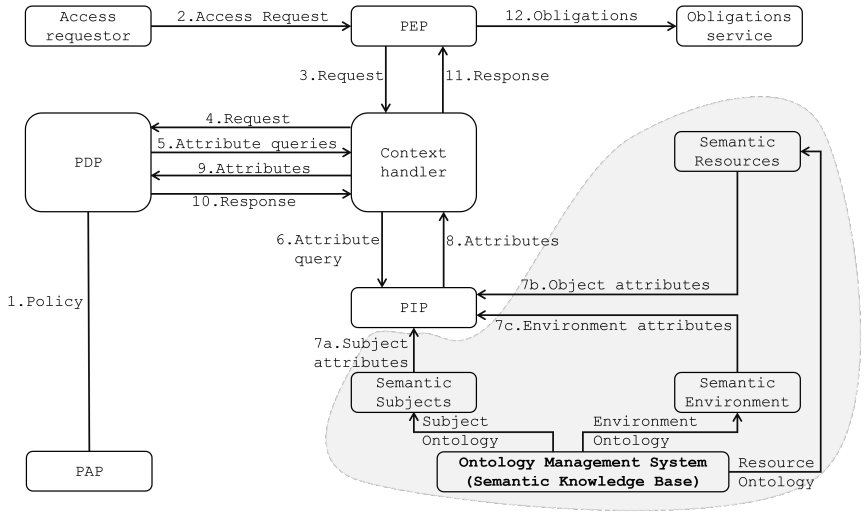


Figure I.8: The scheme proposed by Shen [89]

combining algorithms). Muppavarapu and Chung used Shibboleth [24], which is an attribute authorization service, to protect the privacy of subjects when collecting subjects attributes from the PIP. The case study of Muppavarapu and Chung's scheme was part of a project called OGSA-DAI[1]; however, no implementation was made.

Shen [89] also extended the XACML architecture by adding an ontology management system as shown in Fig. I.8. The new component is intended to create subject, object, and environment ontologies (based on OWL), analyze the created ontologies using an inference engine (i.e., using Jess and SWRL rules), and store the inferred knowledge to help the PDP to make semantic-aware decisions for web services. Shen's scheme uses the XACML policy language and XACML combining algorithms for the specification of access control policies and conflict resolution, respectively. It is not clear what methodology was used for developing and evaluating the ontologies. Besides, the performance of Shen's scheme was not evaluated as it was neither implemented nor supported by a case study. Shen's scheme is somehow similar to the scheme proposed by Priebe et al. [81]. However, Priebe et al.'s scheme performs semantic reasoning at the time of making a decision whereas Shen's scheme does that in advance (probably at the system initialization). Therefore, Shen's scheme makes an access decision much faster than Priebe et al.'s one.

Shen and Cheng [90] updated the scheme proposed in [89] by using SWRL as the policy specification language and replacing the ordinary XACML-based PDP with a Jess inference engine. They developed a simple OWL ontology and

[1]https://www.epcc.ed.ac.uk/projects-portfolio/ogsa-dai-solutions-distributed-data-access-and-management

used SPARQL to query the ontology. Unlike the scheme proposed in [89], the updated scheme does not utilize semantic technologies to only find synonyms of attributes. In the updated scheme, the policy language is changed to SWRL to enhance the expressiveness (of the policy specification), but conflict resolution is not addressed.

Durbeck et al. [36] proposed a semantic security architecture named Access-eGov Solution, with a preliminary version presented in [65]. As represented in Fig. I.9, the proposed architecture looks the same as the architecture proposed by Shen [89] since the semantic component/part is connected to the PIP (the target domain is also web services). However, the process for making an access decision is different. In Durbeck et al.'s scheme, first, the access requester, which acts on behalf of the subject, retrieves the access control policy concerning the requested resource from the PAP to find the list of required attributes for accessing the resource. Then, the access requester retrieves the required user attributes from the PIP. If there exists an attribute that the PIP does not know (since there may be semantically similar attributes with different names), the PIP sends that attribute to the inference engine to get the semantically relevant attributes based on the domain ontologies that the service provider and PIP use. After getting all the required subject attributes, the access requester sends an XACML request to the PDP. Next, the PDP may query the PIP for the values of the object, action, and environment attributes and then make a decision based on the received information. The PIP may call the inference engine again if required.

Durbeck et al. employed the existing ontologies for web services in their proposal, i.e., WSMO [85], which was modeled using the Web Service Modeling Language (WSML) [17]. The proposed architecture was supported by a case study from a project called Access-eGov[1]; however, it was not implemented and the performance was not evaluated.

In Durbeck et al.'s scheme, a subject gathers all the required subject attributes before sending an access request to the PDP (or PEP) to have a higher chance of getting access. However, it has a negative effect on the performance of the system as extra work needs to be done for every single request. The solution may not be practical if several policies exist for an object. Besides, retrieving a policy, which may contain information about the subject, object, action, or environment, by the access requester may cause leakage of information.

In 2010, Dersingh et al. [32] proposed an SABAC scheme for ubiquitous systems by extending both XACML policy language and architecture (a preliminary version appeared in [31]). On one hand, the policy specification language is extended to incorporate semantic contexts into access control policies. On the other hand, as represented in Fig. I.10, the architecture is extended by changing the functionality of the *Context Handler* and adding a semantic knowledge base, which is under the management of a separate system called the context management system. According to the extended architecture, when the context handler receives an access request from the PEP, it uses a method called

[1] https://cordis.europa.eu/project/id/027020

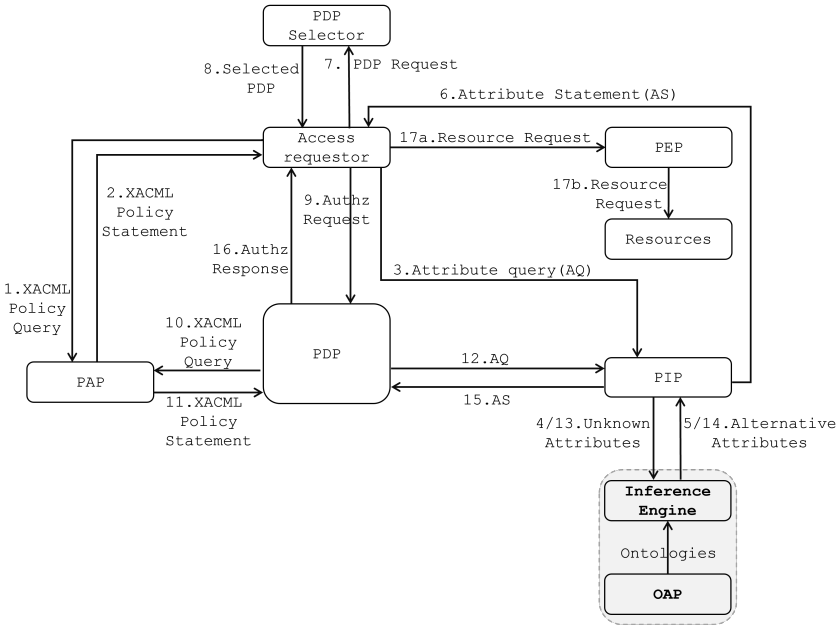


Figure I.9: Durbeck et al.'s scheme [36]

attribute finder (the added method) to obtain the required attributes/contexts from the semantic knowledge base (the added component). After retrieving the required attributes (which is a set of attributes enriched by means of semantic technologies), the context handler forms an XACML request and sends it to the PDP.

The separate context management system not only stores the semantic domain knowledge for the access control system but also is needed to classify and infer new knowledge when a change occurs in the domain. An advantage of such a separation is that it does not rely on a specific policy language, and any policy language supporting the notion of attributes can be used instead of the XACML policy language.

The approach proposed in [32] uses the Racer reasoner [43] as the description logic reasoning engine to classify knowledge (by reasoning over OWL ontologies) and the Jess rule engine to infer new knowledge by processing SWRL rules defined in the ontology over the knowledge that is classified by the Racer reasoner. Dersingh et al. defined a research example as a case study and implemented a prototype of their proposed scheme. However, they neither explained how the ontology was developed nor evaluated the performance of the proposed scheme. Dersingh et al.'s scheme addresses conflict resolution in the same way as the XACML standard, i.e., through combining algorithms.

In 2011, Calvillo et al. [22] proposed an SABAC framework based on the architecture of the XACML standard applied to the healthcare domain. The

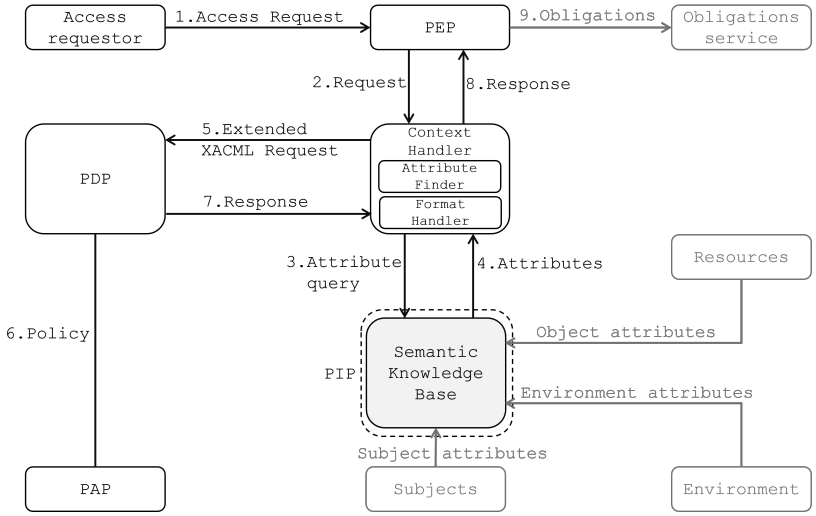


Figure I.10: Dersingh et al.'s scheme [32]

proposed framework includes new components for semantic management as drawn in Fig I.11. The PIP is decomposed into three different PIPs for the subject, object, and environment, and are connected to three related knowledge bases. The functionality of the PDP is also changed into an inference engine (i.e., Jess engine) that makes a decision based on the access control policies specified using SWRL and a sample OWL ontology developed (by Calvillo et al.) for the healthcare domain. When the context handler receives a request from the PEP, it notifies the PDP, which sends back attribute queries to the context handler. Next, the context handler collects the attributes utilizing the three different PIPs. Upon receiving the attributes from the context handler, the PDP uses the received information and the SWRL rules, which exist in the policy semantic knowledge base (i.e., the PAP), for semantic reasoning by the Jess inference engine. The PDP incorporates the inferred axioms to the ontology and then uses the SQWRL [76] to check whether the properties *"actionProhibited"* or *"actionPermitted"* exist between the requester and the requested object (in the ontology). In the case of the existence of the property *"actionProhibited"* (*"actionPermitted"*), the final decision is Deny (respectively Permit). In the case of the absence of both properties, which means there exists no applicable policy, the PDP makes a Deny decision and informs the administrator to define a policy for such an access request. Therefore, we can say that the framework of [22] employes semantic technologies to make decisions in addition to finding synonyms of attributes. Calvillo et al. implemented a prototype of their proposal for the PREDIRCAM[1] project; however, they did not evaluate its performance. Calvillo et al.'s framework resolves conflicts by making a Deny decision when

[1] http://www.gbt.tfo.upm.es/item310&highlight=PREDIRCAM

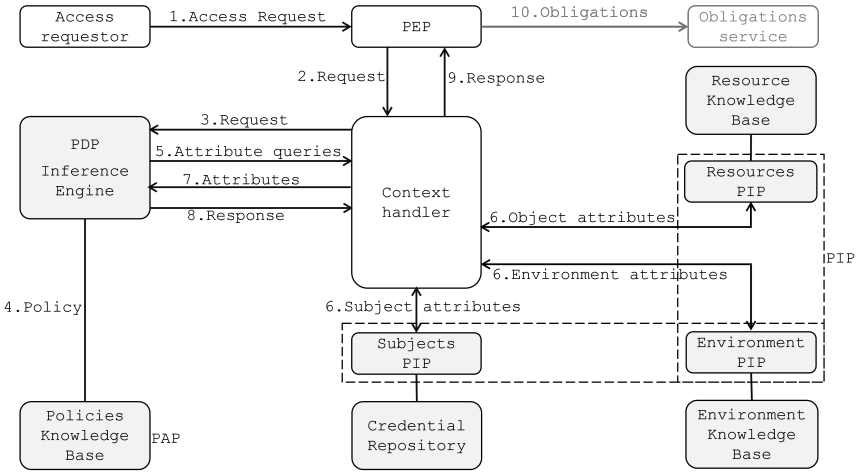


Figure I.11: Calvillo et al.'s scheme [22]

both *"actionProhibited"* and *"actionPermitted"* properties exist at the same time. Since the policy language is changed to SWRL, it can be said that the expressiveness of the policies is enhanced.

Brut et al. [18] added an ontology-based query rewriting mechanism to the XACML standard to enable pervasive healthcare. In Brut et al.'s scheme, a component called the Query Interpreter (QI), which acts as both the PEP and context handler, intercepts an access request and translates it into a normal XACML request. Then, the QI sends the XACML request to a component called the Query Analyzer (QA), i.e., the PDP. The QA evaluates the request and sends the decision back to the QI. If the decision is a Deny, then the QI calls the semantic similarity provider, which is a component added to the XACML architecture, to get similar concepts. It is supposed that a request contains the concepts from International Classification of Diseases, Version 10 (ICD-10) and International Classification of Primary Care, 2nd edition (ICPC-2) ontologies (which are OWL-based ontologies). Then, the QI modifies the access request by replacing attributes (i.e., ontological concepts) with similar ones (those received from the semantic similarity provider) and sends the modified request to the QA. If the QA makes the same denial decision for the modified request, the QI tries to modify the request one more time with the annotated information (it is supposed that the patients can annotate their medical records). Finally, the QA sends the initial Deny decision along with an alternative positive decision (if one of the modified requests gets approved) to the requester.

The main goal of Brut et al.'s scheme was to maximize data sharing while keeping an acceptable level of security. In other words, the aim was to provide medical staff as much as possible access to the patients' medical records in emergency cases. A good point about Brut et al.'s scheme is that existing ontologies, i.e., ICD-10 and ICPC-2, are used. However, the query rewriting

mechanism is quite simple and needs to be improved as only simple ontological relations, i.e., superclass and subclass, are considered. Furthermore, the proposed scheme was not implemented and it is not clear how the performance of a medical data system would be affected by the added query rewriting mechanism. Moreover, since Brut et al.'s scheme employs the XACML policy language, probably conflict resolution will be addressed in the same way as the XACML standard.

Ciuciu et al. [27] addressed the interoperability in distributed environments by augmenting the ABAC with an Ontology Based Interoperation service (OBIS). The proposed architecture, as shown in Fig. I.12, differs from the reference architecture of the XACML standard. Instead of one PDP, several PDPs are employed, i.e., a Master PDP and some local PDPs, which process policies specified in either XACML or PERMIS [26]. The PEP is also doubled as there exists an Application Independent Policy Enforcement Point (AIPEP) in addition to the ordinary PEP. The PEP (the one controlling access to the requested object) intercepts an access request and sends the request to the AIPEP. The attributes in the received request will be validated by means of a Credential Validation Service (CVS). If there is an unknown attribute, then the CVS invokes the OBIS to check how the unknown attribute is related to a known one based on an existing ontology, SecPODE [84]. Then, the CVS sends a validated set of attributes back to the AIPEP. Receiving the set of attributes, the AIPEP sends the request to the Master PDP, which forwards it to a set of local PDPs. If a local PDP cannot make a decision, it may call OBIS to get the semantic similarity value of the received attributes and those of the applicable policies and then makes a proper decision based on the received semantic similarity values.

Ciuciu et al.'s scheme was not implemented and its performance was not evaluated though its case study was the TAS^3 project [1]. Ciuciu et al. did not address conflict resolution explicitly, but it can be assumed that the XACML combining algorithms are used as the policy language is XACML.

In [28], Ciuciu et al. extended their approach from [27] by adding new functionality to the OBIS. The extended OBIS, which is called OBIS Domain Mapper, makes it possible to translate attributes from one domain (e.g., a local PDP) to another domain (the master PDP). Such a translation can be done by means of different ontologies.

In 2012, Zhao and Wang [101] extended the XACML standard with a semantic similarity algorithm and domain ontologies, as shown in Fig. I.13. In Zhao and Wang's scheme, which was proposed for web services, both the application (requester) and provider have their own domain ontologies and the PDP makes a decision based on the semantic similarity of the attributes provided by the requester and attributes required by the provider (the provider's domain ontology shows the attributes required for objects). Zhao and Wang's scheme works as follows: a requester sends its attributes to the PEP. The PEP, which acts as a context handler as well, forms an XACML request and sends it to the

[1] https://cordis.europa.eu/project/id/216287

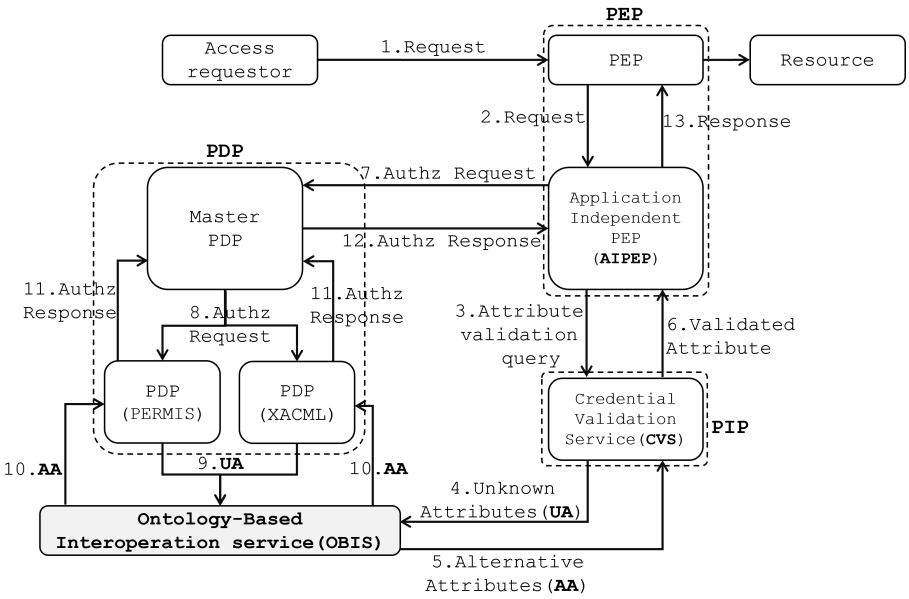


Figure I.12: Ciuciu et al.'s scheme [27]

PDP. The PDP gets the attributes that the provider requires for the requested object through the PIP (and provider's domain ontology). Next, the PDP sends a request to the semantic component to check if these two sets of attributes (i.e., the ones provided by the requester and provider) are similar. The semantic component checks its mapping base, which keeps the results of previous mapping attempts for the sake of efficiency, if the same request was evaluated before. If there is no match in the mapping base, the semantic similarity between attributes (attributes provided by the requester and those required by the provider) will be calculated and the results will be stored in the mapping base for future uses. The PDP grants access if the requester's attributes are similar to those required by the provider. It is not clear what semantic technologies, e.g., ontologies and the inference engine, were used in Zhao and Wang's scheme. Moreover, they did not implement a prototype of their proposed scheme, and accordingly, its performance was not validated. Although Zhao and Wang did not address conflict resolution, it can be assumed that their scheme relies on the XACML combining algorithms.

Zhang et al. [100] also aimed to facilitate multi-domain interoperability by means of an ontology and an attribute mapping mechanism. In Zhang et al.'s scheme, which we draw in Fig. I.14, each domain has its own ontology developed by the experts of the domain. However, it is not clear based on what methodology they developed ontologies and how they validated them. The ontology (which is OWL-based) is an attribute library for access control elements. If a user (subject)

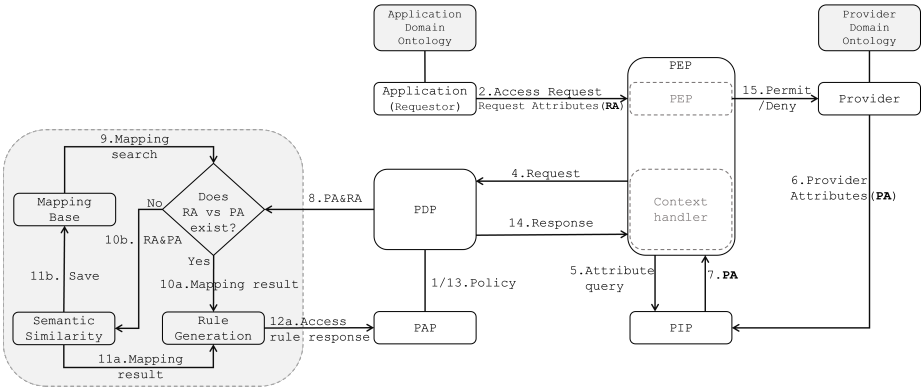


Figure I.13: Zhao and Wang's SABAC [101]

from a domain, say domain A, wants to access an object of another domain, say domain B, Zhang et al.'s scheme works as follows: the PEP (and probably the context handler) sends the request to the PDP of the domain B. As the received request is a cross-domain request, the PDP asks the semantic component (which includes a domain ontology and an attribute mapping mechanism) to map the attributes in the request to the related ones in domain B. The attributes related to the subject and object in domain B will be extracted from domain B's ontology using Jena API. If the similarity value of each attribute is more than a predefined threshold, then the semantic component maps the attributes in the request to the local attributes and sends the local ones to the PDP. Next, the PDP can make a proper decision based on the local attributes and policies. If the semantic component cannot map the provided attributes to the local ones, then the request would be rejected. The idea, which is based on the semantic similarity of attributes, is straightforward. However, the accuracy of the attribute mapping is questionable since the algorithm for calculating the attribute similarity is not mature enough. In addition, the proposed scheme was not implemented and not supported by a case study, and the performance was not evaluated thoroughly. We assume that Zhang et al.'s scheme also addresses conflict resolution in the same way as the XACML standard utilizing the XACML combining algorithms.

In 2013, Hsu [52] proposed an SABAC scheme by extending the XACML architecture as follows. The context handler is replaced with a transformation engine, which is responsible for the transformation from one format to another (i.e., XACML → SWRL → Jess facts and vice versa). The PDP is replaced with a Jess inference engine. The PAP contains *annotated* XACML policies as well as the normal ones. Instead of a PIP, there exists an *XML-based repository* including an ontology base, an SWRL base, and a style sheet base.

Hsu's scheme works based on annotated XACML requests and policies. Hence, first, the policy administrator manually annotates the XACML policies. In XACML, each attribute has a data type that is defined in XACML, XPath [13], or XML Scheme [37]. The data type of attributes is used for annotating the XACML

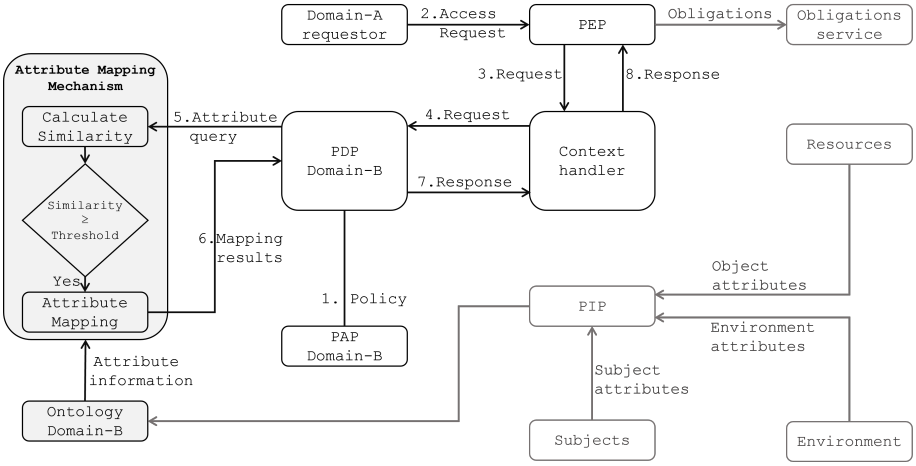


Figure I.14: Zhang et al.'s scheme [100]

policies and requests (by mapping to concepts in the ontology). The policy administrator finds the relevant classes (for the subject and object attributes) in the ontology (using the subject ID and resource ID that exist in each policy) and then sets the Uniform Resource Identifier (URI) of the found concepts as the value of the data type of the relevant attributes. When all policies are annotated, the transformation engine converts them to SWRL rules (using the XACML2SWRL.xsl XSLT style sheet) and stores the obtained rules into the SWRL base.

Hsu's scheme works as follows: when a PEP receives an access request (which is annotated), it forwards the request to the transformation engine, which acts as the context handler. The transformation engine finds all the instances of the subject and object attributes in the ontology (based on the subject ID and resource ID that exist in the request) and transforms them into Jess facts. The transformation engine also finds the relevant ontologies (in the ontology base) using the DataType of the subject and object in the request. Next, it converts the relevant ontologies (which are OWL-based) to Jess rules using another XSLT style sheet called OWL2Jess.xsl. This transformation extends OWL models via rules. It also uses the found ontologies to find the relevant SWRL rules (in the SWRL base) and then converts them to the Jess rules using the SWRL2JESS.xsl XSLT style sheet. Then, the Jess inference engine uses the Jess rules and facts to infer new Jess facts, which show the decision for the received request. However, the decision needs to be returned in the XACML format. Hence, the transformation engine (i.e., the context handler) converts the Jess facts, which are received from the inference engine, into the XACML format using JESS2XACML.xsl XSLT style sheet and then sends the response to the PEP.

Hsu defined a research example as a case study and implemented a prototype of his proposed scheme to evaluate the performance. However, it is not clear

what happens if the inferred new Jess facts, which show the access decision, are in conflict with each other, i.e., they show both Deny and Permit for the same request.

In 2014, Calvillo-Arbizu et al. [20] proposed an SABAC scheme addressing the interoperability in the healthcare domain by means of an ontology of attributes (concepts in the ontology represent attributes). The idea was to have an attribute ontology as a reference ontology. The context handler keeps and uses the reference ontology. When there is an access request, the context handler collects the required attribute values from information providers (the PIP) and translates the collected attribute values to the reference ontology through mapping rules. Next, it sends the request to the PDP, which is an inference engine. The PDP retrieves applicable policies from a policy provider (the PAP), converts the retrieved policies into SWRL rules, and then makes a decision based on the SWRL rules and attributes. The PDP solves conflicting decisions by choosing the most conservative decision. Calvillo-Arbizu et al.'s scheme was neither supported by a case study nor implemented to evaluate its performance. They did not use a standard methodology for developing the attribute ontology. Besides, it is not clear what inference engine (and how) converts the XACML policies into SWRL rules.

In 2016, Drozdowicz et al. [35] extended the XACML architecture by replacing the PIP with another component called SemanticPIP (first proposed in [34]). The new component is an integration of the PIP and the HL7 security and privacy ontology [12], which describes and interrelates important security and privacy concepts in the e-Health domain. The SemanticPIP functions as follows: when the context handler queries for attribute values, the SemanticPIP creates OWL concepts and OWL individuals for the subject, object, action, and environment attributes that exist in the request. It also converts the attribute values existing in the request to data property axioms of the appropriate types. In other words, it translates an access request to an ontology. Then, it merges the constructed ontology with the domain ontology and uses a semantic reasoner to obtain extra information (semantically relevant attributes). After that, it retrieves the attribute values by issuing SPARQL queries on the ontology for the properties recognized by the reasoner. Finally, it returns the retrieved attribute values to the context handler. The context handler updates the request by adding the inferred information and sends it to the PDP, which makes a decision based on the XACML policies and the updated request.

Drozdowicz et al. implemented a prototype of their proposed scheme and provided a research example to show that the proposed scheme works properly. However, the performance was not evaluated and the semantic reasoner was not specified. Although the strategy for conflict resolution was not determined, we assume Drozdowicz et al.'s scheme addresses conflict resolution in the same way as the XACML standard. Two different applications of the scheme proposed in [35] are presented in [33, 93].

Liu and Wang [68] proposed an access control scheme called FCAC for linked health data. FCAC employs XACML for specifying access control policies and

SWRL for expressing complex semantic relationships. FCAC employs four different ontologies, i.e., the subject, object, action, and environment ontologies. The action and environment ontologies, which most of the existing SABAC schemes lack, enhance the semantic awareness of the access control system. In FCAC, semantic technologies, i.e., OWL-based ontologies, SWRL rules, and Jess semantic reasoner, are employed to manage access control policies. FCAC reduces the number of XACML policies based on the results of the semantic reasoning on the ontologies and SWRL rules. Semantic reasoning helps to detect rules which are semantically similar and remove redundant and conflicting rules/policies. Liu and Wang implemented a prototype of FCAC and evaluated its performance thoroughly based on a dataset taken from Linked Clinical Trials [47]. They formally specified the basic elements of their proposal; however, it is not clear if they developed (and evaluated) the ontologies based on standard methodologies.

In 2017, Hilia et al. [48] augmented the XACML architecture with a module called semantic finder having two functions namely *attribute finder* and *policy finder*. The goal was to enable contextual access control in cloud environments as a part of the ComVantage project[1]. The semantic finders are used to obtain contextual (semantic) information from various sources (that are usually heterogeneous) such as local RDF files, web services, semantic reasoners, PAP, and SPARQL endpoints. The attribute finder finds attributes such as physical measures and environment variables, and policies corresponding to a given context can be found using the policy finder.

In Hilia et al.'s scheme, the RDF data model is used to store the data (as RDF triples). The PDP retrieves the required attribute values from PIPs using the semantic finders, which is in contrast to the XACML standard where the context handler is responsible for retrieving attribute values. Conflict resolution is not addressed in Hilia et al.'s scheme; however, since the policy language is XACML, we assume their scheme resolves conflicts using the XACML combining algorithms.

Hilia et al. implemented a prototype of the proposed scheme; however, they did not evaluate the performance of their proposal.

I.4.2 New policy languages

Another line of research proposes new policy specification languages that incorporate semantic technologies.

In 2006, Jin and Fang-Chun [56] proposed an ontology-based ABAC by means of description logics to decrease the complexity of the policy specification and improve interoperability. The main idea was to specify access control policies and represent attributes using a description logic language. They used the restricted $ALL(D)$ [9] description logic language as the basis for expressing access control policies and attributes (of the subject and environment). The $ALL(D)$-based access control policies and attributes form the TBox and ABox

[1] https://cordis.europa.eu/project/id/284928

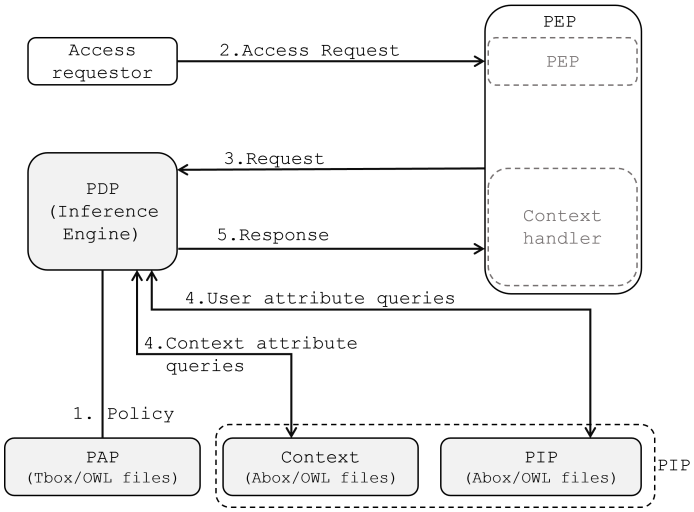


Figure I.15: Jin and Fang-Chun's scheme [56]

of the DL knowledge base, respectively. An inference engine, i.e., Racer reasoner, is used for making access decisions and checking the consistency of access control policies. As shown in Fig. I.15, Jin and Fang-Chun's scheme processes access requests as follows: the PEP sends an access request to the PDP, which is a Racer reasoner, for the evaluation. Receiving a request, the PDP sends queries to the PIP and the context component to obtain the subject and environment (context) attributes, respectively. The PIP and context component issue assertions about the values of the requested attributes and send them to the PDP, which makes a decision by means of the reasoning process based on the received assertions and the policies, which are represented using TBox axioms of the knowledge base.

Jin and Fang-Chun claimed that they implemented their proposed scheme; however, they neither provided any detail about that nor evaluated the performance of their proposal. Besides, they did not address conflict resolution. They might rely on the consistency check done by the Racer reasoner. However, it is not clear how the reasoner can find and resolve conflicts between access control policies for different possible requests (with different sets of attributes). Though Jin and Fang-Chun's scheme improves the expressiveness of the access control policies, it might be difficult to specify very complex policies without losing the decidability.

In 2010, Amini and Jalili [3] proposed an SABAC model for distributed environments based on the $MA(DL)^2$ logic [2], which is a combination of deontic and description logics and supports the specification and inference of access control policies. Accordingly, they suggested an architecture based on the XACML standard as represented in Fig. I.16. In a multi-domain setting, each domain has a PEP, a PDP, a PAP, an $MA(DL)^2$ knowledge base, a context handler, and a credential verifier, where the PDP is an $MA(DL)^2$ inference

engine (Jena is used for reasoning over TBox, i.e., subsumption inference). The $MA(DL)^2$ knowledge base includes access control policies, assertions about the individuals, the current information about the context, and subjects, objects, and actions ontologies. The PAP of a domain may use the policies of other domains (which are kept by other PAPs). The credential verifier is used to verify the validity of the credentials existing in access requests. The context handler is used to gather the required context information from context sensors, create contextual propositions about them, and add the propositions to the knowledge base.

Amini and Jalili's scheme works as follows: when a PEP receives an access request, it checks if the request is about a valid action on an object that is registered in its domain (based on the description of the object that exists in the OWL-S ontology [71]). It also verifies the attributes provided in the request through the credential verifier. If all the checks go through, the PEP forwards the request to the PDP. Then, the PDP uses the credentials (attributes) existing in the request to generate and add more assertions to the ABox of the $MA(DL)^2$ knowledge base. The PDP also updates the context information of the knowledge base utilizing the context handler. Next, the PDP checks the conflicts that may exist between access control policies and removes the conflicting policies from the knowledge base (temporarily). After that, the PDP makes a decision in a two-stage procedure: 1) make a ground-level decision based on the individual data (based on the attributes provided in the original request) and 2) make a conceptual-level decision based on the relationships between ontological concepts (based on semantic relationships). Finally, after making a decision, the PDP deletes the added contextual facts and assertions from the knowledge base. It also adds the deleted conflicting policies to the knowledge base again.

Amini and Jalili provided a formal specification of their proposed authorization model. They also implemented a prototype of their proposed framework and evaluated its performance on a case study (which was a research example defined by them). They developed sample ontologies describing subjects, objects, and actions. The results of the performance evaluation demonstrated that real-time reasoning was time-consuming. Hence, they suggested remedies such as benefiting from parallelization in tableaux system and offline (in advance) materialization of inferred relations. They addressed conflict resolution using an approach that works based on the notion of potential conflict graph.

In 2013, Trivellato et al. [95] proposed an ontology-based context-aware policy specification language called POLIPO inspired from Datalog with constraints [67]. A POLIPO policy includes a set of Horn clauses as follows: $H \leftarrow B_1, \ldots, B_n, c$.

The consequent (head) part, H, determines the decision based on positive or negative atoms, $B_1, \ldots, B_n$, and constraints, c, specified in the antecedent (body) part.

A policy can be constructed using the following elements:

- *Authorization atoms*, representing subject permissions, `perm(S, A, O)`, where `S`, `A`, and `O` denote, respectively, the unique identifier of a subject,

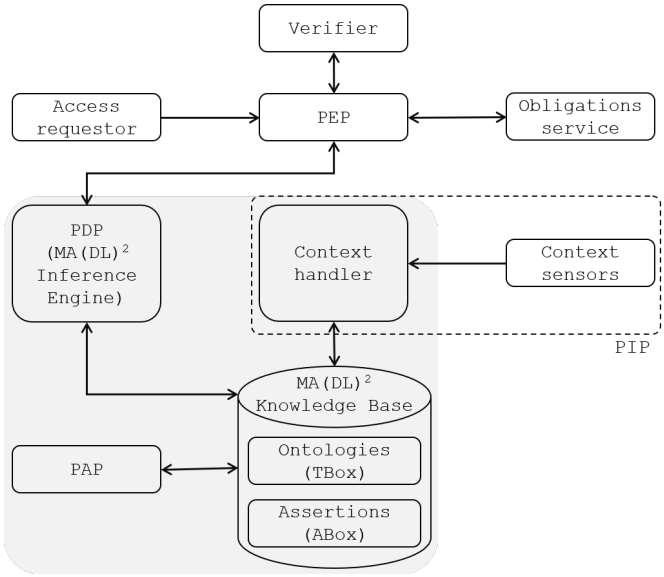


Figure I.16: The framework of the SABAC model proposed in [3]

the URI of an action in an ontology, and the identifier of an object.

- *Credential atoms*, representing certificates issued for attributes of subjects , `cred(I, Att, S)`, where `I` and `S` represent unique identifiers of the issuer and subject, respectively, and `Att` represents the URI of an attribute in an ontology.
- *Ontology atoms*, representing a concept `cURI(ia)` or a relationship `rURI(ib, ic)` in an ontology. Ontology atoms can be used to query the knowledge base. `cURI(ia)` and `rURI(ib, ic)` hold if `ia` is an instance of the concept `cURI`, and the instance `ib` has a relationship `rURI` with the instance `ic` in the ontology.
- *Constraints*, representing conditions through quantifier-free formulae.

The following provides an example policy and its specification using POLIPO:

"Researchers working on a European project can access public information about all ongoing research projects in Europe."

Ontologies:
prefix(Onto1, http://localhost/ontologies/Onto1).
prefix(Onto2, http://localhost/ontologies/Onto2).

Authorizations:
`perm(X,Onto2:Read,Y)`←`cred(EU,Onto1:EU_Project,Z)`, `cred(Z, Onto1:Partner,X)`,`not(Onto1:hasClassification(Y,secret) )`,`cred(EU,Onto1:EU_Project,Y)`

They also proposed a framework, which is composed of a PEP, two PDPs for the access control and trust management (i.e., AC PDP and TM PDP, respectively), a PAP, a semantic alignment evaluator (to align different ontologies), and a knowledge base, which is connected to several ontologies. There are two types of requests: credential requests (specified in SAML) and access requests (specified in XACML).

Trivellato et al.'s framework is a combination of the trust management and context-aware access control as follows: when a PEP receives an access request, it forwards the request to the AC PDP, which retrieves the authorization clauses related to the request (through the PAP). If more credentials are required to make an access decision, the PEP gathers the missing credentials from the requester or a third party and then sends them to the TM PDP. The TM PDP retrieves the relevant credential clauses from the PAP, evaluates the received credentials using the GEM algorithm [94], and sends the verified credentials to the AC PDP through the PEP. Next, the AC PDP, which is a Jena inference engine, makes an access decision and forwards it to the PEP to be enforced. The knowledge base service and semantic alignment evaluator can be used for retrieving the relevant domain and context information and calculating the similarity between concepts and instances of different ontologies.

Trivellato et al.'s framework makes access decisions based on only the subject attributes and context information, and does not consider attributes of objects and actions, which are also important when making a decision. Trivellato et al. implemented a prototype of their proposed framework (for systems-of-systems environments) and used existing RDF-based ontologies like SEM [45]. Their case study was EU NAVFOR[1]. They neither evaluated the performance of their proposal nor addressed conflict resolution.

In [21], access control policies are specified using SWRL rules, and access decisions are made using a Jess inference engine based on OWL ontologies from the healthcare domain. In order to improve the usability of the proposed approach, a metamodel and an application called Me-As-An-Admin (M3A) were developed by which people can specify their intended access control policies easily. A user can easily drag and drop elements of a policy as defined in the metamodel, and then the M3A editor transforms such user-defined policies into SWRL rules. The produced SWRL rules are stored into the knowledge base for evaluation of access requests.

The access control scheme proposed in [21] works as follows: receiving a request, the PDP combines a developed ontology called POVO with the SWRL rules applicable to the received request and performs the reasoning process (PDP is a Jess inference engine). Next, the axioms that are inferred (as a result of the reasoning process) will be incorporated into the POVO ontology. After that, the PDP queries the ontology using SQWRL to check the existence of either Permit or Deny properties between the requester and the requested object in the ontology. If it finds only a Permit property between the requester and the requested object, it grants the access. Otherwise, the access request will

[1] https://eunavfor.eu/

be rejected. Therefore, conflict resolution is addressed by choosing the most protective decision, Deny. A prototype of the proposed scheme was implemented. However, it was neither supported by a case study nor validated by a performance evaluation.

Iqbal and Noll [55] proposed a semantic-attribute based access manager for cloud services. Their proposed access manager has three different components, i.e., a Token Generator, a Token Validator, and a PDP. The access manager is also connected to an attribute knowledge base, which contains attributes represented using the Uniform Resource Identifier (URI) and RDF. The main goal of representing attributes using URI and RDF (in an ontology) was to make them globally identifiable and improve interoperability. The Token Generator issues tokens to subjects. A token includes a set of attributes and a subject holds a token. A token can be validated using the Token Validator. The PDP makes access decisions based on tokens and policies. In order to improve the expressiveness, the SWRL is used for the specification of policies. The PDP is a Pellet reasoner and checks if all the conditions of the antecedent part of applicable SWRL rules are satisfied based on the received token (attributes). Iqbal and Noll implemented their proposed access manager without evaluating its performance (and without any case study). They developed an OWL ontology of access control concepts, which needs to be extended and evaluated based on standard methodologies. Iqbal and Noll's access manager does not address conflict resolution.

Lu et al. [69] utilized semantic technologies for specifying access control policies, making access control decisions (by checking the compliance between access requests and access control policies), and discovering knowledge. They demonstrated that policy concepts (XACML policy elements) can be formalized in OWL and then policies can be dynamically generated using inference results. In other words, by using semantic technologies, it is possible to generate dynamic policies by associating and dissociating policy elements. Lu et al. provided several semantic rules by which access control policies can be defined based on the existing elements in XACML. They also demonstrated that checking the compliance between access requests and access control policies can be done using semantic technologies by modeling the access request using semantic concepts.

Lu et al. employed a case study associated with ADDN[1] and AURIN[2] projects to validate their proposal in the healthcare domain. However, they neither implemented their proposed scheme nor evaluated its performance.

The policy specification language in Lu et al.'s scheme is SWRL, which enhances the expressiveness. However, decidability issues arise with complex and several access control policies. Besides, Lu et al.'s scheme does not address conflict resolution. Lu et al. used OWL ontologies; however, they did not provide any information about the employed ontologies and inference engine.

[1] http://www.addn.org.au/
[2] http://www.aurin.org.au/

In 2019, Verginadis et al. [97] proposed an innovative security-by-design framework called PaaSword, which extends the XACML with semantic technologies to support the federation of access control policies in cloud environments. The semantic technologies are used to (1) infer new knowledge from the contextual information (attributes) in access requests, (2) check the consistency of policies, and (3) make access decisions. PaaSword provides a toolset allowing developers to specify access control policies, and a middleware for managing (including enforcing) the specified policies.

PaaSword employs a generic policy model, which allows the specification of access control policies using ontological concepts. In this model, a rule (a policy is a set of rules) can be specified using the following template:

```
[actor] has [authorization] for [action] on [controlled
object] when [context expression]
```

The template comprises of `actor` (the subject), `authorization` (the effect of a rule, e.g., Permit or Deny), `action`, `controlled object`, and `context expression` (the environment attributes) ontological concepts that allow semantic representation of access control policies. In other words, the template models the knowledge that exists in the access control policies, which in turn makes the management of policies easy. Semantic reasoning can be used for the detection of any inconsistencies between policies.

PaaSword employs an inference engine as the PDP that makes decisions based on the existing policies, and facts and assertions about the attributes provided in access requests. A prototype of PaaSword was implemented and its performance, in terms of the policy evaluation time and RAM consumption, was evaluated thoroughly. However, it was not supported by a case study. PaaSword's policies are based on ontological concepts; however, it is not clear what ontology was used for the performance evaluation.

I.4.3 Hybrid models

Another line of research combines two different access control models, i.e., an ABAC and a SBAC. SBAC is an access control model that works based on semantic technologies. In SBAC, all the entities in a given domain are modeled in subject, object, and action ontologies (all these can also be modeled in a single ontology). The subject ontology represents subjects and relationships between them, e.g., a patient has a GP, which is a medical staff. Similarly, the object ontology defines objects and relationships between them (e.g., credit cards and debit cards are bank cards), and the action ontology demonstrates different actions and the relationship between them, e.g., read, access, and open are the same. SBAC evaluates an access request using a semantic reasoner based on the ontologies and SWRL rules, which define more complex relationships between the involved entities.

In 2015, an SABAC scheme [1] was proposed by a combination of the ABAC and SBAC models for virtual organizations (distributed environments).

The goal was to take advantage of both ABAC and Semantic-Based Access Control (SBAC) models.

The scheme proposed in [1] is a two-stage access control model. In the first stage, which is used for controlling access inside each organization (i.e., intra-organizational access control), an ABAC is used. Hence, the policy administrator of each organization defines the XACML policies in the authorization server of their organization. In the second stage, an SBAC model is employed, where concepts, individuals, and taxonomies are defined in an OWL-based ontology, and access control policies are specified using SWRL rules. The ontology used in the second stage has two basic concepts (Subject and Object) and two basic relations (Permission and Prohibition). The policy administrators can use this ontology to specify high-level access control policies without considering the details of objects in each organization.

The ontology of subjects, which is a part of the ontology used in the second stage to represent subjects attributes, is also used for inferring implicit information (i.e., finding semantic synonyms of attributes in the first stage). Therefore, the ABAC model used in the first stage is enhanced by semantic technologies. However, the enhanced ABAC model uses the ontology of only subjects and not objects, actions, or environment.

As illustrated in Fig. I.17, the SABAC scheme proposed in [1] works as follows: when a PEP receives a request, it forwards the request to the PDP, which belongs to the same organization. Upon receiving the request, the local PDP evaluates the request (using the ABAC, which is extended by the ontology of subjects and an inference engine) and sends the request to the global PDP (there exists one PDP for all organizations that evaluates requests based on the SBAC model). The global PDP, which is a Pellet inference engine, also evaluates the request and returns the decision to the local PDP. Finally, the local PDP makes a final decision based on the first and second stages decisions and using XACML combining algorithms (to manage conflicting decisions). The authors provided the formal specification of the basic elements of their proposed SABAC scheme. A prototype of this scheme was implemented to evaluate its performance based on a case study which was a research example.

In [54], Husain et al. proposed an SABAC scheme for Geo-spatial domains. They combined the ABAC with a domain-specific ontology (an ontology for subjects, objects, and actions in the domain) and an algorithm, which was also proposed in [54]. The ontology contains and defines all the concepts that exist in all collaborating organizations. Husain et al.'s scheme introduces a new effect for the rules (in addition to Deny and Permit effects) called *Partial Permit*, which grants an access request *partially* and is suitable for the Geo-spatial domain. For example, a requester may request to view the map of a specific area. However, some parts of that area may be protected zones that should not be presented to everyone. Having only Deny and Permit rule effects, the request will be rejected; however, by the new *Partial Permit* rule effect, the requester can view the map of the area except the protected zones.

In Husain et al.'s scheme, when a PDP receives a request, it checks the

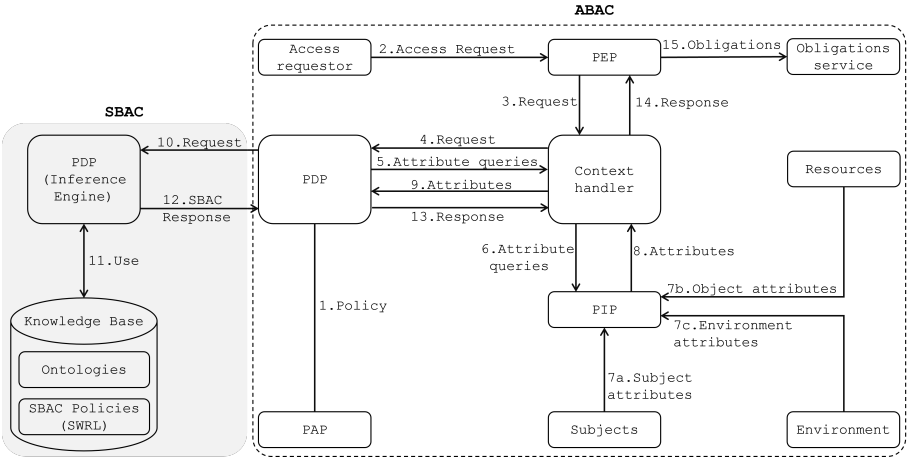


Figure I.17: A hybrid SABAC scheme

semantic difference between the subjects in both the request and policies and then selects the rule(s) whose subject has the minimum semantic difference with that of the request (in the ontology). If there exist several rules with the same semantic difference, it checks the semantic difference between the objects of the request and the selected rules and does the same as before. In the case of the existence of more rules, it does the same for the actions in both the request and selected rules (policies). If two or more rules are selected, then the XACML combining algorithms will be used to make a decision as follows. After selecting the applicable rules, an algorithm calculates the matching similarity score between the attributes in the received request and selected rule(s). If the score is greater than a predefined threshold (*full-effect threshold*), the decision would be the effect that is specified in the rule (the XACML policy). However, if it is greater than another predefined threshold (*partial-effect threshold*), it returns a *Partial Permit*. Otherwise, if it is smaller than the *partial-effect threshold*, it returns a Deny.

Husain et al. created a simple domain ontology and implemented a prototype of their proposal to validate it based on DFW maps[1]. However, they did not provide any detail about the ontology, inference engine, and performance of the proposed scheme.

I.5 Discussion

This section compares and discusses the existing SABAC schemes based on the research questions given in Section I.3.1. Based on these discussions, we extract five properties for, what we call, *the ideal SABAC*, which are denoted as **P1** to

[1]https://www.dfwmaps.com/

Table I.1: A summary of the contributions of the SABAC schemes.

Schemes	Contributions
Damiani et al. [29]	Extending the XACML policy language by modifying the `AttributeValue` (to use RDF assertions as value for attributes) and `MatchId` (by introducing a new function, called metadataQuery) elements.
Priebe et al. [81]	Adding and connecting a semantic component, which includes an inference engine and an ontology administration point, to the *Context Handler* (of the XACML architecture) as shown in Fig. I.7.
Muppavarapu & Chung [75]	Extending the list of attributes provided in access requests using a semantic component similar to [81].
Shen [89]	Adding and connecting a semantic component, which includes an inference engine and subject, object, and environment ontologies, to the PIP as shown in Fig. I.8.
Durbeck et al. [36]	Adding and connecting a semantic component to the PIP similar to [89] but with a different process for making decisions as represented in Fig. I.9.
Dersingh et al. [32]	Extending the XACML policy language to incorporate semantic contexts into policies. Changing the functionality of the *Context Handler* and replacing the PIP with a semantic knowledge base (Fig. I.10).
Calvillo et al. [22]	Decomposing PIP into three PIPs that are connected to three related knowledge bases (Fig I.11). Replacing the PDP with an inference engine. Specifying policies using SWRL.
Brut et al. [18]	Adding a *semantic similarity provider* to the XACML architecture. A component called the Query Interpreter (acting as PEP and *Context Handler*) may update a request (by replacing attributes with similar ones) a few times to get a positive decision from a component called the Query Analyzer, which acts as the PDP.
Ciuciu et al. [27]	Replacing the PDP with a Master PDP and some local PDPs as shown in Fig. I.12. An extra PEP, i.e., Application Independent Policy Enforcement Point, is also added to the architecture. A semantic component called OBIS is connected to both PIP and PDP(s).
Shen and Cheng [90]	Updating [89] by replacing the PDP with an inference engine and using SWRL as the policy language.
Zhao & Wang [101]	Adding and connecting a semantic component to the PDP and PAP as shown in Fig. I.13.
Zhang et al. [100]	Adding and connecting a semantic component, which maps the attributes in a request to those in a domain by means of an ontology and an attribute mapping mechanism, to the PDP as shown in Fig. I.14.
Hsu [52]	Replacing the *Context Handler* with a transformation engine (XACML↔SWRL↔Jess facts). Replacing the PDP with an inference engine and using annotated XACML policies. Replacing the PIP withan *XML-based repository* including an ontology base, a SWRL base, and a style sheet base.
Calvillo-Arbizu et al. [20]	An attribute ontology is added to the *Context Handler*. The *Context Handler* maps the attributes in the received request to those in the ontology through mapping rules. The PDP, which is replaced with an inference engine, converts the retrieved policies into SWRL rules before making a decision.
Drozdowicz et al. [35]	Replacing the PIP with a SemanticPIP (i.e., a semantic component is connected to the PIP).
Liu and Wang [68]	Adding and connecting a semantic component to the PIP. Employing four different ontologies, i.e., the subject, object, action, and environment ontologies.
Hilia et al. [48]	Adding a module (to the PDP) called semantic finder, which has two functions namely *attribute finder* and *policy finder*, for obtaining contextual (semantic) information.
Jin and Fang-Chun [56]	Specifying access control policies and representing attributes using a description logic (the restricted $ALL(D)$) language. Adding a new component, which stores environment (context) attributes, in addition to the PIP as shown in Fig. I.15. Replacing the PDP with an inference engine.
Amini & Jalili [3]	Connecting the PIP (*context handler* in this scheme), PDP, and PAP to a $MA(DL)^2$ knowledge base as shown in Fig. I.16. Replacing the policy language with the $MA(DL)^2$ logic [2], which is a combination of deontic and description logics. Replacing the PDP with an inference engine.
Iqbal and Noll [55]	Introducing and adding a Token Generator, a Token Validator, and an attribute knowledge base. The Token Generator issues tokens, a set of attributes, to subjects, and the Token Validator validates the tokens (i.e., verifies the attributes). The PDP is an inference engine, and policies are SWRL rules.
Trivellato et al. [95]	Replacing the policy language with an ontology-based language inspired by Datalog with constraints. Adding an extra PDP for trust management, which assists the normal PDP in collecting attributes. Both PDPs are connected to a knowledge base, including ontologies and a semantic alignment evaluator.
Calvillo et al. [21]	Adding and connecting a semantic component, including SWRL rules (i.e., policies) and the POVO ontology, to the PDP, which is an inference engine.
Lu et al. [69]	Formalizing XACML policy elements in OWL. Generating dynamic policies by associating and dissociating policy elements and using inference results. Decision-making through checking the compliance between access requests (modeled using semantic concepts) and access control policies.
Verginadis et al. [97]	Employing a generic policy model, for the specification of policies using ontological concepts, which allow the semantic representation of policies. Replacing the PDP with an inference engine.
Amini & Arasteh [1]	Combining a Semantic-Based Access Control model with ABAC as shown in Fig. I.17.
Husain et al. [54]	Extending the functionality of the PDP. The PDP selects a rule as an applicable rule based on the semantic difference of the attributes in both the request and policies (based on an ontology). The PDP makes a decision based on the similarity score of the attributes in the request and selected rule(s).

P5 in Section I.5.1. These evaluations also motivate the open problems that we outline in Section I.6, namely **OP1** to **OP9**.

Answering **RQ1**, SABAC can be realized by extending the XACML standard as seen in [18, 20, 22, 27, 29, 32, 35, 36, 48, 52, 68, 75, 89, 90, 100, 101], proposing a new policy language and accordingly a new architecture as presented in [3, 21, 55, 56, 69, 95, 97], and/or combining different access control models, e.g., ABAC and SBAC, as proposed in [1, 54].

Regarding **RQ2**, in most of the existing schemes (see Table I.2 and Fig. I.5), semantic technologies are used to improve the interoperability by means of finding semantic synonyms of attributes. Different methods are used for finding synonyms, e.g., if the semantic difference of two concepts (attributes) is less than a predefined threshold, then they are considered *semantically similar* as done in [27, 28, 54, 95, 100, 101]. Another way of finding *synonyms of attributes* is based on the relationships between concepts in an ontology as seen in [1, 18, 28, 29, 32, 35, 36, 75, 89, 97].

Some works employed semantic technologies also for enhancing the expressiveness of the policy specification language as seen in [21, 22, 55, 56, 69, 90]. Other works used the semantic technologies for the policy evaluation (i.e., the decision making) as presented in [1, 3, 20–22, 48, 52, 55, 56, 69, 90, 97], policy creation as demonstrated in [68, 69, 97], and managing policy redundancies as used in [68].

Regarding **RQ3**, Table I.2 shows only three SABAC schemes [1, 3, 68] that have formal specifications. They provide a formal specification of the basic elements and the authorization model. However, such simple formal specifications are not suitable for formal verification of properties such as security or deadlock freeness. The formal specifications provided in [1, 3, 68] just define the proposed SABAC schemes and they are not used for formal verification. Therefore, we highlight the property **P4** of the ideal SABAC described in Section I.5.1 and formulate the open problem **OP8** in Section I.6, calling for more research on formal specification and verification.

Regarding **RQ4**, the schemes proposed by Jin and Fang-Chun [56], Amini and Jalili [3], and Trivellato et al. [95] use the new policy specification languages $ALL(D)$, $MA(DL)^2$, and POLIPO, receptively. In the schemes proposed by Shen and Cheng [90], Calvillo et al. [22], Lu et al. [69], Calvillo et al. [21], and Iqbal and Noll [55] the policy specification language is SWRL and an inference engine works as the access control engine (the PDP), making the decisions based on the employed ontologies and SWRL rules. Although SWRL increases the expressiveness of the policy specification, it also increases the complexity of the SABAC due to computational overhead from semantic reasoning. In the schemes of Liu and Wang [68] and Amini and Arasteh [1], access control policies are specified using both the XACML policy language and SWRL rules. In Hsu's scheme [52], the XACML policies and requests are annotated to map attributes to ontological concepts (the data type of attributes is used for the mapping). It can be concluded that Jin and Fang-Chun [56], Amini and Jalili [3], and Trivellato et al. [95] partially address the property **P1** of the ideal SABAC by proposing new policy specification languages. However, it is still not possible to do formal

Table I.2: A comparison of the SABAC schemes

RQ1	Schemes	RQ2	RQ3	RQ4	RQ5	RQ6	RQ7	RQ8	RQ9	RQ10	RQ11
C1	Damiani et al. [29]	SA	No	XACML RDF	Healthcare	No	RDFS	NS	No	Research example	No*
	Priebe et al. [81]	SA	No	XACML	NS	No	OWL, Jena SPARQL	EX	Yes	-	No*
	Muppavarapu & Chung [75]	SA	No	XACML	Data Grids	No	OWL, SWRL	EX	No	OGSA-DAI project	No*
	Shen [89]	SA	No	XACML	Web services	No	OWL, Jess SWRL	EX	No	-	No*
	Durbeck et al. [36]	SA	No	XACML	Web services	No	WSML	WSMO	No	Access-eGov project	No*
	Dersingh et al. [32]	SA	No	XACML	Ubiquitous Systems	No	OWL, Jess, SWRL, Racer, Jena	EX	Yes	Research example	No*
	Calvillo et al. [22]	DM, EXP	No	SWRL	Healthcare Grids	No	OWL, Jess, SWRL, SQWRL	EX	Yes	PREDIRCAM project	Yes
	Brut et al. [18]	SA	No	XACML	Healthcare	No	OWL	ICD-10, ICPC-2	No	-	No*
	Ciuciu et al. [27]	SS	No	XACML, PERMIS	Distributed env.	No	NS/NA	SecPODE	No	TAS^3 project	No*
	Shen and Cheng [90]	DM, EXP	No	SWRL	Mobile web services	No	OWL, Jess, SPARQL, SWRL	EX	No	-	No
	Zhao & Wang [101]	SS	No	XACML	Web services	No	NS	NS	No	-	No*
	Zhang et al. [100]	SS	No	XACML	NS	No	OWL, Jena	EX	No	-	No*
	Hsu [52]	DM	No	Annotated XACML	NS	Yes	OWL, Jess, SWRL	NS	Yes	Research example	No
	Calvillo-Arbizu et al. [20]	DM	No	XACML	Healthcare	No	OWL, SWRL	EX	No	-	Yes
	Drozdowicz et al. [35]	SA	No	XACML	eHealth	No	OWL, SPARQL	HL7 ontology	Yes	Research example	No*
	Liu and Wang [68]	PC, PR	Yes	XACML, SWRL	Healthcare	Yes	OWL, Jess, SWRL	NS	Yes	LinkedCT	Yes
	Hilia et al. [48]	DM	No	XACML	Cloud env.	No	RDF, SPARQL	NA	Yes	ComVantage project	No*
C2	Jin and Fang -Chun [56]	DM, EXP	No	$ALL(D)$	NS	No	$ALL(D)$, Racer, OWL	NS	Yes	-	No
	Amini & Jalili [3]	DM	Yes	$MA(DL)^2$	Distributed env.	Yes	$MA(DL)^2$, Jena	EX, OWL-S	Yes	Research example	Yes
	Iqbal and Noll [55]	DM, EXP	No	SWRL	Cloud services	No	RDF, OWL, SWRL, Pellet	EX	Yes	-	No
	Trivellato et al. [95]	SS	No	POLIPO	Systems of Systems	No	RDF, Jena	SEM	Yes	EU NAVFOR	No
	Calvillo et al. [21]	DM, EXP	No	SWRL	Healthcare	No	OWL, Jess, SWRL, SQWRL	EX	Yes	-	Yes
	Lu et al. [69]	DM, EXP, PC	No	SWRL	eHealth	No	OWL, SWRL	NS	No	ADDN and AURIN	No
	Verginadis et al. [97]	SA, PC, DM	No	RDF	Cloud services	Yes	RDF	NS	Yes	-	Yes
C3	Amini & Arasteh [1]	DM, SA	Yes	XACML, SWRL	Virtual Org.	Yes	OWL, Pellet SWRL	EX	Yes	Research example	Yes
	Husain et al. [54]	SS	No	XACML	Geo-spatial domain	No	NS	EX	Yes	DFW maps	Yes

C1: Extensions of XACML; **C2**: New policy languages; **C3**: Hybrid models; **SA**: Extending the list of attributes by adding Synonyms of Attributes to the request; **DM**: Decision Making; **EXP**: Improving the expressiveness; **SS**: Semantic Similarity; **PC**: Policy creation; **PR**: Diminishing policy redundancy; **NS**: Not Specified; **EX**: A simple new example; **NA**: Not Applicable;
No*: It is not addressed; however, it seems that the XACML combining algorithms can be used

verification of access control policies (justifying the open problem **OP1**).

As represented in Fig. I.6 and listed in Table I.2, the existing SABAC schemes are proposed for various domains (addressing **RQ5**). Damiani et al. [29], Brut et al. [18], Calvillo et al. [22], Calvillo-Arbizu et al. [20], Drozdowicz et al. [35], Liu and Wang [68], Lu et al. [69], and Calvillo et al. [21] proposed SABAC schemes for the healthcare domain. The schemes proposed by Durbeck et al. [36], Shen [89], Shen and Cheng [90], and Zhao and Wang [101] are intended for web services. The schemes of Ciuciu et al. [28], Muppavarapu and Chung [75], Ciuciu et al. [27], and Amini and Jalili [3] are proposed for the grid and distributed environments. The schemes by Hilia et al. [48], Iqbal and Noll [55], and Verginadis et al. [97] are developed for cloud services. The target domain of the schemes proposed by Dersingh et al. [32], Trivellato et al. [95], Amini and Arasteh [1], and Husain et al. [54] is ubiquitous systems, systems of systems, virtual organizations, and geo-spatial domain, respectively. However, the target domain of the schemes provided in [52, 56, 81, 100] is not specified. The results of addressing **RQ5** justify the property **P5** of the ideal SABAC, which suggests a generic modular framework for SABAC so that an SABAC scheme can be used in several domains just by changing the domain ontology (by adding the target domain ontology as a plug-in).

Regarding **RQ6**, Table I.2 shows that the performance of most of the proposals was not evaluated. Only Hsu [52], Liu and Wang [68], Amini and Jalili [3], Verginadis et al. [97], and Amini and Arasteh [1] evaluated the performance of their proposals. However, they did not compare the performance of their proposals with that of other SABAC or ABAC schemes. Besides, they did not perform the experiments with the same parameters and experimental environment. Hence, it is difficult to determine what scheme has a better performance. Only Verginadis et al. [97] compared their proposed scheme to one of XACML implementations, i.e., an ABAC and not an SABAC, to show the overhead of adding semantic technologies to the ABAC. Addressing **RQ6** confirms the lack of an evaluation framework for evaluating not only the performance, but also the security, usability, transparency, and scalability (according to the property **P3** of the ideal SABAC) as described further in the open problems **OP5** and **OP6**.

Addressing **RQ7**, it can be seen that most of the existing SABAC schemes [1, 18, 20–22, 32, 35, 52, 55, 68, 69, 75, 81, 89, 90, 100] used OWL, which is a W3C standard, as the data modeling language for representing the domain knowledge in ontologies. However, some others used their own languages [3, 56], RDFS [29] and RDF [48, 95, 97]. As listed in Table I.2, various inference engines are used such as Jena by [3, 32, 81, 95, 100], Jess by [21, 22, 32, 52, 68, 89, 90], Racer by [32, 56], Pellet by [1, 55], and proprietary ones in [3]. The schemes proposed in [35, 48, 81, 90] employed SPARQL as a query language to retrieve the knowledge from ontologies. SWRL is used in [1, 20–22, 32, 52, 55, 68, 69, 75, 89, 90] as the rule markup language for specifying more complex relationships.

Considering **RQ8**, it can be seen that the ontologies employed in [18, 27, 35, 36, 95] are based on existing ontologies such as ICD-10, ICPC-2, WSMO, SecPODE, SEM, and HL7, whereas the schemes presented in [1, 3, 20–22, 32, 54,

55, 75, 81, 89, 90, 100] employed/developed a simple small ontology. However, it is not mentioned how the employed ontologies are developed (the methodology for developing). Furthermore, it is not stated how the quality, performance, and usefulness of these ontologies are evaluated.

The schemes proposed in [1, 3, 21, 22, 32, 35, 48, 52, 54–56, 68, 81, 95, 97] have a prototype implementation as a proof of concept. However, none of them exists as a commercial tool (**RQ9**). Besides, the source codes of these prototypes are not publicly available (**RQ9**).

Most of the existing SABAC schemes are supported by a case study (**RQ10**) which was either a research project [22, 27, 36, 48, 54, 68, 69, 75, 95] or a research example defined by the authors [1, 3, 29, 32, 35, 52].

Table I.2 shows that resolving conflicts between policies (**RQ11**) was not that important to the researchers, because only Calvillo-Arbizu et al. [20], Amini and Jalili [3], Verginadis et al. [97], Calvillo et al. [21], Liu and Wang [68], Calvillo et al. [22], Husain et al. [54], and Amini and Arasteh [1] addressed this issue (justifying the property **P2** of the ideal SABAC and the open problem **OP7**).

I.5.1 An Ideal SABAC

Based on our literature review, we define below the properties that an ideal SABAC should have. As with any ideal, one often falls short of achieving it, but striving for such an ideal is always fruitful.

P1 An ideal SABAC has an expressive language that considers the semantic relationships between different involved elements and entities (including the attributes of subjects, objects, actions, and the environment) for specifying not only fine-grained access control policies but also obligations, thus offering an advanced access control mechanism that is at the same time dynamic.

P2 An ideal SABAC allows distributed specification of access control policies. Each domain (in a multi-domain setting) can specify and enforce its own access control policies while collaborating with other domains. In other words, all the inconsistencies between access control policies (explicit and implicit, and inter- and intra-domain policies), and incorrect and incomplete policies can be detected and fixed when specifying/merging them. It has a way to deal with conflicts before runtime in contrast to the approach of the XACML standard which fixes inconsistencies by means of combining algorithms at the time of making decisions (i.e., at runtime).

P3 An ideal SABAC not only provides functional requirements like security and privacy, but also it addresses non-functional requirements like performance (better response time), transparency (hiding all the added complexities of distribution), scalability (handling a large number of attributes, requests, policies, semantic relationships, enforcement points, etc.), usability (easy

to deploy/use the system and to specify/manage policies), flexibility (easy adaptation to changes in, e.g., system components), and so on.

P4 An ideal SABAC has a formal foundation that allows formal verification of various properties. Complex interactions between different components of SABAC that are (typically) distributed across networks and organizations can be verified before the actual deployment. It is possible to verify that the system works (restrict/ensure access to resources) properly in all situations, and for example, no combination of attributes and policies will lead to a deadlock. It is possible to verify that appropriate access is always possible, i.e., for every access request, there will be a valid response from the policy decision point (liveness). It can be verified that a malicious requestor can never forge attributes (or pretend to have some attributes) and get access to resources that are not authorized (safety). Formal verification of the above-mentioned (and other applicable) properties leads to well-proven access control.

P5 An ideal SABAC has a modular framework and a prototype implementation with at least one realistic application to a real-world case study demonstrating the claimed features.

The analysis of the existing SABAC schemes demonstrates that the schemes proposed in [3, 8, 21, 22, 55, 56, 90, 95, 97] somehow addressed **P1**. However, the languages proposed/used in these schemes still need improvement to satisfy **P1**. Only a few SABAC schemes [1, 3, 20–22, 54, 68, 97] have addressed the conflict resolution (**P2**), but this is done at the time of decision making (i.e., in an online manner) and not at the time of policy creation or combination. The rest of the existing schemes did not address conflict resolution. Probably the schemes that extended the XACML rely on the combining algorithms of the XACML standard. Nevertheless, it should be noted that semantic technologies may lead to a situation, due to the inference, which cannot be managed by the XACML combining algorithms. In five schemes [1, 3, 52, 68, 97], the performance was evaluated. However, the performance of the proposed schemes was not compared to that of other SABAC schemes. The schemes presented in [75, 89, 90] tried to address to some extent privacy, for example, by using the Shibboleth service for the collection of attributes. Other requirements such as security, transparency, scalability, and usability were not addressed at all as desired in **P3**. Three of the existing schemes have a formal specification [1, 3, 68], which is a good attempt toward **P4**. However, the provided formal specifications are not sufficient to do formal verification of properties. Several existing works have a prototype [1, 3, 21, 22, 31, 34, 48, 52, 54–56, 68, 81, 95, 97] and some benefited from real-world case studies [22, 27, 36, 48, 54, 68, 69, 75, 95]. To be in line with **P5**, there is still a need for a modular generic framework allowing, for example, plugging in different domain ontologies and making the access control scheme suitable for different domains.

I.6 Open problems

This section describes open problems and research directions based on the discussion provided in the previous section and the desired properties of the ideal SABAC.

OP1 Policy specification language

In order to specify access control policies, several policy languages such as XACML, XRBAC [57], and Ponder [30] have been proposed. However, they do not consider semantics and thus are not suitable for SABAC. Other languages such as KAoS [96], Rei [58], Rein [59], and EXAM-S [38] work based on the knowledge represented by ontologies while complex relationships cannot in general be represented by any ontology. Most of the existing SABAC schemes use the XACML policy language along with a rule markup language like SWRL.

In order to improve SABAC schemes, there is a need for a common language for the specification of both access control policies (plus obligations) and semantic relationships together (judging from the results of addressing **RQ4** and aiming for **P1**). If the language is based on a logic that supports the specification and inference of access control policies, then it can also be used to detect semantically incomplete or incorrect access control policies.

OP2 Cryptographic solutions for SABAC

The security of SABAC, like every access control mechanism, relies on a trusted reference monitor that checks all accesses against predefined access control policies. However, such a reference monitor can be easily bypassed, for example, by getting direct access to the data on a storage device. Besides, SABAC is a centralized solution, i.e., an engine is used to make access decisions in each domain (or maybe one for all domains). In response, cryptographic mechanisms have been augmented for achieving the main goal of the access control, i.e., protecting resources, while maintaining the flexibility given by the use of attributes [60]. For instance, ABE schemes are developed to protect resources in a fine-grained manner based on a set of attributes and access structures (i.e., access control policies) [14, 19, 53, 99]. ABE might be considered as a cryptographic replacement of ABAC, but it does not achieve the goals of SABAC. Therefore, developing cryptographic solutions for SABAC that provide protection for multi-domain environments in a decentralized manner could be considered as an open problem as well.

OP3 Developing SABAC ontologies

Most of the existing SABAC schemes work based on some ontologies, e.g., domain-specific ontologies, access control ontologies, etc. However, there exists no reference ontology that can be used for the SABAC. Hence, there is a need for developing an ontology, based on standard methodologies,

covering essential elements of the SABAC, including entities, attributes, environmental parameters, obligations, duty separation, and so on. Such an ontology can be used as a reference ontology along (or combined) with domain-specific ontologies to provide a more sophisticated SABAC scheme. Using an existing reference ontology not only improves the SABAC schemes (in terms of security, performance, etc.), but also saves time and energy as developing an ontology is time-consuming and needs special expertise that security administrators, usually, do not have.

OP4 Privacy

Privacy needs to be considered from two different aspects: privacy of users (subjects) and privacy of data (objects). SABAC schemes control access to the protected objects based on the attributes of subjects and other involved entities and do not work based on the identity of subjects, in contrast to the conventional identity-based access control schemes. It is supposed that the privacy of the subjects is guaranteed as the user identity is not involved in the decision-making process. However, subject attributes that represent the characteristics of a user may identify a subject even though a group of users may share the same set of attributes. For example, a few people may hold a specific set of attributes in a hospital. Hence, it is not difficult to identify such users when they issue a request.

The existing research about privacy in ABAC [6–8, 11, 16, 64, 73, 83, 86] can be used as a guide to address privacy in SABAC as well. For instance, one suggestion for considering privacy when making decisions could be benefiting from a privacy ontology like PrOnto [78].

OP5 Evaluation framework

In order to check if an access control scheme achieves its goals, it needs to be evaluated using a standard evaluation framework. There exists no standard evaluation framework for access control. In the absence of a standard evaluation framework, each scheme may be evaluated based on self-defined criteria and parameters. Hence, it is difficult to say what access control scheme is more suitable for a specific context.

Therefore, based on **P3** and the results of addressing **RQ6**, there is a need for developing a standard evaluation framework for SABAC. Such a framework should be able to check the realization of the functional and non-functional requirements of the access control such as security, performance, usability, transparency, and scalability. In particular, **OP6** describes the scalability in more detail.

OP6 Scalability

As described throughout the paper, SABAC schemes employ several different components, including ontologies and reasoners, which might be deployed in a distributed manner, and are supposed to be used in multi-domain environments. An increase in the number of domains escalates the

number of entities (i.e., subjects and objects, and associated attributes), domain-specific ontologies, and access control policies (both explicit and implicit ones). However, the scalability is not addressed in the existing SABAC schemes. Realistic scalability testing through real-world related case studies is required for SABAC schemes.

OP7 Conflict resolution

Inconsistencies in policies can cause availability (denying authorized requesters access) and safety (granting access to unauthorized requesters) problems. Most of the existing SABAC schemes either rely on the XACML combining algorithms for conflict resolution (those that extended the XACML standard) or did not address conflict resolution at all. Therefore, based on **P2** and the results of addressing **RQ11**, resolving the conflicts between policies (for instance, conflicts between inferred policies and existing ones) is very important and researchers are suggested to consider this issue as well.

OP8 Formal specification

The existing SABAC schemes lack a formal foundation for formal verification of properties like security, safety, consistency, liveness, and deadlock-freeness. Formal specification of the access control systems and policies help to formally prove whether policies are gap-free (cover all the access requests), conflict-free (especially when integrating policies from different domains), and overlap-free. Taking into account the semantics of attributes in SABAC schemes may make some policies redundant (similar to the other policies). Optimizing policies by removing redundancies improves the overall performance of SABAC. Formal methods, e.g., model checking, make it possible to detect redundant policies provided that they are specified formally. Therefore, based on **P4** and the results of addressing **RQ3**, it can be concluded that more research is required towards a formal foundation for the SABAC.

OP9 Obligations

ABAC has a powerful mechanism implemented by the PEP called obligations, which are meant to enforce extra constraints like writing logs, sending notifications, or asking for confirmations. Obligations are greatly desired especially in eHealth, because of the accountability and highly interactive style of work where many types of actions must be logged, various authorizations are needed from experts (i.e., confirmations), or simply sending notifications to relevant parties (like to family/guardians) are required by law. Nevertheless, obligations are not addressed in almost all of the existing SABAC schemes. Semantic technologies as an integral component of SABAC schemes can help to capture the semantics of obligations as well. For instance, an obligation required by a policy can be replaced by a semantic similar obligation if it is not enforceable for any reason.

I.7 Related work

As already pointed out, there exists no survey on SABAC. However, there are several review papers on access control that may provide more general information about the access control. Hence, this section lists some of the related survey papers.

In 2017, Servos and Osborn [88] conducted a literature review of the existing studies on ABAC. They also outlined several open research problems related to ABAC such as delegation, separation of duties, auditability, Hierarchical ABAC, and problems related to sharing and storage of attributes which are not considered in this paper to avoid redundancy. Such issues are relevant for SABAC schemes as well.

Lazouski et al. [66] reviewed works on Usage Control (UCON) [79, 80]. UCON is close to ABAC as it also works based on attributes. However, UCON evaluates an access request continuously. It means that a decision about an access request may change even after granting/denying the access request as the UCON continuously evaluates the access requests. For example, a Permit decision may be changed to a Deny after a period of time or after retrieving a specific amount of data.

Another relevant survey is provided by Kirrane et al. [61]. They analyzed the existing access control mechanisms aiming at securing the Linked Data infrastructure. In other words, the access control schemes that are by/for RDF data, which is the basis of the linked data, are reviewed in [61].

Paci et al. [77] reviewed the existing approaches for access control in community-centered collaborative environments. They focused on the governance (i.e., policy combination and conflict resolution) as resources may be under the administration of several entities, usability and transparency (which include the generation, comprehension, and configuration of policies and feedback generation), and evaluation methods, e.g., controlled experiments and performance evaluation.

In a recent work, Qiu et al. [82] conducted a literature review on the existing access control mechanisms for the Internet of Things (IoT) search. The main function of the IoT search technology is to find correct information about physical entities from IoT and sensor networks in real time. The focus of Qiu et al.'s survey was on policy composition (and accordingly conflict resolution), policy mining, and authorization models, i.e., ABAC, RBAC, UCON, Organizational-Based Access Control (OrBAC), Open Authorization (OAuth), and Capability-based access control (CapBAC).

I.8 Conclusion

In this paper, we have conducted a systematic literature review of semantic attribute-based access control schemes until May 2020. We have provided a comprehensive summary of the conducted research efforts on SABAC, and identified the gaps and trends in this field, to help researchers, developers, and

professionals who want to improve or employ SABAC systems. We have classified the existing SABAC schemes based on a set of research questions to have a general overview of the existing approaches. We have also introduced the notion of an "ideal SABAC" by describing important desired properties of semantic attribute-based access control. Based on the results of the literature review and the proposed ideal SABAC, we have identified several open problems that need to be addressed in order to achieve an ideal SABAC.

References

[1] Amini, M. and Arasteh, M. "A combination of semantic and attribute-based access control model for virtual organizations". In: *ISC International Journal of Information Security* vol. 7, no. 1 (2015), pp. 27–45.

[2] Amini, M. and Jalil, R. *MA (DL) 2 Logical Language Family for Policy Specification and Inference*. Tech. rep. Sharif University of Technology, 2010.

[3] Amini, M. and Jalili, R. "Multi-level authorisation model and framework for distributed semantic-aware environments". In: *IET Information Security* vol. 4, no. 4 (2010), pp. 301–321.

[4] Anderson, A. et al. "eXtensible Access Control Markup Language (XACML) Version 1.0". In: *OASIS Standard* (2003).

[5] Antoniou, G. and Harmelen, F. van. *A semantic web primer*. MIT Press, 2004.

[6] Ardagna, C. A. et al. "A privacy-aware access control system". In: *Journal of Computer Security* vol. 16, no. 4 (2008), pp. 369–397.

[7] Ardagna, C. A. et al. "Offline Expansion of XACML Policies Based on P3P Metadata". In: *Web Engineering, 5th International Conference, ICWE 2005, Sydney, Australia, July 27-29, 2005, Proceedings*. Ed. by Lowe, D. B. and Gaedke, M. Vol. 3579. Lecture Notes in Computer Science. Springer, 2005, pp. 363–374.

[8] Ardagna, C. A. et al. "Towards Privacy-Enhanced Authorization Policies and Languages". In: *Data and Applications Security XIX, 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security, Storrs, CT, USA, August 7-10, 2005, Proceedings*. Ed. by Jajodia, S. and Wijesekera, D. Vol. 3654. Lecture Notes in Computer Science. Springer, 2005, pp. 16–27.

[9] Baader, F. and Hanschke, P. "A Scheme for Integrating Concrete Domains into Concept Languages". In: *Proceedings of the 12th International Joint Conference on Artificial Intelligence. Sydney, Australia, August 24-30, 1991*. Ed. by Mylopoulos, J. and Reiter, R. Morgan Kaufmann, 1991, pp. 452–457.

[10] Baader, F. et al., eds. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[11] Belaazi, M., Rahmouni, H. B., and Bouhoula, A. “Towards a Legislation Driven Framework for Access Control and Privacy Protection in Public Cloud”. In: *SECRYPT 2014 - Proceedings of the 11th International Conference on Security and Cryptography, Vienna, Austria, 28-30 August, 2014*. Ed. by Obaidat, M. S., Holzinger, A., and Samarati, P. SciTePress, 2014, pp. 463–468.

[12] Blobel, B. et al. “HL7 Version 3 Standard: Security and Privacy Ontology, Release 1”. In: (2014).

[13] Boag, S. et al. “XML path language (XPath) 2.0”. In: *W3C, W3C Recommendation, Jan* (2007).

[14] Bobba, R. et al. “Attribute-Based Messaging: Access Control and Confidentiality”. In: *ACM Transactions on Information and System Security (TISSEC)* vol. 13, no. 4 (2010), 31:1–31:35.

[15] Brickley, D. “Resource Description Framework (RDF) Schema Specification”. In: *http://www.w3.org/TR/rdf-schema* (2000).

[16] Brown, K. P. et al. “Fine-grained filtering to provide access control for data providing services within collaborative environments”. In: *Concurr. Comput. Pract. Exp.* Vol. 27, no. 6 (2015), pp. 1445–1466.

[17] Bruijn, J. de et al. “The Web Service Modeling Language WSML: An Overview”. In: *The Semantic Web: Research and Applications, 3rd European Semantic Web Conference, ESWC 2006, Budva, Montenegro, June 11-14, 2006, Proceedings*. Ed. by Sure, Y. and Domingue, J. Vol. 4011. Lecture Notes in Computer Science. Springer, 2006, pp. 590–604.

[18] Brut, M. et al. “APHR: Annotated Personal Health Record for Enabling Pervasive Healthcare”. In: *12th IEEE International Conference on Mobile Data Management, MDM 2011, Luleå, Sweden, June 6-9, 2011, Volume 2*. Ed. by Zaslavsky, A. B. et al. IEEE Computer Society, 2011, pp. 73–79.

[19] Buehrer, D. J. and Wang, C. “CA-ABAC: Class Algebra Attribute-Based Access Control”. In: *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, Macau, China, December 4-7, 2012*. IEEE Computer Society, 2012, pp. 220–225.

[20] Calvillo-Arbizu, J., Roman-Martinez, I., and Roa-Romero, L. M. “Standardized access control mechanisms for protecting ISO 13606-based electronic health record systems”. In: *Proceedings of IEEE-EMBS International Conference on Biomedical and Health Informatics, BHI 2014, Valencia, Spain, June 1-4, 2014*. IEEE, 2014, pp. 539–542.

[21] Calvillo-Arbizu, J., Román, I., and Roa, L. M. “Empowering citizens with access control mechanisms to their personal health resources”. In: *International journal of medical informatics* vol. 82, no. 1 (2013), pp. 58–72.

[22] Calvillo-Arbizu, J. et al. “Privilege Management Infrastructure for Virtual Organizations in Healthcare Grids”. In: *IEEE Transactions on Information Technology in Biomedicine* vol. 15, no. 2 (2011), pp. 316–323.

[23] Cantor, S. et al. *Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0*. 2005.

[24] Carmody, S. "Shibboleth overview and requirement". In: *http://shibboleth.internet2.edu/internet2-shibboleth-requirements-01.html* (2001).

[25] Carroll, J. J. et al. "Jena: implementing the semantic web recommendations". In: *Proceedings of the 13th international conference on World Wide Web - Alternate Track Papers & Posters, WWW 2004, New York, NY, USA, May 17-20, 2004*. Ed. by Feldman, S. I. et al. ACM, 2004, pp. 74–83.

[26] Chadwick, D. W. et al. "PERMIS: a modular authorization infrastructure". In: *Concurrency and Computation: Practice and Experience* vol. 20, no. 11 (2008), pp. 1341–1357.

[27] Ciuciu, I. et al. "Ontology Based Interoperation for Securely Shared Services: Security Concept Matching for Authorization Policy Interoperability". In: *4th IFIP International Conference on New Technologies, Mobility and Security, NTMS 2011, Paris, France, February 7-10, 2011*. IEEE, 2011, pp. 1–5.

[28] Ciuciu, I. et al. "Ontology-Based Matching of Security Attributes for Personal Data Access in e-Health". In: *On the Move to Meaningful Internet Systems: OTM 2011 - Confederated International Conferences: CoopIS, DOA-SVI, and ODBASE 2011, Hersonissos, Crete, Greece, October 17-21, 2011, Proceedings, Part II*. Ed. by Meersman, R. et al. Vol. 7045. Lecture Notes in Computer Science. Springer, 2011, pp. 605–616.

[29] Damiani, E. et al. "Extending Policy Languages to the Semantic Web". In: *Web Engineering - 4th International Conference, ICWE 2004, Munich, Germany, July 26-30, 2004, Proceedings*. Ed. by Koch, N., Fraternali, P., and Wirsing, M. Vol. 3140. Lecture Notes in Computer Science. Springer, 2004, pp. 330–343.

[30] Damianou, N. et al. "The Ponder Policy Specification Language". In: *Policies for Distributed Systems and Networks, International Workshop, POLICY 2001 Bristol, UK, January 29-31, 2001, Proceedings*. Ed. by Sloman, M., Lobo, J., and Lupu, E. Vol. 1995. Lecture Notes in Computer Science. Springer, 2001, pp. 18–38.

[31] Dersingh, A., Liscano, R., and Jost, A. "Context-aware access control using semantic policies". In: *Ubiquitous Computing And Communication Journal (UBICC) Special Issue on Autonomic Computing Systems and Applications* vol. 3 (2008), pp. 19–32.

[32] Dersingh, A. et al. "Utilizing Semantic Knowledge for Access Control in Pervasive and Ubiquitous Systems". In: *Mob. Networks Appl.* Vol. 15, no. 2 (2010), pp. 267–282.

[33] Drozdowicz, M., Ganzha, M., and Paprzycki, M. "Semantic Access Control for Privacy Management of Personal Sensing in Smart Cities". In: *IEEE Transactions on Emerging Topics in Computing* (2020).

[34] Drozdowicz, M., Ganzha, M., and Paprzycki, M. "Semantic Policy Information Point - preliminary considerations". In: *ICT Innovations 2015 - Emerging Technologies for Better Living, Ohrid, Macedonia, 1-4 October, 2015*. Ed. by Loshkovska, S. and Koceski, S. Vol. 399. Advances in Intelligent Systems and Computing. Springer, 2015, pp. 11–19.

[35] Drozdowicz, M., Ganzha, M., and Paprzycki, M. "Semantically Enriched Data Access Policies in eHealth". In: *Journal of Medical Systems* vol. 40, no. 11 (2016), 238:1–238:8.

[36] Dürbeck, S. et al. "A Semantic Security Architecture for Web Services The Access-eGov Solution". In: *2010 International Conference on Availability, Reliability and Security*. IEEE. 2010, pp. 222–227.

[37] Fallside, D. C. and Walmsley, P. "XML schema part 0: primer second edition". In: *W3C recommendation* vol. 16 (2004).

[38] Ferrini, R. "EXAM-S: an Analysis tool for Multi-Domain Policy Sets". PhD thesis. University of Bologna, Italy, 2009.

[39] Forgy, C. L. "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem". In: *Readings in Artificial Intelligence and Databases*. Elsevier, 1989, pp. 547–559.

[40] Friedman-Hill, E. J. *Jess, the java expert system shell*. Tech. rep. Sandia Labs., Livermore, CA (United States), 1997.

[41] Gennari, J. H. et al. "The evolution of Protégé: an environment for knowledge-based systems development". In: *International Journal of Human-Computer Studies* vol. 58, no. 1 (2003), pp. 89–123.

[42] Haarslev, V. and Möller, R. "RACER System Description". In: *Automated Reasoning, First International Joint Conference, IJCAR 2001, Siena, Italy, June 18-23, 2001, Proceedings*. Ed. by Goré, R., Leitsch, A., and Nipkow, T. Vol. 2083. Lecture Notes in Computer Science. Springer, 2001, pp. 701–706.

[43] Haarslev, V. and Möller, R. "Racer: A Core Inference Engine for the Semantic Web". In: *EON2003, Evaluation of Ontology-based Tools, Proceedings of the 2nd International Workshop on Evaluation of Ontology-based Tools held at the 2nd International Semantic Web Conference ISWC 2003, 20th October 2003 (Workshop day), Sundial Resort, Sanibel Island, Florida, USA*. Ed. by Sure, Y. and Corcho, Ó. Vol. 87. CEUR Workshop Proceedings. CEUR-WS.org, 2003.

[44] Haarslev, V. et al. "The RacerPro knowledge representation and reasoning system". In: *Semantic Web* vol. 3, no. 3 (2012), pp. 267–277.

[45] Hage, W. R. van et al. "Design and use of the Simple Event Model (SEM)". In: *Journal of Web Semantics* vol. 9, no. 2 (2011), pp. 128–136.

[46] Harmelen, F. van, Lifschitz, V., and Porter, B. W., eds. *Handbook of Knowledge Representation*. Vol. 3. Foundations of Artificial Intelligence. Elsevier, 2008.

[47] Hassanzadeh, O. et al. “LinkedCT: A Linked Data Space for Clinical Trials”. In: *CoRR* vol. abs/0908.0567 (2009). arXiv: 0908.0567.

[48] Hilia, M. et al. “Semantic Based Authorization Framework For Multi-Domain Collaborative Cloud Environments”. In: *The 8th International Conference on Ambient Systems, Networks and Technologies (ANT 2017) / The 7th International Conference on Sustainable Energy Information Technology (SEIT 2017), 16-19 May 2017, Madeira, Portugal*. Ed. by Shakshuki, E. M. Vol. 109. Procedia Computer Science. Elsevier, 2017, pp. 718–724.

[49] Hill, E. F. *Jess in Action: Java Rule-Based Systems*. USA: Manning Publications Co., 2003.

[50] Horrocks, I. et al. “SWRL: A Semantic Web Rule Language Combining OWL and RuleML”. In: *W3C Member submission* vol. 21 (2004), p. 79.

[51] Hotz, L. et al. “Chapter 6 - Configuration Knowledge Representation and Reasoning”. In: *Knowledge-Based Configuration*. Ed. by Felfernig, A. et al. Boston: Morgan Kaufmann, 2014, pp. 41–72.

[52] Hsu, I. “Extensible access control markup language integrated with Semantic Web technologies”. In: *Information Sciences* vol. 238 (2013), pp. 33–51.

[53] Hur, J. and Noh, D. K. “Attribute-Based Access Control with Efficient Revocation in Data Outsourcing Systems”. In: *IEEE Transactions on Parallel and Distributed Systems* vol. 22, no. 7 (2011), pp. 1214–1221.

[54] Husain, M. F. et al. “Ontology based policy interoperability in geo-spatial domain”. In: *Computer Standards & Interfaces* vol. 33, no. 3 (2011), pp. 214–219.

[55] Iqbal, Z. and Noll, J. “Towards Semantic-Enhanced Attribute-Based Access Control for Cloud Services”. In: *11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2012, Liverpool, United Kingdom, June 25-27, 2012*. Ed. by Min, G. et al. IEEE Computer Society, 2012, pp. 1223–1230.

[56] Jin, P. and Fang-Chun, Y. “Description Logic Modeling of Temporal Attribute-Based Access Control”. In: *2006 First International Conference on Communications and Electronics (ICCE)*. IEEE. 2006, pp. 414–418.

[57] Joshi, J. et al. “Access-Control Language for Multidomain Environments”. In: *IEEE Internet Computing* vol. 8, no. 6 (2004), pp. 40–50.

[58] Kagal, L., Finin, T. W., and Joshi, A. “A Policy Language for a Pervasive Computing Environment”. In: *4th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2003), 4-6 June 2003, Lake Como, Italy*. IEEE Computer Society, 2003, p. 63.

[59] Kagal, L. et al. “Using Semantic Web Technologies for Policy Management on the Web”. In: *Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA*. AAAI Press, 2006, pp. 1337–1344.

[60] Kayem, A. V. D. M., Akl, S. G., and Martin, P. *Adaptive Cryptographic Access Control*. Vol. 48. Advances in Information Security. Springer, 2010.

[61] Kirrane, S., Mileo, A., and Decker, S. “Access control and the Resource Description Framework: A survey”. In: *Semantic Web* vol. 8, no. 2 (2017), pp. 311–352.

[62] Kitchenham, B. “Procedures for Performing Systematic Reviews”. In: *Keele, UK, Keele University* vol. 33, no. 2004 (2004), pp. 1–26.

[63] Kitchenham, B. et al. “Systematic literature reviews in software engineering–A systematic literature review”. In: *Information and Software Technology* vol. 51, no. 1 (2009), pp. 7–15.

[64] Kolter, J., Schillinger, R., and Pernul, G. “A Privacy-Enhanced Attribute-Based Access Control System”. In: *Data and Applications Security XXI, 21st Annual IFIP WG 11.3 Working Conference on Data and Applications Security, Redondo Beach, CA, USA, July 8-11, 2007, Proceedings*. Ed. by Barker, S. and Ahn, G. Vol. 4602. Lecture Notes in Computer Science. Springer, 2007, pp. 129–143.

[65] Kolter, J., Schillinger, R., and Pernul, G. “Building a Distributed Semantic-aware Security Architecture”. In: *New Approaches for Security, Privacy and Trust in Complex Environments, Proceedings of the IFIP TC-11 22nd International Information Security Conference (SEC 2007), 14-16 May 2007, Sandton, South Africa*. Ed. by Venter, H. S. et al. Vol. 232. IFIP. Springer, 2007, pp. 397–408.

[66] Lazouski, A., Martinelli, F., and Mori, P. “Usage control in computer security: A survey”. In: *Comput. Sci. Rev.* Vol. 4, no. 2 (2010), pp. 81–99.

[67] Li, N. and Mitchell, J. C. “DATALOG with Constraints: A Foundation for Trust Management Languages”. In: *Practical Aspects of Declarative Languages, 5th International Symposium, PADL 2003, New Orleans, LA, USA, January 13-14, 2003, Proceedings*. Ed. by Dahl, V. and Wadler, P. Vol. 2562. Lecture Notes in Computer Science. Springer, 2003, pp. 58–73.

[68] Liu, Z. and Wang, J. “A fine-grained context-aware access control model for health care and life science linked data”. In: *Multimedia Tools and Applications* vol. 75, no. 22 (2016), pp. 14263–14280.

[69] Lu, Y. and Sinnott, R. O. “Semantic privacy-preserving framework for electronic health record linkage”. In: *Telematics Informatics* vol. 35, no. 4 (2018), pp. 737–752.

[70] Manola, F., Miller, E., McBride, B., et al. “RDF primer”. In: *W3C recommendation* vol. 10, no. 1-107 (2004), p. 6.

[71] Martin, D. L. et al. “Bringing Semantics to Web Services with OWL-S”. In: *World Wide Web*, no. 3 (2007), pp. 243–277.

[72] McBride, B. “Jena: A Semantic Web Toolkit”. In: *IEEE Internet Computing* vol. 6, no. 6 (2002), pp. 55–59.

[73] Mewar, V. S., Aich, S., and Sural, S. “Access Control Model for Web Services with Attribute Disclosure Restriction”. In: *Proceedings of the The Second International Conference on Availability, Reliability and Security, ARES 2007, The International Dependability Conference - Bridging Theory and Practice, April 10-13 2007, Vienna, Austria*. IEEE Computer Society, 2007, pp. 524–531.

[74] Mishra, R. B. and Kumar, S. “Semantic web reasoners and languages”. In: *Artificial Intelligence Review* vol. 35, no. 4 (2011), pp. 339–368.

[75] Muppavarapu, V. and Chung, S. M. “Semantic-Based Access Control for Grid Data Resources in Open Grid Services Architecture - Data Access and Integration (OGSA-DAI)”. In: *20th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2008), November 3-5, 2008, Dayton, Ohio, USA, Volume 2*. IEEE Computer Society, 2008, pp. 315–322.

[76] O’Connor, M. J. and Das, A. K. “SQWRL: A Query Language for OWL”. In: *Proceedings of the 5th International Workshop on OWL: Experiences and Directions (OWLED 2009), Chantilly, VA, United States, October 23-24, 2009*. Ed. by Hoekstra, R. and Patel-Schneider, P. F. Vol. 529. CEUR Workshop Proceedings. CEUR-WS.org, 2009.

[77] Paci, F., Squicciarini, A. C., and Zannone, N. “Survey on Access Control for Community-Centered Collaborative Systems”. In: *ACM Computing Surveys (CSUR)* vol. 51, no. 1 (2018), 6:1–6:38.

[78] Palmirani, M. et al. “PrOnto: Privacy Ontology for Legal Reasoning”. In: *Electronic Government and the Information Systems Perspective - 7th International Conference, EGOVIS 2018, Regensburg, Germany, September 3-5, 2018, Proceedings*. Ed. by Ko, A. and Francesconi, E. Vol. 11032. Lecture Notes in Computer Science. Springer, 2018, pp. 139–152.

[79] Park, J. “Usage control: a unified framework for next generation access control”. PhD thesis. George Mason University Fairfax, VA, 2003.

[80] Park, J. and Sandhu, R. S. “Towards usage control models: beyond traditional access control”. In: *7th ACM Symposium on Access Control Models and Technologies, SACMAT 2002, Naval Postgraduate School, Monterey, California, USA, June 3-4, 2002*. Ed. by Sandhu, R. S. and Bertino, E. ACM, 2002, pp. 57–64.

[81] Priebe, T., Dobmeier, W., and Kamprath, N. “Supporting Attribute-based Access Control with Ontologies”. In: *Proceedings of the The First International Conference on Availability, Reliability and Security, ARES 2006, The International Dependability Conference - Bridging Theory and Practice, April 20-22 2006, Vienna University of Technology, Austria*. IEEE Computer Society, 2006, pp. 465–472.

[82] Qiu, J. et al. “A Survey on Access Control in the Age of Internet of Things”. In: *IEEE Internet of Things Journal* vol. 7, no. 6 (2020), pp. 4682–4696.

[83] Rahmouni, H. B. et al. “A SWRL Bridge to XACML for Clouds Privacy Compliant Policies”. In: *CLOSER 2014 - Proceedings of the 4th International Conference on Cloud Computing and Services Science, Barcelona, Spain, April 3-5, 2014*. Ed. by Helfert, M. et al. SciTePress, 2014, pp. 27–37.

[84] Reul, Q. and Zhao, G. “Enabling Access to Web Resources through SecPODE-Based Annotations”. In: *On the Move to Meaningful Internet Systems: OTM 2010 Workshops - Confederated International Workshops and Posters: International Workshops: AVYTAT, ADI, DATAVIEW, EI2N, ISDE, MONET, OnToContent, ORM, P2P-CDVE, SeDeS, SWWS and OTMA. Hersonissos, Crete, Greece, October 25-29, 2010. Proceedings*. Ed. by Meersman, R., Dillon, T. S., and Herrero, P. Vol. 6428. Lecture Notes in Computer Science. Springer, 2010, pp. 596–605.

[85] Roman, D. et al. “Web service modeling ontology”. In: *Applied ontology* vol. 1, no. 1 (2005), pp. 77–106.

[86] Rota, A., Short, S., and Rahaman, M. A. “XML secure views using semantic access control”. In: *Proceedings of the 2010 EDBT/ICDT Workshops, Lausanne, Switzerland, March 22-26, 2010*. Ed. by Daniel, F. et al. ACM International Conference Proceeding Series. ACM, 2010.

[87] Seaborne, A. and Prud’hommeaux, E. “SPARQL Query Language for RDF”. In: *W3C recommendation (January 2008)* (2006).

[88] Servos, D. and Osborn, S. L. “Current Research and Open Problems in Attribute-Based Access Control”. In: *ACM Computing Surveys (CSUR)* vol. 49, no. 4 (2017), 65:1–65:45.

[89] Shen, H.-B. “A Semantic- and Attribute-Based Framework for Web Services Access Control”. In: *2010 2nd International Workshop on Intelligent Systems and Applications*. IEEE. 2010, pp. 1–4.

[90] Shen, H. and Cheng, Y. “A Context-Aware Semantic-Based Access Control Model for Mobile Web Services”. In: *Advanced Research on Computer Science and Information Engineering*. Ed. by Shen, G. and Huang, X. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 132–139.

[91] Sirin, E. and Parsia, B. "Pellet: An OWL DL Reasoner". In: *Proceedings of the 2004 International Workshop on Description Logics (DL2004), Whistler, British Columbia, Canada, June 6-8, 2004*. Ed. by Haarslev, V. and Möller, R. Vol. 104. CEUR Workshop Proceedings. CEUR-WS.org, 2004.

[92] Sirin, E. et al. "Pellet: A practical owl-dl reasoner". In: *Web Semantics: science, services and agents on the World Wide Web* vol. 5, no. 2 (2007), pp. 51–53.

[93] Szczekutek, R. et al. "System for semantic technology-based access management in a port terminal". In: *AIP Conference Proceedings*. Vol. 2025. 1. AIP Publishing LLC. 2018, p. 090002.

[94] Trivellato, D., Zannone, N., and Etalle, S. "GEM: A distributed goal evaluation algorithm for trust management". In: *Theory and practice of logic programming* vol. 14, no. 3 (2014), pp. 293–337.

[95] Trivellato, D. et al. "A Semantic Security Framework for Systems of Systems". In: *Int. J. Cooperative Inf. Syst.* Vol. 22, no. 1 (2013).

[96] Uszok, A. et al. "KAoS Policy Management for Semantic Web Services". In: *IEEE Intelligent Systems* vol. 19, no. 4 (2004), pp. 32–41.

[97] Verginadis, Y. et al. "Context-aware Policy Enforcement for PaaS-enabled Access Control". In: *IEEE Transactions on Cloud Computing* (2019).

[98] Wohlin, C. "Guidelines for snowballing in systematic literature studies and a replication in software engineering". In: *18th International Conference on Evaluation and Assessment in Software Engineering, EASE '14, London, England, United Kingdom, May 13-14, 2014*. Ed. by Shepperd, M. J., Hall, T., and Myrtveit, I. ACM, 2014, 38:1–38:10.

[99] Yu, S. et al. "Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing". In: *INFOCOM 2010. 29th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 15-19 March 2010, San Diego, CA, USA*. IEEE, 2010, pp. 534–542.

[100] Zhang, S., Yang, H., and Wang, B. "Realization Distributed Access Control Based on Ontology and Attribute with OWL". In: *Advances in Electronic Engineering, Communication and Management Vol. 1*. Springer, 2012, pp. 583–588.

[101] Zhao, Y. and Wang, X. "Semantic Similarity-Based Web Services Access Control". In: *Autonomous Systems: Developments and Trends*. Ed. by Unger, H., Kyamakya, K., and Kacprzyk, J. Vol. 391. Studies in Computational Intelligence. Springer, 2012, pp. 339–349.

Paper II

Semantic Attribute-Based Encryption: A Framework for Combining ABE schemes with Semantic Technologies

Hamed Arshad, Christian Johansen, Olaf Owe, Pablo Picazo-Sanchez, and Gerardo Schneider

Information Sciences. (2022), DOI: 10.1016/j.ins.2022.10.132.

II

Abstract

Attribute-Based Encryption (ABE) is a cryptographic solution to protect resources in a fine-grained manner based on a set of public attributes. This is similar to attribute-based access control schemes in the sense that both rely on public attributes and access control policies to grant access to resources. However, ABE schemes do not consider the semantics of attributes provided by users or required by access structures. Such semantics not only improve the functionality by making proper access decisions but also enable cross-domain interoperability by making users from one domain able to access and use resources of other domains. This paper proposes a Semantic ABE (SABE) framework by augmenting a classical Ciphertext-Policy ABE (CP-ABE) scheme with semantic technologies using a generic procedure by which any CP-ABE scheme can be extended to an SABE. The proposed SABE framework is implemented in Java and the source code is publicly available. The experiment results confirm that the performance of the proposed framework is promising.

II.1 Introduction

Access Control (AC) is a fundamental security mechanism to restrict access to (sensitive) data. One of the most promising AC models is Attribute-Based Access Control (ABAC) [27, 39], which provides fine-grained protection based on a set of attributes and access control policies. However, ABAC, like every access control mechanism, relies on a trusted reference monitor that checks all access requests against access control policies and can be easily bypassed, e.g., by getting direct access to the data on a storage device. In contrast to ABAC, Attribute-Based

Encryption (ABE) [11, 22] does not rely on a trusted engine (monitor), but uses cryptographic techniques to provide fine-grained data protection based on ASs (i.e., access control policies) represented as a boolean formula over public attributes. Any user who holds a set of public attributes satisfying the AS can decrypt the ciphertext. For instance, if a picture is encrypted under the following AS: $((Friend\_of\_Alice \wedge Age > 30) \vee (Support = TeamA))$, then only friends of Alice who are over 30 years old and those who support *TeamA* can decrypt the picture, where the *Friend_of_Alice* attribute is granted to users that Alice marked as friends. ABE makes it possible to encrypt data not only for a single user (identified by a unique attribute) but also for a group of users (identified by a set of public attributes).

Until now, a considerable number of ABE schemes have been proposed and employed in several domains such as eHealth [35], online social networks [40], hardware security [21], fog computing [28], and storing sensitive data in public clouds [34]. Furthermore, real world companies like Zeutro[1] deploy security systems based on ABE. Moreover, standards like ETSI[2] have been defined (TS 103 458 and TS 103 532) presenting applications to industrial IoT and cloud.

However, the existing ABE schemes are not semantic-aware, i.e., they do not take into account the semantics of attributes. Semantic awareness could improve the functionality of ABE schemes and allow for cross-domain interoperability of systems based on ABE.

Consider the application of ABE in online social networks [6, 40] where platforms (e.g., Twitter, Facebook, and LinkedIn) use different terminologies (for attributes). For instance, Facebook users can share information (e.g., events, photos, videos) with different groups of audiences such as the *Public*, *Friends*, and *Specific Friends*, whereas in Twitter the target audiences can be specified as *Everyone*, *People you follow*, and *Only people you mention*. Furthermore, in LinkedIn the visibility options for posts are *Anyone*, *Connections only*, *Group members*, and *Event attendees*. It is obvious that *Public*, *Everyone*, and *Anyone* have the same meaning, despite being syntactically different. Similarly, *Friends*, *Followers*, and *Connections* are semantic synonyms. Semantic technologies [2, 4, 5, 8, 25] are particularly useful for handling semantic translations, as commonly required for interoperability between different domains.

Interoperability problems are notoriously common also in eHealth where medical staff from different healthcare institutions need to access data like Electronic Health Records (EHR). When coupled with a growing trend of moving medical records into public clouds [7, 50][3], ABE gains even more relevance. The power of semantic technologies goes beyond semantic synonymity and translations by allowing for more types of inferences.

Suppose the EHR of *PatientA* is encrypted based on the following AS: $(GP \vee (Medical\ Doctor \wedge Employer = Emergency\ Hospital))$. A surgeon working at the Emergency Hospital with attributes $\{Surgeon, Employer =$

[1] https://bit.ly/3gvWRGE
[2] https://bit.ly/3xiLoQk
[3] https://ibm.co/2Tz2LxL

Emergency Hospital} will not be able to access the EHR of *PatientA* because she does not hold the *Medical Doctor* attribute. Even though a surgeon is a medical doctor, ABE works syntactically and cannot infer such knowledge. Any basic medical ontology would have *Surgeon* as a subconcept of *Medical Doctor* and would allow an inference engine to infer this information, which can then be added as the extra attribute needed in such emergency cases. It is worth mentioning that the power of semantic technologies is not limited to such simple translations. Semantic technologies allow having more complex inference rules in addition to the basic ones that are based on inheritance. For instance, $HospitalA$ may have an attribute *Senior Surgeon* for surgeons who have worked more than 5 years as a surgeon and hold X and Y certificates. However, $HospitalB$ may not use such an attribute and only use the *Surgeon* attribute. Hence, a surgeon at $HospitalB$ who has worked more than five years and holds both X and Y certificates would not be able to access (decrypt) a file that is protected (encrypted) based on the *Senior Surgeon* attribute at $HospitalA$ as she does not hold such an attribute.

The aim of this paper is to combine ABE schemes with semantic technologies in order to address the two types of semantic enhancements exemplified above. In particular, we achieve *semantic-aware ABE schemes* by making ABE schemes able to use implicit knowledge from an ontology, while facilitating the interoperability between ABE schemes used in different domains. In more detail:

- In Section III.4, we present Semantic Attribute-Based Encryption (SABE), our framework, which can be built around an arbitrary ABE scheme and an arbitrary inference engine working against an arbitrary ontology.
- In Section II.4, we analyze the security of the proposed framework.
- We provide a prototype implementation, detailed in Section III.7, where we use a specific Ciphertext-Policy Attribute-Based Encryption (CP-ABE) scheme [11], and a specific inference engine called Pellet [43] that works with a quite popular semantic language OWL, over a mock ontology that we have made for this implementation; but the ontology, like the other two aspects, can be freely replaced.
- We evaluate, in Section II.6, further properties of SABE in terms of modularity, scalability, extensibility, and generality.

In Section III.3, we introduce some general terms and definitions used in this paper. Section III.8 presents the related work while the paper concludes in Section III.9.

II.2 Preliminaries

This section gives some background information on attribute-based encryption and semantic technologies.

II.2.1 Attribute-Based Encryption

In conventional public-key cryptography, data are encrypted for a particular receiver using the receiver's public key. Hence, if the same data should be encrypted for several receivers, all the public keys of the receivers are needed. This is even more problematic for data that need to be stored encrypted for sharing with future, yet unknown, users. In response to this, Goyal et al. [22] proposed the first ABE scheme by which the encryptor can encrypt a message under a set of public attributes (instead of just an identity as in identity based encryption schemes [13]). Therefore, data can be encrypted for a group of recipients holding the same public attributes.

ABE is a public-key cryptography in which the private key of users and ciphertexts depend upon attributes. The private keys of users are associated to sets of public attributes, whereas each ciphertext has attached an access structure (in CP-ABE). Anyone who has a set of attributes that satisfies the AS of the ciphertext can decrypt it.

When a user joins the system, she claims to have a set of public attributes and a Trusted Authority (TA) is in charge to validate them. If deemed appropriate, the TA provides the user with a private key associated to her attributes. This authentication process is usually out of the scope of ABE schemes since it is assumed that the TA has the knowledge—or the corresponding mechanisms—to prove that users really have the attributes they claim to have. Consequently, in this paper we assume that users cannot cheat the TA and they are provided with the public attributes they actually have.

The advantages of using ABE are multiple: i) different groups of users can be defined according to public attributes; ii) all the encrypted data can be publicly stored in databases because only users that satisfy the AS will retrieve the plaintext, and; iii) security properties such as access control, user collusions and data disclosures are guaranteed by the underlying cryptographic infrastructure.

The main algorithms of a CP-ABE [11] are `Setup`, `KeyGen`, `Encryption`, and `Decryption`. While the first two algorithms are run by the TA, the last two are executed by the users.

Setup(1^λ) This algorithm takes a security parameter as input and generates a master secret key (MK) and a set of public parameters (PP).

KeyGen(MK, S, PP) This method produces a private key (SK) for a provided set of attributes, $S = \{Att_1, ..., Att_N\}$, using the master secret key and public parameters.

Encryption(M, $\mathcal{T}$, PP) It encrypts a message M based on the access structure ($\mathcal{T}$) and public parameters, and returns a ciphertext $CT = (\mathcal{T}, C)$, where C is the encrypted version of M.

Decryption(CT, SK, PP) It decrypts a ciphertext CT using a private key (SK), which is generated for a set of attributes satisfying the access structure included in CT, and public parameters.

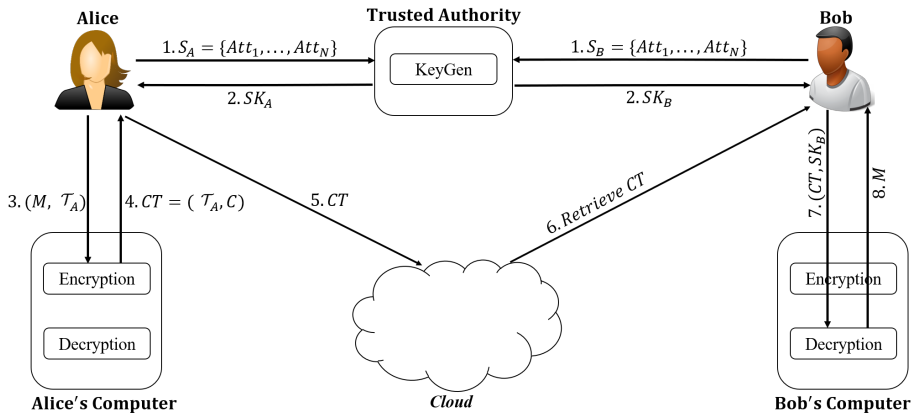


Figure II.1: General architecture of a CP-ABE scheme.

We include a graphical presentation in Figure II.1 where two users, Alice and Bob, join the system. First, they provide their public attributes (S_A and S_B) to the TA (communication 1) and they receive the private keys (SK_A and SK_B) associated to their attributes (2). After that, Alice runs the `Encryption` algorithm producing CT (3 and 4) and sends the ciphertext to the cloud where Bob can get it (5). When Bob retrieves CT from the cloud (6), he runs the `Decryption` algorithm (7) and finally, he obtains the plaintext (8). Figure II.1 depicts all the steps in a CP-ABE scheme; however, it does not mean that all the steps are required for all kinds of operations. For instance, if Alice wants to encrypt a message, she only needs to run the `Encryption` algorithm and provide the message, the desired access structure, and the public parameters (i.e., for encryption, the data owner does not need to get a private key for her attributes).

II.2.2 Semantic technologies

Semantic technologies are a collection of methods, languages, and tools that facilitate advanced data categorization, processing, and relationship discovery across a variety of data sets. Concepts (entities or data) and relationships between them in a certain domain can be described and represented by means of vocabularies. Vocabularies are useful not only for organizing knowledge, but also for resolving ambiguities when integrating different data sets.

RDF Schema (RDFS) [15] was proposed as a language for defining Resource Description Framework (RDF) vocabularies. RDFS is based on the notion of classes and inheritance relationships like those in object-oriented programming languages. RDFS makes it possible to define taxonomies (very simple vocabularies) and perform simple inferences about them. More complex vocabularies, which have thousands of concepts, are called ontologies that can be represented by classes, relations, and instances. The classes can be related to each other by means of relations. For example, in an ontology, class "Medical

Doctor" maybe a *subclass* of a class "Medical Staff" or a relation *"is GP of"* may exist between the class "Medical Doctor" and a class "Patient". There can also be some constraints between the relations among classes that determine which kind of values are allowed. For example, each "Patient" is always a "Person" (a one-to-one relationship) and a "Patient" cannot be a subclass of two "Persons". The classes, relations, and constraints can be combined to form complex statements or assertions expressing the knowledge. In other words, they form the Terminological Knowledge (TBox).

An ontology can be populated by means of instances. In other words, all the classes can contain individuals with some relationships between them. For example, "Alice" can be an instance of the "Medical Doctor" class in the ontology that can have an "ID" and also a relation "is GP of" with another individual "Bob", which is an instance of the class "Patient". The knowledge about the individuals is called Assertional Knowledge (ABox). The terminological knowledge and assertional knowledge form a Knowledge Base, which represents an ontology. The OWL is a standard language for creating/defining ontologies and provides richer semantics than the RDFS.

Inheritance relationships in RDFS and OWL are simple relationships which can be used for performing very simple inferences. However, if some specific relationships need to be held under some conditions, then it is difficult to express them using the ontology markup languages, e.g., RDFS and OWL. For example, it is difficult to specify that a "Chief Physician" is a "Medical Doctor" who was hired more than 10 years ago. Such relationships can be handled using rules.

SWRL [25] is a popular and standard rule markup language that combines Horn logic rules and OWL ontologies to define complex relationships between concepts. Using SWRL it is possible to specify complex inference rules in addition to the basic ones that are based on inheritance. Inference rules make it also possible to infer new knowledge (i.e., inferring implicit relationships from explicit ones). In order to do the inference, a reasoner (i.e., inference engine) is required. A reasoner uses a set of facts and axioms to get new logical consequences.

II.3 SABE: A Semantic ABE Framework

Augmenting ABE schemes with semantic technologies results in semantic-aware schemes by including implicit knowledge in controlling access and improves cross-domain interoperability. We developed a semantic component consisting of 1) a domain ontology; 2) SWRL rules, and; 3) an inference engine. The domain ontology represents the semantic relationships between attributes in a given domain (or domains). The SWRL rules are used to define more complex relationships that are not possible to be defined using ontology data modeling languages. The inference engine performs the reasoning and infers the implicit knowledge.

In the following, we propose two approaches, namely *Semantically-Enriched Key* and *Semantically-Enriched Access Structure*, to augment ABE schemes with semantic technologies. First, we define what an ontology is (see Definition II.1)

as well as the relationships we use between concepts in an ontology (see Definition II.2). Definition II.3 and Definition II.4 define "semantically relevant attributes" (or "semantically relevant concepts" as attributes are represented as concepts in the ontology) in the proposed *Semantically-Enriched Key* and *Semantically-Enriched Access Structure* approaches, respectively.

Definition II.1 (Ontology)**.** *An ontology is a tuple* $O = \langle \boldsymbol{C}, \boldsymbol{R}, \boldsymbol{I} \rangle$*, where* $\boldsymbol{C}$*,* $\boldsymbol{R}$*, and* $\boldsymbol{I}$*, denote, respectively, sets of concepts (classes), relationships between concepts, and instances (individuals) belonging to concepts. Concepts represent the attributes in a domain.*

Definition II.2 (Relationships)**.** *Our proposals use the following relationships between concepts in an ontology:*

- `subClassOf` ($\subseteq$)*: if the semantic scope of a concept* $C_1 \in \boldsymbol{C}$ *is narrower than that of another concept* $C_2 \in \boldsymbol{C}$*, i.e., every instance of* C_1 *is also an instance of* C_2*, then* C_1 *is a subclass of* C_2*. In other words,* $C_1 \subseteq C_2$ *iff* $\forall i \in \boldsymbol{I} : (i \in C_1 \rightarrow i \in C_2)$*, where* $C_1, C_2 \in \boldsymbol{C}$*.*

- `equivalentClass` ($\equiv$)*: if the semantic scope of a concept* $C_1 \in \boldsymbol{C}$ *is equal to that of another concept* $C_2 \in \boldsymbol{C}$*, i.e., both concepts have exactly the same set of individuals, then* C_1 *is an equivalent class of* C_2*. In other words,* $C_1 \equiv C_2$ *iff* $\forall i \in \boldsymbol{I} : ((i \in C_1 \rightarrow i \in C_2) \wedge (i \in C_2 \rightarrow i \in C_1))$*, where* $C_1, C_2 \in \boldsymbol{C}$*.*

Definition II.3 (SR^{Att}_{SEK})**.** *Given a concept* $C_{in} \in \boldsymbol{C}$ *from an ontology* O*, the semantically relevant concepts in SABE-SEK denoted by* $SR^{Att}_{SEK}(C_{in})$ *are defined as the set of all concepts* $C_j \in \boldsymbol{C}$ *such that* $C_{in} < C_j$*, where* $<$ *is defined as the smallest relation satisfying the following rules:*

- $C_2 < C_1$ *if* $C_2 \subseteq C_1$*, i.e.,* C_2 `subClassOf` C_1
- $C_2 < C_1$ *if* $C_2 \equiv C_1$*, i.e.,* C_2 `equivalentClass` C_1
- $C_2 < C_1$ *if* $C_2 < C_3$ *and* $C_3 < C_1$

Definition II.4 (SR^{Att}_{SEAS})**.** *Given a concept* $C_{in} \in \boldsymbol{C}$ *from an ontology* O*, the semantically relevant concepts in SABE-SEAS denoted by* $SR^{Att}_{SEAS}(C_{in})$ *are defined as the set of all concepts* $C_j \in \boldsymbol{C}$ *such that* $C_j < C_{in}$*, where* $<$ *is defined as the smallest relation satisfying the rules provided in Definition II.3.*

II.3.1 SEK: Semantically-Enriched Key

One type of semantic ABE extension, which we call SABE-SEK, is composed of six algorithms: `ABE.Setup`, `ABE.KeyGen`, `ABE.Encryption`, `ABE.Decryption`, `SABE.UpdateAtt`, and `SABE.KeyGen`, of which the first four are identical to those of a conventional CP-ABE scheme as described in Section III.3.1. The last two (extra) algorithms, which the TA runs, are defined below:

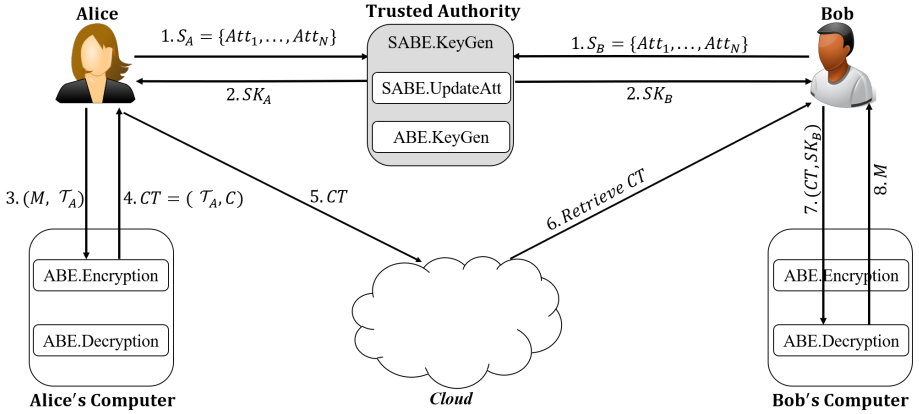


Figure II.2: General architecture of the proposed SABE-SEK.

`SABE.UpdateAtt(`S`,` *Ontology*`)` updates a provided set of attributes, S, based on the semantic relationships defined in a provided ontology, *Ontology*, and returns a new set of attributes, S'.

`SABE.KeyGen(`MK`,` S`,` PP`)` connects the `ABE.KeyGen` algorithm to the semantic component, i.e., the `SABE.UpdateAtt` algorithm. `SABE.KeyGen` first calls `SABE.UpdateAtt` to update the provided set of attributes, and then calls `ABE.KeyGen`(MK, S', PP), which generates a private key, SK, for the updated set of attributes using the master secret key and public parameters.

Note that the TA is trusted to hold the *Ontology* secret, which is part of the initialization process. Otherwise, an adversary may manipulate the ontology to its own advantage.

In the proposed SABE-SEK scheme, we use semantic technologies in the key generation process. The key generation (`ABE.KeyGen`) algorithm generates a private key associated to a set of public attributes that a user provides. The idea is to extend the set of user's attributes by adding all semantically relevant attributes as defined in Definition II.3. Accordingly, the `SABE.KeyGen` algorithm (by calling the `ABE.KeyGen` algorithm) generates a private key based on the extended set of attributes, which in turn means that the user capabilities (in terms of access) will be enhanced as both explicit and implicit knowledge are used in the generation of the user's private key.

Figure II.2 depicts the architecture of the proposed SABE-SEK scheme. Note that SABE-SEK differs from CP-ABE in the key generation process. The other algorithms, i.e., `Setup`, `Encryption`, and `Decryption`, do not need any changes and are identical to the CP-ABE ones, which is why they have an `ABE` prefix.

As demonstrated in Algorithm II.1, when a user submits a set of attributes to a TA, the `KeyGen` algorithm (i.e., the `SABE.KeyGen` algorithm) does not

Algorithm II.1: Pseudocode of SABE-SEK key generation

Input: Master secret key (MK), A set of attributes (S), Public parameters (PP)
Output: A private key (SK)

```
// Updating the provided set of attributes (S) using a domain ontology
   (Ontology). The updated set of attributes is S'.
1 Call SABE.UpdateAtt,
   S' ← SABE.UpdateAtt(S, Ontology)
// Generating a private key using the KeyGen algorithm of a CP-ABE
   scheme.
2 Call ABE.KeyGen,
   SK ← ABE.KeyGen(MK, S', PP)
3 Return SK
```

generate a private key as in the classical CP-ABE schemes. Instead, it (i.e., the `SABE.KeyGen` algorithm) first uses our semantic component, which includes an inference engine and a domain ontology (along with SWRL rules), to obtain all other attributes that are semantically relevant (according to Definition II.3) to those submitted by the user. `SABE.KeyGen` algorithm does this by calling the `SABE.UpdateAtt` algorithm. For example, every *Surgeon* is a *Medical Doctor*; however, every *Medical Doctor* is not necessarily a *Surgeon*. Hence, if a user submits the *Surgeon* attribute, then the semantic component infers that this user implicitly holds the *Medical Doctor* attribute as well based on the domain ontology. After retrieving all the attributes that are semantically relevant to the submitted attributes (i.e., extending the set of attributes in most cases), the `SABE.KeyGen` algorithm calls the `ABE.KeyGen` algorithm, which is the `KeyGen` algorithm of a conventional CP-ABE scheme, to generate a private key for the extended set of attributes (i.e., the attributes that the user submitted and those added by the semantic component) as in the classical CP-ABE schemes. Therefore, a user who has the *Surgeon* attribute would be able to decrypt any ciphertext encrypted under the *Surgeon* attribute or the *Medical Doctor* attribute.

To infer and find the semantically relevant attributes based on Definition II.3, an ontology describing the relationships between the attributes in the given domain is required. For example, we created a proof-of-concept ontology for the healthcare domain (for SABE-SEK), where the concepts represent the attributes. This ontology is provided to the TA during system initialization. When the TA receives a request (a set of attributes) for generating a private key, it calls the semantic component providing the received set of attributes and the ontology. The semantic component finds the semantically relevant attributes according to Definition II.3 and Algorithm II.2. It first assigns a dummy OWL Named Individual to the concepts in the ontology that are related to the received attributes. For example, if a user submits a *Surgeon* attribute, then a dummy Individual, e.g., "TestUser", will be assigned to the concept

Algorithm II.2: Finding semantically relevant attributes in SABE-SEK

Input: S_x, the received set of attributes
Output: $S_x{}'$, updated set of attributes

```
Load the ontology
Create a dummy OWLNamedIndividual, TestUser
for every attribute Att_i in S_x do
    if Att_i is a property, e.g., hasTraveled = 3 then
        Create a property assertion axiom for TestUser as (Att_i,
         TestUser, value). For example, hasTraveled and 3 will be used
         as Att_i and value, respectively, for hasTraveled = 3.
        Add the created axiom to the ontology
    else
        Create a class assertion axiom for TestUser as (Att_i, TestUser)
        Add the created axiom to the ontology
Synchronize the reasoner (inference engine) to obtain inferred axioms
Add inferred axioms to the ontology
Realize the ontology to run SWRL rules
S_x' ← Find all the classes (concepts) in the ontology that TestUser
 belongs.
Return S_x' as the updated set of attributes
```

"Surgeon" in the ontology. In other words, some assertions will be generated (based on the submitted attributes) and inserted into the ABox of the knowledge base. Then, a reasoner (an inference engine) will be executed to infer all the semantically relevant attributes. For example, according to the ontology that we developed for this paper, for the *Surgeon* attribute, the inference engine infers *Medical Doctor*, *Physician*, *Lege*[1], and *Person* attributes as semantically relevant attributes based on Definition II.3. However, some more attributes may be derived in the case of having SWRL rules. Therefore, the semantic component extends the set of submitted attributes and returns the extended set to the key generation algorithm, which generates a private key for the extended set of attributes using the `KeyGen` algorithm of a conventional CP-ABE scheme (i.e., using `ABE.KeyGen`). Finally, the inserted assertions (for the attributes submitted by the user) will be deleted from the ABox.

The following example shows the difference between a CP-ABE scheme and the proposed SABE-SEK scheme.

Example II.1. *Suppose that Alice's EHRs are encrypted based on the following AS:* $(Bob \vee (Medical\ Doctor \wedge Employer = Emergency\ Hospital))$*, where Bob is Alice's GP. If Charlie, who is a surgeon working at the Emergency Hospital with attributes* $\{Surgeon,\ Employer = Emergency\ Hospital\}$ *wants to access*

[1]Lege in Norwegian means Medical Doctor in English

Alice's EHRs, he will not be able to decrypt them using a CP-ABE scheme. This is because the CP-ABE works syntactically and cannot infer that a surgeon is a kind of medical doctor. In the proposed SABE-SEK scheme, however, the semantic component infers that Charlie (holding the Surgeon attribute) implicitly has Medical Doctor, Physician, Lege, and Person attributes. Hence, Charlie's private key will be generated according to the original and inferred attributes, i.e., {Surgeon, Medical Doctor, Physician, Lege, Person, Employer = Emergency Hospital}. Therefore, Charlie can decrypt and access Alice's EHRs using the proposed SABE-SEK scheme.

To enable cross-domain interoperability, we need a common ontology for collaborating domains. For example, we can use a common ontology describing the attributes and relationships between attributes in three online social networks, e.g., Facebook, Twitter, and LinkedIn. When a user submits a *Friend* attribute, the semantic component (in SABE-SEK) infers and returns *Follower* and *Connection* attributes, which have the same meaning on Twitter and LinkedIn, respectively. Then, the user's private key will be generated based on *Friend*, *Follower*, and *Connection* attributes. This approach works properly if we have only one TA for all domains. However, in reality, Facebook, Twitter, and LinkedIn have their own TAs for generating private keys. Accordingly, the `KeyGen` algorithm of each domain uses a different master secret key, and thus, for example, Facebook and Twitter generate different private keys for the same attributes. Hence, it can be said that the proposed SABE-SEK scheme, only makes the ABE schemes semantic-aware and does not enable cross-domain interoperability. To provide cross-domain interoperability, we present a second approach called *Semantically-Enriched Access Structure (SEAS)*, which is described in the next subsection.

II.3.2 SEAS: Semantically-Enriched Access Structure

Our second proposal (see Figure II.3) is called SABE-SEAS, and is composed of seven algorithms: `ABE.Setup`, `ABE.KeyGen`, `ABE.Encryption`, `ABE.Decryption`, `SABE.Setup`, `SABE.UpdateAS`, and `SABE.Encryption`; the first four being identical to those of a conventional CP-ABE scheme, while the last three are described below.

The idea of the SABE-SEAS scheme is to enable cross-domain interoperability by enriching the access structures utilizing semantic technologies. In SABE-SEAS, we add the semantic component to the `Encryption` algorithm of a CP-ABE scheme. When a user wants to encrypt data based on an access structure, the proposed SABE-SEAS scheme updates the provided access structure by including semantically relevant attributes (as defined in Definition II.4, and based on Algorithm II.3 and Algorithm II.4) into the access structure and then encrypts the data based on the updated access structure.

As already mentioned, a common ontology describing the attributes and relationships between attributes in different domains is required to provide cross-domain interoperability. We employ a secure signature scheme [12] to guarantee

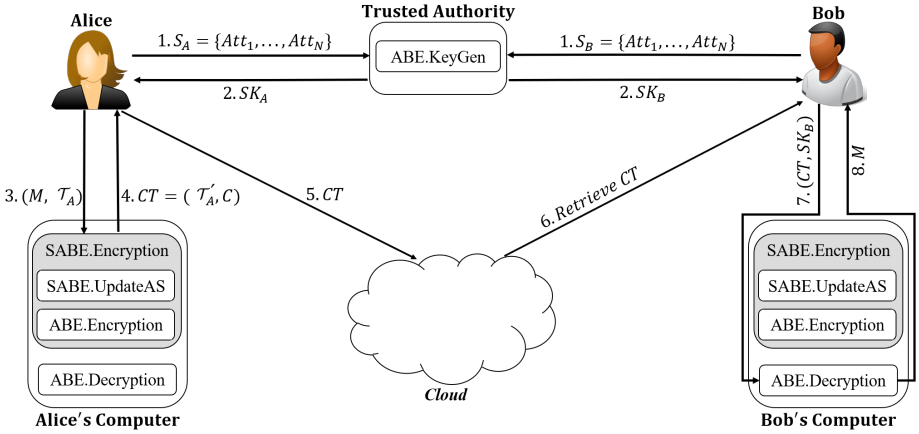


Figure II.3: General architecture of the proposed SABE-SEAS

the validity and authenticity of the ontology. Every TA signs the ontology for its own users using its master secret key and provides the ontology and the corresponding signature to the users as public parameters. The ontology will be validated before updating an access structure by verifying the signature. The goal is to detect unauthorized modifications in the ontology, i.e., the integrity of the ontology, and not the confidentiality of it. Different TAs in SABE-SEAS will generate private keys as usual. A TA, which works independently as in a conventional CP-ABE scheme, does not share any private information or computation with other TAs. In Example II.2, Facebook, Twitter, and LinkedIn do not share any private information with each other (except the common ontology that they agreed upon) and they do not need to trust each other (i.e., external key generators). However, the public keys (or public attributes) of different domains should be publicly available.

Algorithm II.3: Finding semantically relevant attributes in SABE-SEAS

Input: An attribute Att_i
Output: L_i: Attributes that are semantically relevant to Att_i

1 Load the ontology
2 Pre-compute classification (classifying the ontology)
3 Pre-compute instances for each named class in the ontology
4 $L_i \leftarrow$ Find all subclasses of the class Att_i in the ontology
5 $L_i \leftarrow$ Find all named classes that are equivalent to the class Att_i with respect to the set of reasoner axioms
6 Remove the repeated elements from L_i
7 Return L_i as semantically relevant attributes to Att_i

Algorithm II.4: Finding attributes that are semantically relevant to an AND-statement in SABE-SEAS

Input: L_{Input}: Attributes existing in an AND-statement
Output: L_{AND}: Attributes that are semantically relevant to the AND-statement

```
Load the ontology
Create a dummy OWLNamedIndividual, TestUser
for every attribute Att_i in L_Input do
    if Att_i is a property, e.g., property = value then
        Create a property assertion axiom for TestUser as (Att_i,
         TestUser, value), where Att_i is a property.
        Add the created axiom to the ontology
    else
        Create a class assertion axiom for TestUser as (Att_i, TestUser)
        Add the created axiom to the ontology
Synchronize the reasoner (inference engine) to obtain inferred axioms
Add inferred axioms to the ontology
Realize the ontology to run SWRL rules
L_AND ← Find all the classes (concepts) in the ontology that TestUser
 belongs.
Remove superclasses and equivalent classes of the attributes in L_Input
 from L_AND
Return L_AND as semantically relevant attributes to the AND-statement
```

SABE.Setup(1^λ, *Ontology*) is run by the TA and takes as input a security parameter and an ontology (which can be a common ontology for several domains). It calls `ABE.Setup` to generate a master secret key (MK) and a set of public parameters (PP), after which it signs the *Ontology* using a secure signature scheme [12] with the master secret key MK and adds the *Ontology* and the produced signature σ to the public parameters.

SABE.Encryption(M, $\mathcal{T}$, *Ontology*, σ, PP) is run by users, connecting the `ABE.Encryption` algorithm to the semantic component, taking as input a message, M, an access structure, $\mathcal{T}$, a common ontology, *Ontology*, and the ontology's signature generated by the TA. The ontology and the corresponding signature are provided to users as public parameters. As demonstrated in Algorithm II.5, the `SABE.Encryption` algorithm first calls `SABE.UpdateAS` to update the provided access structure based on the semantic relationships defined in the ontology, which is then provided to `ABE.Encryption`(M, $\mathcal{T}'$). Finally, it returns a ciphertext $CT = (\mathcal{T}', C)$, where C is the encrypted version of M.

SABE.UpdateAS($\mathcal{T}$, *Ontology*, σ) is run by users, taking as input an access structure, and the signed ontology. `SABE.UpdateAS` first checks the

Algorithm II.5: Pseudocode of SABE-SEAS encryption

Input: Plaintext(M), Access structure ($\mathcal{T}$), *Ontology*, Signature of the Ontology (σ), Public parameters (PP)

Output: Ciphertext (CT)

```
// Updating the access structure (T) using a domain ontology (after
   validating the authenticity of Ontology using its signature, σ).  T'
   denotes the updated access structure.
```

1 Call `SABE.UpdateAS`,
$\mathcal{T}' \leftarrow$ `SABE.UpdateAS(`$\mathcal{T}$`, Ontology, `σ`)`

```
// Encrypting a plaintext based on the updated access structure using
   the Encryption algorithm of a CP-ABE scheme.
```

2 Call ABE encryption,
$CT \leftarrow$ `ABE.Encryption(`M`, `$\mathcal{T}'$`)`

3 Return CT

signature and then returns an updated access structure, $\mathcal{T}'$, using semantic relationships inferred using the ontology.

For example, suppose that in SABE-SEAS, a user wants to encrypt data based on the following access structure: $(Att_a \vee Att_b) \wedge (Att_c \vee Att_d)$.

Assume that in the common ontology Att_{a1} and Att_{a2} attributes are semantically relevant to Att_a attribute and Att_{b1}, Att_{c1}, and Att_{d1} are semantically relevant to Att_b, Att_c, and Att_d attributes, respectively (according to Definition II.4). Furthermore, there is a SWRL rule stating that $(Att_a \wedge Att_d)$ is the same as Att_e.

The provided access structure will be updated as follows: $((Att_a \vee Att_{a1} \vee Att_{a2} \vee Att_b \vee Att_{b1}) \wedge (Att_c \vee Att_{c1} \vee Att_d \vee Att_{d1})) \vee Att_e$.

Then, the data will be encrypted based on the updated access structure. Note that Att_{a1}, Att_{a2}, Att_{b1}, Att_{c1}, and Att_{d1} could be the attributes of different domains, which are publicly available. It should be noted that for encryption, we only need the name of attributes (or their public keys) and not any private key related to the attributes.

The proposed SABE-SEAS scheme updates access structures based on Algorithm II.6.

The following example, which is based on Figure II.4, demonstrates how the proposed SABE-SEAS scheme enables cross-domain interoperability. Alice is a user who has an account in three social networks, e.g., Facebook, Twitter, and LinkedIn. Bob and Charlie are close friend and friend, respectively, of Alice on Facebook. David and Alice follow each other on Twitter and Emily and Alice are connected on LinkedIn.

Example II.2. *Suppose Alice wants to share a post on Twitter with her followers. Using a CP-ABE scheme, her post will be encrypted based on the following AS:* $(Follower = Alice)$*. It is obvious that Bob, Charlie, and Emily, who are connected to Alice on Facebook and LinkedIn, cannot decrypt and access Alice's*

Algorithm II.6: Pseudocode for updating an access structure

Input: $\mathcal{T}$, the access structure to be updated
Output: $\mathcal{T}'$, updated access structure

```
for every attribute Att_i in T do
    List_A ← Find semantically relevant attributes according to
     Algorithm II.3
    if List_A ≠ ∅ then
        Construct an OR-statement of Att_i and all attributes in List_A
        Replace Att_i with the constructed OR-statement in T
    else
        Keep the attribute Att_i in T as it is
for every AND-statement do
    List_B ← Find attributes that are semantically relevant to the
     AND-statement according to Algorithm II.4
    if List_B ≠ ∅ then
        Construct an OR-statement of the AND-statement and all
         attributes in List_B
        Replace the AND-statement with the constructed statement in T
    else
        Keep the AND-statement in T as it is
for every OR-statement do
    Remove the repeated attributes
Return the result as the updated access structure, T'
```

post on Twitter as they do not have the related private key. However, in the proposed SABE-SEAS scheme, the provided AS: $(Follower = Alice)$ *will be updated as* $((Follower = Alice) \vee (Connection = Alice) \vee (Intimate\ Fried = Alice)) \vee (Friend = Alice) \vee (Close\ Friend = Alice)$ *with the help of the semantic component, where* $(Connection = Alice)$ *is an attribute on LinkedIn and* $(Close\ Friend = Alice)$, $(Intimate\ Friend = Alice)$, *and* $(Friend = Alice)$ *are attributes on Facebook. Then, Alice's post on Twitter will be encrypted based on the updated AS, which means those who are connected to Alice on LinkedIn and Facebook can also decrypt and access Alice's post on Twitter as they have the private keys related to* $(Connection = Alice)$, $(Close\ Friend = Alice)$, $(Intimate\ Friend = Alice)$, *and/or* $(Friend = Alice)$ *attributes.*

Example II.2 shows how SABE-SEAS not only makes the ABE schemes semantic-aware but also enables cross-domain interoperability.

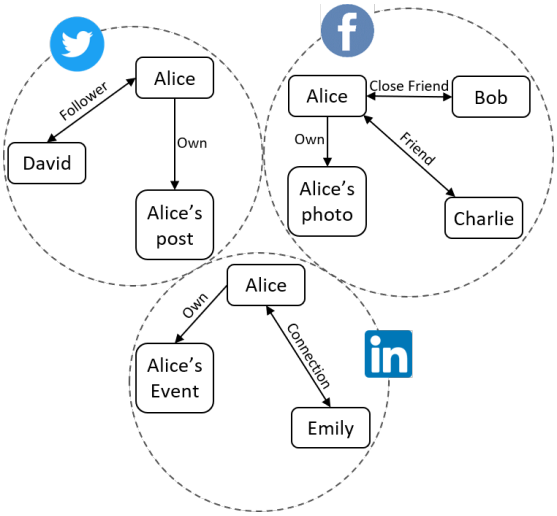


Figure II.4: An example for interoperability

II.4 Security Analysis

II.4.1 Security Assumptions

We consider the following assumptions:

- In conventional CP-ABE schemes, a TA holds a master secret key and generates private keys. In our proposals a TA has the same trust level; particularly, in SABE-SEK, the TA keeps also the ontology secret.

- We assume that the ontology that is provided to a TA during system initialization can be considered trusted in the sense that it demonstrates the correct relationships between attributes (concepts).

- In SABE-SEAS, when different domains form a federation, all the parties, i.e., TAs, should agree on a common ontology. Thus, one TA cannot change the common ontology alone and without the consent of other TAs.

II.4.2 Security Model

The security model for both SABE-SEK and SABE-SEAS is defined using the following game, which is based on the classical indistinguishable encryption against chosen-plaintext attacks (IND-CPA) and mainly considers the confidentiality of the ciphertext.

- **Init phase**: An adversary $\mathcal{A}$ chooses an access structure $\mathcal{T}$ and sends it to a challenger $\mathcal{C}$.
- **Setup phase**: $\mathcal{C}$ generates the master secret key MK and public parameters PP by running the `Setup` algorithm. Then, $\mathcal{C}$ sends the public parameters PP to $\mathcal{A}$ and keeps the master secret key MK secret, whereas particularly for SABE-SEK $\mathcal{C}$ keeps also the ontology secret. However, in the proposed SABE-SEAS scheme, $\mathcal{C}$ generates a signature for the provided ontology and gives the ontology and the corresponding signature to $\mathcal{A}$ as part of the public parameters PP.
- **Phase 1**: $\mathcal{A}$ asks (like any user) from $\mathcal{C}$ private keys related to any sets of attributes. In SABE-SEK, $\mathcal{C}$ runs the `SABE.KeyGen` to generate private keys using updated attributes, whereas in SABE-SEAS it only runs the `ABE.KeyGen`.
- **Challenge phase**: $\mathcal{A}$ submits two messages M_0 and M_1 of equal length. $\mathcal{C}$ selects a random bit $b \in \{0, 1\}$, encrypts M_b under the access structure $\mathcal{T}$, and returns the produced ciphertext CT^*. Note that in SABE-SEAS, $\mathcal{C}$ encrypts M_b under an updated access structure generated by `SABE.UpdateAS`.
- **Phase 2**: $\mathcal{A}$ repeats **Phase 1** multiple times.
- **Guess phase**: $\mathcal{A}$ outputs its guess $b' \in \{0, 1\}$.

$\mathcal{A}$ wins the game if (i) $b = b'$ and (ii) none of the sets of attributes that were requested by $\mathcal{A}$ (including, in the case of SABE-SEK, also attributes inferred through `SABE.UpdateAtt`) satisfy the access structure that was used for encryption (where in the case of SABE-SEAS is the one updated through `SABE.UpdateAS`). The advantage of the adversary $\mathcal{A}$ is defined as the quantity

$$Adv_{\mathcal{A}}^{IND-CPA} = |Pr[b = b'] - \frac{1}{2}|.$$

Definition II.5 (IND-CPA Secure)**.** *An SABE framework (both SABE-SEK and SABE-SEAS schemes) is* IND-CPA secure *iff* $Adv_{\mathcal{A}}^{IND-CPA}$ *is negligible for any probabilistic polynomial time (PPT) adversary.*

For the SABE-SEAS scheme we need something more because here an adversary may also modify the common ontology. Since we employ a secure signature scheme [12], for SABE-SEAS we consider, in addition to the IND-CPA game, also the following game that is based on the Existential Unforgeability under Chosen Message Attacks (EUF-CMA) [14, 20].

- **Setup phase**: exactly as in the previous game.
- **Queries phase**: $\mathcal{A}$ repeatedly requests signatures for chosen messages $(M_1, \ldots, M_j)$, and receives corresponding signatures $(\sigma_1, \ldots, \sigma_j)$ from $\mathcal{C}$. Here we treat ontologies as messages.

- **Forgery phase**: In the end, $\mathcal{A}$ outputs a message M^* (i.e., an ontology) and a signature σ^*.

$\mathcal{A}$ *wins* the game if (i) M^* was not among those messages requested by $\mathcal{A}$ in the **Queries phase**, and (ii) the signature σ^* can be verified correctly using the public key of the trusted authority. The advantage of the adversary in this game is defined as the quantity

$$Adv_{\mathcal{A}}^{EUF-CMA} = Pr[\mathcal{A}\ wins].$$

Definition II.6 (EUF-CMA Secure)**.** *An SABE-SEAS scheme is EUF-CMA secure iff* $Adv_{\mathcal{A}}^{EUF-CMA}$ *is negligible for any PPT adversary.*

II.4.3 Security Proofs

The security of our SABE-SEK and SABE-SEAS schemes can be proved by reduction to the underlying CP-ABE [11] and signature [12] schemes, i.e., if there is an attack against SABE, then the same attack can be used to break the underlying CP-ABE and/or signature schemes (but these have already been proven to be secure).

Theorem II.1. *Both SABE-SEK and SABE-SEAS schemes are IND-CPA secure provided that the underlying CP-ABE scheme is IND-CPA secure.*

Proof. Assume that there exists a PPT adversary $\mathcal{A}$ that can break the proposed SABE with advantage ϵ. We can construct a simulator $\mathcal{B}$ to break the underlying CP-ABE scheme with the same advantage ϵ as follows, where $\mathcal{B}$ will play two roles at the same time: (1) the challenger for the adversary $\mathcal{A}$ in the IND-CPA game for SABE; and (2) the adversary for the challenger in the IND-CPA game for the underlying CP-ABE scheme.

- **Init phase**: $\mathcal{B}$ receives an access structure $\mathcal{T}$ from $\mathcal{A}$ and sends it to $\mathcal{C}$ (in the CP-ABE scheme).
- **Setup phase**: $\mathcal{C}$ generates the master secret key MK and public parameters PP by running the `Setup` algorithm of the underlying CP-ABE scheme. Then, $\mathcal{C}$ sends PP to $\mathcal{B}$ and keeps MK secret. In SABE-SEK, $\mathcal{C}$ keeps the provided ontology secret as well. Then, $\mathcal{B}$ forwards PP to $\mathcal{A}$. However, in SABE-SEAS, $\mathcal{C}$ generates also a signature for the provided common ontology and gives both to $\mathcal{B}$ as part of the public parameters. Therefore, in SABE-SEAS, $\mathcal{B}$ removes the received common ontology and its signature from the public parameters and sends only the rest to $\mathcal{A}$.
- **Phase 1**: When $\mathcal{B}$ receives a private key query for a set of attributes from $\mathcal{A}$, in SABE-SEAS, it forwards the received set of attributes to $\mathcal{C}$ to get the corresponding private keys from the underlying CP-ABE scheme. However, in SABE-SEK, $\mathcal{B}$ first updates the received set of attributes by calling the `SABE.UpdateAtt` algorithm, then sends the updated set of attributes to $\mathcal{C}$.

In response, $\mathcal{C}$ generates the corresponding private keys using the `KeyGen` algorithm of the underlying CP-ABE scheme and returns the generated private keys to $\mathcal{B}$. Then, $\mathcal{B}$ forwards the received private keys to $\mathcal{A}$ in response to $\mathcal{A}$'s original query.

- **Challenge phase**: $\mathcal{A}$ sends two messages of equal length, M_0 and M_1, to $\mathcal{B}$ who forwards them to $\mathcal{C}$. Then, $\mathcal{C}$ selects a random bit $b \in \{0, 1\}$, encrypts M_b under the access structure that was provided in the **Init phase**, and returns the produced ciphertext CT^* (the output of the `Encryption` algorithm of the underlying CP-ABE scheme) to $\mathcal{B}$ who forwards it to $\mathcal{A}$. Note that in SABE-SEAS, $\mathcal{B}$ asks $\mathcal{C}$ to perform the encryption based on the updated access structure (the result of calling the `SABE.UpdateAS` algorithm).
- **Phase 2**: The same as **Phase 1** multiple times.
- **Guess phase**: $\mathcal{A}$ outputs a guess $c' \in \{0, 1\}$, and then $\mathcal{B}$ sends c' to $\mathcal{C}$.

Based on this simulation game, it is clear that if $\mathcal{A}$ has an advantage ϵ in the IND-CPA game against the proposed SABE schemes, then $\mathcal{B}$ can attack the underlying CP-ABE scheme with the same advantage ϵ. However, the underlying CP-ABE has been proven to be IND-CPA secure [11].

To explain more, recall that the proposed SABE schemes directly use the algorithms of the underlying CP-ABE scheme, i.e., we do not change the functionality of `Setup`, `KeyGen`, `Encryption`, and `Decryption` algorithms of the underlying CP-ABE scheme in any way. Indeed, only the input of the `KeyGen` and `Encryption` algorithms of the underlying CP-ABE may be changed as some more attributes may be added to the set of attributes submitted by the user for the key generation (in SABE-SEK) or to the access structure for the encryption (in SABE-SEAS). However, the type of input is still the same, and thus these two algorithms would function in the same way with the same security guarantees. ■

Even if IND-CPA secure, the SABE-SEAS scheme depends also on the security of the signature scheme [12] employed to defeat ontology modification attacks. Recall that in SABE-SEAS the provided access structure may be updated based on a common ontology. An adversary may launch ontology modification attacks by adding new concepts (or relationships) in such a way that makes the attributes of the attacker be semantically related to (possibly all) other concepts, which in turn would allow to decrypt (possibly any) ciphertext.

Theorem II.2. *The proposed SABE-SEAS scheme is secure provided that the underlying CP-ABE scheme is IND-CPA secure and the employed signature scheme is EUF-CMA secure.*

Proof. In SABE-SEAS, the access structure that is updated based on the common ontology is used only after the authenticity and validity of the ontology is checked using the employed signature scheme. If an adversary modifies the common

ontology, then it can be detected before using the access structure. Suppose there exists a PPT adversary $\mathcal{A}$ that can attack the proposed SABE-SEAS scheme by launching ontology modification attacks. We can build a simulator $\mathcal{B}$ that can break the employed signature scheme by using the same actions as $\mathcal{A}$. In other words, the security of the proposed SABE-SEAS can be reduced to the security of the employed signature scheme, which has been proven to be EUF-CMA secure in [12]. ■

II.5 Implementation and Evaluation

We implemented the proposed SABE framework based on both approaches, i.e., Semantically-Enriched Key and Semantically-Enriched Access Structure, in Java. To extend a set of attributes (in SABE-SEK) or update access structures (in SABE-SEAS) using semantic technologies, two simple ontologies are created using Protégé [37] editor (version 5.5.0) and based on the OWL 2 data modeling language. The OWL API [24] (version 5.1.17) is used to deal with the created ontology, e.g., loading the ontology, adding assertions into the ontology, and inferring extra knowledge (implicit knowledge). It means that the semantic component in the proposed framework provides an API by which the set of submitted attributes (for the generation of a private key) or the access structures (for encryption of a data item) may be updated (based on Definition II.3 and Definition II.4).

The default reasoner of the OWL API is the HermiT [19] reasoner. However, the HermiT reasoner does not support SWRL built-in atoms, e.g., *swrlb:greaterThan*. Hence, the Pellet [43] reasoner provided by the Openllet library [18] (version 2.6.1) is used as the inference engine because we used some SWRL rules including SWRL built-in atoms.

As shown in Figure II.5, we implemented SABE in a modular way, being possible to easily replace a module or extend it to any CP-ABE scheme. We made the source code of our implementation publicly available at https://github.com/haamedarshad/SABE-code.

The performance of the proposed SABE framework is evaluated by running the implementation on an Intel Core i7-8550U CPU at 1.80GHz with 32 GB RAM and Windows 10 (64-bit) computer. We added the semantic component to the `KeyGen` algorithm (in SABE-SEK) and `Encryption` algorithm (in SABE-SEAS) of a classical CP-ABE scheme [11, 48]. We executed each of the `KeyGen`, `Encryption`, and `Decryption` algorithms of both the proposed SABE framework (both SABE-SEK and SABE-SEAS) and the underlying CP-ABE scheme [11, 48] 100 times to get the average execution times of the mentioned algorithms. The same input data (with the size of 1MB, 100MB, and 1GB) and access structure are used for encryption and decryption experiments.

Table II.1 shows the average execution times (in milliseconds) of SABE (both SABE-SEK and SABE-SEAS) and the classical CP-ABE scheme. The differences between the execution times (in milliseconds and with 95% confidence intervals) of the `KeyGen` and `Encryption` algorithms of SABE and those of the classical

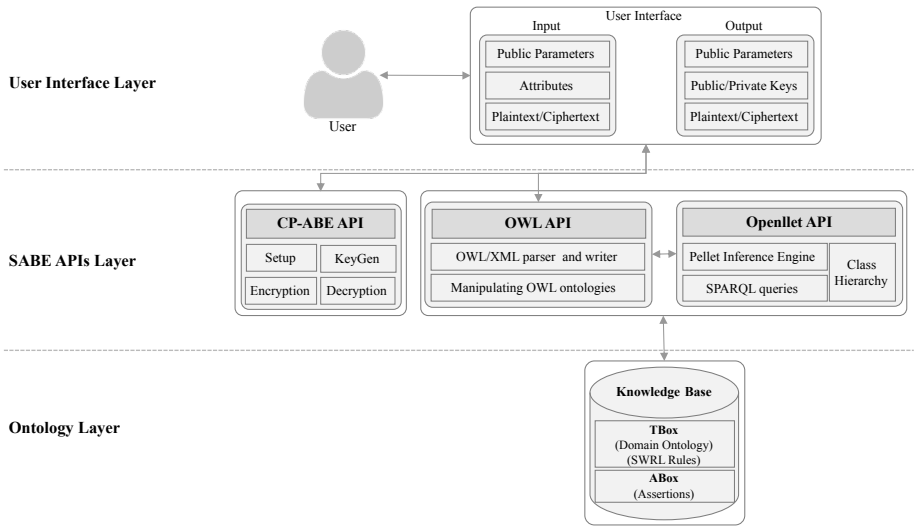


Figure II.5: SABE architecture.

CP-ABE scheme are demonstrated in Figures II.6 and III.9, respectively. Note that the execution time of the `Decryption` algorithm is almost the same for both SABE and CP-ABE.

As can be seen in Table II.1 and Figures II.6 and III.9, the `KeyGen` algorithm in SABE-SEK takes about 5 milliseconds (on average) more than those of the underlying CP-ABE scheme and SABE-SEAS. This difference is because of using the semantic technologies when generating a private key. As explained before, in SABE-SEK, when a request for generating a private key arrives, the set of

Table II.1: Execution time (in milliseconds) of SABE (SEK and SEAS) vs CP-ABE

Algorithm	Input Size	CP-ABE [11]	SABE	
			SEK	SEAS
Key Generation	-	279.97	303.03	279.97
Encryption	1 MB	94.12	94.12	140.49
	100 MB	28.76	228.76	275.13
	1 GB	1595.18	1595.18	1641.55
Decryption	1 MB	33.72	33.83	33.76
	100 MB	381.85	383.42	382.27
	1 GB	3224.38	3266.48	3241.57

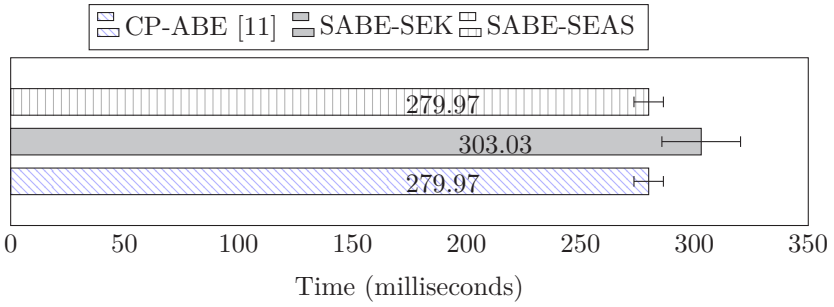


Figure II.6: Execution time for key generation (with 95% confidence intervals).

submitted attributes will be sent to the semantic component, which loads an ontology, adds a few assertions to the ontology (all the attributes submitted by a user will be temporarily added as assertions for a dummy individual into the ABox of the knowledge base), and then performs the semantic reasoning by means of an inference engine to obtain all other attributes that are semantically relevant (according to Definition II.3) to the submitted ones. Next, the semantically relevant attributes will be added to the list of attributes submitted by the user and then the `KeyGen` algorithm generates a private key based on the updated set of attributes. Therefore, the key generation process in SABE-SEK takes 5 more milliseconds, which is negligible.

The execution time of the `Encryption` algorithm in both SABE-SEK and the underlying CP-ABE is the same (see Table II.1). This is because SABE-SEK uses the `Encryption` algorithm of the underlying CP-ABE scheme without any changes. Besides, the input of the `Encryption` algorithm in SABE-SEK is also the same as that in the underlying CP-ABE. However, the `Encryption` algorithm of SABE-SEAS takes in average 46 milliseconds more than those of SABE-SEK and the underlying CP-ABE. This is because in SABE-SEAS, the semantic component will be called to update the provided access structure for each encryption.

In our experiments, there were six attributes in the updated access structure that we used for encryption (in SABE-SEAS). In real scenarios and with large ontologies, more semantically relevant attributes may be inferred from the ontology, and accordingly, the updated access structure may include more attributes. Hence, we performed the encryption (in SABE-SEAS) using an access structure including 50 attributes. The average encryption time for a 1 GB input was 4.2 seconds, which is almost one second more than that with an access structure including six attributes. Therefore, large ontologies may result in bigger access structures (in SABE-SEAS), which in turn the encryption time will be increased. However, the overhead is not very high. Besides, it may be less probable to have more than 50 attributes in an updated access structure.

The size of ciphertexts may increase a little bit (only a few kilobytes as the size of an attribute is a few bytes) in SABE-SEAS as the updated access

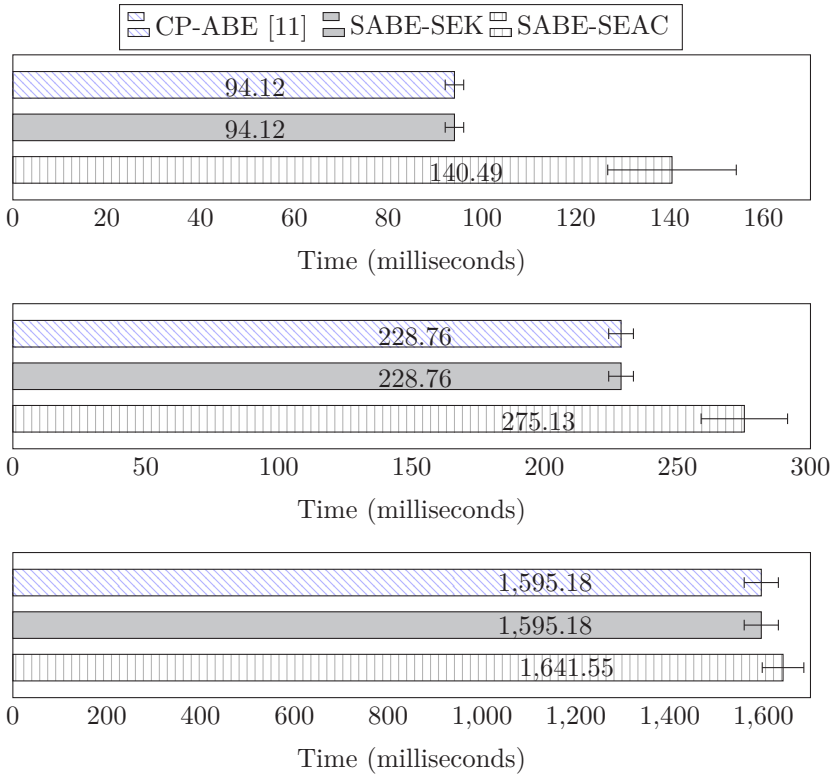


Figure II.7: Execution time for encryption (with 95% confidence intervals). The input size for the top, middle, and bottom charts is 1 MB, 100 MB, and 1 GB, respectively.

structures may include more attributes. The size of private keys in SABE-SEK may also increase in the same way.

Table II.1 also demonstrates that the `Decryption` algorithm of the proposed framework (both SABE-SEK and SABE-SEAS) takes a few milliseconds more than that of the underlying CP-ABE scheme. That might be due to the fact that the private keys (in SABE-SEK) and access structures (in SABE-SEAS) may contain more attributes, and thus checking the compliance between attributes in the private key and the access structure of the ciphertext takes a few more milliseconds. Nevertheless, the difference between the execution time of the `Decryption` algorithm in the proposed framework (both SABE-SEK and SABE-SEAS) and the underlying CP-ABE scheme is negligible.

As a conclusion, it can be said that the overheads associated with adding semantic technologies to the CP-ABE scheme are reasonable. In other words, the overall experiment results are encouraging as such overheads are almost negligible.

II.6 Discussion

In this paper, we proposed two different approaches for augmenting ABE schemes with semantic technologies, i.e., Semantically-Enriched Key (SABE-SEK) and Semantically-Enriched Access Structure (SABE-SEAS). The SABE-SEK scheme makes the ABE schemes semantic-aware; but, it does not facilitate cross-domain interoperability. However, the proposed SABE-SEAS scheme provides both. SABE-SEK increases the time required for generating private keys (as the semantic component will be called for updating a submitted set of attributes) whereas SABE-SEK affects the encryption time (as a provided access structure will be updated utilizing the semantic component). SABE-SEK could be considered more advantageous since: 1) in real-life scenarios, we perform encryption much more frequently than key generation; 2) a TA, which has more resources than a user, runs the `KeyGen` algorithm, whereas the users run the `Encryption` algorithm; 3) in some applications, we may not need cross-domain interoperability so SABE-SEK would be enough (instead of the SABE-SEAS).

In the proposed SABE framework, if the common ontology changes, which is rather infrequent since an ontology describes the relationships between attributes and not users, then the affected private keys (in SABE-SEK) can be revoked [1, 41, 52] and new private keys should be generated according to the new ontology. Besides, ontology changes in SABE-SEAS may affect the existing ciphertexts that can be managed by re-encryption of ciphertexts or other methods [9, 10, 30]. Therefore, aspects regarding the *dynamicity* of the ontology have not been treated in this paper as we consider that they would easily be solved by existing techniques.

The proposed SABE framework (both SABE-SEK and SABE-SEAS) is "CP-ABE agnostic", which means different CP-ABE schemes can be used instead of the one that we used in our implementations. As represented in Figure II.2, SABE-SEK includes `SABE.UpdateAtt` in addition to `ABE.KeyGen` (in the key generation). As demonstrated in Algorithm II.1, `SABE.UpdateAtt` is added for updating the provided set of attributes (based on the semantic relationships between attributes) before calling the `KeyGen` algorithm of a classical CP-ABE scheme (i.e., `ABE.KeyGen`). Figure II.3 and Algorithm II.5 demonstrate that SABE-SEAS includes `SABE.UpdateAS`, which is added for updating access structures before calling the `Encryption` algorithm of a classical CP-ABE scheme (i.e., `ABE.Encryption`). The functionality of `ABE.KeyGen` and `ABE.Encryption` is not changed in any way in the proposed SABE framework. `ABE.KeyGen` and `ABE.Encryption`, respectively, in Algorithm II.1 and Algorithm II.5 are generic constructs that can be realized using different CP-ABE schemes. Besides, as explained throughout the paper, `SABE.UpdateAtt` and `SABE.UpdateAS` are independent of the underlying CP-ABE scheme. They only may update the provided set of attributes, which is an input of `ABE.KeyGen`, and the provided access structure, which is an input of `ABE.Encryption`, by including (in most cases) more attributes based on semantic relationships between attributes. Therefore, any CP-ABE scheme can be extended to a SABE.

II.6.1 Further System Properties

Property 1. *The proposed framework is modular.*

Here we refer to the modularity of the software implementation and architecture for the SABE. As illustrated in Figure II.5, the different modules of SABE are: CP-ABE API, OWL API, Reasoner (Openllet API), User Interface, and a domain ontology. This allows to replace, e.g., the underlying CP-ABE scheme, for reasons of security or performance, without changing other modules. However, we talk about static modularity, for otherwise, if we change the underlying CP-ABE scheme when the framework is in use, i.e., at runtime, then we may not be able to use the updated framework (with a new underlying CP-ABE scheme) for the decryption of the existing ciphertexts because every CP-ABE scheme generates different private keys for the same set of attributes. Nevertheless, this holds for every ABE schemes and that is not a limitation of our proposed framework.

Property 2. *The proposed framework is scalable.*

As described in Section III.4, a common ontology is used to facilitate the interoperability between Facebook, Twitter, and LinkedIn. More online social networks can be added to the scenario by updating the common ontology. The only thing that needs to be done is creating a new ontology, which defines the semantic relationships between attributes of new domains in addition to what exists in the current ontology. Then, thanks to the modularity of the proposed framework, the current ontology can be replaced with the new ontology easily. Therefore, it can be said that the proposed framework is scalable in terms of the number of organizations/domains (in our case, online social networks) collaborating with each other. However, an issue with practical scalability could be the increase in the number of attributes existing in the updated access structures that may increase the encryption time as discussed in Section III.7.

Property 3. *The proposed framework is extensible.*

As explained in Property 1, the proposed framework is modular and every single module can be easily changed. Hence, it is possible to extend the functionalities of the proposed framework by changing the modules. For example, the underlying CP-ABE scheme can be replaced with a new CP-ABE scheme, which may offer extra features like accountable decryption or enforceable obligations. Besides, the semantic component (OWL API and Openllet API) can be extended to provide more implicit knowledge when generating private keys or updating access structures. For instance, more SWRL rules and more advanced relationships can be defined.

Property 4. *The proposed framework is generic.*

SABE can be used in different environments and domains, e.g., eHealth, education, eGovernment, hardware security, cloud computing, etc., other than

online social networks. This can be done by changing only the common ontology that is used in the proposed framework.

II.7 Related Work

In 2005, Sahai and Waters [42] introduced the concept of Attribute-Based Encryption. They proposed a new type of Identity-Based Encryption (IBE) through which one can encrypt a piece of data for a group of recipients enabling multicast encryption [44]. After a year, Goyal et al. [22] proposed a Key-Policy Attribute-Based Encryption (KP-ABE) in which ciphertexts are associated with a set of attributes and private keys are generated based on access structures. Hence, a ciphertext can be decrypted if the access structure of a private key satisfies the attributes required by a ciphertext. Bethencourt et al. [11] proposed the first CP-ABE scheme in which private keys are associated with a set of attributes and the ciphertexts are produced based on access structures. Till now, a considerable number of KP-ABE [29, 38] and CP-ABE [23, 31, 49] schemes have been proposed.

A combination of KP-ABE and CP-ABE, to have both types of ABE at the same time, was proposed by Attrapadung and Imai [3]. Müller et al. [36] proposed a CP-ABE scheme for distributed environments, where several authorities manage attributes and generate private keys. Yu et al. [51] employed proxy re-encryption and lazy re-encryption techniques to improve the efficiency of KP-ABE.

ABE schemes rely on a trusted authority in generating private keys for attributes. The trusted authority, which has full power on private keys, may behave maliciously. Thus, ABE schemes suffer from the key escrow problem. There are a huge number of research studies in the literature [26, 53] addressing the key escrow problem in ABE schemes. For instance, in [16], the key escrow problem was addressed by incorporating several TAs cooperating to generate private keys. However, such a multi-authorities ABE scheme may be susceptible to the collusion of TAs. Hu et al. [26] proposed a multi-authorities CP-ABE scheme addressing the key escrow problem and collusion attacks (i.e., the collusion of the authorities). Zhang et al. [53] proposed a multi-authorities KP-ABE scheme addressing collusion attacks and user privacy. Recently, Zhang et al. [54] proposed a novel CP-ABE scheme addressing the key escrow problem and user revocation. Li et al. [32] proposed a CP-ABE scheme that provides accountability in white-box model and addresses the privacy issues through policy hiding.

Tang and Ji [45] added a verification property to both single-authority and multi-authorities versions of KP-ABE, by which users can verify the correctness of the received private keys as errors may occur during creation or transmission of the keys. Wang et al. [46, 47] combined a hierarchical IBE scheme and a CP-ABE scheme to address the revocation problem in ABE schemes (revoking access rights from users who are no longer legitimate).

There are other research studies [17, 33] reducing the decryption overhead by means of decryption sharing and outsourcing.

II.8 Conclusions

In this paper, we have proposed the first semantic-aware attribute-based encryption framework called SABE, described in Section III.4. We have proposed two different schemes: Semantically-Enriched Key (SEK) and Semantically-Enriched Access Structure (SEAS) using CP-ABE as a baseline scheme for both SABE-SEK and SABE-SEAS. In SABE-SEK, we have modified the key generation process by adding the support for semantic reasoning. The goal was to make CP-ABE schemes semantic-aware by taking into account the semantics of attributes. Algorithm II.1 and Figure II.2 demonstrate that any CP-ABE scheme can be extended to a SABE-SEK scheme by calling `SABE.UpdateAtt`, which updates the provided set of attributes based on the semantic relationships between attributes as defined in a domain ontology, before calling the `KeyGen` algorithm of a classical CP-ABE scheme. The proposed SABE-SEK scheme makes CP-ABE schemes semantic-aware as demonstrated in Example II.1; however, it does not enable cross-domain interoperability as different domains have different trusted authorities with different master secret keys. To provide cross-domain interoperability, we have presented the SABE-SEAS scheme in Section II.3.2. In SABE-SEAS, we have used semantic technologies to update the access structures by including semantically relevant attributes in the access structures as illustrated in Algorithm II.6. In other words, any CP-ABE scheme can be extended to a SABE-SEAS scheme by updating access structures (through `SABE.UpdateAS`) before encryption using the `Encryption` algorithm of a classical CP-ABE scheme, as demonstrated in Figure II.3 and Algorithm II.5. Example II.2 demonstrates that SABE-SEAS not only makes CP-ABE schemes semantic-aware but also facilitates cross-domain interoperability.

We have formally verified the security of the proposed SABE-SEK and SABE-SEAS schemes. We have also implemented a prototype of both schemes by extending a classical CP-ABE scheme in a modular way as demonstrated in Figure II.5. The source codes have been made publicly available for further research.

We have evaluated the effectiveness of the proposed SABE framework by comparing the performance of the proposed SABE-SEK and SABE-SEAS schemes and the underlying CP-ABE scheme in Section III.7. The results of our experiments, as shown in Table II.1, demonstrate that the key generation in SABE-SEK and encryption in SABE-SEAS take a few milliseconds (on average) more than those in the underlying CP-ABE. To assess the effect of the number of attributes on the performance, we have performed the encryption in SABE-SEAS using an access structure including 50 attributes. The results show that the average encryption time for a 1 GB input with a 50-attribute access structure is almost one second more than that with an access structure including six attributes. The results show also that the size of ciphertexts in SABE-SEAS and the size of private keys in SABE-SEK are increased (only a few kilobytes as the size of an attribute is a few bytes) compared to those in the underlying CP-ABE. Therefore, the overall experiment results confirm that SABE improves interoperability and functionality with negligible overheads.

For future work, we plan to incorporate the social network graphs in the domain ontology. Relationships between users change dynamically and sometimes quickly; thus, taking into account the dynamicity of the relationships between users in the social networks improves the quality of the cryptographic access control systems. Besides, we will include the contextual information surrounding users and resources in the social networks to provide a more fine-grained protection.

References

[1] Al-Dahhan, R. R. et al. "Survey on Revocation in Ciphertext-Policy Attribute-Based Encryption". In: *Sensors* vol. 19, no. 7 (2019), p. 1695.

[2] Antoniou, G. and Harmelen, F. van. *A semantic web primer*. MIT Press, 2004.

[3] Attrapadung, N. and Imai, H. "Dual-Policy Attribute Based Encryption". In: *International Conference on Applied Cryptography and Network Security*. Vol. 5536. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2009, pp. 168–185.

[4] Baader, F. et al. *Introduction to description logic*. Cambridge University Press, 2017.

[5] Baader, F. et al., eds. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[6] Baden, R. et al. "Persona: An Online Social Network with User-defined Privacy". In: *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*. New York, NY, USA: ACM, 2009, pp. 135–146.

[7] Barua, M., Lu, R., and Shen, X. "SPS: Secure personal health information sharing with patient-centric access control in cloud computing". In: *IEEE global communications conference (GLOBECOM)*. Atlanta, GA, USA: IEEE, 2013, pp. 647–652.

[8] Bechhofer, S. et al. "OWL web ontology language reference". In: *W3C Recommendation* vol. 10, no. 02 (2004).

[9] Belguith, S., Kaaniche, N., and Russello, G. "PU-ABE: Lightweight attribute-based encryption supporting access policy update for cloud assisted IoT". In: *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. IEEE. 2018, pp. 924–927.

[10] Belguith, S. et al. "PROUD: Verifiable privacy-preserving outsourced attribute based signcryption supporting access policy update for cloud assisted IoT applications". In: *Future Generation Computer Systems* vol. 111 (2020), pp. 899–918.

[11] Bethencourt, J., Sahai, A., and Waters, B. "Ciphertext-Policy Attribute-Based Encryption". In: *IEEE symposium on security and privacy (SP'07)*. Berkeley, CA, USA: IEEE, 2007, pp. 321–334.

[12] Boneh, D. and Boyen, X. “Short Signatures Without Random Oracles and the SDH Assumption in Bilinear Groups”. In: *J. Cryptol.* Vol. 21, no. 2 (2008), pp. 149–177.

[13] Boneh, D. and Franklin, M. “Identity-based encryption from the Weil pairing”. In: *Annual international cryptology conference*. Berlin, Heidelberg: Springer, 2001, pp. 213–229.

[14] Boneh, D., Shen, E., and Waters, B. “Strongly Unforgeable Signatures Based on Computational Diffie-Hellman”. In: *Public Key Cryptography*. Vol. 3958. Lecture Notes in Computer Science. Springer, 2006, pp. 229–240.

[15] Brickley, D. “RDF vocabulary description language 1.0: RDF schema”. In: *http://www. w3. org/TR/rdf-schema/* (2004).

[16] Chase, M. “Multi-authority Attribute Based Encryption”. In: *Theory of cryptography conference*. Berlin, Heidelberg: Springer, 2007, pp. 515–534.

[17] Chen, N. et al. “Efficient CP-ABE Scheme With Shared Decryption in Cloud Storage”. In: *IEEE Transactions on Computers* vol. 71, no. 1 (2022), pp. 175–184.

[18] Galigator, A. *Openllet: An Open Source OWL DL reasoner for Java*. https://github.com/Galigator/openllet. 2020.

[19] Glimm, B. et al. “HermiT: an OWL 2 reasoner”. In: *Journal of Automated Reasoning* vol. 53, no. 3 (2014), pp. 245–269.

[20] Goldwasser, S., Micali, S., and Rivest, R. “A "Paradoxical" Solution To The Signature Problem”. In: *25th Annual Symposium onFoundations of Computer Science, 1984*. 1984, pp. 441–448.

[21] Gorbunov, S., Vaikuntanathan, V., and Wee, H. “Attribute-based encryption for circuits”. In: *Journal of the ACM (JACM)* vol. 62, no. 6 (2015), pp. 1–33.

[22] Goyal, V. et al. “Attribute-based Encryption for Fine-grained Access Control of Encrypted Data”. In: *Proceedings of the 13th ACM Conference on Computer and Communications Security*. CCS ’06. Alexandria, Virginia, USA: ACM, 2006, pp. 89–98.

[23] Goyal, V. et al. “Bounded ciphertext policy attribute based encryption”. In: *International Colloquium on Automata, Languages, and Programming*. Berlin, Heidelberg: Springer, 2008, pp. 579–591.

[24] Horridge, M. and Bechhofer, S. “The owl api: A java api for owl ontologies”. In: *Semantic web* vol. 2, no. 1 (2011), pp. 11–21.

[25] Horrocks, I. et al. “SWRL: A Semantic Web Rule Language Combining OWL and RuleML”. In: *W3C Member submission* vol. 21 (2004), p. 79.

[26] Hu, S., Li, J., and Zhang, Y. “Improving Security and Privacy-Preserving in Multi-Authorities Ciphertext-Policy Attribute-Based Encryption”. In: *KSII Transactions on Internet & Information Systems* vol. 12, no. 10 (2018), pp. 5100–5119.

[27] Hu, V. C. et al. “Guide to Attribute Based Access Control (ABAC) Definition and Considerations”. In: *NIST Special Publication (SP)* vol. 800, no. 162 (2014), pp. 1–47.

[28] Jiang, Y. et al. “Ciphertext-policy attribute-based encryption against key-delegation abuse in fog computing”. In: *Future Generation Computer Systems* vol. 78 (2018), pp. 720–729.

[29] Lewko, A., Sahai, A., and Waters, B. “Revocation systems with very small private keys”. In: *IEEE Symposium on Security and Privacy*. Oakland, CA, USA: IEEE, 2010, pp. 273–285.

[30] Li, J. et al. “An efficient attribute-based encryption scheme with policy update and file update in cloud computing”. In: *IEEE Transactions on Industrial Informatics* vol. 15, no. 12 (2019), pp. 6500–6509.

[31] Li, J., Chen, N., and Zhang, Y. “Extended File Hierarchy Access Control Scheme with Attribute-Based Encryption in Cloud Computing”. In: *IEEE Transactions on Emerging Topics in Computing* vol. 9, no. 2 (2021), pp. 983–993.

[32] Li, J. et al. “Attribute Based Encryption with Privacy Protection and Accountability for CloudIoT”. In: *IEEE Transactions on Cloud Computing* vol. 10, no. 2 (2022), pp. 762–773.

[33] Li, J. et al. “Full Verifiability for Outsourced Decryption in Attribute Based Encryption”. In: *IEEE Transaction on Services Computing* vol. 13, no. 3 (2020), pp. 478–487.

[34] Li, J. et al. “Secure attribute-based data sharing for resource-limited users in cloud computing”. In: *Computers & Security* vol. 72 (2018), pp. 1–12.

[35] Liu, J., Huang, X., and Liu, J. K. “Secure sharing of Personal Health Records in cloud computing: Ciphertext-Policy Attribute-Based Signcryption”. In: *Future Generation Computer Systems* vol. 52 (2015). Special Section: Cloud Computing: Security, Privacy and Practice, pp. 67–76.

[36] Müller, S., Katzenbeisser, S., and Eckert, C. “Distributed attribute-based encryption”. In: *International Conference on Information Security and Cryptology*. Seoul, Korea: Springer, 2008, pp. 20–36.

[37] Musen, M. A. “Protégé ontology editor”. In: *Encyclopedia of Systems Biology* (2013), pp. 1763–1765.

[38] Ostrovsky, R., Sahai, A., and Waters, B. “Attribute-based encryption with non-monotonic access structures”. In: *Proceedings of the 14th ACM conference on Computer and communications security*. CCS ’07. Alexandria, Virginia, USA: ACM, 2007, pp. 195–203.

[39] Parducci, B., Lockhart, H., and Rissanen, E. “Extensible access control markup language (XACML) version 3.0”. In: *OASIS Standard* vol. 2013, no. 1 (2013), pp. 1–154.

[40] Picazo-Sanchez, P., Pardo, R., and Schneider, G. “Secure photo sharing in social networks”. In: *IFIP International Conference on ICT Systems Security and Privacy Protection*. Rome, Italy: Springer, 2017, pp. 79–92.

[41] Premkamal, P. K., Pasupuleti, S. K., and Alphonse, P. “Dynamic traceable CP-ABE with revocation for outsourced big data in cloud storage”. In: *International Journal of Communication Systems* vol. 34, no. 2 (2021), e4351.

[42] Sahai, A. and Waters, B. “Fuzzy identity-based encryption”. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Aarhus, Denmark: Springer, 2005, pp. 457–473.

[43] Sirin, E. et al. “Pellet: A practical owl-dl reasoner”. In: *Web Semantics: science, services and agents on the World Wide Web* vol. 5, no. 2 (2007), pp. 51–53.

[44] Sookhak, M. et al. “Attribute-based data access control in mobile cloud computing: Taxonomy and open issues”. In: *Future Generation Computer Systems* vol. 72 (2017), pp. 273–287.

[45] Tang, Q. and Ji, D. “Verifiable Attribute Based Encryption”. In: *IJ Network Security* vol. 10, no. 2 (2010), pp. 114–120.

[46] Wang, G., Liu, Q., and Wu, J. “Hierarchical attribute-based encryption for fine-grained access control in cloud storage services”. In: *Proceedings of the 17th ACM conference on Computer and communications security*. CCS ’10. Chicago, Illinois, USA: ACM, 2010, pp. 735–737.

[47] Wang, G. et al. “Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers”. In: *Computers & Security* vol. 30, no. 5 (2011), pp. 320–331.

[48] Wang, J. *Java Realization for Ciphertext-Policy Attribute-Based Encryption*. https://github.com/junwei-wang/cpabe/. 2012.

[49] Waters, B. “Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization”. In: *International Workshop on Public Key Cryptography*. Taormina, Italy: Springer, 2011, pp. 53–70.

[50] Yeh, L. et al. “Cloud-Based Fine-Grained Health Information Access Control Framework for Lightweight IoT Devices with Dynamic Auditing and Attribute Revocation”. In: *IEEE Transactions on Cloud Computing* vol. 6, no. 2 (2018), pp. 532–544.

[51] Yu, S. et al. “Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing”. In: *INFOCOM 2010. 29th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 15-19 March 2010, San Diego, CA, USA*. IEEE, 2010, pp. 534–542.

[52] Yu, S. et al. “Attribute based data sharing with attribute revocation”. In: *Proceedings of the 5th ACM symposium on information, computer and communications security*. 2010, pp. 261–270.

[53] Zhang, L., Liang, P., and Mu, Y. “Improving privacy-preserving and security for decentralized key-policy attributed-based encryption”. In: *IEEE Access* vol. 6 (2018), pp. 12736–12745.

[54] Zhang, R. et al. “Key Escrow-free Attribute Based Encryption with User Revocation”. In: *Information Sciences* vol. 600 (2022), pp. 59–72.

Paper III

Attribute-Based Encryption with Enforceable Obligations

Hamed Arshad, Pablo Picazo-Sanchez, Christian Johansen, Gerardo Schneider

Journal of Cryptographic Engineering. DOI: 10.1007/s13389-023-00317-1.

Abstract

Attribute-Based Encryption (ABE) is a cryptographic mechanism that provides fine-grained access control to encrypted data, which can thus be stored in, e.g., public clouds. However, ABE schemes lack the notion of *obligations*, which is common in Attribute-Based Access Control systems such as eXtensible Access Control Markup Language and Usage Control. Obligations are used to define and enforce extra constraints that happen before approving or denying an access request. In this paper, we propose Attribute-Based Encryption with enforceable OBligations (OB-ABE), a system for extending any classical ABE with enforceable obligations. Our system architecture has as core component trusted hardware enclaves, implemented with Intel Software Guard Extensions (SGX), used for enforcing obligations. We employ ProVerif to formally model OB-ABE and verify its main property called "enforceable obligations", i.e., if a message is encrypted along with an obligation, then the message can be decrypted only after enforcing the attached obligation. OB-ABE has two more properties: (i) OB-ABE is a "conservative extension" of the underlying ABE scheme, preserving its security properties; (ii) OB-ABE is "backward compatible" in the sense that any ciphertext produced by an ABE scheme can be decrypted by its extended OB-ABE version, and moreover, a ciphertext produced by an OB-ABE scheme can be decrypted by its underlying ABE scheme provided that the ciphertext does not have obligations attached. We also implement in C using Intel SGX a prototype of an OB-ABE extending the well-known Ciphertext-Policy ABE.

III.1 Introduction

Attribute-Based Encryption (ABE) [109] is a public-key encryption scheme that allows storage of sensitive data in a secure manner in untrusted locations such as public clouds. ABE provides fine-grained encryption using an access structure, usually represented as a Boolean formula where the variables are public attributes.

Only users with the public attributes that satisfy the access structure are able to retrieve the plaintext. For instance, if the data is encrypted under the following access structure: $((doctors \vee \text{caregiver}_{\text{id}} = 31415) \vee (researcher \wedge Norway))$, then only doctors, a specific caregiver, or researchers working in Norway, can decrypt the data. One distinctive feature of ABE is that it can encrypt data for a single user (identified by a unique public attribute, such as $\text{caregiver}_{\text{id}}$) as well as for a group of users (identified by a set of public attributes).

ABE has been applied, in multiple variants, e.g., Ciphertext-Policy Attribute-Based Encryption (CP-ABE) or Key-Policy Attribute-Based Encryption (KP-ABE), in many domains such as: hardware security [42], social networks [11, 101], public clouds [12, 71], fog computing [58] or eHealth [72, 76]. Additionally, real world companies like Zeutro[1] have started to deploy security systems based on ABE. Standards have also been defined, for example by ETSI[2] (TS 103 458 and TS 103 532) proposing applications of ABE to, for instance, industrial IoT and cloud.

Attributes have also been used to achieve fine-grained access control to data—without the encryption part—for instance in systems commonly called Attribute-Based Access Control (ABAC) [53, 98, 138] or Usage Control (UCON) [14, 99, 100, 141]. ABAC is the successor of Role-Based Access Control (RBAC) [110] that has reached the maturity of OASIS standards with Security Assertion Markup Language (SAML) 2.0 [80] and eXtensible Access Control Markup Language (XACML) [98]. For instance, XACML provides a fine-grained and declarative policy language, as well as a distributed architecture for ABAC implementations. In ABAC, access decisions are made not only based on the public attributes of the users, similar to what ABE does, but also based on attributes of the data (i.e., the requested object, like confidentiality level), the desired action (e.g., read/write), and conditions of the environment (e.g., time of day), according to predefined access control policies involving all these types of attributes [124, 133, 139].

ABAC has a powerful mechanism implemented by the Policy Enforcement Point (PEP) (one of the XACML components that is responsible for enforcing authorization results) called *obligations* to enforce extra constraints that cannot be managed through normal policies (e.g., writing logs, sending notifications, or asking for confirmations). Obligations are greatly desired, e.g., in eHealth, because of the accountability and highly interactive style of work where many types of actions must be logged, various authorizations are needed from experts (i.e., confirmations, e.g., from the doctor on duty), or simply sending notifications to relevant parties (like to family/guardians) are required by law. ABAC is a popular choice in eHealth systems for managing access control, both for the obligations mechanism as well as for the flexibility given by the attributes and the way they cater for fine-grained access control also in emergencies [5, 48, 57, 83, 87, 106, 107].

[1] https://github.com/zeutro/openabe

[2] https://www.etsi.org/newsroom/news/1328-2018-08-press-etsi-releases-cryptographic-standards-for-secure-access-control

Nevertheless, ABAC systems are not based on cryptographic techniques. Besides, all their components such as Policy Enforcement Point (PEP), Policy Decision Point (PDP), Policy Information Point (PIP), and Policy Administration Point (PAP) are considered as trusted entities. ABAC, like every access control mechanism, relies on a trusted reference monitor that enforces the access control policies (specified by a security administrator) onto data objects. The reference monitor can however be bypassed, e.g., by getting direct access to the data on a storage device. ABE, on the other hand, achieves the same granularity of access control to data by means of encryption, but does not implement obligations. In ABE (more specifically CP-ABE), access structures are specified by data owners and not by a security administrator. Furthermore, in ABE besides dispensing of trusted components (such as the PEP, PAP, and PDP of ABAC), the encryption and decryption operations are distributed among users, thus removing the single point of failure that ABAC systems suffer from.

The aim of this paper is to provide a general way of augmenting ABE schemes with *enforceable obligations*, thus bringing one important feature from ABAC into ABE. We define a system architecture having Intel Software Guard Extensions (SGX) as the core component used for enforcing obligations added on top of any ABE scheme. Our proposal is meant to be general and extensible, and as such, we define a language for describing complex obligations (similar to of XACML [98]) by combining basic actions (obligations) like "sendTextMessage", "sendEmail" or "writeLog" implemented inside SGX. In other words, the Intel SGX is used to provide a Trusted Execution Environment (TEE), which is called a hardware enclave in SGX terms. Even though there are still controversies about SGX [24, 43, 62, 81, 91, 117], the concept of trusted hardware enclaves is what we rely on in this paper, and we assume it is appropriately implemented, in our case by the Intel SGX.

An important aspect of our proposal is that it allows any existing ABE scheme to be extended in the same way. We call such augmented ABE schemes by Attribute-Based Encryption with enforceable OBligations (OB-ABE). In order to demonstrate that the extended versions are compatible with the underlying (baseline) versions, we define and prove a property called *backward compatibility* (see Property III.2 on page 166). In addition, we also prove that such an extension does not affect the security of the underlying ABE scheme (see Property III.3 on page 168).

Our contributions can be summarized as follows:

- We propose OB-ABE as a general extension with obligations of any ABE scheme (Sections III.4.2 to III.4.4).
- For this, we first define in Section III.4.1 a formal language to write complex obligations that must be enforced by a trusted hardware enclave before releasing a plaintext.
- We verify three key properties of OB-ABE, i.e., enforceable obligations, backward compatibility, and conservative extension (Section III.6).

- We provide a prototype implementation of OB-ABE based on Intel SGX (Section III.7).

The rest of the paper is structured as follows: Section IV.4 explains a real-world use-case for OB-ABE. Section III.3 gives needed background information about ABE, Intel SGX, and ProVerif. Section III.5 analyzes the security of the proposed OB-ABE scheme. Related work is presented in Section III.8 and we conclude in Section III.9.

III.2 Motivating use case

We use throughout the paper examples from a pilot on Assisted Living and Community Care Systems (ALCCS) from a project called SCOTT[1]. The ALCCS pilot develops a system, similar to the one pictured in Figure III.1, where Alice, an elderly woman, lives alone in a smart house equipped with various sensors, e.g., to detect whether she is lying down on the floor. Additionally, Alice wears an Elderly UI patch on her body, which continuously monitors the activity level and periodically transmits the measurements to a home edge system (i.e., the Elderly Context Derivation (ECD) in Figure III.1; an instance of fog computing) for storage and further processing. The Elderly UI patch also has a panic button that can be used by Alice if needed. There are also sensors which measure Alice's physiological parameters and send them to a health data broker (a cloud storage). The ECD service uses the information provided by both the Elderly UI and the medical sensors to determine whether or not to automatically raise an emergency. If help is needed, then the corresponding caregivers (these can be both professionals as well as relatives or neighbors, depending on the situation) will be informed. Once a caregiver accepts to help, she will be given temporary access to Alice's home (through a smart door lock).

Due to the sensitive information, the Electronic Patient Record (EPR) of Alice should be stored in an encrypted manner, e.g., there is an increasing trend for hospitals to move their infrastructure and data into clouds. Therefore, one would think of using ABE for storing the EPR of Alice, including any information coming from the daily activities of Alice and her caregivers.

Alice may want to have a policy (i.e., access structure, the same as in our example from the Introduction) as described in Example III.1. This example demonstrates why ABE schemes need to be extended with obligations.

Example III.1. *Doctors and Alice's caregiver (a specific caregiver that is assigned to Alice in advance) are allowed to access Alice's EPRs, but upon access, a text message should be sent to Alice and this access should be logged for audit purposes. Furthermore, researchers in Norway can access Alice's EPRs, but besides notifying Alice and logging the access, an email should be sent to the security department of the research institution (i.e., the employer of the researcher).*

[1] EU Horizon 2020 ECSEL Joint Undertaking project SCOTT – Secure COnnected Trustable Things (https://scottproject.eu/)

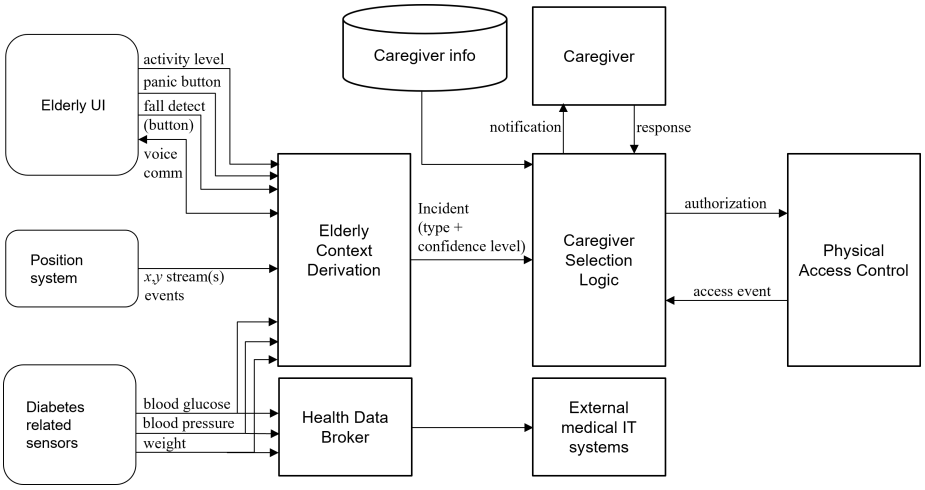


Figure III.1: Architecture of the Assisted Living and Community Care pilot

In this example, there are extra constraints, i.e., sending a text message to Alice, writing a log, and sending an email to the security department, that are not possible to manage using access structures in classical ABE schemes, but can be managed using obligations. Hence, we want to extend ABE schemes to support obligations. As mentioned before, obligations can be used in several scenarios such as medical environments, governmental administrative procedures or public-sector services. For instance, in a medical system one can define an obligation for writing a log for each action taken by the system (which is very useful, and in many cases also required by law, to keep track of "who did what treatment and when"), or an obligation to send notifications and possibly also waiting for an acknowledgment before granting access (e.g., access by a nurse may need acknowledgment from a doctor).

III.3 Preliminaries

III.3.1 Background on Attribute-Based Encryption

In conventional public-key cryptography, data are encrypted for a particular receiver using the receiver's public key. Hence, if the same data should be sent to several receivers, all the public keys of the receivers are needed in advance to encrypt the data for each of them. In response to this, Boneh and Franklin [17] proposed the first practical implementation of IBE in 2001 and a few years later, the first ABE scheme was published by Sahai and Waters [109] by which the encryptor can encrypt a message under a set of public attributes (instead of just an identity as in IBE schemes). Therefore, data can be encrypted for a group of recipients holding the same set of public attributes.

More concretely, ABE is a kind of public-key cryptography in which the

private key of a user and the ciphertext are dependent upon attributes. In ABE (more specifically CP-ABE), an access structure is attached to a ciphertext and the keys of users are associated to sets of public attributes. Hence, anyone who has a subset of attributes that satisfies the access structure of the ciphertext can decrypt it and get the plaintext.

When a user joins the system, she claims to have a set of public attributes and a TA is in charge to validate them. If deemed appropriate, the TA provides her with the private key associated to the public attributes she holds. This authentication process is usually out of the scope of ABE schemes since it is always assumed that the TA has the knowledge—or the corresponding mechanisms—to prove that users really have the attributes they claim to have. Consequently, in this paper we assume that users cannot cheat TA and they are provided with the public attributes they actually have.

A ciphertext can thus be decrypted only if users have a set of public attributes that satisfy the access structure defined by the encryptor and attached to the ciphertext. For example, if a doctor has the following public attributes $S = \{doctor, hospitalA\}$, then she can decrypt ciphertexts encrypted under the following access structure: $\mathcal{T} = ((doctor \vee \text{caregiver}_{\text{id}} = 31415) \vee (researcher \wedge Norway))$. In other words, the set S satisfies $\mathcal{T}$. On the contrary, a researcher who works in Sweden cannot access the data (decrypt the ciphertext) because the attribute set S of the researcher does not satisfy $\mathcal{T}$.

The advantages of using ABE are multiple: i) different groups of users can be defined according to public attributes; ii) all private information can be stored in public databases and it will only be decrypted by users who satisfy the access structure $\mathcal{T}$; and iii) security properties such as access control, user collusions, and data disclosures are guaranteed by the underlying cryptographic infrastructure.

In what follows, we briefly describe the main algorithms that a CP-ABE [15] is made up of: Setup, KeyGen, Encryption, and Decryption. While the first two algorithms are run by the TA, the other two (i.e., Encryption and Decryption) are executed by the users of the system. In more detail:

Setup(1^λ) This algorithm takes a security parameter as input and generates a master secret key, MK, and a set of public parameters, PP.

KeyGen(MK, S, PP) The key generation algorithm produces a private key, SK, for a provided set of attributes, $S = \{Att_1, ..., Att_N\}$, using the master secret key and the public parameters.

Encryption(M, $\mathcal{T}$, PP) encrypts a message M based on the provided access structure, $\mathcal{T}$, and returns a ciphertext $CT = (\mathcal{T}, C)$.

Decryption(CT, SK, PP) decrypts a ciphertext using a provided private key, SK, which is related to a set of attributes satisfying the access structure included in CT.

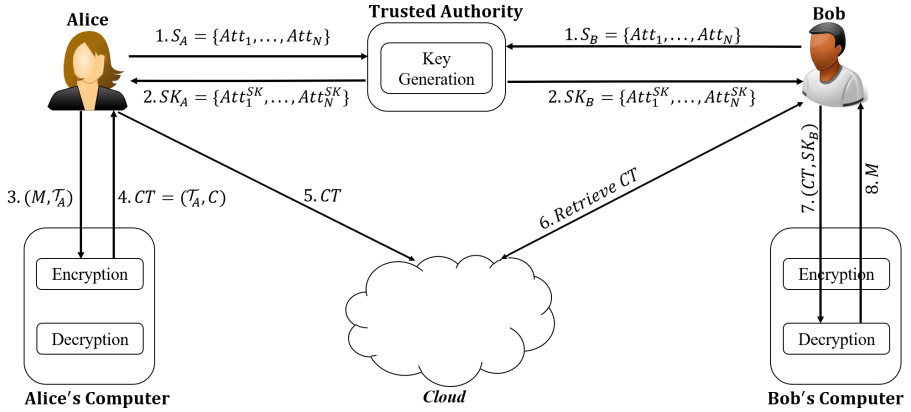


Figure III.2: General architecture of the CP-ABE scheme [15].

Figure III.2 shows an example where two users, Alice and Bob, join the system. First, they provide their public attributes (S_A and S_B) to the TA (1) and they receive the private keys (SK_A and SK_B) associated to theirs attributes (2). After that, Alice runs the Encryption algorithm producing CT (3 and 4) and sends the ciphertext to the cloud where Bob can get it (5). When Bob retrieves CT from the cloud (6), he runs the Decryption algorithm (7) and finally, he obtains the plaintext (8).

III.3.2 Background on Intel Software Guard Extensions

Intel SGX [82] is a set of extensions for secure computation available on Intel's new generation of CPUs (6^{th} generation and later). The goal of SGX is to protect the confidentiality and integrity of the execution code against unauthorized accesses by privileged software such as the operating system, BIOS, or Hypervisor. In fact, it provides a kind of a TEE by means of transparent encryption and enforcing strict hardware access control. Such a trusted (isolated and protected) execution environment is called *enclave* in SGX terminology, where no one can see the computations and secrets inside it. Therefore, the SGX handles secrets and executes software in a trustworthy environment on an untrusted system (the operating system and memory).

The main functionalities of an enclave are *isolation*, *sealing*, and *attestation*. An enclave provides an isolated environment such that the data and code inside an enclave cannot be accessed by other processes. It has a hardware-resident key for sealing (encrypting and authenticating) data passed to the host environment. The code, data, and metadata of an enclave, and the results of computations performed inside the enclave, can be signed and attested by means of local as well as remote attestation. As represented in Figure III.3, an SGX-enabled application includes two parts: trusted and non-trusted, where the trusted part resides encrypted in the memory (i.e., in an enclave).

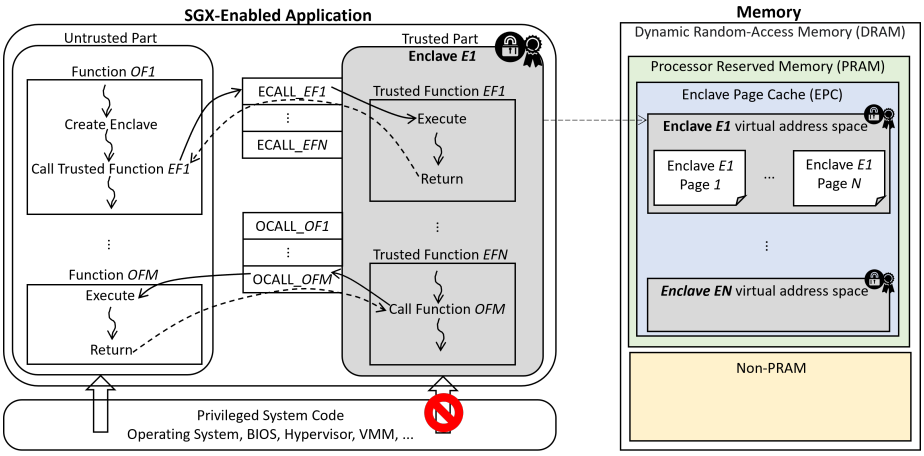


Figure III.3: Structure of an SGX-enabled application and the memory layout

Isolation: An enclave resides in the Enclave Page Cache (EPC), which is a part of the memory guarded by the hardware as shown in Figure III.3. The maximum size of EPC is 120 MB out of which only 90 MB can be used by an application and 4 KB is used for page chunks. The data and code of an enclave will be copied into the EPC (pages inside it) when loading an enclave program and the EPC pages can be accessed only when the processor is running in *enclave mode*. The hash measurement of the page contents, which is called MRENCLAVE, also will be stored along with the data and code inside the EPC. Each enclave will be assigned an identity by which the hardware can control access to the contents of the enclave. The hardware can ensure that pages of an enclave can be accessed only by executable code pages of the same enclave (with the same enclave identity).

Sealing: There is a secret key inside every Intel SGX CPU called the *root seal key* that can be used by enclaves to generate another key for sealing (encrypting and authenticating) their data and code. An enclave generates the sealing key, which is called the *seal key*, using EGETKEY instructions, then it encrypts/authenticates both its data and code with the seal key and stores them in the untrusted memory. The seal key is unique for each enclave, which means that other enclaves (even the ones on the same platform) cannot generate/get the same *seal key*. Therefore, the sealed data can only be accessed/retrieved by the enclave that owns them.

Attestation: The Intel SGX provides local and remote attestations by which different enclaves (on the same or different platforms) can attest each other. By means of the attestation, an enclave can be assured that it is dealing with the right code and data. Secret materials (e.g., secret keys) can then be provided to an enclave (remotely) in a secure manner by means of the attestation process.

The local attestation can be used between two enclaves on the same platform

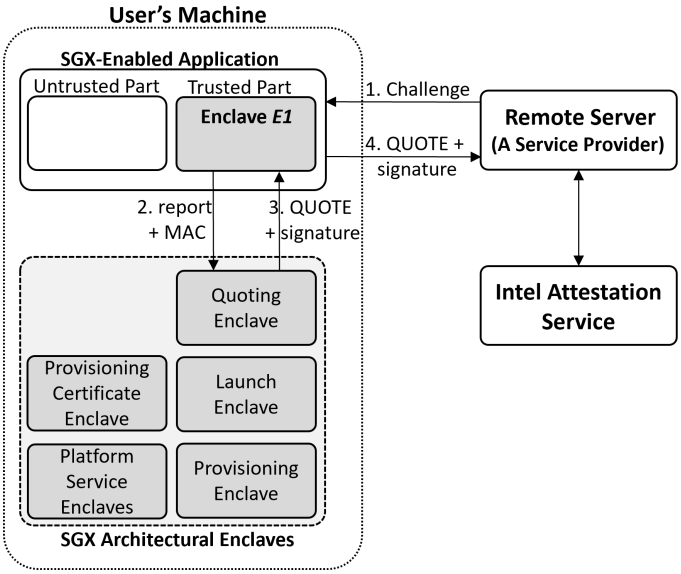


Figure III.4: SGX Remote Attestation. Architectural Enclaves are Intel provided enclaves.

(the same CPU on the same machine). As mentioned before, every Intel SGX processor has a unique root seal key that is the same for all enclaves on the same platform. Different enclaves on the same platform can use that key for the authentication process. An enclave can generate a Message Authentication Code (MAC)—which is called a *report*—for the hash value of its contents (MRENCLAVE) and its metadata with a *Report Key* which is a common key that all SGXs can generate.

The remote attestation provides MAC reports to be verified by enclaves on other platforms (e.g., remote machines). As represented in Figure III.4, when an SGX-enabled application on a user's machine wants to receive some services (for example, secret keys) from a service provider (a remote server) or when a remote server wants to verify that an untampered enclave is running on a legitimate CPU (or to provide, for instance, secret keys to an SGX-enabled application), a challenge message including a nonce will be sent to the SGX-enabled application through a established secure communication (message 1 in Figure III.4). The SGX-enabled application's enclave (in this case, Enclave *E1*) generates a local report, which includes the challenge's response and a temporary key that can be used for key exchange and securing subsequent communication, and a MAC for the report. Then, the SGX-enabled application's enclave provides the generated report and MAC to a special enclave, called Quoting Enclave, for verification and signing (2). The Quoting Enclave, which is one of the Intel provided SGX architectural enclaves and exists on the CPU, verifies the provided report, creates a QUOTE structure for the verified report, signs the QUOTE with a private

key for an anonymous group signature scheme called Intel Enhanced Privacy ID (EPID), and then sends the QUOTE and signature to the SGX-enabled application's enclave (3). The SGX-enabled application forwards the received QUOTE and signature to the remote service provider, i.e., the challenger (4). The remote service provider can verify the received information and the response to the initial challenge using the relevant EPID public key certificate and revocation information obtained through the Intel Attestation Service. The remote service provider can use the temporary key provided by the SGX-enabled application for key exchange.

Since the SGX encrypts both the memory and the plaintext of the secrets available inside the CPU, it is assumed that no one can open the CPU content. SGX has been used in many different contexts to achieve a higher level of security [21, 39, 86, 120, 130].

Interactions: The code executed inside an SGX enclave is considered as the trusted part of an application and can interact with the untrusted part of the application through Enclave Calls (ECALLs) and Outside Calls (OCALLs) as illustrated in Figure III.3. The trusted functions can be invoked using ECALLs. Functions inside an enclave can call untrusted functions (those that are outside SGX enclaves) through OCALLs. In other words, we enter an enclave using an ECALL and inside the enclave we can use an OCALL to do, for example, I/O operations.

III.3.3 Background on ProVerif

We employ the ProVerif formal verification tool [1, 16] to formally verify the main property of our proposed scheme in Section III.6. ProVerif supports symmetric and asymmetric encryption schemes, digital signatures, and hash functions. Such cryptographic primitives can be specified as equations or rewrite rules. ProVerif does not have limitation on the number of sessions (including parallel ones) and message space. In order to formally verify a protocol, the protocol and its desired properties need to be specified in the *typed applied π-calculus* [2, 3]. ProVerif takes the specification as input, converts it into Horn clauses, and then tries to check if the protocol satisfies the desired properties. It can verify different types of properties such as secrecy, authentication, strong secrecy, and equivalence. If a property is not satisfied, then the output shows the trace leading to the violation of that property. ProVerif may return false attacks; however, if it states that the specification satisfies a property, then this is indeed the case.

Table 1 reproduces the syntax of the process calculus employed by ProVerif. A term A (or B) basically contains names (e.g., a, b to denote atomic data items like nonces or keys) and variables (e.g., x, y). It may also contain tuples and constructor/destructor applications (f denotes a function name), each with an arity. Symbols =, <>, &&, ||, and **not**() denote equality, inequality, conjunction, disjunction, and negation, respectively.

Protocols should be represented as processes with the following forms: 0 denotes the null process that performs nothing. In a protocol, several entities

Table III.1: Syntax of the process calculus employed by ProVerif

$A, B ::=$	terms
a, b	names
x, y	variables
$(A_1, \ldots, A_i)$	tuple
$f(A_1, \ldots, A_i)$	constructor/destructor
$A = B$	equality between two terms
$A <> B$	inequality between A and B
$A \&\& B$	conjunction
$A \| B$	disjunction
$\mathbf{not}(A)$	negation
$P, Q ::=$	processes
0	null process
$P \| Q$	parallel composition
$!P$	replication
$\mathbf{new}\ n : t$	name restriction
$\mathbf{in}(ChannelA, m : t)$	receiving a message
$\mathbf{out}(ChannelB, A)$	sending a message
$\mathbf{if}\ A\ \mathbf{then}\ P\ \mathbf{else}\ Q$	condition
$\mathbf{let}\ y = h(A_1, \ldots, A_i)\ \mathbf{in}\ P\ \mathbf{else}\ Q$	term evaluation
$E(A_1, \ldots, A_i)$	macro

(e.g., a server and a client) run in parallel. The parallel composition operator | can be used to combine different processes and model the parallel execution of entities in a protocol. The symbol ! denotes replication of a process as there might be several instances of a process (for example, several clients). The name restriction **new** $n : t$, where t shows the type of the name n, is defined to model fresh random numbers (such as nonces and keys) and secret channels. Since different participants in a protocol communicate through sending and receiving messages, input and output processes are defined. The statement $\mathbf{in}(ChannelA, m : t)$ denotes that the process receives a message m of type t from the channel $ChannelA$ and $\mathbf{out}(ChannelB, A)$ states that the process sends A out on the channel $ChannelB$. The statement **if** A **then** P **else** Q means that the process behaves as the process P if A holds; otherwise, it behaves as the process Q. A statement of the form $\mathbf{let}\ y = h(A_1, \ldots . A_i)\ \mathbf{in}\ P\ \mathbf{else}\ Q$ can capitalize the power of destructors. It evaluates $h(A_1, \ldots . A_i)$ and if it does not fail, then it bounds y to the result of $h(A_1, \ldots . A_i)$ and runs the process P; otherwise, it only runs the process Q. However, if Q is a null process, then the statement can be simplified as $\mathbf{let}\ y = h(A_1, \ldots . A_i)\ \mathbf{in}\ P$. Finally, $E(A_1, \ldots, A_i)$ denotes a macro E with arguments $A_1, \ldots, A_i$.

III.4 The OB-ABE Scheme

In this section, we present OB-ABE, an extension of ABE schemes with obligations. In order to define OB-ABE, we build it upon an existing ABE scheme (in this paper, we use CP-ABE) by only modifying its `Encryption` and `Decryption` algorithms while `Setup` and `KeyGen` algorithms remain unchanged.

OB-ABE is compatible with ABE in the sense that if a ciphertext CT is produced by an ABE, it can also be decrypted by its extended version, OB-ABE. At the same time, a user can produce a ciphertext CT using OB-ABE and users who hold the set of public attributes that satisfies the access structure $\mathcal{T}$ can decrypt CT with ABE provided that the ciphertext does not have obligations to enforce. This *backward compatibility* property is defined and verified in Section III.6 (as Property III.2).

In the following, we explain how we define obligations. Then, we present our proposed OB-ABE scheme by providing the general architecture and giving a more detailed explanation of both the `Encryption` and `Decryption` algorithms (processes) as well as our threat model.

III.4.1 Obligations

Obligations are means by which we can define and enforce extra constraints that are not easy to be specified by an access structure (in classical ABE schemes) and should happen before approving an access request (i.e., before releasing the results of the decryption).

Definition III.1. *An obligation is an operation specified by the data owner (encryptor) that should be performed (enforced) before releasing the results of a decryption.*

Remark III.1 (Types of obligations)**.** *We focus in this paper on what is called pre-obligations in the XACML standard [98]. Pre-obligations need to be performed before the* `Decryption` *algorithm returns a plaintext, e.g., "completing a registration step by the user who decrypts the ciphertext by entering her email address". For post- or ongoing-obligations it is not clear what mechanism to use to enforce them. Post-obligations need to be performed after returning the decrypted result, e.g., "data should be removed within 20 days". Ongoing-obligations have to be performed during data accessing, e.g., anyone who decrypts a ciphertext, has to "keep open a window showing an advertisement during the usage".*

The first part of the policy from Example III.1 (i.e., "Doctors and Alice's caregiver are allowed to access to Alice's EPRs" and "researchers in Norway can access Alice's EPRs"), can be easily specified and enforced by encrypting Alice's EPRs using ABE and a suitable access structure involving attributes of doctors, the caregiver, e.g., assuming the id of Alice's caregiver is 31415, and researchers. Alice's EPRs should be encrypted under the following access structure $\mathcal{T} = ((doctor \vee \text{caregiver}_{\text{id}} = 31415) \vee (researcher \wedge Norway))$. However, the second part (i.e., "a text message should be sent to Alice and this access

$$
\begin{array}{lcl}
OB & ::= & A \mid OB; OB \mid OB + OB \mid OB^* \mid OB \blacktriangleright OB \\
A & ::= & \langle Con, BOB \rangle \mid 1 \\
Con & ::= & BC \mid Con \wedge Con \mid Con \vee Con \mid \neg Con \mid \top \mid \bot \\
BOB & ::= & \textbf{email} \mid \textbf{SMS} \mid \textbf{log} \mid \ldots \\
BC & ::= & isSGXenabled \mid Time \mid \ldots
\end{array}
$$

Table III.2: Language for defining obligations; written in BNF grammar

should be logged for audit purposes" and "but besides notifying Alice and logging the access, an email should be sent to the security department of the research institute") states additional constraints that cannot be expressed with normal access structures. In other words, the first part of the policy expresses *who can access what*, and the second part defines *what must happen before granting the access (i.e., before returning the results of the decryption)*.

We define a language in Table III.2 for forming complex obligation terms out of basic obligations. This is standard in electronic contracts logics [84, 103] or in dynamic logics for reasoning about programs [13, 47], and it is close to regular expressions and Kleene algebras [35, 64]. Since we also use "tests", or conditionals, this obligation language is quite expressive, being able to capture while-programs [65]. We explain shortly our formal notation.

An obligation OB can be: a conditional action (A); a combination of obligations using the following operators: sequential composition (;); branching, or non deterministic choice (+); iteration, or Kleene star (*); *reparations* ($\blacktriangleright$). The skip action (1), is used to make the algebraic theory nice as it is the neutral element for the sequential composition, and used in the unrolling of the iteration (this is what we mean that there is no obligation, i.e., the empty/skip action). Conditional actions (i.e., A) are basic obligations (BOB) prefixed by a condition (Con), which is defined classically using a Boolean algebra over a set of basic conditions (BC). The basic obligations (with examples given in bold font) should be thought as a library of SGX code/programs that we consider as trusted code and that can be updated over time if new basic obligations are needed (that is what we intended to represent by the trailing "...").

Remark III.2. *For the sake of simplicity, we assume the existence of the BOB library of trusted SGX programs that implement various standard actions that one would want to enforce as obligations. However, one can also think of using the attestation mechanism of SGX so that the encryptor can program whatever SGX code as basic obligations in her obligation definition, and then be assured that the decryptor's SGX is executing exactly that code.*

When working with obligations it is common to define *reparations* ($\blacktriangleright$). For example, $OB_1 \blacktriangleright OB_2$ states that if the main obligation, OB_1, is not performed successfully, then the reparation obligation, OB_2, should be executed instead. Reparations can be complex and arbitrarily nested. An obligation, like sending an email to the administrator, can fail due to external factors (e.g., the mail server is down) and then the access request would be denied as the obligation

could not be enforced. Hence, reparations are means to specify ways to treat such situations.

Example III.2 (Complex obligations)**.** *The sequential composition can be used to define a set of obligations all of which need to be performed successfully before access can be granted (before the results of decryption can be returned). We just stack the obligations one after the other. Strictly speaking this is not a set, but a string, because there is also a strict order in which these obligations must be executed.*

One can also define alternatives using the choice operator, where only one of the alternative obligations need to be executed successfully in order for access to be granted. This can be used as a way to mitigate DoS attacks, where the adversary tries to make one of the external services, like the email server, unresponsive in order to disrupt the decryption process (reparations can also be used with the same goal).

Example III.3 (Conditional obligations)**.** *Obligations need to be enforced only if their attached condition holds. If we use* $\langle \top, BOB_1 \rangle$*, then* BOB_1 *is always enabled and must be executed since the condition* $\top$ *is always true. Simple conditions can be, e.g.:* BC_1 *="if the requester is a doctor",* BC_2 *="if the requester is the caregiver*$_{id}$*=31415",* BC_3 *="if the requester is a researcher from Norway", or* BC_4 *="if the current time is between 18:00 and 07:00". Complex conditions can be defined using the Boolean operators, e.g.,* $(BC_1 \wedge BC_2) \vee BC_3$*.*

We can define complex conditional obligations for Example III.1 as follows: "If the requester is a doctor or Alice's caregiver (i.e., $caregiver_{id} = 31415$*), then a text should be sent to Alice and the access should be logged; otherwise, if the requester is a researcher from Norway and it is outside working hours, then besides notifying Alice and writing a log, an email should be sent to the security department" using a term:*

$\langle (BC_1 \vee BC_2), \mathbf{SMS} \rangle; \langle (BC_1 \vee BC_2), \mathbf{log} \rangle + \langle (BC_3 \wedge BC_4), \mathbf{SMS} \rangle; \langle (BC_3 \wedge BC_4), \mathbf{log} \rangle; \langle (BC_3 \wedge BC_4), \mathbf{email} \rangle$.

III.4.2 Architecture of OB-ABE

We present here the OB-ABE architecture and highlight the differences between CP-ABE [15] and OB-ABE. Concretely, we differentiate the classical CP-ABE algorithms with the prefix **ABE** whereas the OB-ABE ones are introduced with the prefix **SGX**.

Figure III.5 shows the architecture of OB-ABE scheme which is made up of seven main algorithms, where `ABE.Setup`, `ABE.KeyGen`, `ABE.Encryption`, and `ABE.Decryption` algorithms are the same as those (Setup, KeyGen, Encryption, and Decryption, respectively) in Section III.3.1.

SGX.Encryption(M**,** OB**)** The trusted encryption algorithm encrypts a message M along with a provided set of obligations, OB, with a secret key, which is

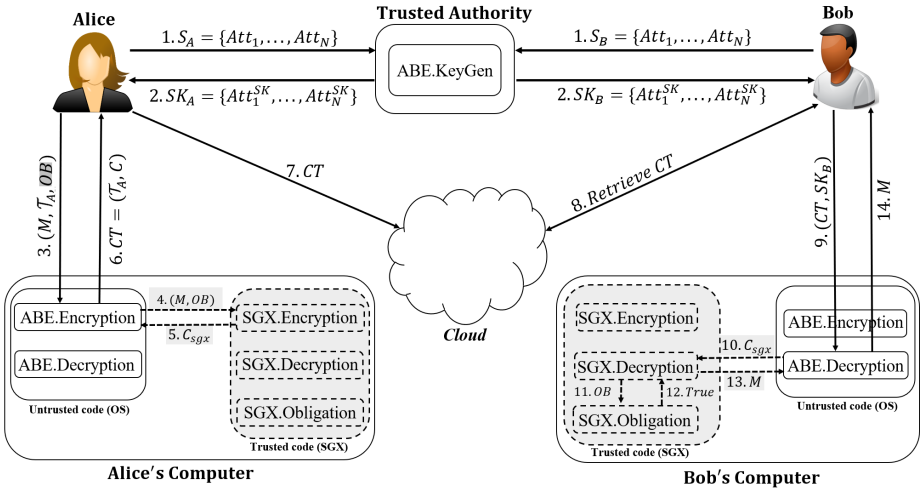


Figure III.5: Architecture of the proposed OB-ABE scheme.

only known to the trusted hardware enclave, using a symmetric encryption scheme, to get a ciphertext C_{sgx}.

SGX.Decryption(C_{sgx}, SK) The trusted decryption (or SGX Decryption) algorithm decrypts a ciphertext, C_{sgx}, with a secret key that is only known to the trusted hardware enclave. After decryption and getting (M, OB), it calls the obligation algorithm (`SGX.Obligation`) to enforce obligations OB. If obligations are enforced successfully, then it returns back the message M; otherwise, it removes the M and returns an ERROR message.

SGX.Obligation(OB) The obligation algorithm takes a set of obligations, OB, as input and enforces them. If the process is executed correctly, it returns True otherwise it returns False.

Differences with CP-ABE:

Comparing our proposed OB-ABE scheme with CP-ABE, there are two main differences. In OB-ABE, both `Encryption` and `Decryption` algorithms are split into two parts, i.e., trusted (`SGX.Encryption`, `SGX.Decryption`) and untrusted (`ABE.Encryption`, `ABE.Decryption`) parts. The trusted code/part runs in an enclave provided by the Intel SGX or other relevant technologies (i.e., in a secure environment in the CPU) and has access to the application's secrets. On the other hand, the operating system manages the untrusted code/part. In order to allow communication between trusted and untrusted parts, the latter can call enclave functions through *ecalls* and the trusted parts can call the untrusted ones using *ocalls*.

The other difference is the inclusion of the new `SGX.Obligation` algorithm, used to enforce extra constraints defined in the form of a complex obligations term. Note that in a standard ABE setting, a user that has the set of attributes satisfying the access structure and knows the decryption algorithm can easily ignore any obligations attached to a ciphertext. Therefore, for OB-ABE we need a trusted component on the user side to enforce such obligations, which in our case is the Intel SGX.

OB-ABE procedure:

We include a graphical schema of our proposed OB-ABE, where the aforementioned differences are marked by dashed-line boxes and gray shaded areas in Figure III.5. In particular, the first two steps are the same as those in CP-ABE from Figure III.2, i.e., Alice and Bob provide their attributes to a TA and receive the corresponding private keys (SK_A and SK_B). After that, Alice encrypts a message, M, providing an access structure $\mathcal{T}_A$ and a set of obligations, OB, under which she wants to encrypt the message (communication 3). The `ABE.Encryption` algorithm calls the trusted `SGX.Encryption` algorithm running inside the enclave and provides both the message M and the obligations OB (4). The trusted `SGX.Encryption` algorithm attaches the obligations, OB, to M, and encrypts the result by using a symmetric encryption algorithm with a secret key that is only known by the trusted hardware enclave (the secret key of the SGX, K_{SGX}). Then, it returns the result, i.e., C_{sgx}, back to the `ABE.Encryption` algorithm (5) to be encrypted with the `ABE.Encryption` algorithm of the CP-ABE, giving the result (CT) back to Alice (6). Alice then stores CT on a public repository (7).

When Bob retrieves CT from the cloud (8), he executes the `ABE.Decryption` algorithm by providing CT and SK_B, which is his private key that satisfies the access structure $\mathcal{T}_A$ (9). The untrusted `ABE.Decryption` algorithm decrypts CT and calls the trusted `SGX.Decryption` algorithm by providing the result of the decryption, i.e., C_{sgx} (10). The trusted `SGX.Decryption` algorithm decrypts the received C_{sgx} with the secret key of the SGX (i.e., K_{SGX}) and obtains the attached obligations, OB. Then, it calls the `SGX.Obligation` algorithm to enforce the obligations (11). If everything goes well, it returns True (12); otherwise, it returns False. After receiving True from the `SGX.Obligation` algorithm, the `SGX.Decryption` returns the plaintext M to the untrusted decryption algorithm, `ABE.Decryption` (13). Finally, the untrusted `ABE.Decryption` algorithm sends back M to Bob (14).

Key provisioning:

There are two ways to provide the SGX secret key, K_{SGX}, to all enclaves (i.e., both the encryptor and decryptor share the same SGX secret key). The first option is to store it in a variable inside the enclave when developing the application. The second option is to provision K_{SGX} by a remote server after developing the application and when running the application through a secure communication link. As illustrated in Figure III.4 and explained in Section III.3.2, the Intel SGX remote attestation enables an enclave (on a user's machine) to establish a private channel with a remote entity and receive secret keys from

the remote entity in a secure manner. Please note that different users (enclaves) do not use different SGX secret keys. Hence, even for multiple recipients, only one SGX secret key is required. In CP-ABE schemes, we do not know the exact recipients in advance as anyone holding a set of attributes satisfying the access structure can decrypt the ciphertext (i.e., with CP-ABE we encrypt for future, yet unknown, users). If we would perform the SGX encryption based on different SGX secret keys, then we would undermine this advantage of CP-ABE schemes.

Since the SGX encrypts both the memory and the plaintext of the secrets available inside the CPU (i.e., inside enclaves), no one (except the relevant hardware enclaves) can obtain the SGX secret key. Even if the SGX secret key gets compromised, the security of the ABE part would not be affected. A compromised SGX secret key does not help an adversary to decrypt a ciphertext without holding the required attributes. If an adversary obtains an SGX secret key, she still needs to decrypt a ciphertext using ABE by providing the attributes (the private key) that satisfy the ciphertext's access structure. If the adversary has the required attributes for ABE decryption and has extracted the SGX secret key, then she can only ignore the obligations attached to the ciphertext that are supposed to be enforced by the SGX. Nevertheless, in order to minimize the impact of leakage of an SGX secret key, i.e., to limit the number of ciphertexts that can be affected by a key disclosure, the SGX secret key can be updated periodically. Periodic update of the encryption key in symmetric encryption schemes is a way to address the key disclosure related issues [4, 61, 69, 94].

In what follows we describe the encryption and decryption algorithms (processes) of OB-ABE in more detail.

III.4.3 Encryption

Suppose Alice wants to encrypt and publish a message, M, along with some obligations OB. As shown in Figure III.6, the encryption process is as follows:

Step 1: Alice chooses either "encryption with obligations" or "encryption without obligations". If no obligations are chosen, then the process goes to Step 5. Otherwise, it goes to Step 2.

Step 2: Alice selects the desired set of obligations, OB.

Step 3: The obligations OB are concatenated to the message, i.e., (M, OB).

Step 4: The result of Step 3 is encrypted with the SGX secret key (K_{SGX}) using a symmetric encryption scheme as $C_{sgx} = E_{K_{SGX}}(M, OB)$. Since the SGX secret key is only known by the SGX, this encryption is performed by the SGX.

Step 5: Alice specifies the access structure $\mathcal{T}$ in a Boolean formula, e.g., $((Att_1 \wedge Att_2) \vee Attr_3)$.

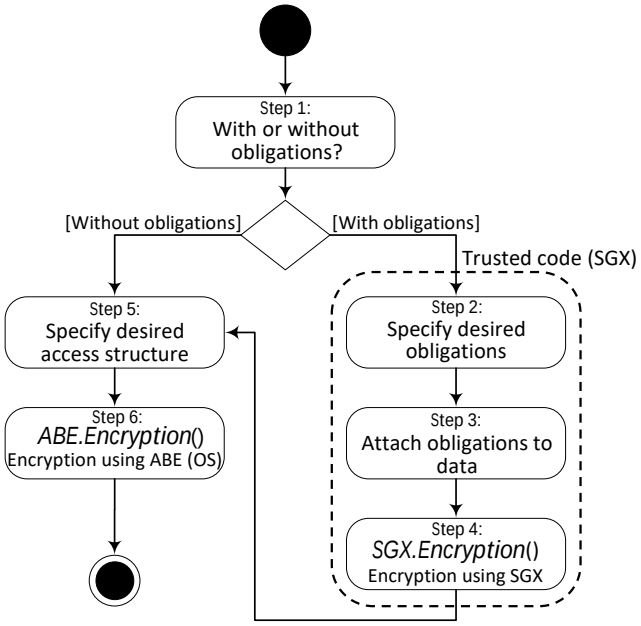


Figure III.6: Encryption process of the OB-ABE scheme.

Step 6: The result of Step 4, i.e., C_{sgx}, is encrypted using `ABE.Encryption` to get the ciphertext C and accordingly $CT = (\mathcal{T}, C)$. Note that if Alice chooses the second option (i.e., "encryption without obligations"), then the encryption algorithm in this step, takes M as input (instead of C_{sgx}).

III.4.4 Decryption

Following with the example, suppose Bob holds a set of attributes satisfying the access structure $\mathcal{T}$ previously defined by Alice and wants to decrypt the ciphertext $CT = (\mathcal{T}, C)$, provided by Alice. As shown in Figure III.7, the decryption process is as follows:

Step 1: Bob provides his required private key (attributes) according to the access structure $\mathcal{T}$.

Step 2: The ciphertext $CT = (\mathcal{T}, C)$ is decrypted using the `ABE.Decryption` algorithm with the provided attributes (i.e., Bob's private key) to obtain either $C_{sgx} = E_{K_{SGX}}(M, OB)$ or M.

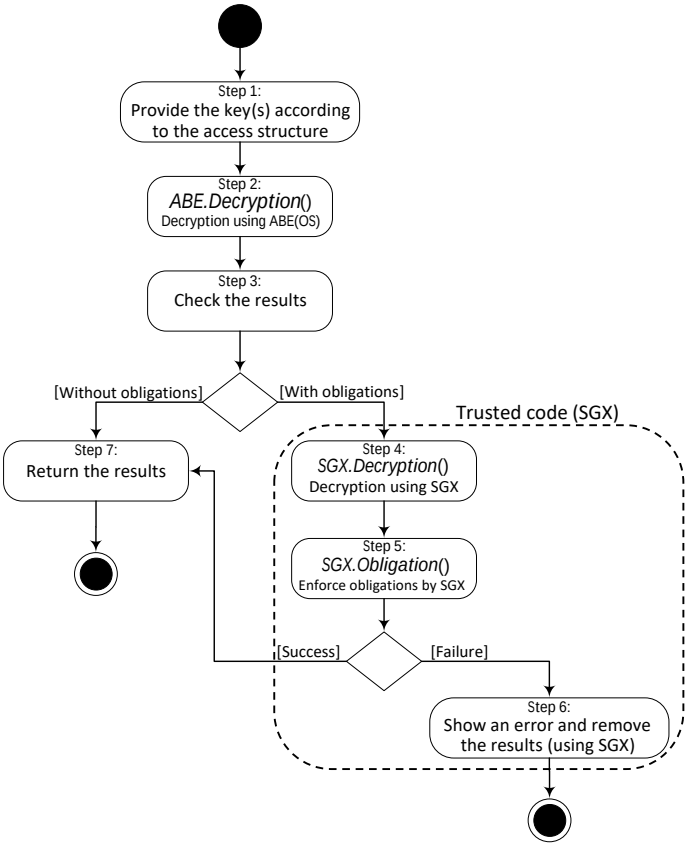


Figure III.7: Decryption process of the OB-ABE scheme.

Step 3: The result of the previous Step is checked to see if there are attached obligations.[1] If there are no obligations then the process goes to Step 7. Otherwise, it goes to Step 4.

Step 4: The SGX decrypts the result of the previous Step (i.e., $C_{sgx} = E_{K_{SGX}}(M, OB)$) using the `SGX.Decryption` algorithm with the SGX secret key, K_{SGX}, as $D_{K_{SGX}}(C_{sgx}) = (M, OB)$.

Step 5: Next, the SGX checks the attached obligations and enforces them through the `SGX.Obligation` algorithm. If the obligations are successfully

[1] This can be done in various ways, e.g., in our implementation, we use the file extension, e.g., a ciphertext having the `.obabe` extension has obligations (the `.cpabe` extension is used for ciphertexts without obligations). Note that there is no reason for an adversary to manipulate the file extension because she would not gain the plaintext as the SGX would not perform the decryption.

enforced, then the SGX returns the plaintext, M, to Bob (Step 7). Otherwise, the SGX performs the reparation obligations (if any), which are alternative obligations that will be enforced if there is a problem in enforcing the specified obligations (see Section III.4.1 for more information about the reparation obligations). If the SGX cannot enforce the obligations (normal and reparation ones), it does not return the plaintext to Bob and goes to Step 6.

Step 6: The SGX shows an error message and removes the results of the decryption.

Step 7: The result, i.e., M, is given to Bob.

III.4.5 Threat model

We assume a malicious adversary with root privileges on the same machine as the enclave. The adversary has full control over the machine, which means she has the ability to control the entire software environment including the OS, hypervisor, other running processes, and low-level firmware. The adversary is able not only to start, stop, and terminate enclave software, but also to start multiple identical enclaves at the same time. The adversary has also full control (read, change, reply, and delay) over all messages transmitted between applications and enclaves. Also, she can run memory attacks (e.g., cold boot attacks) and rollback attacks (the adversary can try to replace the sealed data with an old version).

The data and source code in enclave and the Intel Attestation Service, which can be used for the attestation process, are trusted. Hence, the adversary cannot compromise the hardware enclaves to get enclave keys (i.e., the secret keys for attestation, sealing, and so on). It is also supposed that the adversary cannot break the cryptographic primitives used by the SGX implementation nor the primitives used by the trusted enclave module.

We do not consider side channel attacks because they depend on the implementations [18, 60, 96, 127, 128, 131]. For instance, if the enclave programs are data-oblivious, i.e., do not have memory access patterns or control flow branches that depend on the values of sensitive data, then there is no leakage of the information through side-channels [39]. A more detailed list of SGX vulnerabilities and corresponding countermeasures is provided in Section III.A.

Since we assume that the user's machine is under the control of the adversary (except for the SGX enclave code) then any ABE scheme that the user would run would be under the control of the adversary. This is a quite unrealistically strong adversary model, which we purposely make in order to prove the enforceable obligations property of OB-ABE. Any weaker adversary (as assumed by a particular ABE) would preserve the validity of our property.

Moreover, we "fully" trust SGX, i.e., both to enforce the obligations as well as to release the plaintext. In particular, the first layer of encryption is that of SGX, which thus protects the plaintext from the adversary who does the ABE encryption. On the decryption side, the plaintext is again protected by the SGX

from an adversary who would want to get away without executing the required obligations, i.e., the adversary would decrypt the ABE encryption (assuming the necessary attributes are in place), but then only SGX can release the plaintext upon successful execution of the obligations.

One could think of reducing the trust in SGX by only relying on it to enforce the obligations but not to release/see the plaintext. This would imply that the ABE scheme is trusted, i.e., that it is running in a trusted environment. This can only be in another separate enclave of the SGX, for the rest of the machine is under the control of the adversary (for if it would not, then there is no point in using the SGX technology, as we would trust the user's machine). We however find no reason for such a separation of trust.

The reverse order of encryptions would allow a data owner to encrypt a message first by an ABE scheme, then only use SGX to add the obligations and encrypt them along with the ABE ciphertext, thus catering for the split trust mentioned above, where SGX does not see the plaintext. However, the decryptor will attempt to decrypt an ABE ciphertext only after the successful execution of the obligations. But this is wrong in the case when the decryptor does not have the necessary attributes as the obligation actions would have been executed without reasons (e.g., write in a log about the decryption, but in fact no decryption was possible).

III.5 Security Analysis

III.5.1 Security Assumptions

The proposed OB-ABE scheme is based on the following assumptions:

- TA that holds the master secret key is considered trusted like conventional CP-ABE schemes.
- SGX, i.e., hardware enclaves provided by SGX that contain the SGX secret key, enforce obligations, and release plaintexts after enforcing obligations, is considered trusted and secure.

III.5.2 Security Model

The security model for OB-ABE is defined using the following game, which is based on the classical indistinguishable encryption against chosen-plaintext attacks (IND-CPA) and mainly considers the confidentiality of the ciphertext.

- **Init phase**: An adversary $\mathcal{A}$ chooses an access structure $\mathcal{T}$ and sends it to a challenger $\mathcal{C}$.
- **Setup phase**: $\mathcal{C}$ generates the master secret key MK and public parameters PP using the `ABE.Setup` algorithm. $\mathcal{C}$ also initializes the SGX

to get the SGX secret key, K_{SGX}. Next, $\mathcal{C}$ sends the public parameters PP to $\mathcal{A}$ and keeps the master secret key MK and the SGX secret key K_{SGX} secret.

- **Phase 1**: $\mathcal{A}$ asks (like any user) from $\mathcal{C}$ private keys related to any sets of attributes. $\mathcal{C}$ generates the requested private keys by running the `ABE.KeyGen` algorithm.
- **Challenge phase**: $\mathcal{A}$ submits two messages M_0 and M_1 of equal length along with two obligations O_0 and O_1, where the obligations have also the same length and size. In other words, $\mathcal{A}$ submits (M_0, O_0) and (M_1, O_1) pairs. $\mathcal{C}$ selects a random bit $b \in \{0,1\}$, encrypts (M_b, O_b) using the `SGX.Encryption` algorithm to get C_{sgx}. Then, $\mathcal{C}$ encrypts C_{sgx} under the access structure $\mathcal{T}$ using the `ABE.Encryption` algorithm, and returns the resulted ciphertext CT^*.
- **Phase 2**: $\mathcal{A}$ repeats **Phase 1** several times.
- **Guess phase**: $\mathcal{A}$ outputs its guess $b' \in \{0,1\}$.

$\mathcal{A}$ wins the game if (i) $b = b'$ and (ii) none of the sets of attributes that were requested by $\mathcal{A}$ satisfy the access structure that was used for encryption. The advantage of the adversary $\mathcal{A}$ is defined as the quantity

$$Adv_{\mathcal{A}}^{IND-CPA} = |Pr[b = b'] - \frac{1}{2}|.$$

Definition III.2 (IND-CPA Secure)**.** *The proposed OB-ABE scheme is* IND-CPA secure *iff* $Adv_{\mathcal{A}}^{IND-CPA}$ *is negligible for any probabilistic polynomial time (PPT) adversary.*

III.5.3 Security Proof

The security of the proposed OB-ABE scheme is proved by reduction to the underlying CP-ABE [15] scheme. In other words, it is demonstrated that if there exists an attack against OB-ABE, then the same attack can be used to break the underlying CP-ABE, which has been proven to be IND-CPA secure.

Theorem III.1. *The proposed OB-ABE scheme is IND-CPA secure provided that the underlying CP-ABE scheme is IND-CPA secure.*

Proof. Assume that there exists a PPT adversary $\mathcal{A}$ that can break the proposed OB-ABE with advantage ϵ. A simulator $\mathcal{B}$ can be constructed to break the underlying CP-ABE scheme with the same advantage ϵ as follows. Note that $\mathcal{B}$ plays two roles at the same time: (1) the challenger for the adversary $\mathcal{A}$ in the IND-CPA game for OB-ABE; and (2) the adversary for the challenger in the IND-CPA game for the underlying CP-ABE scheme.

- **Init phase**: $\mathcal{B}$ receives an access structure $\mathcal{T}$ from $\mathcal{A}$ and sends it to $\mathcal{C}$ (in the CP-ABE scheme).

- **Setup phase**: $\mathcal{C}$ generates the master secret key MK and public parameters PP using the `Setup` algorithm of the underlying CP-ABE scheme (i.e., `ABE.Setup`). Then, $\mathcal{C}$ sends PP to $\mathcal{B}$ and keeps MK secret. After that, $\mathcal{B}$ initializes the SGX to get the SGX secret key, K_{SGX}, which will be kept secret by $\mathcal{B}$. Next, $\mathcal{B}$ forwards PP to $\mathcal{A}$.

- **Phase 1**: When $\mathcal{B}$ receives a private key query for a set of attributes from $\mathcal{A}$, it forwards the received set of attributes to $\mathcal{C}$ to get the corresponding private keys from the underlying CP-ABE scheme. In response, $\mathcal{C}$ generates the corresponding private keys using the `KeyGen` algorithm of the underlying CP-ABE scheme (i.e., `ABE.KeyGen`) and returns the generated private keys to $\mathcal{B}$. Next, $\mathcal{B}$ forwards the received private keys to $\mathcal{A}$ in response to $\mathcal{A}$'s original query.

- **Challenge phase**: $\mathcal{A}$ sends (M_0, O_0) and (M_1, O_1) pairs to $\mathcal{B}$, where M_0 and M_1 are two messages of equal length and O_0 and O_1 are two obligations that also have the same length and size. $\mathcal{B}$ encrypts (M_0, O_0) and (M_1, O_1) using the `SGX.Encryption` algorithm to get C_{sgx-0} and C_{sgx-1} for, respectively, (M_0, O_0) and (M_1, O_1). After that, $\mathcal{B}$ sends C_{sgx-0} and C_{sgx-1}, which have the same length and can be considered as two plain messages, to $\mathcal{C}$. Then, $\mathcal{C}$ selects a random bit $b \in \{0, 1\}$, encrypts C_{sgx-b} under the access structure that was provided in the **Init phase**, and returns the produced ciphertext CT^* (the output of the `Encryption` algorithm of the underlying CP-ABE scheme, i.e., `ABE.Encryption`) to $\mathcal{B}$ who forwards it to $\mathcal{A}$.

- **Phase 2**: The same as **Phase 1** several times.

- **Guess phase**: $\mathcal{A}$ outputs a guess $c' \in \{0, 1\}$, and then $\mathcal{B}$ sends c' to $\mathcal{C}$.

Based on this simulation game, it is clear that if $\mathcal{A}$ has an advantage ϵ in the IND-CPA game against the proposed OB-ABE scheme, then $\mathcal{B}$ can attack the underlying CP-ABE scheme with the same advantage ϵ. However, the underlying CP-ABE has been proven to be IND-CPA secure [15].

As explained in **Property III.3** (in Section III.6), the proposed OB-ABE scheme does not change the functionality of `Setup`, `KeyGen`, `Encryption`, and `Decryption` algorithms of the underlying CP-ABE scheme in any way. Indeed, only the input of the `Encryption` and `Decryption` algorithms of the underlying CP-ABE will be changed. However, the type of input is still the same, and thus these two algorithms would function in the same way with the same security guarantees. ■

III.6 Verification

Properties of the proposed OB-ABE scheme are discussed and verified in this section. OB-ABE ensures that only those users having the required public attributes can access the original resources provided that the accompanying obligations are satisfied. Hence, the first property of OB-ABE is defined as follows.

Property III.1. ***Enforceable obligations****: If a message M is encrypted along with some obligations OB, then the decryption algorithm must return the message M only after enforcing all the attached obligations OB.*

In what follows, we detail our encoding into ProVerif. First, we specify the communication channels, functions, types, and variables as follows. We define *skey* and *pkey* types for secret keys and public keys, respectively. We also define the *obligation* type for obligations. The secret key of the SGX, which is only known by the SGX that provides a trusted execution environment (that is why it is specified as private), is represented as *Ksgx*. Each user has an ABE private key related to her set of attributes that is represented as *SKABE* (for the sake of simplicity, we assume that the user is in possession of such a key and can use it to perform ABE operations).

Please note that normally in ABE schemes, *SKABE* would be specified as secret, i.e., a secret key on which the properties of the ABE scheme depend. However, since in our threat model (Section III.4.5) the adversary has full control over the operating system that runs the ABE algorithms and thus is able to get the ABE private key of the user, we decided to specify *SKABE* as a public parameter in our formalisation. Besides, we do not want to verify the security properties of the underlying ABE scheme. In our formalisation we purposely make very few assumptions, and only assume the SGX secret key to be private, hence SGX is our only trust factor. This, moreover, shows that the three properties of the OB-ABE depend solely on the SGX.

```
type skey.
type pkey.
type obligation.
const Ksgx: skey [private].
free SKABE: skey.
```

We assume that the user encrypts a plaintext using the untrusted part of the application which is managed by the operating system, and the SGX (i.e., the trusted part of the application managed by a trusted hardware enclave providing the trusted execution environment), and finally stores the result, CT, on a public cloud. When a user wants to decrypt a message, she retrieves CT from the cloud and then uses her ABE private key to decrypt it. We specify three communication channels between the user and the operating system, *chUOS*, the operating system and the SGX, *chOSSGX*, and the user and the cloud, *chUserCloud*, as follows.

```
free chUserOS: channel.
free chOSSGX: channel.
free chUserCloud: channel.
```

The functions that are used in our proposed scheme are specified as follows:

```
fun h(bitstring): bitstring.
fun penc(bitstring, pkey):bitstring.
reduc forall m:bitstring, k:skey; pdec(penc(m, pk(k)), k) = m.
fun senc(bitstring, skey):bitstring.
reduc forall m:bitstring, k:skey; sdec(senc(m, k), k) = m.
fun pk(skey):pkey.
fun execOBG(obligation):bool.
table tCT(bitstring, bitstring).
```

fun h, **fun** penc, and **fun** senc represent the hash function, public key encryption, and symmetric encryption, respectively. Note that pdec and sdec represent the public key decryption and symmetric decryption, respectively, which are specified using rewrite rules. There is a function for computing the public key related to a secret key that is represented as **fun** pk and another one for enforcing (executing) obligations represented as **fun** execOBG, which takes a set of obligations as input and returns a Boolean, i.e., whether the execution was successful or not. As ciphertexts are stored in a cloud database, **table** tCT is defined as a database for persistent storage.

In order to verify the desired property, we specify the following events and query:

```
event ObligationsRequired(bitstring, obligation).
event ObligationsEnforced(bitstring, obligation).
event Decrypted(bitstring, obligation).
query i:bitstring, j:obligation; event(Decrypted(i, j)) ==>
event(ObligationsEnforced(i, j)) ==> event(ObligationsRequired(i, j)).
```

The ObligationsRequired event occurs after encrypting a message along with some obligations, the event ObligationsEnforced happens after enforcing obligations using the SGX, and Decrypted event occurs after returning the result of the decryption (the plaintext). The query states that if the event Decrypted(i, j) is executed for a message (data item) i with an obligation set j, then the event ObligationsEnforced(i, j) is executed before that for the same data item and obligation set, and the ObligationsEnforced(i, j) event is executed after the event ObligationsRequired(i, j) with the same arguments i and j.

The following provides the specification for each of the participating entities, i.e., the user, the untrusted part of the application (OS), the trusted part of the application (SGX), and the cloud, which are considered as different processes.

let $Uenc$ = **let** $PKABE$ = pk(SKABE) **in**
new m: bitstring;
new ob: obligation;
out(chUserOS, (m, ob, $PKABE$));
in(chUserOS, rCT:bitstring);
out(chUserCloud, rCT);
in(chUserCloud, $ridCT$:bitstring).
let $Udec$ = **new** $xidCT$: bitstring;
out(chUserCloud, $xidCT$);
in(chUserCloud, rCT:bitstring);
out(chUserOS, (rCT, SKABE));
in(chUserOS, $wrxym$:bitstring).
let $User$ = $Uenc|Udec$.

The $User$ process is composed of the user encryption (*Uenc*) and user decryption (*Udec*) processes. According to the user encryption process, the user computes the public key related to her ABE key. Then, she selects a plaintext, m, and a desired set of obligations, *ob*, and sends them along with her ABE public key to the OS encryption process (i.e., *Oenc*) through channel *chUserOS* (i.e., provides such information to the untrusted part of the application). After receiving the result of the encryption (the result of encryption with both OS and SGX), the user (i.e., *Uenc* process) sends the received ciphertext (rCT) to the *Cloud* process (in fact to the cloud store process, *Cstore*) through channel *chUserCloud* and receives the index ($ridCT$) for the submitted ciphertext. When the user wants to decrypt a ciphertext, she retrieves the ciphertext from the cloud by sending the index of that ciphertext, $xidCT$, to the *Cloud* process (i.e., to the cloud retrieve process, *Cretrieve*). Then, she sends the retrieved ciphertext and her ABE key (*SKABE*) to the OS decryption process (i.e., *Odec*) and in response receives the plaintext ($wrxym$).

The untrusted part of the application is represented by the OS process, which is composed of the OS encryption (*Oenc*) and OS decryption (*Odec*) processes. When *Oenc* receives a plaintext, a set of obligations, and an ABE public key, it forwards the received plaintext and set of obligations to the trusted part (*SGX* and more specificity to the SGX encryption process, *SGXenc*) through channel *chOSSGX*. Upon receiving the result of encryption by the trusted hardware enclave from the trusted part (i.e., receiving *rCsgx* from *SGXenc*), the *Oenc* process re-encrypts the received ciphertext with the user's ABE public key (*rPKABE*), and then sends the result (CT) to the user (*Uenc*) through the channel *chUserOS*. For the decryption, when the *Odec* receives a request from the *Udec* process, it decrypts the received ciphertext ($xrCT$) with the received ABE's key (*rSKABE*) and sends the result (*Csgx*) to the SGX decryption process (i.e., *SGXdec*) for the second decryption and for enforcing the attached obligations. Finally, when the *Odec* process receives the plaintext ($rxym$) from the *SGXdec* process, it forwards the plaintext to the *Udec* process through channel *chUserOS*.

let $Oenc =$
in(chUserOS, (rm:bitstring, rob:obligation, $rPKABE$:pkey));
out(chOSSGX, (rm, rob));
in(chOSSGX, $rCsgx$: bitstring);
let CT = penc($rCsgx$, $rPKABE$) **in**
out(chUserOS, CT).
let $Odec$ = **in**(chUserOS, ($xrCT$:bitstring, $rSKABE$:skey));
let $Csgx$ = pdec($xrCT$, $rSKABE$) **in**
out(chOSSGX, $Csgx$);
in(chOSSGX, $rxym$:bitstring);
out(chUserOS, $rxym$).
let $OS = Oenc|Odec$.

The trusted part of the application is represented as SGX process, which consists of the SGX encryption ($SGXenc$) and SGX decryption ($SGXdec$) processes. When the $SGXenc$ process receives a plaintext (xrm) and a set of obligations ($xrob$) from the $Oenc$ process, an event ObigationsRequired for the received message (i.e., the plaintext, xrm, and the set of obligations, $xrob$) occurs. Then, the $SGXenc$ process encrypts the plaintext and the set of obligations with its secret key $Ksgx$ using the symmetric encryption function $senc$, and sends back the result ($Csgx$) to the $Oenc$ process through channel $chOSSGX$. The $SGXdec$ process works as follows: when it receives a ciphertext, it decrypts it with its secret key ($Ksgx$) and enforces the attached obligations ($xyrob$). After enforcing (executing) obligations, the event ObigationsEnforced occurs, parametrized with the corresponding ciphertext (xym) and obligations ($xyrob$). Finally, the $SGXdec$ process sends the plaintext to the $Odec$ process and the event Decrypted occurs, parametrized with the same plaintext and set of obligations.

let $SGXenc =$
in(chOSSGX, (xrm:bitstring, $xrob$:obligation));
event ObigationsRequired(xrm, $xrob$);
let $Csgx$ = senc(concat(xrm, $xrob$), Ksgx) **in**
out(chOSSGX, $Csgx$).
let $SGXdec$ =**in**(chOSSGX, $rCsgx$:bitstring);
let (xym:bitstring, $xyrob$:obligation) = sdec($rCsgx$, Ksgx) **in**
let $eobg$ = execOBG($xyrob$) **in**
if $eobg$ **then**(**event** ObigationsEnforced(xym, $xyrob$);
out(chOSSGX, xym);
event Decrypted(xym, $xyrob$)).
let $SGX = SGXenc|SGXdec$.

The $Cloud$ process consists of the cloud store ($Cstore$) and cloud retrieve ($Cretrieve$) processes. When the $Cstore$ process receives a ciphertext ($xrCT$) from the $Uenc$ process through channel $chUserCloud$, it generates an index for

the ciphertext, adds both into its database, and sends the index ($idCT$) to the *User* process. When the *Cretrieve* process receives an index from the *Udec* process, it retrieves the related ciphertext from its database and sends it back to the *Udec* process.

let *Cstore* = **in**(chUserCloud, $xrCT$:bitstring);
new $idCT$:bitstring;
insert tCT($idCT$, $xrCT$);
out(chUserCloud, $idCT$).
let *Cretrieve* = **in**(chUserCloud, $xxidCT$:bitstring);
get tCT(= $xxidCT$, CT) **in**
out(chUserCloud, CT).
let *Cloud* = *Cstore*|*Cretrieve*.

Since in OB-ABE several instances of these entities may execute in parallel and at the same time, the proposed scheme is defined as follows:

process !*User* |!*OS* |!*SGX* |!*Cloud*

We used version 2.00 of the ProVerif tool to verify the the above specifications. The output of the ProVerif, as shown in Figure III.8, confirms that the **Property III.1** is satisfied.

As mentioned earlier, another important property of OB-ABE is *Backward Compatibility*, which is defined as follows:

Property III.2. ***Backward compatibility****: OB-ABE is compatible with ABE according to the two following lemmas.*

The following two lemmas prove that our approach satisfies Property III.2.

Lemma III.1. *A ciphertext produced by an ABE scheme can be decrypted by its extended version, OB-ABE.*

Proof. If a user encrypts a message using ABE, it would be encrypted using the `ABE.Encryption` algorithm. The resulting ciphertext can be decrypted using the `ABE.Decryption` algorithm with a private key related to the access structure.

Suppose a user wants to decrypt the ciphertext produced by ABE using the extended version of ABE, OB-ABE. As demonstrated in Figure III.7 and Algorithm III.1, the decryption process of OB-ABE decrypts a ciphertext using `ABE.Decryption` and then checks if there is any obligations attached to the result or not. Since there is not any obligations attached to the CT (because it is encrypted using ABE, which does not support obligations), it returns the result of `ABE.Decryption` to the user (Figure III.7: Step 1 → Step 2 → Step 3 → Step 7). Therefore, it can be said that ciphertexts produced by ABE can be decrypted by OB-ABE. ■

```
Process:
(   {1}!
    (   {2}let PKABE: pkey = pk(SKABE) in
        {3}new m_21: bitstring;
        {4}new ob: obligation;
        {5}out(chUserOS, (m_21,ob,PKABE));
        {6}in(chUserOS, rCT: bitstring);
        {7}out(chUserCloud, rCT);
        {8}in(chUserCloud, ridCT: bitstring)
    ) | (
        {9}new xidCT: bitstring;
        {10}out(chUserCloud, xidCT);
        {11}in(chUserCloud, rCT_22: bitstring);
        {12}out(chUserOS, (rCT_22,SKABE));
        {13}in(chUserOS, wrxym: bitstring)
    )
) | (
    {14}!
    (   {15}in(chUserOS, (rm: bitstring,rob: obligation,rPKABE: pkey));
        {16}out(chOSSGX, (rm,rob));
        {17}in(chOSSGX, rCsgx: bitstring);
        {18}let CT: bitstring = penc(rCsgx,rPKABE) in
        {19}out(chUserOS, CT)
    ) | (
        {20}in(chUserOS, (xrCT: bitstring,rSKABE: skey));
        {21}let Csgx: bitstring = pdec(xrCT,rSKABE) in
        {22}out(chOSSGX, Csgx);
        {23}in(chOSSGX, rxym: bitstring);
        {24}out(chUserOS, rxym)
    )
) | (
    {25}!
    (   {26}in(chOSSGX, (xrm: bitstring,xrob: obligation));
        {27}event ObligationsRequired(xrm,xrob);
        {28}let Csgx_23: bitstring = senc((xrm,xrob),Ksgx) in
        {29}out(chOSSGX, Csgx_23)
    ) | (
        {30}in(chOSSGX, rCsgx_24: bitstring);
        {31}let (xym: bitstring,xyrob: obligation) = sdec(rCsgx_24,Ksgx) in
        {32}let eobg: bool = execOBG(xyrob) in
        {33}if eobg then
        {34}event ObligationsEnforced(xym,xyrob);
        {35}out(chOSSGX, xym);
        {36}event Decrypted(xym,xyrob)
    )
) | (
    {37}!
    (   {38}in(chUserCloud, xrCT_25: bitstring);
        {39}new idCT: bitstring;
        {40}insert tCT(idCT,xrCT_25);
        {41}out(chUserCloud, idCT)
    ) | (
        {42}in(chUserCloud, xxidCT: bitstring);
        {44}get tCT(=xxidCT,CT_26: bitstring) in
        {43}out(chUserCloud, CT_26)
    )
)
-- Query event(Decrypted(i,j)) ==> (event(ObligationsEnforced(i,j)) ==>
event(ObligationsRequired(i,j)))
Completing...
Starting query event(Decrypted(i,j)) ==> (event(ObligationsEnforced(i,j))
==> event(ObligationsRequired(i,j)))
RESULT event(Decrypted(i,j)) ==> (event(ObligationsEnforced(i,j)) ==>
event(ObligationsRequired(i,j))) is true.
```

Figure III.8: Output of the ProVerif tool

Algorithm III.1: Pseudocode of OB-ABE decryption

```
   Input: Ciphertext, Private Key (which is related to the Access
          Structure)
   Output: Plaintext
 1 Call ABE decryption,
   M ← ABE.Decryption(Ciphertext, Private Key);
 2 if there is any Obligations in M then
 3 |   Call the trusted decryption function,
   |    (Plaintext, Obligations) ← SGX.Decryption(M);
 4 |   Enforce Obligations;
 5 |   if Obligations are enforced then
 6 |   |   Return Plaintext;
 7 |   else
 8 |   |   Remove Plaintext and show an ERROR Message;
 9 |   end
10 else
11 |   Plaintext ← M;
12 |   Return Plaintext;
13 end
```

Lemma III.2. *A ciphertext produced by an OB-ABE scheme can be decrypted by its underlying ABE scheme provided that the ciphertext does not have obligations to enforce.*

Proof. Suppose a user wants to encrypt a data item using OB-ABE and she does not want to have any obligations for that data item. As there are no obligations, according to Figure III.6 and Algorithm III.2, the data item would be encrypted using the encryption algorithm of the underlying ABE scheme (Figure III.6: Step 1 → Step 5 → Step 6) by calling `ABE.Encryption` (see lines 6-8 of Algorithm III.2). Hence, the result would be the same as the result of encrypting that data item using the underlying ABE scheme. Therefore, ciphertexts produced by OB-ABE without obligations can be considered as those produced by the underlying ABE scheme. Consequently, such ciphertexts can be decrypted using the underlying ABE scheme. ■

Property III.3. ***Conservative extension in terms of security****: OB-ABE preserves any security property of the underlying ABE scheme.*

Proof. As described before, an ABE scheme can be extended to an OB-ABE scheme by changing the `encryption` and `decryption` processes, i.e., by adding one more encryption operation and consequently one more decryption operation. As represented in Figures III.2 and III.5, OB-ABE includes `SGX.Encryption` and `SGX.Decryption` in addition to `ABE.Encryption` and `ABE.Decryption` functions of ABE. These two algorithms are added to support obligations (in addition to `SGX.Obligations`, which can be merged with `SGX.Decryption`).

Algorithm III.2: Pseudocode of OB-ABE encryption

Input: *Plaintext*, *Access Structure*, *Obligations*
Output: *Ciphertext*
if *there is any Obligations* **then**
  Attach *Obligations* to *Plaintext*, $M \leftarrow (Plaintext, Obligations)$;
  Call the trusted encryption function/algorithm,
  $C_{sgx} \leftarrow SGX.Encryption(M)$;
  Call ABE encryption,
  $Ciphertext \leftarrow ABE.Encryption(C_{sgx}, AccessStructure)$;
  Return *Ciphertext*;
else
  Call ABE encryption,
  $Ciphertext \leftarrow ABE.Encryption(Plaintext, Access\ Structure)$;
  Return *Ciphertext*;
end

The functionality of `ABE.Encryption` and `ABE.Decryption` is not changed in any way. Instead, the input of these two algorithms is changed as follows.

If a user wants to encrypt a data item along with some obligations, then, according to Figure III.6 and Algorithm III.2, obligations would be attached to the data item and the result would be encrypted using a trusted encryption algorithm running in a trusted hardware enclave (i.e., `SGX.Encryption`). Then, the result of the encryption using the trusted encryption algorithm should be encrypted using the ABE encryption (i.e., `ABE.Encryption`) being the same as in the ABE scheme. It is clear that only the input of the encryption algorithm of ABE is changed (however, the type of the input is still a bitstring). In other words, the plaintext (input of the encryption algorithm of ABE) is replaced with a ciphertext (the plaintext is encrypted with a secret key which is only known by the trusted hardware enclave).

Besides, if a user wants to decrypt a ciphertext, which has some obligations attached, using OB-ABE, then, as represented in Figure III.7 and Algorithm III.1, first the ciphertext would be decrypted using the ABE decryption algorithm (`ABE.Decrption` from the underlying ABE). The result is not the plaintext (the requested data item); instead, it is an encrypted version of the requested data item (it is encrypted with a secret key which is only known to the trusted hardware enclave). In order to get the requested data item, the result of the first decryption needs to be decrypted using the trusted decryption algorithm `SGX.Decryption` and the attached obligations should be enforced. Therefore, the data item can be retrieved only after enforcing obligations.

Moreover, in our implementation we use the same template for all obligations, which implies that the obligations would have the same length and size.[1] Besides, the OB-ABE attaches the obligations to the original plaintext (i.e., forms a

[1]Having the same length for different obligations can be achieved in various ways, e.g., i) considering the same length for all obligations and using padding for those that consume less

new plaintext) and then encrypts the result using SGX and ABE. In particular, OB-ABE does not attach the obligations to the ciphertexts, in which case an adversary could infer information about the attached obligations based on their length or size. An adversary may try to distinguish which plaintext is encrypted along with which obligations. However, the adversary cannot succeed as the underlying CP-ABE scheme has been proven to be IND-CPA secure [15].

As OB-ABE does not change the functionality of the underlying ABE scheme algorithms, it can be concluded that OB-ABE does not open up new security issues. Therefore, security properties provided by the ABE still hold. ■

III.7 Implementation and Evaluation

We implemented a prototype of the proposed scheme developed in C using the Intel(R) SGX SDK 2.2 for Linux. We evaluated the performance of the proposed scheme on a machine with an Intel Core i7-8550U CPU at 1.80GHz with 32 GB RAM and Ubuntu 16.04 LTS (64-bit). Since we only modified the encryption and decryption algorithms (processes), we focused on the evaluation of execution times of these algorithms. We also compared the execution times of these two algorithms with the corresponding ones of the CP-ABE. In our experiments we did not take into account the time required to enforce obligations (e.g., sending an email) which vary from machine to machine and it does not directly depend on the proposed scheme. We extended the baseline CP-ABE [15] by adding support for obligations and SGX. Our implementation is available at https://github.com/haamedarshad/OB-ABE.

Encryption: If the user does not want to add obligations, then we use the encryption of the original CP-ABE scheme, otherwise, another function for choosing the desired obligations is run. Then, the algorithm attaches the selected obligations to the plaintext and calls `SGX.Encryption` algorithm (`ecall_enc()`). This function runs inside an enclave and encrypts the received message with K_{SGX}, the SGX's secret key, using Advanced Encryption Standard (AES) (`sgx_aes_ctr_encrypt()`) which is provided by Intel SGX trusted cryptography library. Upon receiving the resulted C_{sgx} from the enclave, the `ABE.Encryption` algorithm encrypts C_{sgx} under the access structure provided by the user. The result of encryption is returned as the ciphertext with obligations, CT.

Decryption: The user runs `ABE.Decryption` algorithm in order to decrypt CT with the private key corresponding to her public attributes. Then, if there are obligations, `ABE.Decryption` calls `SGX.Decryption` algorithm, the trusted code (`ecall_dec()`), which is running inside an enclave and takes the results of the first decryption as input, i.e., C_{sgx}. `SGX.Decryption` decrypts C_{sgx} using the function `sgx_aes_ctr_decrypt()`, which is provided by Intel SGX trusted cryptography library, by using K_{SGX}—the secret key that is only known to the enclave. After the decryption, `ecall_dec()` calls another function to enforce

space; or ii) having a table of obligations-codes inside the SGX and attaching only the code of obligations when encrypting a plaintext along with obligations.

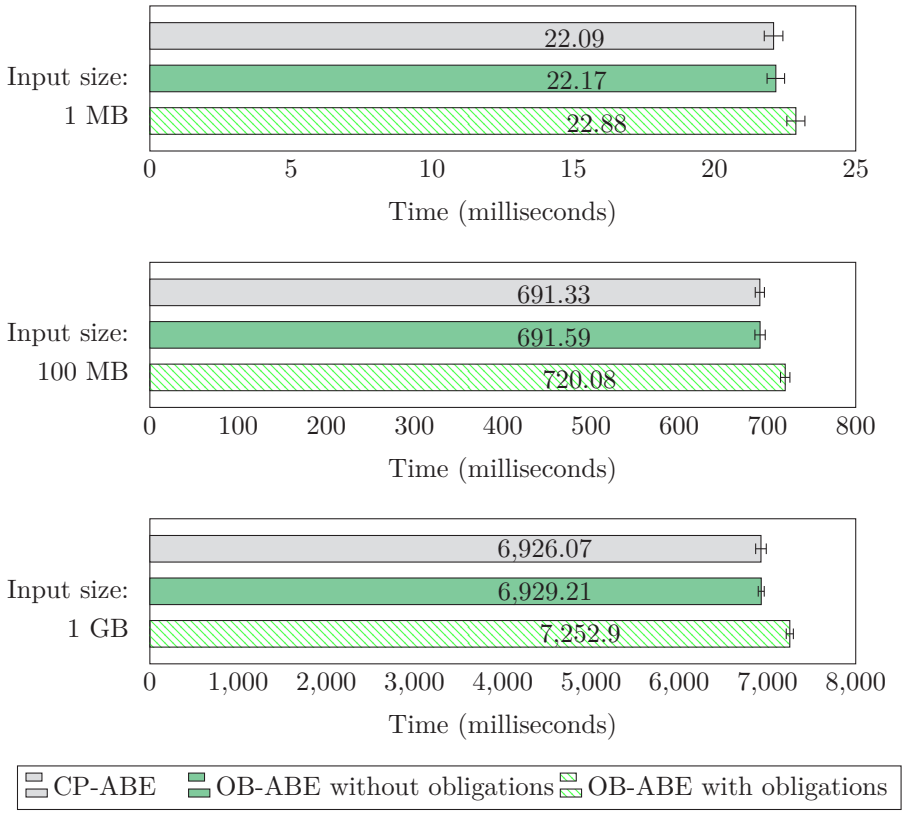


Figure III.9: Execution time for encryption (with 95% confidence intervals).

the attached obligations. Finally, `SGX.Decryption` returns the plaintext to `ABE.Decryption` and to the user.

In order to illustrate the performance overhead associated with adding obligations to ABE schemes, the average execution times of both the proposed OB-ABE scheme and the underlying CP-ABE scheme are estimated. Each experiment, i.e., encryption and decryption with both OB-ABE (with and without obligations) and CP-ABE, is executed 100 times for different input sizes, i.e., 1 MB, 100 MB, and 1 GB. The same access structure and obligations are used for all experiments. Table III.3 lists mean execution times (in milliseconds) of both OB-ABE and CP-ABE schemes. Figure III.9 represents the difference between execution time (in milliseconds and with 95% confidence intervals) of encryption of the proposed OB-ABE scheme and that of the underlying CP-ABE scheme. The difference between execution times of the decryption operations for both schemes is represented in Figure III.10.

As shown in Table III.3, the mean execution times of OB-ABE for encryption and decryption with obligations are more than those of the underlying CP-ABE

Table III.3: Execution times of OB-ABE vs CP-ABE (times for performing the actual obligations are not counted in).

Algorithm	Input size	Scheme		
		OB-ABE		CP-ABE
		With Obligations	Without Obligations	
Encryption	1 MB	22.88 ms	22.17 ms	22.09 ms
	100 MB	720.08 ms	691.59 ms	691.33 ms
	1 GB	7252.90 ms	6929.21 ms	6926.07 ms
Decryption	1 MB	9.56 ms	9.45 ms	9.40 ms
	100 MB	934.60 ms	928.85 ms	927.30 ms
	1 GB	9699.58 ms	9648.73 ms	9644.74 ms

scheme because the proposed OB-ABE scheme encrypts the input data one more time using SGX (when there is some obligations) and accordingly in decryption with obligations, there is one more decryption (by SGX). In OB-ABE without obligations, only the encryption and decryption algorithms of the underlying CP-ABE scheme will be called. However, the experiments show that the encryption and decryption (without obligations) of the proposed OB-ABE scheme have higher execution times than those of the underlying CP-ABE scheme. The overhead is due to the fact that OB-ABE creates encryption and decryption enclaves when we run the program (we have developed a single software including all the algorithms) regardless of whether it has obligations or not. Moreover, the OB-ABE scheme executes some extra checks (e.g., to see whether obligations are present), slightly increasing the execution time. Nevertheless, the overhead is negligible, being just a few milliseconds for a 1 GB input.

The overall experimental results are encouraging as the overheads associated with adding obligations to the base-line ABE scheme are reasonable.

III.8 Related Work

Attribute-Based Encryption: The concept of attribute-based encryption is introduced by Sahai and Waters [109] in 2005 where a new type of IBE is proposed by which one can encrypt a data under a set of attributes, i.e., data can be encrypted for a group of recipients (holding the set of attributes). Therefore, multicast encryption can be achieved by means of ABE schemes [125].

In 2006, Goyal et al. [44] proposed a new ABE scheme, called KP-ABE in which each ciphertext, CT, is annotated with a set of attributes and each private key has an access structure $\mathcal{T}$. Therefore, a CT can be decrypted with a private key that its $\mathcal{T}$ satisfies the annotated set of attributes. In 2007, Bethencourt et al. [15] proposed another type of ABE called CP-ABE in which the private key is associated with the set of attributes and the CT has a $\mathcal{T}$.

Afterwards, several researchers proposed different ABE schemes with the aim of improving security and efficiency. However, since a CP-ABE scheme is used in

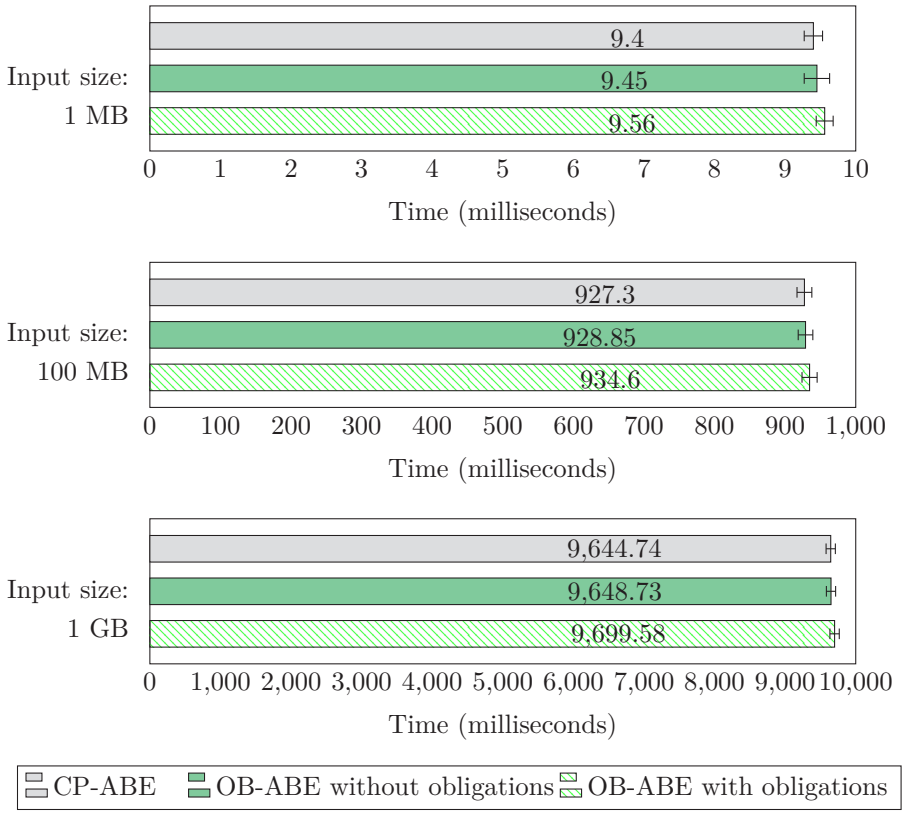


Figure III.10: Execution time for decryption (with 95% confidence intervals).

the design of our proposed scheme and CP-ABE is more suitable for facilitating fine-grained access control, we focus on some of the CP-ABE schemes that have been proposed recently.

In [6], Agrawal and Chase proposed a practical and efficient CP-ABE scheme that requires a constant number of pairing (Type-III) operations for decryption. In other words, the number of attributes in access structures does not affect the decryption time. Furthermore, Agrawal and Chase's scheme allows any attribute names and does not bound the size of attribute sets or access structures.

In addition to the improvement of security and efficiency, another line of research focuses on enhancing ABE schemes by adding more functionalities to the classical CP-ABE schemes. In classical CP-ABE schemes, a group of users who share the same attributes would hold the same private key. Hence, if one of them loses her private key or discloses it for financial benefits, she cannot be detected and traced. Besides, a TA may also generate a private key for an unauthorized user without being detected. In order to address this issue, *accountable* CP-ABE schemes [70, 77, 78], which can be classified into white-box accountability and black-box accountability [78, 92, 142], were proposed.

Recently, Liu et al. [79] proposed a black-box accountable CP-ABE scheme for eHealth. Liu et al. demonstrated that their proposed scheme not only can trace a user from a leaked private key but also can identify TA's malicious activity. Although accountable CP-ABE schemes help to identify the owner of a leaked private key, they do nothing for revoking the leaked private key. Hence, CP-ABE schemes with user/attribute revocation mechanisms emerged [9, 10, 126]. There are two types of revocation in CP-ABE schemes: direct and indirect [142]. In direct revocable CP-ABE schemes, only the revoked users will be affected and the private keys of non-revoked users do not need to be updated. However, indirect revocable CP-ABE schemes require updating the private keys of all non-revoked users and the affected ciphertexts [137]. Li et al. [74] proposed a CP-ABE scheme with white-box accountability and direct revocation. To improve the performance, Li et al. designed their scheme in a way that facilitates constant ciphertext size and requires a constant number of pairing operations for decryption. In a recent work [140], Zhang et al. proposed a CP-ABE scheme that not only provides user revocation but also addresses the key escrow problem by involving both users and TA in the key generation process. Zhang et al.'s scheme [140] falls into indirect revocable CP-ABE schemes because it requires updating the private keys of non-revoked users and also affected ciphertexts.

Access structures in CP-ABE schemes may include sensitive information about those who can decrypt ciphertexts. For example, consider a ciphertext that is generated for patients with a specific disease. Those who do not hold the required attributes cannot decrypt such a ciphertext; however, they can find out what disease those who can decrypt it are suffering from. To address such privacy issues, policy-hidden CP-ABE schemes [66, 70, 93], classified as partially hidden and fully hidden [142], were proposed. Partially policy-hidden CP-ABE schemes consider attributes as name-value pairs and hide only the value part of attributes. However, fully hidden schemes hide both name and value of attributes, which makes it difficult and computationally expensive for a user to check if her private key satisfies the access structure of a ciphertext. In 2021, Zhang et al. [144] proposed a partially policy-hidden CP-ABE scheme that is secure against attribute value guessing attacks. Zhang et al.'s scheme outsources decryption testing, which checks if a private key satisfies an access structure, to a cloud server for the sake of efficiency. In [143], a fully policy-hidden CP-ABE scheme is proposed utilizing hidden vector encryption (for hiding access structures) and blockchain (for decryption testing). Recently, Yang et al. [136] proposed another fully policy-hidden scheme based on the concept of private set intersection. Yang et al.'s scheme [136] supports large universes and outsources heavy computations required for encryption, policy hiding, and decryption testing to multiple edge servers.

Another important feature of modern access control systems like ABAC that is missing in ABE schemes is *obligations*, which allow to enforce extra constraints that cannot be addressed by normal access control policies (i.e., access structures). Hence, this paper augments ABE schemes with obligations by utilizing Intel SGX, which provides trusted hardware enclaves. Since CP-ABE is more suitable for realizing fine-grained access control, a CP-ABE is augmented with obligations in

this paper. However, other types of ABE can also be augmented with obligations based on the approach proposed in this paper.

Intel SGX applications: Since SGX provides a trusted execution environment, researchers have tried to use it in different contexts to achieve a higher level of security [21, 32, 39, 75, 86, 120, 130].

The SGX is also used to provide an accountable and trustworthy function as a service (FaaS) [7]. The trusted execution environment provided by the SGX ensures both the integrity of the outsourced computations, and the correctness of the measured usage information (for correct billing). It also minimizes the leakage of the information to the service provider. In [104], Qiang et al. enhance the security (confidentiality and integrity) of serverless computing services with the help of Intel SGX. The remote attestation capability of the SGX is used to provide integrity (verifying the integrity of the function modules) and the SGX enclave is used to provide confidentiality (protecting the core modules of the API gateway). Deployment of cloud micro services suitable for dealing with sensitive data is very important and difficult. In [20], Intel SGX is integrated into micro services to address the security and privacy concerns related to sensitive data.

PRESAGE [27] is a framework for genomic data outsourcing (i.e., genetic testing) that uses Intel SGX to provide acceptable levels of security and privacy. The Intel SGX is used not only for sealing the genomic data (to be stored on public clouds) but also for providing secure genetic query matching by means of the remote attestation. In [67], a privacy preserving approach is proposed for DNA processing. The Intel SGX plays a key role in the proposed approach as it makes the read alignment (finding the position of DNA sequences) secure. It is demonstrated that the proposal is much faster than other related approaches thanks to Intel SGX. Shaon et al. [121] also used SGX to propose a secure framework for genetic data analytics in untrusted cloud setups.

Tramer and Boneh [129] proposed a framework for efficient and secure execution of deep neural networks using the Intel SGX, where DNN computations are divided between untrusted and trusted entities. The DNN inference is outsourced, performing the computations on a faster (untrusted) co-located processor, which are then verified in a trusted execution environment. Cheng et al. [31] improved the security of blockchains by means of hardware-based trusted execution environments, with a solution based on the Intel SGX.

Pires et al. [102] claimed that querying search engines endangers the privacy of web users as well as existing privacy-preserving solutions are not secure against user re-identification attacks, and used Intel SGX to preserve the privacy of search engines' users. In [36], a secure and privacy-preserving architecture (called Fidelius) for web browsers is proposed using Intel SGX. Fidelius protects user secrets even if the underlying browser and operating system are fully under control of an adversary. PubSub-SGX [8] is a content-based publish/subscribe system in which SGX is incorporated to guarantee the integrity and confidentiality of data and preserve the privacy of users.

In [49], SGX is used to process data streams in the Internet of Things environments securely or to develop secure in-memory database engines managing

confidential data in rack-scale environments [111].

Obligation Specification: There are several access control policy specification languages that support also the specification of obligations, e.g., XACML [98], Rei [59], and Ponder [34]. However, they only offer syntactic elements for obligation specification, and do not cover different types of obligations, thus none provide a concrete model for specifying obligations. For instance, it is not possible to specify conditional obligations, pre-obligations, or repetitive ones using Rei and Ponder.

XACML is an OASIS standard providing an XML-based policy specification language, with no formal foundation, and a reference architecture. The policy language allows specifying obligations as part of policies. An obligation includes two elements `ObligationID` and `FulfillOn`, where the second determines when the obligation is mandatory, with value being either `Permit` or `Deny`. XACML does not support different types of obligations, and obligations are black boxes. XACML has introduced *advices* as optional obligations, which can be ignored if it is not possible to fulfil them.

Rei [59] is a general-purpose policy language based on deontic logic [41], which supports security, management, and conversation policies. Obligations are not the first-class entities in Rei. However, Rei has mechanisms to detect and resolve conflicts between prohibition policies and non-complex obligations.

Ponder [34] is an object-oriented language for the specification of security and management (of network and distributed systems) policies. In Ponder, obligations are considered as condition-action rules that can be triggered by events. It is claimed that complex obligations can also be specified by means of concurrency operators of Ponder. Ponder addresses the conflicts between obligations (actions of obligations) and policies by halting the target application, which may not be an acceptable solution.

SPL [108] is a policy specification language that defines obligations as events that should be performed in the future and after performing the current event. If obligations cannot be performed (e.g., because of conflicts between obligations and policies), then everything will be rolled back to the state before performing the event triggering the obligation. The expressiveness of the SPL is limited as it does not support different types of obligations.

OSL [51] is an obligation specification language based on linear temporal logic (LTL) for distributed usage control. Obligations in OSL are either "usage restrictions", prohibiting given usages under certain conditions, or "action requirements", mandating the execution of certain actions conditionally (based on time, purpose, event-defined, environment, and cardinality) or unconditionally. The syntax and semantics of OSL are formalized in the formal language Z. OSL obligations can be converted to XrML and ODRL [55], which are digital right management (DRM) specification languages, and then be enforced by existing DRM mechanisms.

Irwin et al. [56] presented a formal metamodel to model and analyze a system from the obligations point of view. They provided formal definitions of secure states (the states without unfulfilled obligations) and accountable states for

obligation management based on their metamodel. Accountable states refer to the states that identify those who did not fulfill obligations. Irwin et al.'s approach is rather restricted and does not cover different types of obligations such as conditional, reoccurring, and pre- obligations. Irwin et al.'s approach is more about the analysis of the obligation enforcement rather than the specification of different types of obligations.

Li et al. [73] extended the XACML standard by designing a language for the specification of obligations. They modeled obligations as state machines communicating with the policy enforcement point (a component of the XACML architecture that is in charge of enforcing access decisions and obligations) through events. Based on Li et al.'s proposal, an obligation can mandate the occurrence of a series of events (and not only one event), which may be dependent on each other.

PoCo [38] is an enforcement system and a language based on the simply-typed lambda-calculus for the composition of policies containing obligations. PoCo makes it possible to compose complex atomic obligations. It prevents insecure situations, which may happen due to incomplete execution of obligations or execution of obligations violating other policies, by allowing policies to check the obligations associated with other policies before their execution.

Ni et al. [89] proposed an obligation model for Privacy-aware Role Based Access Control [90]. The obligation model addresses the undesired issues that may happen due to conflicts between obligations and policies, e.g., an obligation mandates performing an action that is prohibited by another policy. They also provided solutions to determine the relationships between obligations, for instance, an obligation may cover another one. Ni et al. claim that their proposal needs to be improved by addressing unfulfilled obligations, providing a solution for the accountability problem, and providing a mechanism for the optimization of the execution order of obligations.

Hilty et al. [50] provided a formal model for the specification of policies and obligations based on distributed temporal logics. In their model, obligations can be expressed as formulas without past time temporal operators. Hilty et al.'s model supports post-obligations and addresses the observability problem, i.e., helping the reference monitor to be able to check if a post-obligation is fulfilled.

III.9 Conclusions

In this work, we proposed OB-ABE, an Attribute-Based Encryption with enforceable OBligations scheme based on a real-world eHealth case study, the SCOTT project. OB-ABE is as an extension of classical ABE schemes with obligations that are enforced before retrieving a plaintext. In particular, we used CP-ABE as a base-line scheme and modified both the `encryption` and the `decryption` processes to include obligations in such a way that a user might define a set of operations to be executed before returning a plaintext. We formally verified security properties, i.e., enforceable obligations, compatibility, and conservative extension. Finally, we provided a prototype of the OB-ABE

scheme based on the Intel SGX architecture and freely released the source for future research on the topic. The results of our evaluation are encouraging, having obtained reasonable overhead when compared with the underlying CP-ABE scheme.

An alternative could have been to rely on an SGX-enabled cloud provider that implements an ABAC mechanism inside SGX, and thus, the required obligations (if any) would still be enforced by SGX. However, ABAC is not based on cryptographic techniques, but relies on a trusted reference monitor to control access requests. Since this can be bypassed, e.g., by getting direct access to the data on a storage device (in the cloud), one could think of adding a layer of encryption before, or after, the ABAC mechanism, which could again be handled by SGX. But such a solution would now get close to ABE, only that ABE (and accordingly OB-ABE) does not rely on a trusted reference monitor. With ABE a user can encrypt her data based on given attributes and then store the ciphertext on the public clouds. In the above alternative, however, a security administrator specifies access control policies to protect resources (data) of an organization (or a domain). Furthermore, having an access control engine on the cloud causes a single point of failure as all the access requests should be examined and checked by the access control engine. With ABE and the extended version OB-ABE, there would not be such a single point of failure as the decryption operations (as well as encryption operations, attribute compliance checks, and enforcing obligations) would be performed on the users' machines. Moreover, we do not need to trust a cloud provider, instead, we trust a trusted hardware on the users' machines.

Another alternative approach could have been running only ABE directly inside SGX. Such an alternative approach (henceforth "ABE-SGX") would not require performing each encryption and decryption two times as the extra encryption (in OB-ABE) would not be needed anymore. However, ABE-SGX would have a much higher performance overhead than OB-ABE. Since ABE schemes are public-key cryptographic schemes and mostly require expensive pairing-based cryptographic primitives, ABE-SGX would require performing time-consuming operations by SGX, which in turn several page evictions from the EPC to the untrusted memory (in an encrypted format) might be required because of the SGX's limited EPC page size. However, OB-ABE uses SGX for lightweight symmetric encryption/decryption. In other words, ABE-SGX would use SGX for public-key cryptography, which is much slower (heavyweight) than the symmetric encryption required by OB-ABE. In addition, OB-ABE uses the SGX only for ciphertexts with obligations, whereas ABE-SGX would use SGX for every ciphertext, regardless of having obligations or not. Besides, OB-ABE uses SGX's built-in cryptographic library for symmetric encryption. However, SGX's cryptographic library does not support the cryptographic primitives required for ABE. Therefore, new cryptographic libraries need to be developed for SGX to be able to run ABE inside SGX. Developing custom libraries for SGX not only is very complicated but also requires many OCALLs, as SGX does not support system calls, which would negatively impact performance. Besides, in order to force users to run ABE inside SGX (to enforce obligations by SGX), ABE

algorithms, i.e., Setup, KeyGen, Encryption, and Decryption, would need to be modified and bonded to SGX. Otherwise, users who have the required private keys (i.e., attributes) can run ABE (which its algorithms are public) on a non-SGX machine and ignore the attached obligations. Modifying ABE algorithms might open up new security issues. However, as discussed in Property III.3 (in Section III.6), OB-ABE does not affect the security of the underlying ABE scheme.

Acknowledgments

This work was partially supported by the Swedish Foundation for Strategic Research (SSF) and the Swedish Research Council (VR). The authors would like to thank Sergiu Bursuc at the Security and Trust of Software Systems group, University of Luxembourg, for his help and support in verifying the properties of the proposed scheme.

Compliance with Ethical Standards

Conflict of interest. The authors declare that they do not have conflict of interests.

Ethical approval. This article does not contain any studies with human participants or animals performed by any of the authors.

Appendix III.A Intel SGX Security

Intel SGX has been under scrutiny since its early releases and several vulnerabilities have been discovered. These are often published along with corresponding countermeasures. The following provides a list of attacks against Intel SGX and possible remedies.

III.A.1 Vulnerabilities

Xu et al. [135] introduced controlled-channel attacks against SGX-enabled applications by which sensitive information of the protected applications can be extracted. In such attacks, an adversary causes intentional page-faults to get sequence information of page faults, which helps the adversary to infer the data that exists in the protected memory (i.e., the enclave's data). In [25], it is demonstrated that enclave data can be revealed even without intentional page-faults and by monitoring the page table entry (PTE) status, which shows all page read/write attempts. A framework called SGX-Step was presented in [22] to attack SGX enclaves. The SGX-Step makes it possible to configure Advanced Programmable Interrupt Controller (APIC) timer interrupts and then track PTE status. Several attacks [23, 24, 46, 54, 88, 132] have been built on top of the SGX-Step. Another similar attack is the Sneaky Page Monitoring

(SPM) attack [134]. This memory-based attack works based on manipulation of the page's accessed flag in PTE and does not need enclave interrupts contrary to the page-fault attacks. The granularity of the SPM attack can be improved using a cache timing method called Prime+Probe [97], which has been used in cache attacks [19, 33, 43, 85, 117] as well.

Cache attacks are based on the time gap between main memory access and CPU cache access. In order to avoid cache attacks, Intel SGX restricts access to the Enclave Page Cache memory from a non-enclave code. Nevertheless, several cache attacks against SGX-protected programs have been presented. For instance, Brasser et al. [19] extracted 70% of a 2048-bit RSA key during the decryption. Brasser et al.'s attack is applicable when the attacker's process and the enclave process share probes cache line and use the same CPU core. Schwarz et al. [117] presented another cache attack, by creating an enclave and attacking a victim enclave using the Prime+Probe method, which does not require the attacker's process and the enclave process to use the same core. Götzfried et al. [43] presented a cache attack to extract the secret key of an AES algorithm from an enclave. Lee et al. [68] presented a branch prediction attack through branch shadowing, which is possible because the Branch Target Buffer (BTB) address information can be shared between an enclave and the untrusted part (outside the enclave). The goal of the attack was to infer the control flow inside enclaves. BranchScope [37] is another similar attack that does not work based on the BTB. The BranchScope attack can reveal secret information without any knowledge of the internal predictor organization. Bluethunder [54] is also a branch predictor attack for extracting secret information of enclaves. It is demonstrated that the Bluethunder attack is faster than the BranchScope attack as the first one uses a 2-level directional predictor.

SgxPectre [29] is a kind of speculative execution attack that reveals enclave secret information by affecting the branch prediction of the enclave code to change the control flow of the program inside the enclave. The aim of changing the control flow is to perform instructions that make cache-state changes observable. Monitoring the cache-state changes helps to extract enclave secrets. Koruyeh et al. [63] presented a similar attack called SpectreRSB by exploiting the return stack buffer (RSB) instead of the branch predictor unit.

Recently, several new attacks such as RIDL [114], CacheOut [113], ZombieLoad [118], SGAxe [115], and CrossTalk [105] have been presented that are classified as Microarchitectural Data Sampling (MDS) attacks. Rogue In-Flight Data Load (RIDL), CacheOut, and ZombieLoad attacks exploit internal CPU buffers, i.e., the Line Fill Buffers (LFBs) between the L1 Data caches and the L2 caches. RIDL can leak data from loads that are not already in the L1D cache. However, using ZombieLoad, the results of memory loads (the requested data may already exist in the L1D cache) can be leaked. While RIDL and ZombieLoad cannot control which data is loaded into the LFBs and thereafter revealed, CacheOut makes it possible to control the leakage by forcing cache contention on the intended L1D data. The SGAxe attack uses the CacheOut to extract the attestation key from the Quoting enclave and then sign fake attestation quotes. Most of the presented attacks can be mitigated by isolating the victim

enclaves and the attackers on separate CPU cores. However, CrossTalk [105] makes it possible to attack enclaves even with core separation and leak data among different cores using a global staging buffer.

III.A.2 Countermeasures

Most of the presented vulnerabilities either do not expose all the information that exists inside an enclave or need a powerful attacker with significant effort. However, even small leakages can become dangerous over time. Fortunately, Intel and researchers have provided and proposed countermeasures to overcome the attacks exploiting such vulnerabilities. For instance, Intel addresses many weaknesses by changing the hardware of new CPUs, patching the microcode of CPUs (which changes the way that a CPU performs its tasks), and upgrading the system software. ZigZagger, which proposed a remedy for the branch prediction attack presented in [68], converts conditional branches into non-conditional jumps to trampolines (indirect jump vectors). The main idea of ZigZagger was to hide the control flow. Branch shadowing attacks are also addressed by means of control flow randomization [52]. In other words, in [52], conditional branches are eliminated, and the targets of non-conditional branches are hidden through run-time code randomization and modifications at compile-time. Both ZigZagger and the solution proposed in [52] were implemented as extensions for the LLVM compiler [1]. Chen et al. [28] also proposed a solution that restricts access to other CPU cores by creating shadow threads. Shih et al. [122] and Chen et al. [30] addressed controlled-channel attacks using Intel Transactional Synchronization Extensions (TSX) that hide page fault events from OS by handling faults as transaction abort. Strackx et al. [127] proposed a remedy for page-table based controlled-channel attacks utilizing TSX transactions. On the other hand, Fu et al. [40] presented SGX-LAPD as a fix for systems that do not support TSX.

A compiler-assisted solution for page-fault based attacks was proposed by Shinde et al. [123]. Gruss et al. [45] provided a software library that wraps secret handling code and secret data in TSX transactions. Chandra et al.'s solution [26] was to add random noise, e.g., add accesses to dummy data, to the algorithms. Sasy et al. [112] and Ohrimenko et al. [95] proposed solutions based on Oblivious RAM (ORAM), which removes access patterns by encrypting and shuffling them. Still other solutions exist that are based on the randomization of the address space layout [119] and decryption of the code at runtime [116].

References

[1] Abadi, M., Blanchet, B., and Comon-Lundh, H. "Models and Proofs of Protocol Security: A Progress Report". In: *Computer Aided Verification*. Vol. 5643. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2009, pp. 35–49.

[1] https://llvm.org/

[2] Abadi, M., Blanchet, B., and Fournet, C. “The applied pi calculus: Mobile values, new names, and secure communication”. In: *Journal of the ACM (JACM)* vol. 65, no. 1 (2017), pp. 1–41.

[3] Abadi, M. and Fournet, C. “Mobile Values, New Names, and Secure Communication”. In: *Proceedings of the 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. Vol. 36. POPL ’01. London, United Kingdom: ACM, 2001, pp. 104–115.

[4] Abdalla, M. and Bellare, M. “Increasing the Lifetime of a Key: A Comparative Analysis of the Security of Re-keying Techniques”. In: *Advances in Cryptology — ASIACRYPT 2000*. Ed. by Okamoto, T. Vol. 1976. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2000, pp. 546–559.

[5] Afshar, M., Samet, S., and Hu, T. “An Attribute Based Access Control Framework for Healthcare System”. In: *Journal of Physics: Conference Series* vol. 933 (2017), p. 012020.

[6] Agrawal, S. and Chase, M. “FAME: Fast Attribute-Based Message Encryption”. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’17. Dallas, Texas, USA: Association for Computing Machinery, 2017, pp. 665–682.

[7] Alder, F. et al. “S-FaaS: Trustworthy and Accountable Function-as-a-Service using Intel SGX”. In: *Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop*. New York, NY, USA: ACM, 2019, pp. 185–199.

[8] Arnautov, S. et al. “PubSub-SGX: Exploiting Trusted Execution Environments for Privacy-Preserving Publish/Subscribe Systems”. In: *37th Symposium on Reliable Distributed Systems (SRDS)*. Salvador, Brazil: IEEE Computer Society, 2018, pp. 123–132.

[9] Attrapadung, N. and Imai, H. “Attribute-Based Encryption Supporting Direct/Indirect Revocation Modes”. In: *Cryptography and Coding*. Ed. by Parker, M. G. Vol. 5921. Lecture Notes in Computer Science. Springer, 2009, pp. 278–300.

[10] Attrapadung, N. and Imai, H. “Conjunctive Broadcast and Attribute-Based Encryption”. In: *Pairing-Based Cryptography – Pairing 2009*. Ed. by Shacham, H. and Waters, B. Vol. 5671. Lecture Notes in Computer Science. Springer, 2009, pp. 248–265.

[11] Baden, R. et al. “Persona: An Online Social Network with User-defined Privacy”. In: *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*. New York, NY, USA: ACM, 2009, pp. 135–146.

[12] Barua, M., Lu, R., and Shen, X. “SPS: Secure personal health information sharing with patient-centric access control in cloud computing”. In: *IEEE global communications conference (GLOBECOM)*. Atlanta, GA, USA: IEEE, 2013, pp. 647–652.

[13] Beckert, B., Hähnle, R., and Schmitt, P. H. *Verification of Object-Oriented Software. The KeY Approach*. Berlin, Heidelberg: Springer, 2007.

[14] Bertino, E., Bonatti, P. A., and Ferrari, E. "TRBAC: A temporal role-based access control model". In: *ACM Transactions on Information and System Security (TISSEC)* vol. 4, no. 3 (2001), pp. 191–233.

[15] Bethencourt, J., Sahai, A., and Waters, B. "Ciphertext-Policy Attribute-Based Encryption". In: *IEEE symposium on security and privacy (SP'07)*. Berkeley, CA, USA: IEEE, 2007, pp. 321–334.

[16] Blanchet, B., Abadi, M., and Fournet, C. "Automated verification of selected equivalences for security protocols". In: *The Journal of Logic and Algebraic Programming* vol. 75, no. 1 (2008), pp. 3–51.

[17] Boneh, D. and Franklin, M. "Identity-based encryption from the Weil pairing". In: *Annual international cryptology conference*. Berlin, Heidelberg: Springer, 2001, pp. 213–229.

[18] Brasser, F. et al. "DR. SGX: Automated and adjustable side-channel protection for SGX using data location randomization". In: *Proceedings of the 35th Annual Computer Security Applications Conference*. New York, NY, USA: ACM, 2019, pp. 788–800.

[19] Brasser, F. et al. "Software Grand Exposure: SGX Cache Attacks Are Practical". In: *WOOT*. VANCOUVER, BC, CANADA: USENIX Association, 2017, pp. 1–12.

[20] Brenner, S. et al. "Secure cloud micro services using Intel SGX". In: *IFIP International Conference on Distributed Applications and Interoperable Systems*. Cham: Springer, 2017, pp. 177–191.

[21] Brenner, S. et al. "SecureKeeper: Confidential ZooKeeper using Intel SGX". In: *Middleware Conference*. New York, NY, USA: ACM, 2016, p. 14.

[22] Bulck, J. V., Piessens, F., and Strackx, R. "SGX-Step: A Practical Attack Framework for Precise Enclave Execution Control". In: *SysTEX@SOSP*. SysTEX'17. Shanghai, China: ACM, 2017, 4:1–4:6.

[23] Bulck, J. V. et al. "A Tale of Two Worlds: Assessing the Vulnerability of Enclave Shielding Runtimes". In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. CCS '19. London, United Kingdom: ACM, 2019, pp. 1741–1758.

[24] Bulck, J. V. et al. "Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution". In: *USENIX Security Symposium*. BALTIMORE, MD, USA: USENIX Association, 2018, pp. 991–1008.

[25] Bulck, J. V. et al. "Telling Your Secrets without Page Faults: Stealthy Page Table-Based Attacks on Enclaved Execution". In: *USENIX Security Symposium*. VANCOUVER, BC, CANADA: USENIX Association, 2017, pp. 1041–1056.

[26] Chandra, S. et al. “Securing Data Analytics on SGX with Randomization”. In: *ESORICS (1)*. Vol. 10492. Lecture Notes in Computer Science. Cham: Springer, 2017, pp. 352–369.

[27] Chen, F. et al. “PRESAGE: privacy-preserving genetic testing via software guard extension”. In: *BMC medical genomics* vol. 10, no. 2 (2017), p. 48.

[28] Chen, G. et al. “Racing in Hyperspace: Closing Hyper-Threading Side Channels on SGX with Contrived Data Races”. In: *IEEE Symposium on Security and Privacy*. San Francisco, CA, USA: IEEE Computer Society, 2018, pp. 178–194.

[29] Chen, G. et al. “SgxPectre: Stealing Intel Secrets From SGX Enclaves via Speculative Execution”. In: *IEEE Secur. Priv.* Vol. 18, no. 3 (2020), pp. 28–37.

[30] Chen, S. et al. “Detecting Privileged Side-Channel Attacks in Shielded Execution with Déjà Vu”. In: *AsiaCCS*. ASIA CCS ’17. Abu Dhabi, United Arab Emirates: ACM, 2017, pp. 7–18.

[31] Cheng, R. et al. “Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contracts”. In: *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. Stockholm, Sweden: IEEE, 2019, pp. 185–200.

[32] Coppolino, L. et al. “A comparative analysis of emerging approaches for securing java software with Intel SGX”. In: *Future Generation Computer Systems* vol. 97 (2019), pp. 620–633.

[33] Dall, F. et al. “CacheQuote: Efficiently Recovering Long-term Secrets of SGX EPID via Cache Attacks”. In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* Vol. 2018, no. 2 (2018), pp. 171–191.

[34] Damianou, N. et al. “The Ponder Policy Specification Language”. In: *POLICY*. Vol. 1995. Lecture Notes in Computer Science. Baltimore, Maryland, USA: Springer, 2001, pp. 18–38.

[35] Desharnais, J., Möller, B., and Struth, G. “Kleene algebra with domain”. In: *ACM Transactions on Computational Logic (TOCL)* vol. 7, no. 4 (2006), pp. 798–833.

[36] Eskandarian, S. et al. “Fidelius: Protecting user secrets from compromised browsers”. In: *2019 IEEE Symposium on Security and Privacy (SP)*. San Francisco, CA, USA: IEEE, 2019, pp. 264–280.

[37] Evtyushkin, D. et al. “BranchScope: A New Side-Channel Attack on Directional Branch Predictor”. In: *ASPLOS*. ASPLOS ’18. Williamsburg, VA, USA: ACM, 2018, pp. 693–707.

[38] Ferguson, D. et al. “PoCo: A Language for Specifying Obligation-Based Policy Compositions”. In: *Proceedings of the 2020 9th International Conference on Software and Computer Applications*. ICSCA 2020. Langkawi, Malaysia: Association for Computing Machinery, 2020, pp. 331–338.

[39] Fisch, B. et al. “Iron: functional encryption using Intel SGX”. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’17. Dallas, Texas, USA: ACM, 2017, pp. 765–782.

[40] Fu, Y. et al. “Sgx-Lapd: Thwarting Controlled Side Channel Attacks via Enclave Verifiable Page Faults”. In: *RAID*. Vol. 10453. Lecture Notes in Computer Science. Cham: Springer, 2017, pp. 357–380.

[41] Gabbay, D. et al. *Handbook of deontic logic and normative systems*. Milton Keynes, UK: College Publication, 2013.

[42] Gorbunov, S., Vaikuntanathan, V., and Wee, H. “Attribute-based encryption for circuits”. In: *Journal of the ACM (JACM)* vol. 62, no. 6 (2015), pp. 1–33.

[43] Götzfried, J. et al. “Cache attacks on Intel SGX”. In: *Proceedings of the 10th European Workshop on Systems Security*. EuroSec’17. Belgrade, Serbia: ACM, 2017, pp. 1–6.

[44] Goyal, V. et al. “Attribute-based Encryption for Fine-grained Access Control of Encrypted Data”. In: *Proceedings of the 13th ACM Conference on Computer and Communications Security*. CCS ’06. Alexandria, Virginia, USA: ACM, 2006, pp. 89–98.

[45] Gruss, D. et al. “Strong and Efficient Cache Side-Channel Protection using Hardware Transactional Memory”. In: *USENIX Security Symposium*. VANCOUVER, BC, CANADA: USENIX Association, 2017, pp. 217–233.

[46] Gyselinck, J. et al. “Off-Limits: Abusing Legacy x86 Memory Segmentation to Spy on Enclaved Execution”. In: *ESSoS*. Vol. 10953. Lecture Notes in Computer Science. Cham: Springer, 2018, pp. 44–60.

[47] Harel, D., Tiuryn, J., and Kozen, D. *Dynamic Logic*. Cambridge, MA, USA: MIT Press, 2000.

[48] Hathaliya, J. J. and Tanwar, S. “An exhaustive survey on security and privacy issues in Healthcare 4.0”. In: *Computer Communications* vol. 153 (2020), pp. 311–335.

[49] Havet, A. et al. “Securestreams: A reactive middleware framework for secure data stream processing”. In: *Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems*. DEBS ’17. Barcelona, Spain: ACM, 2017, pp. 124–133.

[50] Hilty, M., Basin, D. A., and Pretschner, A. “On Obligations”. In: *ESORICS*. Vol. 3679. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2005, pp. 98–117.

[51] Hilty, M. et al. “A Policy Language for Distributed Usage Control”. In: *ESORICS*. Vol. 4734. Lecture Notes in Computer Science. Dresden, Germany: Springer, 2007, pp. 531–546.

[52] Hosseinzadeh, S. et al. “Mitigating Branch-Shadowing Attacks on Intel SGX using Control Flow Randomization”. In: *CoRR* vol. abs/1808.06478 (2018), pp. 1–7.

[53] Hu, V. C. et al. “Guide to Attribute Based Access Control (ABAC) Definition and Considerations”. In: *NIST Special Publication (SP)* vol. 800, no. 162 (2014), pp. 1–47.

[54] Huo, T. et al. “Bluethunder: A 2-level Directional Predictor Based Side-Channel Attack against SGX”. In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* Vol. 2020, no. 1 (2020), pp. 321–347.

[55] Iannella, R. “The Open Digital Rights Language: XML for Digital Rights Management”. In: *Information Security Technical Report* vol. 9, no. 3 (2004), pp. 47–55.

[56] Irwin, K., Yu, T., and Winsborough, W. H. “On the modeling and analysis of obligations”. In: *CCS*. CCS ’06. Alexandria, Virginia, USA: ACM, 2006, pp. 134–143.

[57] Al-Issa, Y., Ottom, M. A., and Tamrawi, A. “eHealth Cloud Security Challenges: A Survey”. In: *Journal of Healthcare Engineering* vol. 2019 (2019), pp. 1–15.

[58] Jiang, Y. et al. “Ciphertext-policy attribute-based encryption against key-delegation abuse in fog computing”. In: *Future Generation Computer Systems* vol. 78 (2018), pp. 720–729.

[59] Kagal, L., Finin, T. W., and Joshi, A. “A Policy Language for a Pervasive Computing Environment”. In: *POLICY*. Lake Como, Italy: IEEE Computer Society, 2003, p. 63.

[60] Kim, D. et al. “SGX-LEGO: Fine-grained SGX controlled-channel attack and its countermeasure”. In: *computers & security* vol. 82 (2019), pp. 118–139.

[61] Klooß, M., Lehmann, A., and Rupp, A. “(R)CCA Secure Updatable Encryption with Integrity Protection”. In: *Advances in Cryptology – EUROCRYPT 2019*. Ed. by Ishai, Y. and Rijmen, V. Springer International Publishing, 2019, pp. 68–99.

[62] Kocher, P. et al. “Spectre attacks: Exploiting speculative execution”. In: *IEEE Symposium on Security and Privacy*. San Francisco, CA, USA: IEEE, 2019, pp. 1–19.

[63] Koruyeh, E. M. et al. “Spectre Returns! Speculation Attacks using the Return Stack Buffer”. In: *WOOT @ USENIX Security Symposium*. BALTIMORE, MD, USA: USENIX Association, 2018, pp. 1–12.

[64] Kozen, D. “A completeness theorem for Kleene algebras and the algebra of regular events”. In: *Information and computation* vol. 110, no. 2 (1994), pp. 366–390.

[65] Kozen, D. “Kleene algebra with tests”. In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* vol. 19, no. 3 (1997), pp. 427–443.

[66] Lai, J., Deng, R. H., and Li, Y. “Expressive CP-ABE with Partially Hidden Access Structures”. In: *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*. ASIACCS ’12. Seoul, Korea: Association for Computing Machinery, 2012, pp. 18–19.

[67] Lambert, C. et al. “MaskAl: Privacy Preserving Masked Reads Alignment using Intel SGX”. In: *37th Symposium on Reliable Distributed Systems (SRDS)*. Salvador, Brazil: IEEE, 2018, pp. 113–122.

[68] Lee, S. et al. “Inferring Fine-grained Control Flow Inside SGX Enclaves with Branch Shadowing”. In: *USENIX Security Symposium*. VANCOUVER, BC, CANADA: USENIX Association, 2017, pp. 557–574.

[69] Lehmann, A. and Tackmann, B. “Updatable Encryption with Post-Compromise Security”. In: *Advances in Cryptology – EUROCRYPT 2018*. Ed. by Nielsen, J. B. and Rijmen, V. Vol. 10822. Lecture Notes in Computer Science. Springer International Publishing, 2018, pp. 685–716.

[70] Li, J. et al. “Privacy-aware attribute-based encryption with user accountability”. In: *International Conference on Information Security*. Pisa, Italy: Springer, 2009, pp. 347–362.

[71] Li, J. et al. “Secure attribute-based data sharing for resource-limited users in cloud computing”. In: *Computers & Security* vol. 72 (2018), pp. 1–12.

[72] Li, M. et al. “Scalable and Secure Sharing of Personal Health Records in Cloud Computing Using Attribute-Based Encryption”. In: *IEEE Transactions on Parallel and Distributed Systems* vol. 24, no. 1 (Jan. 2013), pp. 131–143.

[73] Li, N., Chen, H., and Bertino, E. “On practical specification and enforcement of obligations”. In: *CODASPY*. San Antonio Texas, USA: ACM, 2012, pp. 71–82.

[74] Li, Q. et al. “TRAC: Traceable and Revocable Access Control Scheme for mHealth in 5G-Enabled IIoT”. In: *IEEE Transactions on Industrial Informatics* vol. 18, no. 5 (2022), pp. 3437–3448.

[75] Li, X. et al. “A survey on the security of blockchain systems”. In: *Future Generation Computer Systems* vol. 107 (2020), pp. 841–853.

[76] Liu, J., Huang, X., and Liu, J. K. “Secure sharing of Personal Health Records in cloud computing: Ciphertext-Policy Attribute-Based Signcryption”. In: *Future Generation Computer Systems* vol. 52 (2015). Special Section: Cloud Computing: Security, Privacy and Practice, pp. 67–76.

[77] Liu, Z., Cao, Z., and Wong, D. S. “Blackbox traceable CP-ABE: how to catch people leaking their keys by selling decryption devices on ebay”. In: *CCS*. ACM, 2013, pp. 475–486.

[78] Liu, Z., Cao, Z., and Wong, D. S. "White-Box Traceable Ciphertext-Policy Attribute-Based Encryption Supporting Any Monotone Access Structures". In: *IEEE Transactions on Information Forensics and Security* vol. 8, no. 1 (2013), pp. 76–88.

[79] Liu, Z. et al. "Black-Box Accountable Authority CP-ABE Scheme for Cloud-Assisted E-Health System". In: *IEEE Systems Journal* (2022), pp. 1–10.

[80] Lockhart, H. and Campbell, B. "Security assertion markup language (SAML) v2. 0 technical overview". In: *OASIS Committee Draft* vol. 2 (2008), pp. 94–106.

[81] Matetic, S. et al. "ROTE: Rollback Protection for Trusted Execution". In: *USENIX Security Symposium*. VANCOUVER, BC, CANADA: USENIX Association, 2017, pp. 1289–1306.

[82] McKeen, F. et al. "Innovative Instructions and Software Model for Isolated Execution". In: *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy*. HASP '13. Tel-Aviv, Israel: ACM, 2013.

[83] Meddah, N., Jebrane, A., and Toumanari, A. "Scalable lightweight ABAC scheme for secure sharing PHR in cloud computing". In: *International Conference on Advanced Information Technology, Services and Systems*. Tangier, Morocco: Springer, 2017, pp. 333–346.

[84] Meyer, J.-J. C. "A different approach to deontic logic: deontic logic viewed as a variant of dynamic logic". In: *Notre Dame Journal of Formal Logic* vol. 29, no. 1 (1988), pp. 109–136.

[85] Moghimi, A., Irazoqui, G., and Eisenbarth, T. "CacheZoom: How SGX Amplifies the Power of Cache Attacks". In: *CHES*. Vol. 10529. Lecture Notes in Computer Science. Taipei, Taiwan: Springer, 2017, pp. 69–90.

[86] Mokhtar, S. B. et al. "X-search: revisiting private web search using Intel SGX". In: *Middleware Conference*. Middleware '17. Las Vegas, Nevada: ACM, 2017, pp. 198–208.

[87] Mukherjee, S. et al. "Attribute based access control for healthcare resources". In: *Proceedings of the 2nd ACM Workshop on Attribute-Based Access Control*. ABAC '17. Scottsdale, Arizona, USA: ACM, 2017, pp. 29–40.

[88] Murdock, K. et al. "Plundervolt: Software-based fault injection attacks against Intel SGX". In: *2020 IEEE Symposium on Security and Privacy (SP)*. San Francisco, CA, USA: IEEE, 2020, pp. 1466–1482.

[89] Ni, Q., Bertino, E., and Lobo, J. "An obligation model bridging access control policies and privacy policies". In: *SACMAT*. SACMAT '08. Estes Park, CO, USA: ACM, 2008, pp. 133–142.

[90] Ni, Q. et al. "Privacy-Aware Role-Based Access Control". In: *IEEE Secur. Priv.* Vol. 7, no. 4 (2009), pp. 35–43.

[91] Nilsson, A., Bideh, P. N., and Brorsson, J. *A Survey of Published Attacks on Intel SGX*. 2020. arXiv: 2006.13598 [cs.CR].

[92] Ning, J. et al. "White-Box Traceable Ciphertext-Policy Attribute-Based Encryption Supporting Flexible Attributes". In: *IEEE Transactions on Information Forensics and Security* vol. 10, no. 6 (2015), pp. 1274–1288.

[93] Nishide, T., Yoneyama, K., and Ohta, K. "Attribute-Based Encryption with Partially Hidden Encryptor-Specified Access Structures". In: *Applied Cryptography and Network Security*. Ed. by Bellovin, S. M. et al. Vol. 5037. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008, pp. 111–129.

[94] Nishimaki, R. "The Direction of Updatable Encryption Does Matter". In: *Public-Key Cryptography – PKC 2022*. Ed. by Hanaoka, G., Shikata, J., and Watanabe, Y. Springer International Publishing, 2022, pp. 194–224.

[95] Ohrimenko, O. et al. "Oblivious Multi-Party Machine Learning on Trusted Processors". In: *USENIX Security Symposium*. Austin, TX: USENIX Association, 2016, pp. 619–636.

[96] Oleksenko, O. et al. "Varys: Protecting SGX Enclaves from Practical Side-Channel Attacks". In: *USENIX Annual Technical Conference*. BOSTON, MA, USA: USENIX Association, 2018, pp. 227–240.

[97] Osvik, D. A., Shamir, A., and Tromer, E. "Cache Attacks and Countermeasures: The Case of AES". In: *Topics in Cryptology – CT-RSA 2006*. Ed. by Pointcheval, D. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1–20.

[98] Parducci, B., Lockhart, H., and Rissanen, E. "Extensible access control markup language (XACML) version 3.0". In: *OASIS Standard* vol. 2013, no. 1 (2013), pp. 1–154.

[99] Park, J. "Usage control: a unified framework for next generation access control". PhD thesis. George Mason University Fairfax, VA, 2003.

[100] Park, J. and Sandhu, R. "The UCON ABC usage control model". In: *ACM Transactions on Information and System Security (TISSEC)* vol. 7, no. 1 (2004), pp. 128–174.

[101] Picazo-Sanchez, P., Pardo, R., and Schneider, G. "Secure photo sharing in social networks". In: *IFIP International Conference on ICT Systems Security and Privacy Protection*. Rome, Italy: Springer, 2017, pp. 79–92.

[102] Pires, R. et al. "CYCLOSA: Decentralizing Private Web Search Through SGX-Based Browser Extensions". In: *38th International Conference on Distributed Computing Systems (ICDCS)*. Vienna, Austria: IEEE, 2018, pp. 467–477.

[103] Prisacariu, C. and Schneider, G. "A dynamic deontic logic for complex contracts". In: *The Journal of Logic and Algebraic Programming* vol. 81, no. 4 (2012), pp. 458–490.

[104] Qiang, W., Dong, Z., and Jin, H. "Se-Lambda: Securing Privacy-Sensitive Serverless Applications Using SGX Enclave". In: *International Conference on Security and Privacy in Communication Systems*. Singapore, Singapore: Springer, 2018, pp. 451–470.

[105] Ragab, H. et al. "CrossTalk: Speculative Data Leaks Across Cores Are Real". In: *IEEE Symposium on Security & Privacy*. Online: IEEE, 2021, pp. 1–16.

[106] Ray, I. et al. "Applying attribute based access control for privacy preserving health data disclosure". In: *IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*. Las Vegas, NV, USA: IEEE, 2016, pp. 1–4.

[107] Ray, I. et al. "Using attribute-based access control for remote healthcare monitoring". In: *International Conference on Software Defined Systems (SDS)*. Valencia, Spain: IEEE, 2017, pp. 137–142.

[108] Ribeiro, C. et al. "SPL: An Access Control Language for Security Policies and Complex Constraints". In: *NDSS*. San Diego, California: The Internet Society, 2001, pp. 1–19.

[109] Sahai, A. and Waters, B. "Fuzzy identity-based encryption". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Aarhus, Denmark: Springer, 2005, pp. 457–473.

[110] Sandhu, R. S. et al. "Role-based access control models". In: *Computer* vol. 29, no. 2 (Feb. 1996), pp. 38–47.

[111] Sartakov, V. et al. "STANlite–a database engine for secure data processing at rack-scale level". In: *International Conference on Cloud Engineering (IC2E)*. Orlando, FL, USA: IEEE, 2018, pp. 23–33.

[112] Sasy, S., Gorbunov, S., and Fletcher, C. W. "ZeroTrace : Oblivious Memory Primitives from Intel SGX". In: *NDSS*. San Diego, CA, USA: The Internet Society, 2018, pp. 1–15.

[113] Schaik, S. van et al. "CacheOut: Leaking Data on Intel CPUs via Cache Evictions". In: *CoRR* vol. abs/2006.13353 (2020), pp. 1–18.

[114] Schaik, S. van et al. "RIDL: Rogue In-Flight Data Load". In: *IEEE Symposium on Security and Privacy*. San Francisco, CA, USA: IEEE, 2019, pp. 88–105.

[115] Schaik, S. van et al. *SGAxe: How SGX fails in practice*. 2020.

[116] Schuster, F. et al. "VC3: Trustworthy Data Analytics in the Cloud Using SGX". In: *IEEE Symposium on Security and Privacy*. San Jose, CA, USA: IEEE Computer Society, 2015, pp. 38–54.

[117] Schwarz, M. et al. "Malware Guard Extension: Using SGX to Conceal Cache Attacks". In: *DIMVA*. Vol. 10327. Lecture Notes in Computer Science. Bonn, Germany: Springer, 2017, pp. 3–24.

[118] Schwarz, M. et al. “ZombieLoad: Cross-Privilege-Boundary Data Sampling”. In: *CCS*. CCS ’19. London, United Kingdom: ACM, 2019, pp. 753–768.

[119] Seo, J. et al. “SGX-Shield: Enabling Address Space Layout Randomization for SGX Programs”. In: *NDSS*. San Diego, California, USA: The Internet Society, 2017, pp. 1–15.

[120] Sfyrakis, I. and Gross, T. “UniGuard: Protecting Unikernels Using Intel SGX”. In: *IEEE International Conference on Cloud Engineering (IC2E)*. Orlando, FL, USA: IEEE, 2018, pp. 99–105.

[121] Shaon, F. et al. “SGX-BigMatrix: A practical encrypted data analytic framework with trusted processors”. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’17. Dallas, Texas, USA: ACM, 2017, pp. 1211–1228.

[122] Shih, M. et al. “T-SGX: Eradicating Controlled-Channel Attacks Against Enclave Programs”. In: *NDSS*. San Diego, CA, USA: The Internet Society, 2017, pp. 1–15.

[123] Shinde, S. et al. “Preventing Your Faults From Telling Your Secrets: Defenses Against Pigeonhole Attacks”. In: *CoRR* vol. abs/1506.04832 (2015), pp. 1–16.

[124] Smari, W. W., Clemente, P., and Lalande, J.-F. “An extended attribute based access control model with trust and privacy: Application to a collaborative crisis management system”. In: *Future Generation Computer Systems* vol. 31 (2014), pp. 147–168.

[125] Sookhak, M. et al. “Attribute-based data access control in mobile cloud computing: Taxonomy and open issues”. In: *Future Generation Computer Systems* vol. 72 (2017), pp. 273–287.

[126] Staddon, J. et al. “A Content-Driven Access Control System”. In: *Proceedings of the 7th Symposium on Identity and Trust on the Internet*. IDtrust ’08. Gaithersburg, Maryland, USA: Association for Computing Machinery, 2008, pp. 26–35.

[127] Strackx, R. and Piessens, F. “The Heisenberg defense: Proactively defending SGX enclaves against page-table-based side-channel attacks”. In: *arXiv preprint arXiv:1712.08519* vol. abs/1712.08519 (2017), pp. 1–16.

[128] Stubbs, R. *Intel® SGX Technology and the Impact of Processor Side-Channel Attacks*. Fortanix, 10th March 2020. https://fortanix.com/blog/2020/03/intel-sgx-technology-and-the-impact-of-processor-side-channel-attacks/. 2020.

[129] Tramèr, F. and Boneh, D. “Slalom: Fast, Verifiable and Private Execution of Neural Networks in Trusted Hardware”. In: *ICLR*. New Orleans, Louisiana, United States: OpenReview.net, 2019, pp. 1–19.

[130] Tychalas, D., Tsoutsos, N. G., and Maniatakos, M. "SGXCrypter: IP protection for portable executables using Intel's SGX technology". In: *22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*. Chiba, Japan: IEEE, 2017, pp. 354–359.

[131] Van Bulck, J. and Piessens, F. "Tutorial: Uncovering and Mitigating Side-Channel Leakage in Intel SGX Enclaves". In: *Proceedings of the 8th International Conference on Security, Privacy, and Applied Cryptography Engineering (SPACE'18)*. Kanpur, India: Springer, 2018, pp. 1–4.

[132] Van Bulck, J., Piessens, F., and Strackx, R. "Nemesis: Studying Microarchitectural Timing Leaks in Rudimentary CPU Interrupt Logic". In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. CCS '18. Toronto, Canada: ACM, 2018, pp. 178–195.

[133] Veloudis, S. et al. "Achieving security-by-design through ontology-driven attribute-based access control in cloud environments". In: *Future Generation Computer Systems* vol. 93 (2019), pp. 373–391.

[134] Wang, W. et al. "Leaky Cauldron on the Dark Land: Understanding Memory Side-Channel Hazards in SGX". In: *CCS*. CCS '17. Dallas, Texas, USA: ACM, 2017, pp. 2421–2434.

[135] Xu, Y., Cui, W., and Peinado, M. "Controlled-Channel Attacks: Deterministic Side Channels for Untrusted Operating Systems". In: *IEEE Symposium on Security and Privacy*. San Jose, CA, USA: IEEE Computer Society, 2015, pp. 640–656.

[136] Yang, L. et al. "Achieving privacy-preserving sensitive attributes for large universe based on private set intersection". In: *Information Sciences* vol. 582 (2022), pp. 529–546.

[137] Yang, X., Li, W., and Fan, K. "A revocable attribute-based encryption EHR sharing scheme with multiple authorities in blockchain". In: *Peer-to-peer Networking and Applications* (2022), pp. 1–19.

[138] Yuan, E. and Tong, J. "Attributed based access control (ABAC) for web services". In: *IEEE International Conference on Web Services (ICWS'05)*. Orlando, FL, USA: IEEE, 2005, p. 569.

[139] Yüksel, B., Küpçü, A., and Özkasap, Ö. "Research issues for privacy and security of electronic health services". In: *Future Generation Computer Systems* vol. 68 (2017), pp. 1–13.

[140] Zhang, R. et al. "Key escrow-free attribute based encryption with user revocation". In: *Information Sciences* vol. 600 (2022), pp. 59–72.

[141] Zhang, X. et al. "Formal model and policy specification of usage control". In: *ACM Transactions on Information and System Security (TISSEC)* vol. 8, no. 4 (2005), pp. 351–387.

[142] Zhang, Y. et al. "Attribute-Based Encryption for Cloud Computing Access Control: A Survey". In: *ACM Computing Surveys* vol. 53, no. 4 (Aug. 2020).

[143] Zhang, Z. et al. “An Expressive Fully Policy-Hidden Ciphertext Policy Attribute-Based Encryption Scheme With Credible Verification Based on Blockchain”. In: *IEEE Internet of Things Journal* vol. 9, no. 11 (2022), pp. 8681–8692.

[144] Zhang, Z., Zhang, W., and Qin, Z. “A partially hidden policy CP-ABE scheme against attribute values guessing attacks with online privacy-protective decryption testing in IoT assisted cloud computing”. In: *Future Generation Computer Systems* vol. 123 (2021), pp. 181–195.

Paper IV

Process Algebra Can Save Lives: Static Analysis of XACML Access Control Policies using mCRL2

Hamed Arshad, Ross Horne, Christian Johansen, Olaf Owe, Tim A. C. Willemse

In: Mousavi, M.R., Philippou, A. (eds) Formal Techniques for Distributed Objects, Components, and Systems. FORTE 2022. Lecture Notes in Computer Science, vol 13273. Springer, Cham. DOI: 10.1007/978-3-031-08679-3_2.

Abstract

This paper proposes an approach to formally verify XACML policies using the process algebra mCRL2. XACML (eXtensible Access Control Markup Language) is an OASIS standard for access control systems that is much used in health care due to its fine-grained, attribute-based policy definitions, useful in dynamic environments such as emergency wards. A notorious problem in XACML is the detection of conflicts, which arise especially when combining policies, such as when health institutions merge. Our formal translation of XACML policies into mCRL2, using our automated tool XACML2mCRL2, enables us to verify the above property, called consistency, as well as other policy properties such as completeness and obligation enforcement. Verifying policy properties statically allows us to resolve inconsistencies in advance, thus avoiding situations where an access request is denied in a critical situation (e.g., in an ambulance, when lives may be put in danger) just because of incomplete or inconsistent policies. The mCRL2 toolset is especially useful for modeling behaviors of interactive systems, where XACML would be only one part. Therefore, we verify an access control system together with the intended health care system that it is supposed to protect. For this, we exemplify how to verify safety and liveness properties of an assisted living and community care system.

IV.1 Introduction

Access control is a fundamental security mechanism that protects resources based on access control policies. Access control systems are normally part of the

authorization process that checks access requests against policies to ensure that only authorized users get access to resources of a system. One of the more recent and expressive access control models is the Attribute-Based Access Control (ABAC) [17, 24], which provides fine-grained protection based on attributes of subject, object, and action. ABAC offers numerous advantages over conventional access control models and has reached the maturity of OASIS standards with the eXtensible Access Control Markup Language (XACML) [24] and the Security Assertion Markup Language (SAML) [7].

In this paper we focus on e-Health, were ABAC is a popular choice due to the flexibility given by the attributes and the way they cater for fine-grained access control in emergencies [14, 18, 27]. ABAC can handle quite complex access policies, such as for collaborative access control, where multiple subjects are involved (e.g., a doctor needs to be present/logged-in in order for a nurse to perform a procedure). An important feature of the XACML standard architecture is the use of obligations to perform actions before granting/denying access. For example, detailed auditing of health-care processes (such as administering medicines, preparing operation rooms, or home visits) can be done using obligations.

At the heart of ABAC are the access control policies, which can be specified using the policy language provided by the XACML standard. However, developing XACML policies is complex and error-prone because the policies grow in complexity at the same rate as the complexity (not the size) of the systems they are intended to protect. This is exacerbated by the XML-based verbose syntax and the extensive collection of features in XACML. The consistency and completeness of policies are important properties, e.g., a doctor cannot access the medical records of a patient due to inconsistent policies, or a caregiver cannot open the door lock of an elderly in home-care scenarios due to incomplete policies.

Resolving conflicts is currently done at runtime by employing one of the several XACML combining algorithms. For example, the DenyOverrides combining algorithm states that if several applicable rules result in both Permit and Deny, the final decision would be Deny. Such strategies of defaulting to deny access requests may be good for ensuring confidentiality, but they can be detrimental to the availability of the system. In complex and dynamic systems as in e-Health we wish to minimize the number of times that such conflicting situations appear, so to increase the availability of systems where unavailability may put lives at risk. This issue can be addressed by static analysis of access control policies, since the static analysis allows the states to be explored before the system is executed.

This paper presents an approach, and a tool, for verifying XACML policies integrated into their e-Health processes. Our approach is based on a process algebra, called mCRL2 [6, 11, 13], for modeling the behavior of distributed protocols and systems [1, 9]. Using mCRL2 has the advantage of featuring time and (custom) data types, which we use for specifying XACML policies. The mCRL2 process language is accompanied by a powerful toolset, enabling us to simultaneously model XACML policies and e-Health processes subject to such

policies, as well as specifying the properties we wish to verify.

Our goal is to formally verify access control policies and resolve inconsistencies in advance. Static analysis of policies helps to discover inconsistent and incomplete policies before their actual use. Counterexamples generated by the verifier can help security administrators to correct their policies. This paper provides also a tool for automatic transformation of XACML policies into mCRL2 specifications. Our tool, which we call XACML2mCRL2, can be added to the mCRL2 toolset to make it possible to automatically verify XACML policies.

A second goal, which also motivated our choice of process algebra, is to not only verify XACML policies independently, but together with the system that these policies are supposed to control. Since process algebras, like mCRL2, are particularly useful for modeling behaviours of distributed systems, the translations that our tool provides can be combined with models of, e.g., e-Health systems. Formal verification of such systems enables us to prove properties such as liveness or safety, which are relevant for the availability and confidentiality.

Structure of the paper:

Section IV.2 provides basic information about the XACML policy language. Section IV.3 presents our approach for the specification and verification of XACML policies using mCRL2. It first describes the procedure for mapping XACML policies into mCRL2 (Section IV.3.1), then it formulates the desired properties of the XACML policies using the modal μ-calculus (Section IV.3.2), next it explains how to verify the XACML policies (the mCRL2 specifications and properties) using the mCRL2 toolset, and finally, some example policies are analyzed based on the proposed approach. Section IV.4 explains how to analyze the behavior of the systems employing access control schemes using the same approach explained in Section IV.3. Related work is discussed in Section IV.5.

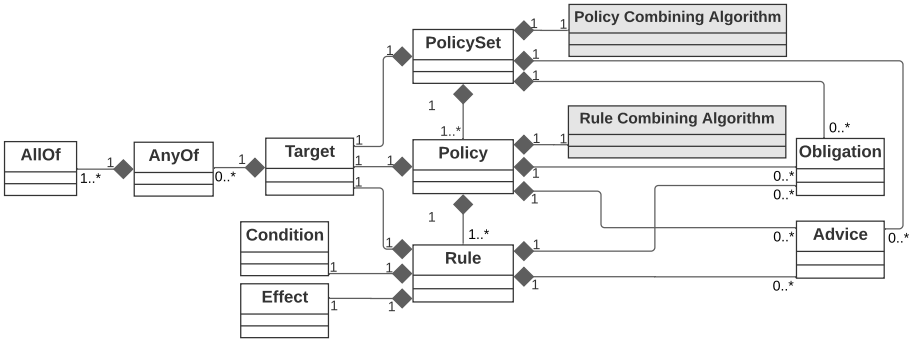


Figure IV.1: XACML Policy language model [24]. As explained in Sec. IV.3.1, the proposed approach does not translate gray boxes to mCRL2.

IV.2 Background on the XACML Policy Language

The XACML standard describes (besides other things such as a reference architecture) a policy specification language, which we will simply refer to as XACML in this paper. As represented in Fig. IV.1, XACML has a hierarchical structure, with the main elements being: `PolicySet`, which includes one or more `Policies` or other `PolicySets`, and `Policy`, which includes one or more `Rules`.

Every `PolicySet`, `Policy`, and `Rule` has a `Target`, which determines the requests to which they are applicable. A `Target` may include a conjunction of `AnyOf` elements, each consisting of a disjunction of `AllOf` elements. An `AllOf` element is a conjunction of pairs `(attribute-name, attribute-value)`, as XACML policies are based on the attributes of subjects, objects, actions, and the environment. The `Target` of a `Rule` may be empty, making the `Rule` applicable to all requests filtered based on the `Targets` of the `PolicySet` and `Policy`.

A `Rule`, normally meant to express a very simple access control policy, has a `Condition` part as well as an `Effect` that is either `Deny` or `Permit`. If the attributes provided in a request match those needed by the `Target` and the `Condition` of a `Rule`, then the `Effect` of the `Rule` will be returned to the parent `Policy`. In the case the attributes do not satisfy the `Condition` or if an error occurs, `NotApplicable` or `Indeterminate`, respectively, will be returned to the parent `Policy`. `Conditions` are more powerful than `Targets` because XACML provides numerous functions (such as *integer-greater-than*, *integer-less-than-or-equal*, *n-of*, *not*, *anyURI-starts-with*) that can be used inside conditions, in addition to the OR/AND constructions that `Targets` are limited to.

`Obligation` or `Advice` expressions may be attached to every `PolicySet`, `Policy`, and `Rule` in order to enforce extra constraints. For example, a `Policy` may use an `Obligation` to require to log successful access to patients medical records. An `Advice` is the same as an `Obligation` with the difference that the `Advice` is optional, i.e., the policy enforcement point (PEP) can ignore an `Advice`, whereas it must always execute all `Obligations`.

Since multiple `Rules` (or `Policies`) with different `Effects` may be applicable to the same request, conflicting rule `Effects` (or decisions) may be reached, which indicates inconsistencies between `Rules` and `Policies`. The XACML standard does not offer any solution to detect such inconsistencies when authoring access control policies. Instead, it provides several combining algorithms, i.e., `Rule Combining Algorithms` and `Policy Combining Algorithms`, to combine contradictory decisions and reach a single decision.

IV.3 Modeling and Analyzing XACML policies

The mCRL2 toolset [6] can be used for analyzing software and concurrent systems [12]. Its main input language is an ACP-inspired process algebra that also has built-in support for frequently-used data types and operations on these, as well as facilities for users to specify their own data types. Properties of systems modeled

in the mCRL2 language can be expressed in the first-order modal μ-calculus. For a detailed account of the mCRL2 language and the property language, we refer to [11–13]; in the remainder of this paper, we explain the relevant syntax and concepts as we go along.

In this section we show how to specify and verify XACML policies using the mCRL2 toolset (version 202106.0). We first explain how we map XACML policies into mCRL2 processes; this is the essence of our tool XACML2mCRL2 [3]. Then, we formally define three desired properties for XACML policies and in the end analyze three example policies.

IV.3.1 Mapping XACML Policies into mCRL2

We encode the XACML components into mCRL2 concepts as follows. An attribute can be considered as a name-value pair. A Rule can be considered as a tuple ⟨*RuleID, Target, Condition, Effect, Obligation*⟩, where *Target* and *Condition* are sets of attributes, and *Effect* can be either *Permit* or *Deny*. The last element is abstracted as a pair of *ObligationID* and a *FulfillOn* element with value either *Permit* or *Deny*, specifying the decision for which the obligation is applicable. A Policy can be abstracted as ⟨*PolicyID, Target, Rules, Obligation*⟩, and similarly a PolicySet as ⟨*PolicySetID, Target, Policies, Obligation*⟩, with both *Rules* and *Policies* being sets. PolicySet, Policy, and Rule can also include advices. However, since the policy enforcement point can ignore enforcing an advice, it can be ignored when analyzing XACML policies.

An mCRL2 specification begins with a declaration of the required data types and actions. We start by defining the following data types for the subject, object, and action attributes:

```
sort SAtt = struct attribute(name:SAttName, value:SAttValue);
sort OAtt = struct attribute(name:OAttName, value:OAttValue);
sort AAtt = struct attribute(name:AAttName, value:AAttValue);
```

The keyword **sort** is used to define new data types. We use the keyword **struct** because these are structured data types (functional data types), defined using other data types for names and values. Normally, all attribute names and values used in the policies that are going to be analyzed need to be listed in the definitions of these sorts. For the sake of simplicity in the text here, we consider the following values for these sorts.

```
sort SAttName   = struct subjectid;
sort SAttValue  = struct CareGiverA|Doctor;
sort OAttName   = struct resourceid;
sort OAttValue  = struct HealthData;
sort AAttName   = struct actionid;
sort AAttValue  = struct Read;
```

We define a data type `Decision` containing the two possible rule effects.

```
sort Decision = struct Permit|Deny;
```

The following sort lists obligation identifiers. Here we keep this list minimal, but our tool populates this with all the obligations found in the XACML files.

```
sort ObgID = struct email|log;
```

We model the main elements of an XACML policy, i.e., PolicySet, Policy, and Rule, as separate processes (starting with keyword **proc**). All the actions performed by these processes need to be declared at the beginning of the mCRL2 specification. If an action is parametrized by some data, then the type of the parameter needs to be stated as well. Below, the `Request` action carries three sets of attributes (i.e., subject, object, and action attributes), whereas a `Response` action carries in addition also a decision. We use finite sets, defined with *FSet*, with the type of the elements specified as parameter.

```
act Request:FSet(SAtt)#FSet(OAtt)#FSet(AAtt);
    Obligation:FSet(SAtt)#FSet(OAtt)#FSet(AAtt)#ObgID;
    Response:FSet(SAtt)#FSet(OAtt)#FSet(AAtt)#Decision;
```

The policy decision point evaluates an access request against the policies and reaches a decision based on the values of attributes provided in the access request. Hence, we define the above-mentioned processes as parametrized processes, where a process carries a set of attributes provided in the request.

The `PolicySet` process checks whether the attributes that exist in the received request match those requested by the target of the policies. If a policy is applicable to a request, the corresponding `Policy` process will be called (i.e., the `PolicySet` process will behave as the corresponding `Policy` process). For example, suppose that a `PolicySet` (e.g., `PolicySet_PLS1`) contains two policies, e.g., `Policy_PL1` and `Policy_PL2`, where the target of the first one is applicable to the requests that contain a subject attribute `attribute(A, B)` and the second one is applicable to the requests containing an object attribute `attribute(C, D)` and an action attribute `attribute(E, F)`.

```
proc PolicySet_PLS1(RS:FSet(SAtt), RO:FSet(OAtt), RA:FSet(AAtt))=
     ((attribute(A, B) in RS) ->
                                  Policy_PL1(RS, RO, RA))
    + ((attribute(C, D) in RO && attribute(E, F) in RA) ->
                                       Policy_PL2(RS, RO, RA));
```

In the above, the `PolicySet` process consists of non-deterministic choice operator `+` and the ternary if-then-else construct `c -> A <>B`, where the evaluation of the Boolean condition `c` determines the behavior. If the condition `c` holds, then the process behaves as the process `A`; otherwise, it behaves as the process `B`. Note that `c ->A` is equivalent to `c ->A <>delta`, where `delta` denotes a process that cannot perform any action. The *in* operator is used to check if an attribute exists in the received request including sets of subject attributes, `RS`, object attributes, `RO`, and action attributes, `RA`. Fig. IV.2 shows how every element of a simplified version of the corresponding XACML specification is transformed to mCRL2 (please note shapes, colors, and labels). Our standard for naming mCRL2 processes is `{PolicySet/Policy /Rule}_{PolicySetId/PolicyId/RuleId}`, where `PolicySetId`, `PolicyId`, and `RuleId` will be extracted from the XACML specification. The red dashed lines box in Fig.IV.2-a represents a logical AND between two conditions; hence, it is transformed to && in mCRL2. In XACML, `access-subject`, `resource`,

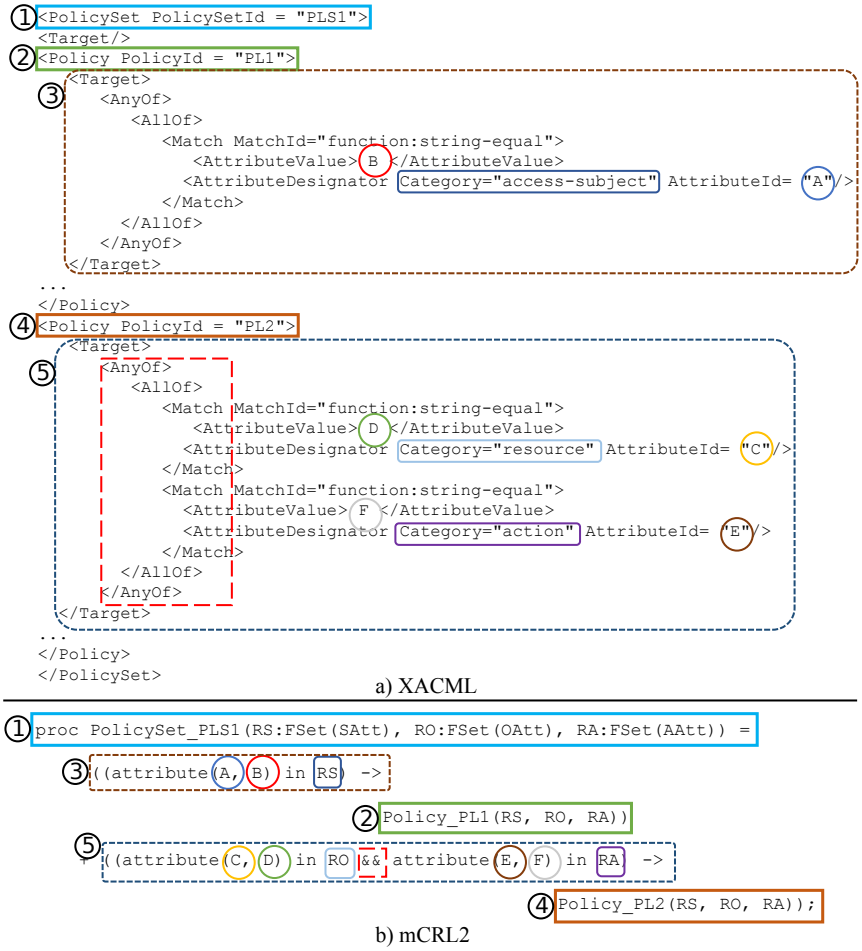


Figure IV.2: A simplified piece of an XACML specification and corresponding mCRL2 specification.

`action` categories are for, respectively, subject, object, and action attributes (`RS`, `RO`, and `RA` in our mCRL2 specifications).

A `Policy` process also checks the target of its rules to call the `Rule` processes that are applicable to the received request. Suppose that `Policy_PL1` has `Rule_R1` and `Rule_R2`, where `Rule_R1` and `Rule_R2` are applicable to the requests containing object attributes, respectively, `attribute(G, H)` and `attribute(I, J)`.

```
1 Policy_PL1(RS:FSet(SAtt), RO:FSet(OAtt), RA:FSet(AAtt)) =
2    ((attribute(G, H) in RO) ->  Rule_R1(RS, RO, RA))
3  + ((attribute(I, J) in RO) ->  Rule_R2(RS, RO, RA));
```

A Rule process checks whether the received request satisfies its condition. If the request satisfies the condition of the rule, then the Rule performs the Response action, which reflects the effect of the rule. However, if there is an Obligation associated with the Rule, then the Rule performs the Obligation action (which carries the request and the obligation ID) before the Response action. In the XACML policy language, an Obligation might be included in a PolicySet, a Policy, or a Rule. In our mapping, we move an Obligation from the PolicySet and Policy levels to the Rule that activates them. For example, suppose Policy_PL1 has an obligation, which mandates writing a log when an access is granted (Obligation ID = log, FulfillOn = Permit). The effect of Rule_R1 and Rule_R2, which are included in Policy_PL1, is Permit and Deny, respectively. Moreover, the condition of Rule_R1 and Rule_R2 includes action attributes attribute(K, L) and attribute(M, N), respectively. In the following specification, the symbol . denotes the sequential composition. Fig. IV.3 also represents the relation between a simplified version of the corresponding XACML specification and the generated mCRL2 specification (follow the shapes, colors, and labels). As shown in Fig. IV.3, the log obligation of Policy_PL1 is moved to the applicable rule, Rule_R1 (see the red box that is labeled with 7).

```
Rule_R1(RS:FSet(SAtt), RO:FSet(OAtt), RA:FSet(AAtt)) =
((attribute(K, L) in RA) ->
                  (Obligation(RS, RO, RA, log).Response(RS, RO, RA, Permit)));

Rule_R2(RS:FSet(SAtt), RO:FSet(OAtt), RA:FSet(AAtt)) =
((attribute(M, N) in RA) ->
                           Response(RS, RO, RA, Deny));
```

An mCRL2 specification ends with the initialization of the system. The system can be initialized by considering all possible combinations of attributes and excluding empty sets for subject, object, and action attributes as follows.

```
init sum RS:FSet(SAtt).sum RO:FSet(OAtt).sum RA:FSet(AAtt).
(RS !={} && RO !={} && RA !={}) ->
                              Request(RS, RO, RA).PolicySet_PLS1(RS, RO, RA);
```

In mCRL2 specifications, the initialization section starts with the **init** keyword. The summation operator, sum, is used for considering all possible values for attributes in RS, RO, and RA. In mCRL2, !, {}, and && denote the negation, empty set, and logical AND, respectively.

We have implemented a prototype, XMACL2mCRL2, for automatic transformation of XACML policies into mCRL2 specification using Java. Since XACML policies are in XML format, XMACL2mCRL2 uses the declarative eXtensible Stylesheet Language Transformations (XSLT) (version 1.0) to define a set of template rules specifying how XACML policies should be transformed into mCRL2. XMACL2mCRL2 may further be integrated into the mCRL2 toolset as a new tool for analyzing XACML policies. The completeness of our transformation tool (and our proposed approach) can be evaluated in terms of the number of XACML policy elements that are covered/supported in the translation from XACML to mCRL2. Our transformation tool covers all elements of the XACML policy model represented in Fig. IV.1 except Policy Combining

`Algorithm` and `Rule Combining Algorithm`. The combining algorithms are intentionally not modeled because one of our goals is to find inconsistencies between policies and fix them when authoring policies, i.e., statically before the access control system is put into production.

The XACML standard offers several functions that can be used in the condition part of rules to form more complex conditions. However, the current version of our implementation may not cover all of them. For example, we can now check whether or not the request contains attribute (Role, Doctor). However, some policies may use other functions, such as *greater-than* or *less-than*, e.g., a policy might be applicable to requests issued by adults (age > 18). Since all the functions offered in the standard can be specified/modeled in mCRL2, the current version of our transformation tool can be improved by considering the remaining functions for the condition part of the rules.

IV.3.2 Specifying the Properties of XACML Policies

This section formulates our desired properties of XACML policies using the first order modal μ-calculus. The properties that we consider are policy-completeness, policy-consistency, and obligation-safety as defined below.

Property P1. *(Policy-Completeness). A set of policies is complete if it covers all the access requests.*

More formally, for every `Request` action, the policy set will inevitably provide a `Response` action. This can be formalized as follows.

```
forall rs:FSet(SAtt), ro:FSet(OAtt), ra:FSet(AAtt).
[Request(rs, ro, ra)]mu X.
([!exists d:Decision.Response(rs, ro, ra, d)]X && <true>true)
```

In the above formula, the symbols `forall` and `&&` denote the conventional first-order logic constructs and can be interpreted as usual. The modal operators `[_]_` and `<_>_` allow to reason about the behavior of the process. A state satisfies formula `[A]`ϕ if all states reached by an action taken from a set of actions `A`, satisfy formula ϕ. Sets of actions are described using first-order logic; for instance, `true` describes the set of all actions, `exists` denotes set union and the `!` operator denotes set complement. The subformula `[Request(rs,ro,ra)]`ϕ captures exactly those states whose `Request`-successor states satisfy ϕ. The operator `<A>`ϕ is dual, and holds true of a state if it has some `a`-transition (with `a` taken from set `A`) leading to a state satisfying ϕ. Note that the state formula `true` holds true in all states. A state therefore satisfies `<true>true` whenever it can execute *some* action. Finally, the subformula shaped `mu X.([!A]X &&<true>true)` describes exactly those states for which executing an action taken from the set `A` is at some point unavoidable; in our case, `A` is the set of `Response` actions with either a `Deny` or `Permit` decision, but attribute sets that match those of the `Request` action.

Property P2. *(Policy-Consistency). A set of policies is conflict-free if there is no inconsistency between policies.*

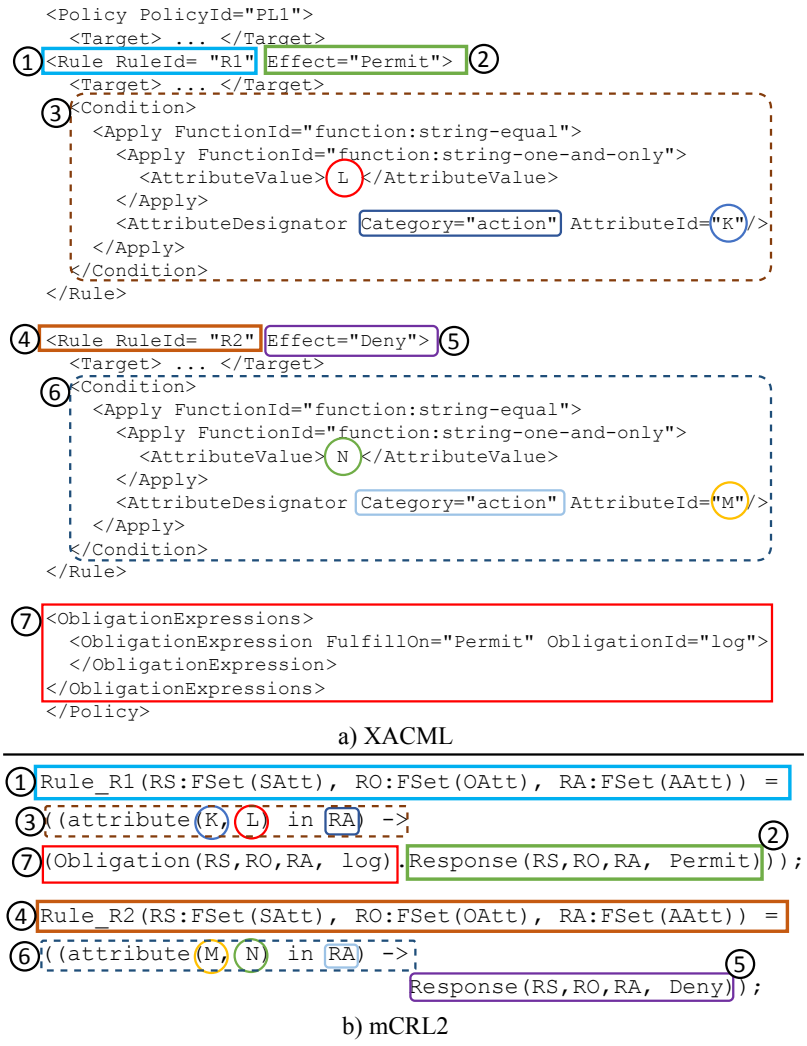


Figure IV.3: a) A simplified XACML specification and; b) the corresponding mCRL2 specification.

This property is important when integrating policies from different domains. Stated more formally, this property requires that executing a `Request` action (with concrete attributes) cannot lead to both a `Deny` and a `Permit` decision. This can be expressed as follows:

```
forall rs:FSet(SAtt), ro:FSet(OAtt), ra:FSet(AAtt).
[Request(rs, ro, ra)]
!(<true*.Response(rs, ro, ra, Deny)> true &&
<true*.Response(rs, ro, ra, Permit)> true)
```

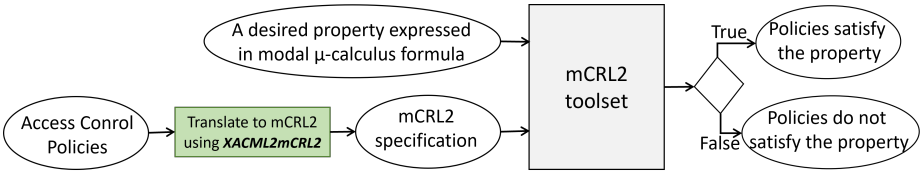


Figure IV.4: Analyzing the specifications using the mCRL2 toolset

Here, the subformulas shaped `<true*.A>true`, using the Kleene * and having the same meaning as `mu X.(<true>X||<A>true)`, describe precisely those states that, in a finite number of steps, can reach a state that can execute an action from the set `A`. Our formula thus describes that after any `Request` action, it is impossible to both reach a state that can execute a `Response` action with a `Deny` decision and a state that can execute a `Response` action with a `Permit` decision.

Property P3. *(Obligation-Safety). A concrete Request either will always yield an Obligation, or it will never yield an Obligation.*

Intuitively, this means that executing a `Request` action cannot lead to a state in which both an `Obligation` action and a non-`Obligation` action are enabled at the same time.

```
forall rs:FSet(SAtt), ro:FSet(OAtt), ra:FSet(AAtt).
[Request(rs,ro,ra)]!(<exists o:ObgID.Obligation(rs,ro,ra,o)>
true && <!exists o:ObgID.Obligation(rs, ro, ra, o)>true)
```

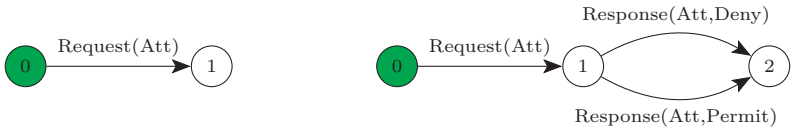
After translating XACML policies into mCRL2 using our *XACML2mCRL2* tool and specifying the desired properties using the modal μ-calculus, the model checker of the mCRL2 toolset can be used to do the verification as shown in Fig. IV.4. If the policies satisfy the desired property, a `true` will be returned as the result. However, if the property is violated, a `false` is returned, along with a counterexample illustrating the violation.

Example IV.1. *Consider a rule that allows Doctors to Read patients' HealthData.*[1]

$$\textbf{Rule 1}\text{: } ((resourceid = HealthData) \wedge (actionid = Read) \wedge (subjectid = Doctor)) \Rightarrow Permit$$

Analyzing the corresponding mCRL2 specification shows that the Policy-Completeness property is not held. The verification engine returns the counterexample represented in Fig. IV.5a showing an access request that this rule does not cover. Policy-Consistency holds because there is no other rule that can cause conflicts, and the same for Obligation-Safety as the rule has no obligation expression.

[1]The XACML version of all the rules and the corresponding mCRL2 specifications generated using our XACML2mCRL2 tool are available in [3]



(a) (b)

Figure IV.5: (a) A counterexample violating the *Policy-Completeness* property in Example IV.1. (b) A counterexample violating the *Policy-Consistency* property in Example IV.4. Att = {attribute(subjectid, CareGiverA)}, {attribute(resourceid, HealthData)}, {attribute(actionid, Read)}.

Example IV.2. *Consider adding to the policy in Example IV.1 a rule that allows Doctors to Read everything and has an obligation for logging all successful accesses.*

Rule 2: $((subjectid = Doctor) \wedge (actionid = Read) \Rightarrow \langle Permit, Obligation(log)\rangle$

Analyzing the updated policy shows that only Policy-Consistency is held. The counterexample in Fig. IV.6 shows that the Obligation-Safety property is violated because there is a request that can be covered by **Rule 2** but it is possible to get a permit response for that (by **Rule 1**) without performing the log obligation.

Example IV.3. *Consider adding to the policy in Example IV.1 another rule to deny access requests to the HealthData of patients if the requester is not a Doctor.*

Rule 3: $((resourceid = HealthData) \wedge (actionid = Read) \wedge \mathbf{NOT}(subjectid = Doctor)) \Rightarrow Deny$

Analyzing this updated policy shows that all our desired properties are held.

Example IV.4. *Update the policy of Example IV.3 by adding a rule to allow CareGiverA (e.g., ambulance nurse) to Read the HealthData.*

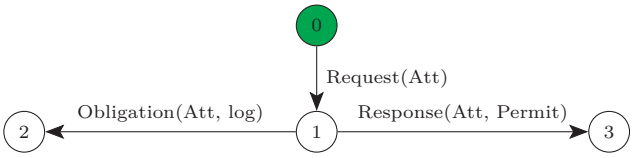


Figure IV.6: A counterexample violating the *Obligation-Safety* property in Example IV.2. Att = {attribute(subjectid, Doctor)}, {attribute(resourceid, HealthData)}, {attribute(actionid, Read)}.

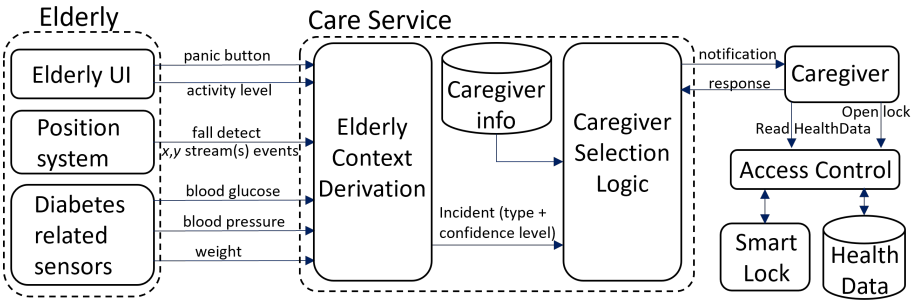


Figure IV.7: Architecture of the Assisted Living and Community Care System.

Rule 4: $((resourceid = HealthData) \wedge (actionid = Read) \wedge (subjectid = CareGiverA)) \Rightarrow Permit$

We can verify that Policy-Completeness and Obligation-Safety hold, but for Policy-Consistency we are shown the counterexample from Fig. IV.5b, which demonstrates a conflict between **Rule 3** and **Rule 4**.

IV.4 System behavior in presence of XACML policies

This section demonstrates how we can use mCRL2 to formally specify and verify also the system around the access control policies, thus allowing to perform XACML policy verification in context.

Use case informal description:

Our use case is taken from a pilot called Assisted Living and Community Care Systems (ALCCS) coordinated by Phillips research in a European project called SCOTT[1]. The goal of the ALCCS pilot is to develop a system for elderly home care with a simplified architecture depicted in Fig. IV.7. Along the way we describe its formalization in mCRL2.

Bob is an elderly person living alone in a smart home equipped with a variety of sensors, e.g., for measuring activity level or blood pressure. Important for us is the panic button, which can be used to get help when needed, and the fall detection sensors. Sensor readings are being sent to a storage and processing unit called Elderly Context Derivation (ECD), which uses these to raise emergency alerts (e.g., if Bob presses the panic button or has fallen).

When there is an emergency, a list of potential caregivers, who can be professionals or neighbors, will be notified. Once a caregiver receives the notification from the ECD (i.e., CareService) and proceeds with the case, the caregiver will be granted access to Bob's house and medical records. In this

[1]EU Horizon 2020 ECSEL Joint Undertaking project SCOTT – Secure COnnected Trustable Things (https://scottproject.eu/)

scenario, it is important that whenever Bob falls or presses the panic button, i.e., when there is an alarm, then Bob should eventually get help from a caregiver, i.e., the alarm should eventually be handled by a caregiver. Moreover, it is also important that only a caregiver who has been assigned to an elderly can open the door lock of the elderly's house in the case of an emergency.

Modeling and verification of the ALCCS:

The interacting components represented in Fig. IV.7 are modeled as separate mCRL2 processes running in parallel. These processes are initialized below, where we restrict the allowed actions to those in the list `Act` and define the synchronization pairs in the list `Com` (see the full specification in [3]). An example of a pair of synchronizing actions is `SndReply|RcvReply -> Reply`, resulting in the action `Reply`.

```
1 init allow(Act, comm(Com,
2 rename({Request->RcvACRequest, Response->SndACResponse},
3 Elderly||CareService({CG1, CG2})||Lock(false, false)||
4 AccessControl||CareGiver({attribute(subjectid, CareGiverA)},
5 {attribute(resourceid, HealthData)},
6 {attribute(actionid, Read)})));
```

In the specification of these processes, the following data types and actions are used, where `EL` represents elderly IDs (it is assumed that there are two caregivers and two elderly patients). In the full specifications from [3] one can see that we also add in the actions and sorts used in the processes related to the access control policies from the previous section.

```
1 sort CareGivers = struct CG1 | CG2;
2 sort EL = struct EID1 | EID2;
3 act Fall, Panic, Read, Willing, Notwilling;
4 SndAlarm, RcvAlarm, Alarm:EL;
5 ...
```

We model a simplified version of the elderly patient, where an alarm is raised by the `Elderly` process when either falling or pressing the panic button. We define recursive processes, where the specified behavior continues indefinitely (unless forced to wait indefinitely for their communicating parties).

```
1 proc Elderly=(Fall + Panic).sum E:EL.SndAlarm(E).Elderly;
```

The `CareService` receives an alarm and notifies caregivers. This has been initialized above with a list of two caregivers `{CG1, CG2}`. The `CareService` assigns a caregiver who is willing to handle the emergencies.

```
1 CareService(L:FSet(CareGivers)) = sum E:EL.RcvAlarm(E).
2 set_emergency(E, true).sum cg:CareGivers.(cg in L) ->
3 SndNotification(E, cg).set_assignment(E, cg, true).
4 RcvFinished(E, cg).set_emergency(E, false).
5 set_assignment(E, cg, false).CareService();
```

The `CareGiver` process either accepts or rejects to be available for handling the emergencies (notice the non-deterministic choice operator `+` on line 6). Recall

that CareGiver was initialized with three sets of attributes. After receiving the assignment message from the CareService, the caregiver is supposed to enter the elderly's house, and thus the CareGiver process sends an OpenLock message to the Lock process of the elderly's house.

```
CareGiver(RS:FSet(SAtt), RO:FSet(OAtt), RA:FSet(AAtt)) =
Willing.sum E:EL, CG:CareGivers.RcvNotification(E, CG).
SndOpenLock(E, CG).((SndACRequest(RS, RO, RA).sum D:Decision.
    RcvACResponse(RS, RO, RA, D).(D == Permit) ->
                                Read.SndFinished(E, CG).CareGiver())
 + (SndFinished(E, CG).CareGiver()))
+ Notwilling.CareGiver();
```

The lock can be opened only if *(i)* there is an emergency for the elderly and *(ii)* the requester is assigned as the caregiver for the elderly (notice the operator && inside the condition to the deterministic conditional choice operator).

```
Lock(ev, av:Bool) = sum E:EL,CG:CareGivers.RcvOpenLock(E,CG).
((ev && av) ->
              LockOpened(E,CG) <> Rejected(E,CG)).Lock(ev, av)
+ sum E:EL. sum ev':Bool.get_emergency(E,ev').Lock(ev', av)
+ sum E:EL, CG:CareGivers.sum av':Bool.get_assignment(E, CG, av').Lock
    (ev, av');
```

After getting inside the elderly's house, the caregiver may try (notice again the non-deterministic choice on line 5 in the CareGiver process) to read the HealthData of the elderly by sending an access request (on line 4). The access request will be evaluated based on the existing access control policies through the AccessControl process, which is defined to make the processes related to the access control policies recursive. Here, Rule 4 from the previous section, is the only policy that we used for evaluation of access requests (the full specification in [3] contains all the policies that were introduced in Sec. IV.3).

```
AccessControl =
(sum RS:FSet(SAtt).sum RO:FSet(OAtt).sum RA:FSet(AAtt).
(RS !={} && RO !={} && RA !={}) ->
         Request(RS,RO,RA).PolicySet_root(RS, RO, RA)).AccessControl;

PolicySet_root(RS:FSet(SAtt), RO:FSet(OAtt), RA:FSet(AAtt)) =
Policy_Policy1(RS, RO, RA);

Policy_Policy1(RS:FSet(SAtt), RO:FSet(OAtt), RA:FSet(AAtt)) =
((attribute(resourceid, HealthData) in RO) &&
(attribute(actionid, Read) in RA)) ->
                                    Rule_Rule4(RS, RO, RA);

Rule_Rule4(RS:FSet(SAtt), RO:FSet(OAtt), RA:FSet(AAtt)) =
(attribute(subjectid, CareGiverA) in RS) ->
                                        Response(RS, RO, RA, Permit);
```

After handling the emergency, the CareGiver informs the CareService that the case is done, which in turn closes the emergency and unassigns the caregiver.

Specifying the system properties:

We exemplify two behavioral properties that we desire of our system above. First we are interested in a form of conditional liveness (i.e., some event will eventually happen under certain conditions), which are sometimes called *response* properties.

Property S1. *(Response). Invariantly, every alarm must eventually be handled by some caregiver.*

```
[true*]forall e:EL.
[SndAlarm(e).(!exists cg:CareGivers.Finished(e,cg))*]
<true*.exists cg:CareGivers.Finished(e,cg)>true
```

Here, the subformula of the form `[A*]`ϕ, which is shorthand for `nu X.([A]X &` `&`ϕ`)`, describes those states from which all states reachable by executing zero or more actions from the set `A`, satisfy property ϕ.

Property S2. *(Safety). Opening the door lock (LockOpened action) is not permitted, as long as there is no emergency for the elderly. Furthermore, only the assigned caregiver can open the door lock.*

This property can be split into the two following requirements:

Property S2-A. *Only a caregiver assigned to an elderly can open the lock.*

```
forall e:EL,cg:CareGivers.nu X.([!assignment(e,cg,true)]X &&
[LockOpened(e, cg)]false && [assignment(e,cg,true)] nu Y.
([!assignment(e,cg,false)]Y && [assignment(e,cg,false)]X))
```

In the above formula, the two alternating greatest fixed points are used to characterize the situation that, along a path either no `assignment` has happened, and so `LockOpened` should not be enabled, expressed by `[LockOpened(e, cg)]false`, *or* an `assignment` has just happened. In the latter case we descend into the fixed point `Y`, where any action is permitted so long as the caregiver is not unassigned. By unassigning a caregiver we again recurse to `X`.

Property S2-B. *An assignment for an emergency must be preceded by the emergency, and is only 'valid' for as long as the emergency is 'active'.*

```
forall e:EL.nu X.([!emergency(e, true)]X &&
[exists cg:CareGivers.assignment(e, cg, true)]false &&
[emergency(e, true)]nu Y.
([!exists cg:CareGivers.assignment(e, cg, false)]Y &&
[exists cg:CareGivers.assignment(e, cg, false)] X))
```

Analyzing the model (with 212 states and 596 transitions) based on **Property S1** and **Property S2** (both **Property S2-A** and **Property S2-B**) show that all the properties are held.

IV.5 Related Work

Bryans [4] used the CSP [15] and the FDR model-checking tool for analyzing access control policies. They expressed RBAC policies in CSP and then used the FDR to analyze the CSP specifications and compare policies (to check if two policies are equivalent). However, Bryans did not consider different properties and did not model the condition part of rules nor obligations.

Kolovski et al. [19] employed the Pellet description logic reasoner to find equivalent, redundant, and incompatible XACML policies. However, Kolovski et al. also did not take into account rule conditions nor obligations.

Ahn et al. [2] used Answer Set Programming (ASP) [20, 22] and ASP solvers for analyzing XACML-based RBAC policies. Ahn et al.'s approach does not handle obligations nor complex conditions and attribute functions.

Fisler et al. [8] proposed a method for analyzing XACML-based RBAC policies by representing the policies using Multi-Terminal Binary Decision Diagrams (MTBDD) [10]. However, the proposed method does not verify all the desired properties of policies and does not take into account all elements of XACML policies. Rao et al. [26] proposed an algebra, called Fine-grained Integration Algebra (FIA), for integration of XACML policies. FIA also uses MTBDDs for representing XACML policies. Policies can be integrated by mapping operations on the policies onto operations on the MTBDDs, which represent policies. After mapping operations, the resulted MRBDD can be traversed to generate an XACML policy that is the result of the integration of two or more policies. Hu et al. [16] proposed a policy-based segmentation method to detect and resolve policy anomalies (conflicts and redundancies). Hu et al.'s method first represents (parses) policies using the Binary Decision Diagram (BDD) [5], then it transfers rules into Boolean expressions. Next, it replaces Boolean expressions with Boolean variables. After that, it identifies anomalies using two proprietary algorithms. Morisset et al. [23] also employed BDDs to address the problem of missing information in ABAC. They proposed a framework for efficient extended evaluation of XACML policies, which checks all the possible outcomes of the evaluation of a given request by considering all possible values for the hidden attributes (i.e., by extending the initial request). Lin et al. [21] proposed a policy analyzer through the combination of MTBDDs (to represent/parse policies) and a SAT solver (to check if two representations are similar). The main goal was to find the similarities between XACML policies.

Turkmen et al. [28] proposed a framework based on satisfiability modulo theories (SMT) for the verification of XACML policies. The goal was to convert policies into SMT formulas and verify the desired properties using SMT solvers.

Another relevant work is the formalization of XACML in terms of multi-valued logics presented in [25]. Ramli et al. [25] provided an abstract syntax for XACML and formalized combining algorithms as operators on a partially ordered set of decisions.

In our proposed approach, mCRL2 is used for specifying all elements of XACML policies (for ABAC), including obligations, and analyzing the most important properties of access control policies. Furthermore, access control

policies are also analyzed *in context*, i.e., together with the system that these policies are supposed to protect. The system surrendering access control policies is also specified and analyzed using mCRL2. The system and access control policies are combined using the parallel composition offered by mCRL2. The use of parallel composition and analysis of access control policies in context distinguish the approach presented in this paper from prior approaches and especially those based on formal methods. It would be interesting for future work to verify the soundness of our translation of XACML into mCRL2 with respect to the formal semantics for XACML provided by Ramli et al. [25].

IV.6 Conclusion

We have presented a methodology supported by a tool for formal verification of access control policies, which is built on top of mCRL2. Our XACML2mCRL2 tool implements the mapping from XACML policies into mCRL2 specifications as described in Sec. IV.3. The mapping covers every element of XACML policies, i.e., policy set, policy, and rule, and allows to formally verify the completeness (**Property P1**) and consistency (**Property P2**) of the XACML policies using the first-order modal μ-calculus. Moreover, in contrast to other related approaches surveyed in Sec. IV.5, XACML2mCRL2 takes into account the obligation expressions; for instance, Example 2 shows a violation of our Obligation-Safety property (**Property P3**). The model checker provided by the mCRL2 toolset automatically generates counterexamples, such as those shown in Fig. 3 and Fig. IV.6, useful for detecting and resolving incomplete and inconsistent policies.

To analyze the XACML policies *in context*, i.e., together with the system that these policies are supposed to protect, we have modeled an e-Health use case represented in Fig. IV.7. We have integrated the mCRL2 specifications generated by XACML2mCRL2 into the use case model, including the Elderly, CareGiver, CareService, and Lock processes. We have formulated and verified liveness and safety properties, **Property S1** respectively **Property S2**, to make sure that an elderly will always receive help in the case of an emergency, and respectively only assigned caregivers can open the door lock of the elderly person's house. Therefore, we can conclude that our methodological approach to formal verification of access control policies can potentially be used to avoid critical problems in, for example, e-Health systems. Indeed, mCRL2 verifies automatically liveness and safety properties for our use case having hundreds of states and transitions, which would be difficult to analyze manually.

Our XMLACL2mCRL2 tool translates only XACML access control policies to mCRL2 specifications. The **P*** formulas (i.e., **Property P1**, **Property P2**, and **Property P3**) presented in the paper are generic formulas for XACML policies. In other words, the provided formulas can be used in different environments for the same properties, i.e., *Policy-Completeness*, *Policy-Consistency*, and *Obligation-Safety*. Therefore, neither the specification of **P*** formulas needs to be automated nor the user needs to understand the generated mCRL2 specifications

(and the provided formulas) for analyzing XACML access control policies.

References

[1] Aceto, L. et al. *Reactive Systems: Modelling, Specification and Verification*. Cambridge University Press, 2007.

[2] Ahn, G. et al. "Representing and Reasoning about Web Access Control Policies". In: *Proceedings of the 34th Annual IEEE International Computer Software and Applications Conference, COMPSAC 2010, Seoul, Korea, 19-23 July 2010*. IEEE Computer Society, 2010, pp. 137–146.

[3] Arshad, H. et al. *GitHub repository for "Process Algebra Can Save Lives: Static Analysis of XACML Access Control Policies using mCRL2"*. https://github.com/haamedarshad/XACML2mCRL2. 2022.

[4] Bryans, J. W. "Reasoning about XACML policies using CSP". In: *Proceedings of the 2nd ACM Workshop On Secure Web Services, SWS 2005, Fairfax, VA, USA, November 11, 2005*. ACM, 2005, pp. 28–35.

[5] Bryant, R. E. "Graph-Based Algorithms for Boolean Function Manipulation". In: *IEEE Trans. Computers* vol. 35, no. 8 (1986), pp. 677–691.

[6] Bunte, O. et al. "The mCRL2 Toolset for Analysing Concurrent Systems - Improvements in Expressivity and Usability". In: *TACAS (2)*. Vol. 11428. Lecture Notes in Computer Science. Springer, 2019, pp. 21–39.

[7] Cantor, S. et al. *Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0*. 2005.

[8] Fisler, K. et al. "Verification and change-impact analysis of access-control policies". In: *27th International Conference on Software Engineering (ICSE 2005), 15-21 May 2005, St. Louis, Missouri, USA*. ACM, 2005, pp. 196–205.

[9] Fokkink, W. *Modelling distributed systems*. Springer Science & Business Media, 2007.

[10] Fujita, M., McGeer, P. C., and Yang, J.-Y. "Multi-Terminal Binary Decision Diagrams: An Efficient Data Structure for Matrix Representation". In: *Formal Methods in System Design* vol. 10, no. 2 (1997), pp. 149–169.

[11] Groote, J. F. and Keiren, J. J. A. "Tutorial: Designing Distributed Software in mCRL2". In: *FORTE*. Vol. 12719. Lecture Notes in Computer Science. Springer, 2021, pp. 226–243.

[12] Groote, J. F. et al. "Modelling and Analysing Software in mCRL2". In: *FACS*. Vol. 12018. Lecture Notes in Computer Science. Springer, 2019, pp. 25–48.

[13] Groote, J. F. et al. “The Formal Specification Language mCRL2”. In: *Methods for Modelling Software Systems (MMOSS), 27.08. - 01.09.2006*. Ed. by Brinksma, E. et al. Vol. 06351. Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2006.

[14] Hathaliya, J. J. and Tanwar, S. “An exhaustive survey on security and privacy issues in Healthcare 4.0”. In: *Computer Communications* vol. 153 (2020), pp. 311–335.

[15] Hoare, C. A. R. “Communicating Sequential Processes”. In: *Commun. ACM* vol. 21, no. 8 (1978), pp. 666–677.

[16] Hu, H., Ahn, G., and Kulkarni, K. “Anomaly discovery and resolution in web access control policies”. In: *16th ACM Symposium on Access Control Models and Technologies, SACMAT 2011, Innsbruck, Austria, June 15-17, 2011, Proceedings*. ACM, 2011, pp. 165–174.

[17] Hu, V. C. et al. “Guide to Attribute Based Access Control (ABAC) Definition and Considerations”. In: *NIST Special Publication (SP)* vol. 800, no. 162 (2014), pp. 1–47.

[18] Al-Issa, Y., Ottom, M. A., and Tamrawi, A. “eHealth Cloud Security Challenges: A Survey”. In: *Journal of Healthcare Engineering* vol. 2019 (2019), pp. 1–15.

[19] Kolovski, V., Hendler, J. A., and Parsia, B. “Analyzing web access control policies”. In: *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*. ACM, 2007, pp. 677–686.

[20] Lifschitz, V. “What Is Answer Set Programming?” In: *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*. AAAI Press, 2008, pp. 1594–1597.

[21] Lin, D. et al. “EXAM: a comprehensive environment for the analysis of access control policies”. In: *Int. J. Inf. Sec.* Vol. 9, no. 4 (2010), pp. 253–273.

[22] Marek, V. W. and Truszczynski, M. “Stable Models and an Alternative Logic Programming Paradigm”. In: *The Logic Programming Paradigm - A 25-Year Perspective*. Artificial Intelligence. Springer, 1999, pp. 375–398.

[23] Morisset, C., Willemse, T. A. C., and Zannone, N. “A framework for the extended evaluation of ABAC policies”. In: *Cybersecur.* Vol. 2, no. 1 (2019), p. 6.

[24] Parducci, B., Lockhart, H., and Rissanen, E. “Extensible access control markup language (XACML) version 3.0”. In: *OASIS Standard* vol. 2013, no. 1 (2013), pp. 1–154.

[25] Ramli, C. D. P. K., Nielson, H. R., and Nielson, F. “The logic of XACML”. In: *Sci. Comput. Program.* Vol. 83 (2014), pp. 80–105.

[26] Rao, P. et al. “An algebra for fine-grained integration of XACML policies”. In: *14th ACM Symposium on Access Control Models and Technologies, SACMAT 2009, Stresa, Italy, June 3-5, 2009, Proceedings*. ACM, 2009, pp. 63–72.

[27] Ray, I. et al. “Applying attribute based access control for privacy preserving health data disclosure”. In: *IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*. Las Vegas, NV, USA: IEEE, 2016, pp. 1–4.

[28] Turkmen, F. et al. “Formal analysis of XACML policies using SMT”. In: *Comput. Secur.* Vol. 66 (2017), pp. 185–203.