IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 23 October 2022, accepted 28 December 2022, date of publication 25 January 2023, date of current version 2 February 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3240000

SURVEY

Attribute-Based Approaches for Secure Data Sharing in Industrial Contexts

ALEX CHIQUITO, ULF BODIN, AND OLOV SCHELÉN, (Member, IEEE)
Department of Computer Science, Electrical and Space Engineering, Luleå University of Technology, 97187 Luleå, Sweden
Corresponding author: Alex Chiquito (alex.chiquito@ltu.se)

This work was supported by the Arrowhead Tools Research Project 826452.

ABSTRACT The sharing of data is becoming increasingly important for the process and manufacturing industries that are using data-driven models and advanced analysis to assess production performance and make predictions, e.g., on wear and tear. In such environments, access to data needs to be accurately controlled to prevent leakage to unauthorized users while providing easy to manage policies. Data should further be shared with users outside trusted domains using encryption. Finally, means for revoking access to data are needed. This paper provides a survey on attribute-based approaches for access control to data, focusing on policy management and enforcement. We aim to identify key properties provided by attribute-based access control (ABAC) and attribute-based encryption (ABE) that can be combined and used to meet the abovementioned needs. We describe such possible combinations in the context of a proposed architecture for secure data sharing. The paper concludes by identifying knowledge gaps to provide direction to future research on attribute-based approaches for secure data sharing in industrial contexts.

INDEX TERMS IoT, access control, NGAC, fine-grained, attribute-based, encryption.

I. INTRODUCTION

The Industry 4.0 revolution relies heavily on data to generate value, innovation, new services, and optimize current processes [1]. The data origins may vary from sensor data to financial statements and even strictly confidential user or business data. Industries and businesses often join forces with other organizations to create better products, comply with regulations, or fulfill the needs of customers. Such relationships and collaborations create ecosystems in which data are often the strategic resource. Hence, data need to be shared among organizations in a data-driven ecosystem for it to be used as a strategic resource for creating desired values, innovations, or process improvements [2].

With data as a strategic resource in ecosystems where collaboration is key for value generation, a conflict appears. While there is a strong need for efficient sharing of data, there is also a need to protect the same, considering its intrinsic value. Furthermore, some data may be inherently more valuable than other data due to its importance in a given process or the insights on a process that it may provide. There may also be data that should be publicly accessible and other data that are imperative to keep private, e.g., to protect business-critical data or personal data that must be protected according to the EU's General Data Protection Regulation (GDPR) [3].

Currently, in addition to storing industrial data in corporate data centers, as part of the efforts to enable data-driven ecosystems and driven by needs such as high availability and extendable capacity, third-party cloud servers and services such as Microsoft's Azure [4] and Amazon's S3 [5] have gained popularity. This allows for enhanced collaboration, agility, and scalability while reducing large upfront investment and the cost of owning and maintaining infrastructure. However, the use of third-party cloud servers raises its own security concerns, as data are now outside the owner's control. This concern acts as a deterrent to many potential adopters, such as industries or organizations. Thus, ensuring the privacy, confidentiality, and integrity of the outsourced data is crucial. [6], [7], [8]. Moreover, concerns related to legal and organizational requirements may need to be considered [9].

The associate editor coordinating the review of this manuscript and approving it for publication was Mansoor Ahmed.

Different levels of data protection are needed, as enforcing very strict policies regarding public data sharing may unnecessarily slow down the data sharing process, while too little enforcement may cause leakages of sensitive data and harm the business [2]. The protection of data based on policies can be achieved using role-based access control (RBAC) [10] or the more recent concept of attribute-based access control (ABAC) [11], [12]. ABAC assembles policy, subject attributes, and object attributes to determine and enforce a set of allowable operations by the subject upon the object [13]. Furthermore, environmental conditions can be used to enforce the allowable operations [14].

Access control strategies based on RBAC or ABAC can control the flow of data while it is being shared inside a trusted domain, where the access control policies can be enforced without any sensitive information leaving the secure domain. Third-party data storage service providers may, however, not be fully trustworthy; hence, those access control strategies may not be sufficient. In addition, data may leak during transfer between source or storage and the user outside the trusted domain. Thus, access control alone cannot prevent data leakage in untrusted domains and in transit over public networks.

A classic approach when dealing with incompletely trusted cloud environments/third-party service providers is to encrypt the data. Common encryption approaches such as public key infrastructure (PKI) can be used [15]. One such approach is Pretty Good Privacy (PGP). PGP uses symmetric key and public key encryption mechanisms to guarantee data privacy while sending and receiving data, and it is widely used in tasks such as texts, emails, and file sharing applications. The general process when preparing data to be sent is to encrypt the message with a random key, called the session key, to create a ciphertext. The ciphertext is then packaged alongside the session key encrypted with the receiver's public key. Finally, the package is sent to the final user, which uses its own private key to obtain the session key to finally obtain the underlying message [16]. This type of approach is, however, costly at runtime, as the encryption task is not performed until the transmission step. Alternatively, multiple copies of the encrypted data could be stored with different public keys for different users. This, however, increases the storage overhead on the cloud, especially when the number of users is large.

Runtime cost and storage issues of per-end-user encryption are addressed by attribute-based encryption (ABE). ABE allows for encryption with attributes, so a data owner needs only to predefine enough attributes to support the desired access granularity and thereby does not need to care about the number of users in the system. Moreover, the encryption can be decoupled from the transmission step by allowing the data owner to initially encrypt the data without complete information of the end users other than their attributes, i.e., their public keys [17]. The attributes used by ABE are similar to those of ABAC, i.e., ABE solutions fit into the ABAC concept naturally due to their attribute nature [14].

Industries typically store data in both corporate data centers and on cloud platforms. Consequently, both access control and encryption are needed to properly protect the data. This raises the question of how these complementary mechanisms can be combined. ABE-enabled ABAC has been explored from the perspective of means of using ABE to realize ABAC [14]. However, the combination of ABE and ABAC, where the latter is used without ABE in trusted domains, needs further investigation. ABE-enabled access control models have, however, been studied [18], [19], [20], [21], [22].

In this paper, we investigate the use of attribute-based approaches for access control to data in trusted domains using ABAC and how such control concepts can be extended with ABE for mobile enforcement of access policies in untrusted domains. We identify key properties for secure data sharing in untrusted domains and survey ABAC and ABE to establish their respective rationales and state-of-the-art (SoTA) in the context of these properties. We pay specific attention to architectural aspects, concepts and principles of use to explore possible ways to combine these attribute-based approaches into a coherent system for secure data sharing that supports both trusted and untrusted domains. We further explore the SoTA methods of combining ABE and access control paradigms, which, as mentioned above, mainly include ABE-enabled access control models over RBAC models.

Our main contributions are as follows:

- The use of SoTA for ABAC, ABE and their combination for secure data sharing.
- Proposal of an architecture combining ABAC and ABE for such use.
- Research gaps related to this combination.

The proposed architecture makes use of general ABAC components to enable encryption and decryption using ABE according to attributes defined in the same way as those used for access control in trusted domains. The most important research gap we have identified is how to translate ABAC policies to ABE access structures. More gaps exist, such as how to efficiently extract ABAC attributes for use in the complementary ABE process.

The paper is structured as follows: in Section II, we discuss the general needs for securing data sharing in trusted and untrusted domains and identify the key properties desired from a system supporting such sharing; in Section III and Section IV, we describe the rationale of ABAC and ABE and identify their respective SoTA; Section V covers SoTA on ABE-enabled access control and presents our combined architecture; our findings are discussed in Section VI, and we conclude the paper in Section VII.

II. KEY PROPERTIES OF SECURE DATA SHARING

Access control mechanisms such as RBAC or ABAC are effective in protecting data from unauthorized access in a trusted and controlled domain. Nevertheless, cloud-based services are untrusted or open to honest-but-curious attacks, making traditional access control approaches insufficient to

ensure data privacy due to the lack of full data security control for the data owners [7], [23].

Cryptographic techniques such as ABE have proven to be a good solution when dealing with outsourced data in a variety of applications. ABE applies attribute-based policies to create ciphertexts from the stored data that only users with matching attributes can decrypt. However, the use of ABE mechanisms to protect industrial data has some challenges. In particular, the creation and management of policies is an important issue with ABE, as it does not use any standard policy language such as those supported by the ABAC standards [24]. This motivates us to investigate how to combine ABE with ABAC. For this investigation, we identify a set of key properties that neither ABE nor ABAC can efficiently cover on their own. We select properties that are important for a secure data sharing scheme relying on outsourced storage and cloud computing for industrial data. In Sections III and IV, we use these properties to investigate the state-of-the-art (SoTA) for ABAC and ABE. We further use the properties to evaluate the combined scheme for ABAC-enabled ABE in Section 4.

Key properties for secure data sharing:

- **Fine-granularity**: The level of granularity on which the access control policies are applied can range from specific files to tables inside a database or measures from specific sensors. The ability to efficiently manage and enforce fine-grained policies, e.g., on specific values in time-series data, is an important property. The granularity of the access policies depends on several other properties such as the expressiveness of the policy language or the ease of attribute management, which are discussed next.
- **Expressiveness**: This property is determined by how simple expressing policies for a policy language is. Some policy languages may provide useful abstraction, whereas some languages may require fully defined elements or the definition of additional elements. For this property, it is important to note that ability of two policy languages to express the same type of policies does not make them necessarily equally expressive. If policy language A requires more effort than policy language B to express the same policy, we consider B to be more expressive than A.
- **Dynamics**: Industrial IoT environments are dynamic. New sensors and users are entering and leaving the network constantly. Access control schemes may or may not be designed to create or update access policies after system initialization. In addition, with the revocation of access, ciphertext re-encryption or a system restart may be needed.
- **Ease of management**: Burdensome administrative overhead is one of the key deterrents of implementing fine-grained access control. The creation of this type of policy may create a large number of attributes that are difficult to efficiently manage. Some tools, such as hierarchical attribute management or automated attribute assignments, may help.
- **Access revocation**: In industrial data sharing applications, events such as the end of collaboration with some partner require efficient access revocation capabilities. Depending on the chosen approach, access revocation may require re-encryption of ciphertexts, keying material redistribution, or the participation of a central authority.
- **Implementation/Deployment**: Industries often already have some type of access control scheme implemented, even if the scheme is not efficient or fully secure. The extensive initial implementation work of an attribute-based approach is one of the reasons why these approaches have not been positioned as industry standards. Solutions such as tools to migrate from already existing access control schemes or even compatibility with existing standards are then needed.

III. ATTRIBUTE-BASED ACCESS CONTROL APPROACHES FOR SECURE DATA SHARING

A. BACKGROUND

1) MAC

In mandatory access control (MAC), the system administrator labels data with confidentiality levels and labels users with clearance levels. The relation between the data labels and the user level, as well as every operation related to them, is strictly managed by the security policy administrator. This means that in a pure MAC system, individual data owners have no direct control of the policies regarding their shared data. However, MAC policies are useful to set organization-wide policies for organizations such as governments or the military. Combining MAC policies with other approaches, such as discretionary access control (DAC), helps to enforce strict sensitive data privacy policies while enabling the sharing of information between organizations [25].

2) DAC

Proposed in the early 1970s, discretionary access control is defined as a means to restrict access to resources based on the identity of the user. In DAC, the owner of a resource typically has control over that resource. The access control is based on the Lampson access matrix model, where resources are set as columns and subjects as rows. The entry in the (subject, resource) in the matrix represents the access privileges that the subject has over the resource. Access control is considered discretionary because the owner or subjects with the correct access rights are able to pass those permissions to any other user under its discretion [26].

3) RBAC

An approach that can enforce both MAC and DAC policies is role-based access control (RBAC). RBAC is founded on the idea that in many organizations, access control decisions are made based on the roles of the users requesting data. Roles are closely related to job functions within an organization,

and they work as a group where users share a set of access permissions to resources. From a management point of view, the idea of assigning access rights to roles rather than to individual users makes sense, as the responsibilities of a role are more stable than those of a user. In contrast, the membership of users in roles tends to change over time. Hence, RBAC allows for simpler security administration, simplifying the process of granting and revoking access rights.

Moreover, RBAC introduces the concept of role hierarchy as a mechanism to align roles with the authority and responsibility level within an organizational chart. Roles at the top of an organizational chart (seniors) tend to be more powerful than roles at the bottom of the chart (juniors). In practice, the top-level roles should inherit the access rights from those under it, in addition to any other permissions assigned to it [10].

While RBAC is useful and simple to maintain in static and small organizations, managing an RBAC scheme within large heterogeneous organizations can be a burdensome task. Role explosion issues can occur when the number of users and roles becomes unmanageable. This is common in large organizations with many specialized employees performing a number of specific tasks. Due to these scalability issues, research has been conducted to develop more flexible and manageable access control schemes to achieve fine granularity in large heterogeneous organizations [27].

B. KEY CONCEPTS AND MECHANISMS

Attribute-based access control (ABAC) is an access control model where policies over subjects performing operations on objects are defined based on assigned attributes. It serves as an alternative to traditional access control models such as discretionary access control (DAC), mandatory access control (MAC), and role-based access control (RBAC). Rather than using the user identity as the base for access control, ABAC uses attributes to construct access policies. These attributes can be in one of the following five categories:

- User attributes describe characteristics of subjects within the system. These can be features such as the job title, name, age, clearance level, etc.
- Object attributes describe the features of resources within the systems. These can include the file type, format, author, security level, etc.
- Environmental attributes describe the current state of the system, including features such as the day of the week, time of the day, number of users, etc.
- Connection attributes describe the current session of the user, such as the location, IP address, session start date and time, etc.
- Administrative attributes describe the configuration properties of the system, such as the maximum session time or minimum trust level required for access.

At its core, ABAC access policies limit user access to resources or objects based on the result of Boolean operations between attributes. The most common elements found in ABAC systems are **Users (U)**, which represent the set of users accessing the system, **Objects (O)**, which are the set of protected objects in the system, **Attributes (A)**, as described previously, **Permissions (PERM)**, which represent the set of possible actions to perform over objects that can be authorized to users, and finally **Policies (P)**, which are the set of policies governing access to the system [28].

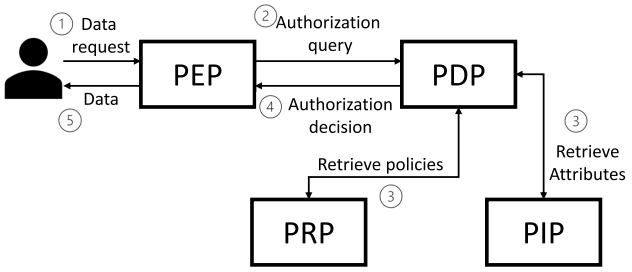


FIGURE 1. Basic ABAC architecture.

The ABAC model connects all these elements via relations; in particular, user attribute assignments (UAAs) assign attributes to users, object attribute assignments (OOAs) assign attributes to resources, and finally, policy permission relations (PRRs) connect policies with the granted access rights. The implementation of these relations and assignments varies based on policy languages, especially the way permissions are assigned to policies.

A typical ABAC architecture, such as the one presented in Figure 1, usually consists of at least one policy enforcement point receiving user data requests. This PEP forwards the request to a policy decision point (PDP), as illustrated in Figure 2, which retrieves relevant policies from the policy retrieval point (PRP) and relevant attributes from the policy information point (PIP). The PDP then evaluates the policies and attributes and generates an authorization decision that is sent back to the PEP as an answer to the original request. Only if the authorization decision is granted does the PEP return the requested data to the user.

Additional parameters can be integrated into the authorization decision process, such as environmental or conditional attributes. Furthermore, different PEPs enforcing policies in different places may produce different authorization decisions given these types of parameters [9]. Industrial settings may take advantage of these features to segregate specific types of data given the nature of the operations performed in any given location.

C. STANDARD POLICY LANGUAGES

For an access control system to be able to enforce policies, the policies and attributes must first be defined. The access control policy language is then a critical part of ABAC models, allowing tasks such as the definition of access control policies, user attribute assignments, or organization and management of the protected objects. ABAC models often use custom-made policy languages suited for specific use cases. However, there exist standardized policy languages and ways to express and enforce access policies in support of diverse data services.

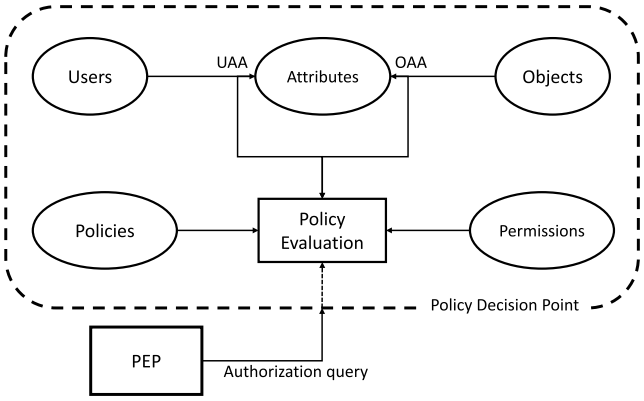


FIGURE 2. ABAC policy decision point internal flow.

One of the most frequently used standard policy languages is the eXtensible Access Control Markup Language (XACML) proposed in 2002 by the Organization for the Advancement of Structured Information Standards (OASIS) [29]. XACML is an XML-based tool designed to express access control policies and to provide mechanisms for querying the policy system and obtaining authorization decision responses. XACML authorization decision queries include user attributes, object attributes, action attributes, and environmental attributes. These attributes are defined as name-value pairs, and they represent characteristics or properties associated with the user, object, action, or environment. While user and object attributes are created by the system administrator and stored in the PIP, environmental attributes depend on the context surrounding each authorization request. These environmental attributes may include characteristics such as the time of the day, location, or date.

In XACML, access policies are defined as sets of policies (called PolicySets) composed of policies and optionally other policy sets. Moreover, the individual policies are composed of rules, which may include Boolean conditions that are dependent on attributes. The result of the condition evaluation inside a given rule influences the policy authorization decision. Furthermore, each PolicySet has a target, indicating the conditions under which the contained policies apply. When computing authorization decisions for a given request, the PDP only evaluates the PolicySets whose target is satisfied by the request attributes. The XACML XML-based language provides flexible and expressive access policy creation. However, defining policies in heterogeneous environments usually results in lengthy textual documents, making policy creation and maintenance a complex task even for simple rules [30].

Another emerging standard ABAC policy framework is next generation access control (NGAC), proposed by the National Institute of Standards and Technology (NIST) [31]. NGAC is a flexible relation-based standard designed to express and enforce ABAC policies. NGAC policies are defined as relations between attributes in a graph. Those attributes can either be user attributes (*ua*) denoting users and their properties and characteristics or object attributes (*oa*) denoting the resources needing protection. Furthermore, attributes can play the role of containers, which are used to group users, objects, and other attributes within a policy. To express policies, NGAC uses the following four different types of relations [30]:

- Assignments denote containment of one attribute in another.
- Associations link user attributes to object attributes with a set of authorized access rights.
- Prohibitions explicitly deny users, attributes, or processes from performing a specific set of actions.
- Obligations control dynamic access control policies based on events and environmental contexts.

Obligations provide an additional level of flexibility and expressiveness that would not be possible to achieve using only associations within a graph. Obligations are defined as a pair of event patterns *ep* and responses *r*. The event pattern contains the conditions that must be matched when an operation is performed over a specified object. If those conditions are matched, then the associated response is triggered. This response contains a set of administrative operations that must be performed after the event occurs. Such administrative operations include actions such as creating prohibitions or new associations. In this way, obligations can be used to prevent sensitive data leakage as well as to enable history-based policies [30].

Authorization decision requests are composed of a triple with elements *(u, op, o)*, querying whether user *u* is authorized to perform operation *op* over object *o*. The NGAC decision algorithm considers the associated privileges and prohibitions that apply to the requested user, with prohibitions taking precedence. For the user to be authorized, there must exist a path containing an association specifying the operation *op* between the user *u* and the object *o* in the policy graph [32]. The graph-based policies expressed in NGAC make it a very flexible and abstract policy language that is suitable for complex environments. Hence, the management, creation, and administration of policies are simpler tasks with NGAC than with XACML [30].

D. SoTA

Tien, et al. [33] proposed an extension to ABAC XACML called eXACML that is designed for flexible and secure data sharing on the cloud. This approach extends XACML to support obligations. This enables the expression and enforcement of value constraints, aggregation, or windowed frame data queries, improving the flexibility of possible access control policies. The proposed architecture contains a proxy server between the users and the cloud server. This proxy server processes the request from the user and enforces the access policies. However, this approach considers a trusted cloud environment, making the approach infeasible for use in public clouds without cryptographical or other privacy mechanisms.

Joshi et al. [34] presented a semantically rich access control scheme designed to protect critical organizational documents stored in the cloud. This approach applies notions of edge computing to enforce access control policies inside the

organization's trusted domain. Furthermore, the authors apply the oblivious RAM encryption mechanism inside the trusted domain before storing data in the cloud. This mechanism allows for the identity of the user accessing data to remain obscure to the third-party cloud service provider, as well as protection of the content of the documents.

The ''access broker'' who manages the document access is based on a complex ontology describing document attributes. This ABAC access broker expresses its access policies using semantic web rule language (SWRL). For the process of accessing data, the user submits a request to the access broker, which then evaluates the policies against the user's and accessed document's attributes. If the authorization decision from the access broker is positive, the file exchange and encryption module retrieves the requested documents from the cloud server and decrypts them for the user to read.

With Joshi's scheme, the request, decision evaluation, document retrieval, encryption, and decryption processes all occur inside the organization's trusted domain. This approach exploits the flexibility of its complex ontology schema to express and enforce complex policies using contextual and environmental attributes. However, the possible access rights enforced by the access broker are limited to reading and writing. Furthermore, this scheme has limitations on collaborative work, where only one user can write on a given document at a time.

Kanwal et al. [35] proposed a privacy-aware relationship semantics-based XACML access control model designed to protect electronic health records (EHRs) in a hybrid cloud called PRSX-AC. This model is an extension of the standard XACML-ABAC with a semantic relationship-based access control (rel-BAC) hybrid approach. In rel-BAC, access rights are represented as relations between users and objects.

XACML-ABAC combined with rel-BAC allows for fine-grained access control and flexible policy creation for EHR applications in hybrid clouds. Depending on the user level, the model uses the ABAC policies and attributes to evaluate access decisions and queries the relationship-reasoner component to interpret the semantic meaning of the patient relationships if needed. Moreover, this approach uses Anatomy [36] to achieve data privacy when publishing EHR records to the cloud. This process transforms the EHR records into two tables, one called the quasi-attribute table (QAT) and a second called the sensitive-attribute table (SAT), separating the quasi-identifier (QI) values from the sensitive values. These processes disassociate the relationship between the two tables, protecting the privacy of EHR data by not allowing the rebuilding of individual records.

IV. ATTRIBUTE-BASED ENCRYPTION

A. BACKGROUND

ABE is a public key encryption mechanism designed with group decryption goals rather than single users. This mechanism evolved from the older identity-based encryption (IBE) schemes, where the public key is usually a string that uniquely identifies one user, such as its social security number or email address. IBE schemes allow owners to send secret messages to other users without knowing their public keys; instead, the owner encrypts the message using the receiver's identity. Only users whose identity matches that of the ciphertext can access the underlying message [14].

The concept of attribute-based encryption was first introduced by Sahai et al. [37] as part of a fuzzy identity-based encryption approach. In this scheme, the identity is described by a set of attributes that are used to encrypt a message. Only users whose assigned attributes overlap with those used to create the ciphertext can decrypt it and obtain the message. The attributes overlap if the sets match by a predefined threshold k, which means that the user must have at least k attributes from the ciphertext attribute set to be able to decrypt the said ciphertext.

The usage of attributes when defining keying material while implementing ABE in an industrial data sharing contexts reduces the computational and administrative cost related to encryption, compared to traditional encryption schemes. ABE allows to encrypt the data once and send it to multiple different recipients instead of encrypting every time a message is sent. Moreover, creating the keying material in terms of attributes allows the creation of ciphertexts without the need to fully define the final users, simplifying the setup process [17].

B. KEY CONCEPTS AND MECHANISMS

In this section, we describe the basic components of ABE.

1) ACCESS STRUCTURE AND ACCESS TREES

In general, in a secret sharing scheme, the owner of a secret wants to share it with a limited collection of parties. This collection is called the access structure. In an encryption scheme, the access structure represents the set of parties that are able to decrypt a message, while any party that does not belong to the access structure cannot reveal any of the information about the shared secret. In particular, for attribute-based encryption, the access structure contains the set of authorized attributes.

Access structures can be represented as access trees, where each nonleaf node represents a threshold gate, and each leaf node represents an attribute. In the nonleaf nodes, threshold gates can be represented with logical OR gates if the threshold is 1 (1-of-n) and logical AND gates if the threshold is equal to the number of child nodes connected to the gate (n-of-n).

Monotone access structures are the most commonly used access structures in ABE schemes. For an access structure to be monotone, any set containing the set of authorized attributes can satisfy the access structure. In other words, if a subset of attributes B can reconstruct the secret, then every superset of B can also reconstruct the secret [38]. One consequence of using monotone access structures is that negative attributes cannot be efficiently used when constructing the access tree; in other words, monotone access structures do not support logical NOT gates. However, it is still possible to inefficiently express negative attributes in monotone access

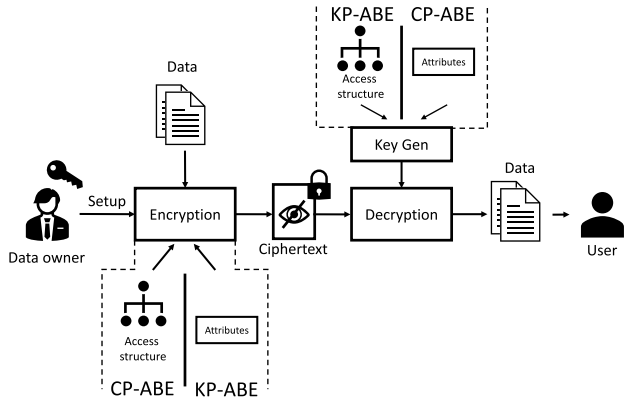


FIGURE 3. ABE general model.

structures using the not-attribute as a completely separate attribute in the tree [39].

2) BASIC CONSTRUCTION ALGORITHMS

Attribute-based encryption schemes follow a basic construction structure based on the following 4 primary algorithms: setup, encryption, key generation, and decryption.

- **Setup**: The algorithm that takes any implicit security parameter and creates public parameters and a master key. In this step, the universe of attributes is defined.
- **Encryption**: The algorithm that applies encryption to a message.
- **Key generation**: The algorithm that generates decryption keys based on a set of attributes.
- **Decryption**: The algorithm that decrypts a ciphertext to obtain the underlying message.

The ABE scheme in use determines where the user attributes and the access structure are associated. This is the main difference between the following two main ABE schemes: key-policy ABE (KP-ABE) and ciphertext-policy ABE (CP-ABE). In KP-ABE, the ciphertexts are labeled with a set of attributes, while the user's private key is the one associated with the access policy. Hence, in this scheme, the access structure that controls whether a user is allowed to decrypt a ciphertext or not is handled in the key generation algorithm. On the other hand, CP-ABE associates the user's private key with a set of attributes, while the ciphertexts are encrypted by an access policy. In CP-ABE, the encryption algorithm handles the access structure, creating a ciphertext in which only those who possess a specific set of attributes that satisfies this access structure will be able to decrypt.

C. SotA

Wang et al. [40] presented an efficient encryption scheme called file hierarchy CP-ABE (FH-CP-ABE) that is based on the hierarchical structure of access policies. The proposed layered model of the access structure aims to solve the problem of multiple hierarchical file sharing by encrypting all the files with a single integrated access structure. With k number of files, M1 being the highest hierarchy and mk the lowest hierarchy, if a user can decrypt M1, the user can also decrypt m2, m3,. . .,mk.

For the encryption process, the data owner chooses a number of content keys and encrypts the files using symmetric encryption. Then, the data owner encrypts the content keys using FH-CP-ABE encryption, obtaining an integrated ciphertext. For the decryption process, the user decrypts the integrated ciphertext and then, using the FH-CP-ABE decryption operation, obtains the content keys. Depending on the user attributes, a number of content keys will be decrypted, allowing the authorized files to be obtained using symmetric decryption. The access structures used in this implementation follow a three-level form. In this access structure, nonleaf nodes are threshold gates, and attributes reside in the leaf nodes [40].

Deng et al. [41] presented ciphertext-policy hierarchical attribute-based encryption CP-HABE, an encryption scheme where users with higher level attributes can delegate their access to users at a lower level. This scheme is designed to enable sharing of encrypted data efficiently without leaking sensitive information about the organization attribute model or hierarchy. This key delegation model allows for data to be shared with a greatly reduced communication and management overhead. This scheme is based on an attribute matrix, where higher-level attributes are on the top, while lower attributes in the hierarchy are at lower levels, creating the notion of attribute vectors. In CP-HABE, access policies are created with the attribute vectors that the users should possess to decrypt the data. This approach greatly reduces the management associated with key distribution and is especially useful for use cases where keys cannot be generated for all users.

Huang et al. [23] described a secure data collaboration scheme for reading and writing cloud-stored data using ABE and attribute-based signatures (ABSs). Furthermore, the authors apply a HABE-based mechanism to relieve the key management task of the attribute authority. The ABS is used to enable the cloud server provider to authenticate and authorize users for writing operations over stored data, evaluating the signature's attributes over the access policies. This scheme uses a full delegation mechanism, with a central authority for top-level attribute domains and a number of independent domain authorities. Finally, the proposed approach uses partial decryption to delegate most of the encryption work to the cloud server, which is suitable for resource-constrained devices.

Li et al. [42] presented a hierarchical ABE scheme with continuous leakage resilience. This resilience is achieved by using a key update algorithm to rerandomize the keys in what the author presents as continuous leakage-resilience hierarchical attribute-based encryption (CLR-HABE). This scheme uses the concept of attribute vectors to achieve hierarchical representation of attributes. It is important to note that the attribute hierarchy only applies to the parties or users.

Bobba et al. [43] proposed a scheme designed to improve the attribute flexibility of CP-ABE called ciphertext-policy attribute set based encryption (CP-ASBE). This scheme organizes user attributes into recursive set-based structures. These structures allow CP-ASBE to support compound attributes without loss of flexibility over the single attributes. Furthermore, CP-ASBE applies efficient revocation mechanisms in the form of expiration time attributes for each set. This is maintained by a key server saving the state, eliminating the need for frequent key generation and distribution tasks.

Wan et al. [44] proposed a hierarchical attribute-set-based encryption (HASBE) scheme, extending the work performed in CP-ASBE [43] to support hierarchical user structures. This extension is proposed to improve scalability and flexibility while maintaining the fine-granular capabilities of CP-ASBE. When a user attempts to decrypt a given ciphertext, it may only use attributes within a set and is not able to combine attribute elements from another set unless allowed by the encryptor in the access structure.

Xu et al. [45] presented an approach for secure data sharing of sensitive information using network encoding and attribute-based encryption. Network encoding allows for intermediate nodes in a network to process incoming message streams instead of just forwarding them. This can increase the throughput of a network when multicasting.

The authors data sharing model is set as a distributed file sharing system, where files are distributed over the network with backups on other network nodes. When a file request is sent to a node, it searches for the file locally and sends the requested file if found. Otherwise, the node sends the request to the controller node, which has lists to locate the file and answer the request.

The authors apply attribute-based encryption to the messages being sent in the network. When multicasting, the starting node will prepare the files and divide them into the needed parts. Those parts are later encrypted with the ABE policy, including a header with the location of each part. After receiving all the file parts, the end node uses its private key to decrypt the segmented parts and join them together according to the parts header [45].

Encrypting data is a good method to ensure confidentiality in an uncontrolled public cloud environment, but it reduces the possibilities of some operations occurring over ciphertexts, such as searching for keywords or specific items within the data. Searchable encryption provides mechanisms to achieve secure searching of encrypted data. In a public cloud environment, searchable encryption enables users to perform keyword-based searches using query trapdoors without revealing secrets to the cloud server [46].

Jingzhang et al. [47] presented a cryptographic retrieval scheme that combines CP-ABE with searchable encryption and supports attribute updates and applied it to medical records. The model contemplates the following four entities: the data owner, user, cloud server and an authority center. The authority center is in charge of generating public attribute keys, public parameters and version attribute keys, as well as assigning private keys to data users.

The data owner encrypts the plaintext record using symmetric key encryption and then uses CP-ABE to encrypt the used symmetric key. The two ciphertexts are then uploaded into the cloud. Meanwhile, the data user provides the attributes of the record he or she wants to access as well as its secret key to compute a trapdoor. The trapdoor is used in the cloud service to search for a symmetric key ciphertext that satisfies the provided attributes in the trapdoor, allowing the medical ciphertext to be found. The data user then decrypts the symmetric key using the CP-ABE decryption algorithm and then uses the resultant symmetric key to decrypt the record ciphertext [47].

For the updating of attributes, the authority center takes the user secret master key, the current version key and a set of public attribute keys and creates a new version key, update key and user secret key. Then, using the new user secret key, it creates a new ciphertext using proxy re-encryption to reduce the calculation burden [47].

Sun et al. [48] presented a scheme based on an attribute-based keyword search with efficient user revocation (ABKS-UR) to enable scalable fine-grained searching of encrypted data in a multiuser multicontributor environment.

This approach uses CP-ABE to enable the generation of trapdoors without needing an always online trusted authority (TA). The cloud server can search encrypted indices and return only matching results to those of the users attributes associated with the trapdoor. The data owners create indices consisting of keywords in a file before outsourcing them to the cloud server using an access structure with the attributes of authorized users. The approach uses proxy re-encryption and lazy re-encryption to delegate as much work as possible to the cloud server (CS), similar to efficient user revocation.

The outsourced data can be encrypted using any secure encryption technique. When searching for data, the user generates a trapdoor of keywords of interest with his private key. The CS searches the authorized datasets and returns results that match the users attributes on the trapdoor with those of the access structure set in the secure indices. The cloud server also keeps a list of authorized users per dataset, acting as a first filter for users trying to perform keyword searches of datasets. To revoke users from the system, the secure indices are re-encrypted, and the secret keys of the remaining authorized users are updated. This process is outsourced to the CS using proxy re-encryption.

Proxy re-encryption is a cryptographic primitive that allows for the delegation of decryption rights for a ciphertext encrypted under a user public key to another user. The ciphertext key translation is performed by a semitrusted proxy, which does not obtain information about the plaintext and secret keys. However, a re-encryption key from Alice to Bob allows Bob to decrypt all files encrypted by Alice.

Zhao et al. [49] presented an attribute-based conditional proxy re-encryption (AB-CPRE) scheme, in which the

re-encryption condition is expressed as an access structure and attribute set. In this scheme, the re-encryption is successfully performed only if the requestor attributes satisfy the re-encryption key access structure. In this proposed scheme, the authors apply KP-ABE techniques to embed the access policies into the re-encryption keys. By utilizing an access structure for the re-encryption key, the owner of the data can enforce attribute-based access control over its encrypted data.

Liang et al. [50] proposed a novel ABE system that supports keyword private search and encrypted data sharing as a searchable key-policy attribute-based proxy re-encryption (KP-ABEPR) scheme. This approach is based on an ABE system with fast decryption, extending it to support asymmetric pairing groups to protect the privacy of keywords inside a ciphertext. The keyword search is performed with the interaction between a private key generator (PKG) and a system user providing a keyword, creating a trapdoor token. This token will not enable the CS to decrypt or obtain information from the data. The authors then combine this keyword search capability with attribute-based proxy re-encryption to achieve secure data sharing. This approach guarantees that the keyword search capability will work even after re-encrypting and sharing the ciphertexts. Finally, the proposed protocol supports keyword updating before the ciphertext is shared with other users [50].

D. IDENTIFIED RESEARCH GAPS

Table 1 presents an analysis of the related work considering our identified requirements. The main gaps illustrated by this comparison are the clear difficulty of implementation and the lack of expressive and easy-to-manage access structures.

Attribute-based encryption adds a whole new level of flexibility and manageability compared to traditional schemes such as public-key or symmetric key encryption. With traditional methods, having a system with multiple users with different decryption capabilities usually requires encrypting the same data under multiple public keys for each user in the system. Hence, these methods are not suitable for industrial data sharing applications in real environments. The capability of ABE to create policies for resources and obtain access to them depending on the given attributes of a user is key for efficient data sharing.

One of the most challenging aspects of ABE is that setting up a complex and secure access control scheme necessitates significant prior setup work. The majority of ABE-based solutions require access policies to be fully defined for each individual data object in need of protection. As a result, minor differences in similar policies may necessitate significant rework when they are defined. Furthermore, ABE policies are typically not expressed in a standard policy language, complicating the transition from another access control scheme.

Furthermore, the lack of expressiveness in policy language is a significant gap. Several approaches to expressive ABE have been proposed, including the use of nonmonotonic access structures or attribute sets. However, those approaches lack the dynamic capabilities required in industrial settings.

The explored solutions share a gap regarding the ease of management, which is an important requirement in industrial settings. The lack of a centralized authority to manage policies and user attributes generates potentially large amounts of administrative work to manage, change, and revoke attributes and policies. Hierarchical approaches attempt to fill that gap by using a single hierarchical policy for different data objects. However, most of the hierarchical techniques are applied exclusively to the user side of the attributes. Applying hierarchical capabilities to the object attributes as well could improve the granularity of access policies while making them easier to manage.

Even though some approaches provide some dynamic capabilities, there is still a gap in efficiently managing dynamic policies and attributes. This challenge is reflected in the access revocation capabilities of ABE schemes. Re-encrypting data or distributing new keys are methods often used to provide access revocation capabilities to a system. However, in industrial settings, these approaches are not effective, since the production downtime caused is costly.

Covering the discussed gaps with a pure cryptographic approach is challenging due to the limited information and mechanisms that can be embedded in ciphertext and keys. Many of the surveyed approaches rely on centralized parties or authorities to add flexibility to access control and manage tasks such as access revocation. The range of tasks of the centralized party can be exploited to add even more flexibility using ABAC-inspired mechanisms to attempt to cover the identified gaps.

V. ABAC-ENABLED ABE

A. RATIONALE

ABE is a promising solution for access control in a cloud environment. However, it has some implementation issues, specifically in attribute policy construction. ABE policies are not usually designed to be created from an RBAC or ABAC base. Moreover, ABE data policy definition and expression do not follow any standard language. The methods used to build ABE policies, such as access structures, lack flexibility and expressiveness.

ABE schemes rely on monotonic access structures to define and express access control policies. These monotonic access structures do not allow for the usage of negative attributes (the usage of a NOT operator) and do not support the hierarchical relation of attributes. All these limitations hinder the expressiveness of ABE policies. Furthermore, attribute revocation and policy change management are still issues with ABE. All of these issues create a significant barrier to implementing ABE as part of industrial data sharing schemes, as they necessitate significant investment during the system's development and policy management.

Access control models such as RBAC and ABAC offer standardized expressive policy languages such as XACML or NGAC, as well as easier ways to efficiently revoke access to attributes and add new ones to existing policies. However, for

TABLE 1. Comparison of State-of-the-art ABE schemes against identified properties. *(✓): Mentioned but not addressed, ✓: Addressed but not in depth, ✓✓: explored in depth.*

Solution	Fine-grained	Expressiveness	Dynamic	Ease of management	Access revocation	Ease of implementation	Technology
[45]	–	–	–	–	–	–	Network encoding
[49]	(✓)	–	✓	–	–	–	Proxy re-encryption ABE
[47]	–	–	✓	✓	✓	–	Searchable Encryption Proxy re-encryption
[41]	–	–	–	✓✓	–	–	H-ABE
[40]	(✓)	–	–	(✓)	–	–	H-ABE
[23]	(✓)	–	✓	✓	–	–	H-ABE
[42]	–	–	–	(✓)	–	–	H-ABE
[46]	(✓)	–	–	–	–	–	Searchable Encryption
[48]	(✓)	–	✓	–	✓	–	Searchable Encryption Proxy re-encryption
[50]	–	–	–	–	–	–	Searchable ABE Proxy re-encryption
[43]	(✓)	✓✓	–	–	✓	–	Attribute-set
[44]	(✓)	✓✓	–	(✓)	✓	–	H-Attribute-set

outsourced storage applications, such as cloud environments, pure RBAC or ABAC schemes are not sufficient. These access control models rely on fully trusted domains where data are expected to pass through access control before reaching the final user. When the data are outsourced to the cloud service, the data owner loses control, and it is susceptible to curious providers or leaks.

Naturally, combining the data confidentiality capabilities of ABE with the expressiveness and management of more traditional access control models can help cover each approach's weaknesses. However, there is a risk of greatly increasing the storage overhead when combining features from access control to ABE. This is a particularly important drawback when dealing with high volumes of dynamic data, as in industrial IoT applications.

B. SOTA

The majority of related work on ABE-enabled access control models has been on RBAC. This is mainly due to the technical limitations of translating more complex policies to ABE access structures.

Zhou et al. [18] presented a role-based encryption scheme that combines RBAC with cryptographic techniques to enforce RBAC policies over encrypted data in public clouds. RBAC policies can be enforced using role hierarchies. Furthermore, this scheme allows for efficient user revocation; this revocation does not affect other users or roles, and the data do not need to be re-encrypted. The proposed RBE cloud storage architecture is based on a hybrid cloud infrastructure, where data are stored in the public cloud while a private cloud is used to store the organization's sensitive structure information. The private cloud is designed to only communicate with the system administrator and the role managers, reducing the attack surface of the cloud.

Zhu et al. [19] presented an RBAC-compatible ABE model for secure cloud storage. The paper introduces a combination of RBAC with ABE to achieve user-friendly and easy-to-manage security mechanisms to protect user data in the cloud. To achieve compatible RBAC hierarchy policies to be implemented into ABE, the author presents a new ABE scheme called attribute-based encryption with attribute lattices (ABE-AL). In ABE-AL, attribute lattices or hierarchies define seniority relations among all the values of an attribute. This means that if a user possesses the senior attribute value, it then possesses the permissions of juniors. The proposed system allows users who already use RBAC to access protected cloud resources in a transparent manner. The system requires administrators to provide a transformation from RBAC to ABE, while the user needs to deploy an RBAC to ABE module in its terminal. The author also presents cryptographic mechanisms to achieve comparison operations on attribute lattices to improve constraint expressions and reduce computational overheads.

Lang et al. [20] presented an extended CP-ABE scheme (ECP-ABE) to achieve self-contained data protection. This scheme tries to improve the expressiveness of traditional CP-ABE access structures to enable it to express any ABAC policy. Self-contained data protection is a mechanism that ensures the integrity and confidentiality of data without depending on other parties. This scheme introduces the idea of extended leaf nodes to allow CP-ABE access structures to support logical and arithmetic comparison operators. The original CP-ABE leaf node is replaced by an operator node with the following two children: an attribute name node and an attribute value node. These three new nodes compose an extended leaf node.

The usage of extended leaf nodes allows the access structure to express comparison attributes, interval operators and logical operators. The user creates access policies using extended trees that are transformed into standard access policy trees in the encryption algorithm. The original extended tree is attached in the ciphertext. To decrypt the data, the user sends his basic attribute set and the extended parts of the access tree to the key generation entity. The PKG reconstructs

the extended tree with the embedded information from the ciphertexts and evaluates the set of attributes from the user to create a decryption key if the tree is satisfied.

Lang et al. [21] described a flexible and self-contained approach for data protection using a combination of RBAC and CP-ABE. This scheme allows data to ensure its own security without depending on the storage servers. The authors describe an extension to RBAC, called data-centric RBAC (DC-RBAC), which allows for access control policies to be bounded to the data it protects. Furthermore, the authors integrate the proposed DC-RBAC with the ECP-ABE scheme proposed in [20]. ECP-ABE is extended to support role assignment and inheritance. DC-RBAC differs from RBAC in that the extension can support both positive and negative role assignments, improving expressiveness. Furthermore, the extension also presents two new constraints related to attribute-based access control in the form of user attributes and environmental constraints. User attribute constraints work with values associated with user attributes, such as age, name, security level, etc. Environmental attributes evaluate contextual information from the environment, such as access time, day of the week, or location.

To protect the data, the data owner first creates a DC-RBAC policy that is then mapped to an ECP-ABE access tree. The data access process is performed by the decryption party, and it is divided into two indivisible subprocesses, private key request and decryption. In the first process, the decryption party sends the leaf and extended leaf nodes to the PKG. The PKG then evaluates whether the user attributes and roles satisfy the extended attributes, generates a key if authorized and sends it to the user. Then, if the user attributes, roles and environmental attributes satisfy the DC-RBAC policy, the data are decrypted. This scheme achieves more flexible and fine-grained access control, while having the data itself determine which users are authorized to decrypt it without relying on other parties.

Mahmood et al. [22] presents a scheme to provide data security in the cloud that combines role-based access control (RBAC) with cryptographic techniques. In this scheme, the data are encrypted using an improved RSA encryption algorithm. This algorithm introduces a new valuation key used by third parties to execute operations on the encrypted data, while the data are encrypted and decrypted with a private key that is only known to the data owner. The proposed scheme uses XACML policy language to enable an RBAC layer. The access control layer provides guarantees that unauthorized users cannot make inappropriate changes in the system.

In this architecture, when the user sends a data request to the access manager, the data owner sends a request to a third party to perform operations over the encrypted data. The third party then handles the operation only if the user is authorized by the RBAC layer and then returns the data and the private key for the user to decrypt. This architecture relies on traditional access control infrastructure such as an access manager, policy decision (PDP) and enforcement points (PEP), and policy management points (PMP).

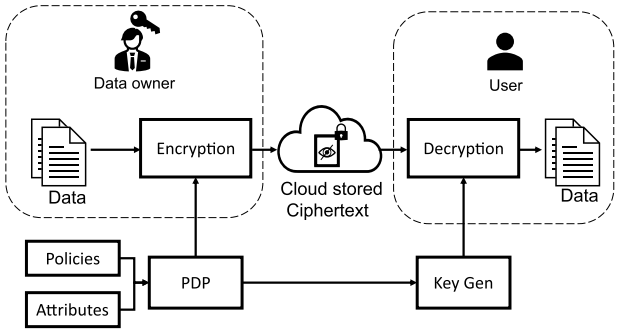


FIGURE 4. ABAC and ABE combined architecture proposal.

Most state-of-the-art encryption-enabled access control schemes are based on RBAC. This may mainly be due to the extended adoption of RBAC within organizations, industry, and operating systems. Role hierarchy is a great addition to ABE, as the capacity to reuse policies and facilitate role and policy management is important in a data sharing environment. Later proposals have made use of standard policy languages such as XACML to express policies. However, the policies are still limited to what a role-based scheme can enforce, not taking advantage of the expressiveness of the policy language. Table 2 shows the evaluation of the SotA against the identified properties. One of the main challenges for this approach is to achieve an expressive access control policy definition while having easy-to-manage policies with access revocation capabilities. Attribute-based access control paradigms can be applied as a logical next step for this type of encryption enabled access control scheme. Moreover, related work has focused mainly on role membership, hierarchy, and inheritance only for users. The usage of hierarchical groups and attributes for data can further contribute to the ease of management and work needed to setup and maintain the initial policies.

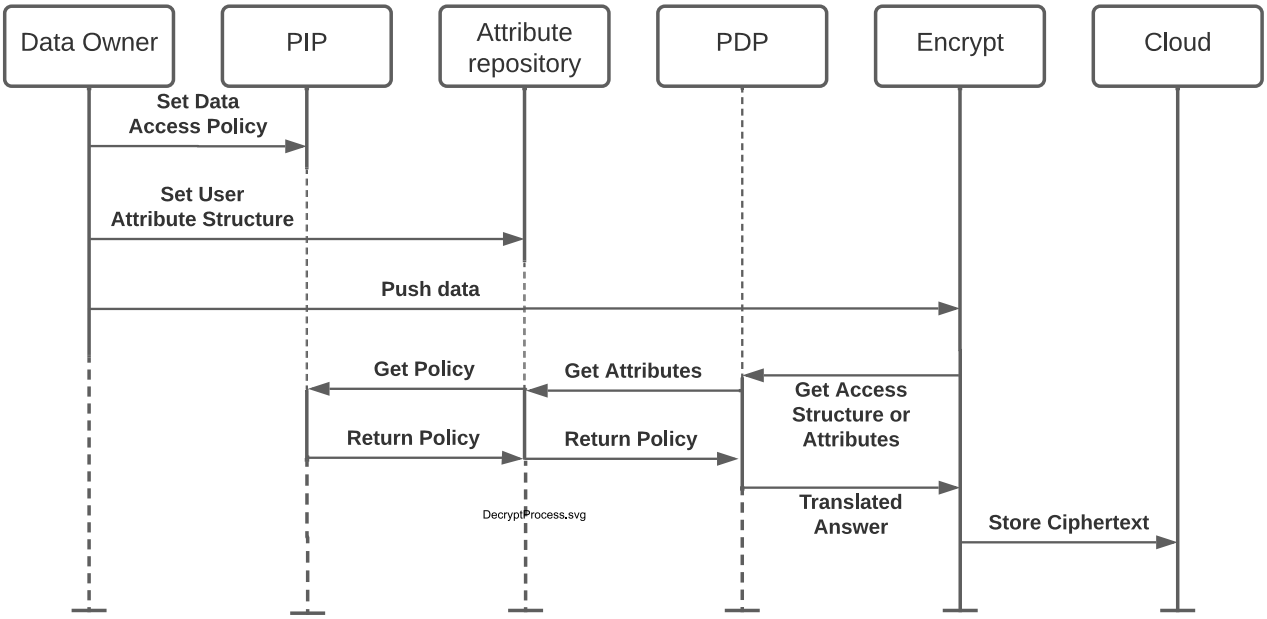
C. ARCHITECTURE

ABE has some shortcomings, particularly in regard to its practical use. Incorporating ABAC infrastructure ideas into ABE can help to solve some of its individual challenges. One of the more valuable elements in an ABAC scheme is the decision point and its communication to the policy and attribute repository. There has been the development of powerful standard policy languages for ABAC, such as NGAC or XACML. Furthermore, having a centralized policy and attribute repository makes the creation, distribution and management of policies and attributes a simpler task compared to the implications of key distribution in ABE.

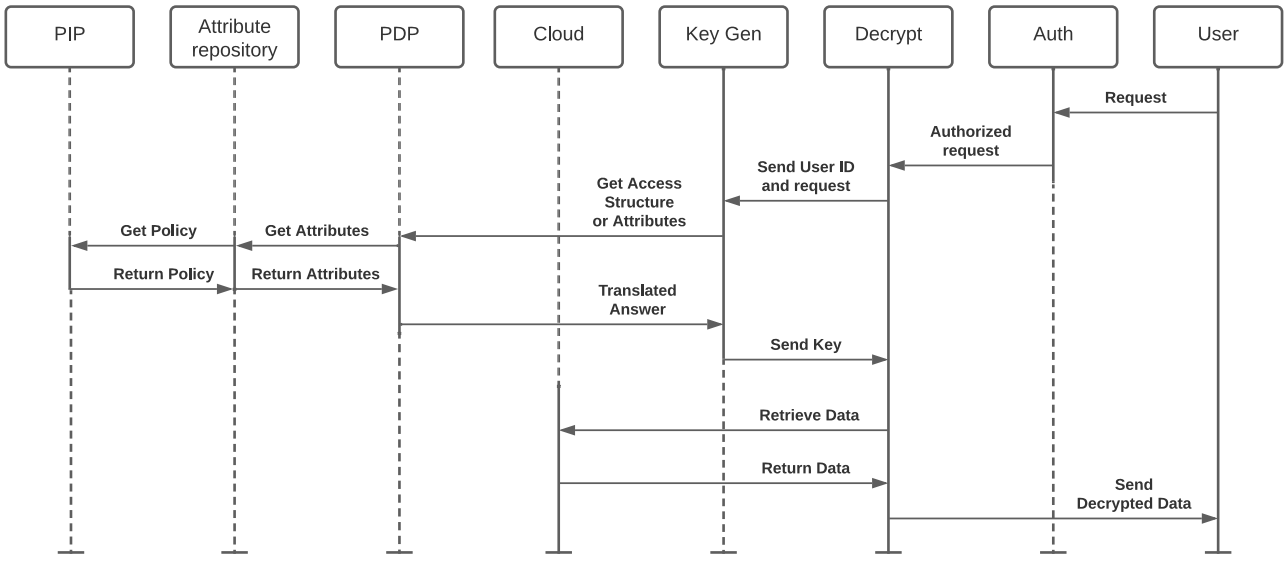
The complete encryption and decryption process is illustrated in Figure 5. The encryption process is triggered by the data owner pushing data into the encryption algorithm. The encryption algorithm then fetches the needed parameters from the PDP, depending on the ABE model used (either KP-ABE or CP-ABE), and generates the ciphertext. For the decryption process, any authorized user can send data requests to the decryption algorithm.

TABLE 2. Comparison of state-of-the-art ABE schemes against identified properties. *(✓): Mentioned but not addressed, ✓: Addressed but not in depth, ✓✓: explored in depth.*

Solution	Fine-grained	Expressiveness	Dynamic	Ease of management	Access revocation	Ease of implementation	Technique
[18]	–	–	(✓)	(✓)	✓✓	(✓)	RBAC-ABE
[19]	–	✓✓	–	✓	–	✓✓	RBAC-ABAC-ABE
[20]	–	✓✓	–	–	–	–	ABAC policies
[21]	✓✓	✓✓	–	–	–	–	ABAC policies
[22]	✓	✓✓	–	✓	–	✓✓	RBAC-ABE



(a) Encryption process



(b) Decryption process

FIGURE 5. Signaling diagram for the complete data sharing process.

The decryption algorithm will forward the request to the key generation process, which fetches the needed parameters from the PDP. Given the returned parameters, the key generation process creates the decryption key and is used to return the original data to the requester. A decryption key is generated whether the user is authorized to the resource or not. If the attributes and policies provided for the key generation does not satisfy the ciphertext requirements, the generated key will fail to retrieve the original data.

Figures 4 and 5 show the operational steps in encrypting and decrypting data in our architecture. In Figure 4 the flow of data from the data owner to the final user is shown. First, the data owner triggers the encryption process. The keying material needed for this process is generated by the PDP from the existing policies and attributes. Moreover, the decryption process is triggered whenever the final user requests access to an encrypted resource. In a similar way to how the encryption keying material is generated, the keying material for the decryption is derived from the relevant user attributes related to the target resource policies.

In this architecture, both the encryption and key generation algorithms depend directly on the output from an ABAC decision point, and we can discuss the implications. First, from the ABAC point of view, Figures 4 and 5 show that there is no need for a dedicated PEP. The PEP functions are absorbed by the decryption algorithm, where the policies and attributes can allow or reject the attempt to obtain the underlying data. Moreover, the PDP now shifts from policy evaluation to policy information. As illustrated in Figure 5, for both the encryption and decryption processes, the critical steps in the process depend directly on information provided by the PDP. Depending on the chosen ABE approach, the PDP will either summarize the access policies to extract the access structure of a specific resource or obtain the attributes related to the resource. Furthermore, the key generation algorithm is affected in a similar manner. When the user submits a data request to the system, the PDP would need to retrieve either the summarized user's access structure or obtain the attributes related to that user.

In this architecture, policy creation and attribute management are simplified through the ABAC infrastructure and standard policy language. The data owner is in charge of attribute creation, distribution, and management. This, in conjunction with an authorization system, enables the system to ensure that users have access to only its authorized attributes. Furthermore, the key distribution is improved as keys can now be generated at the request time, opening opportunities for efficient revocation models.

Figure 6 shows the process for access revocation in our architecture. At any time, the data owner or system administrator can update either the resource policy or the attribute structure. These changes are made directly to the centralized policy information point (PIP) or the attribute repository respectively. In our architecture, decryption keys are generated whenever there is an attempt to access the data. Hence, users whose access rights were revoked by any update

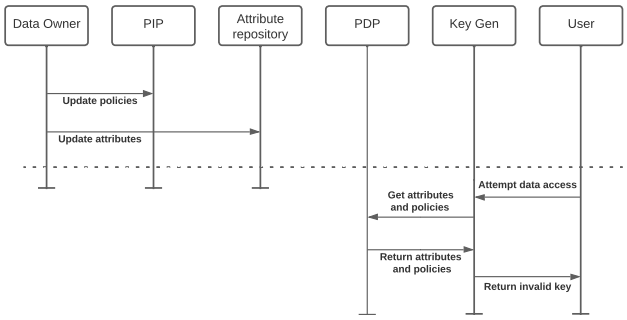


FIGURE 6. Access revocation process.

on policies or attributes will not be able to obtain a valid decryption key at the next attempt to access the resource.

Finally, even though highly expressive policies can be defined with the typically used monotonic access structures, they become so complex that it is unfeasible to expect data owners to create such policies for their resources. Standard policy languages can improve the expressiveness of common ABE access structures while lowering the complexity, making it ideal to solve the implementation complexity challenges of ABE.

D. EVALUATION

We evaluate this architecture considering the desired properties identified in Chapter 2.

- **Fine-granularity**: One of the greatest deterrents to implementing fine-grained access policies is the challenge of creating complete and sound policies for a large number of resources. In typical ABE schemes, access structures need to be fully defined for each individual resource. Consequently, defining the application of different policies to similar resources, even if they share part of their access structures, includes duplicated work, making it difficult to scale. ABAC standard policy languages such as NGAC or XACML can support the efficient creation of fine-grained access control policies thanks to their hierarchical properties. This capability can be translated into ABE with the help of modified encryption and key generation algorithms, generating the needed cryptographic parameters from information coming from a dedicated PDP.
- **Expressiveness**: One of the main technical challenges when improving the expressiveness of access structures in ABE schemes is translating the hierarchical attribute and role hierarchy into a mathematical form [51]. In our architecture, we delegate policy processing work to the ABAC PDP to generate adequate access structures for both the encryption and key generation algorithms. In this way, the architecture can take advantage of the expressiveness properties of ABAC policy languages.
- **Dynamic**: Our architecture is reactive in the sense that the decryption key generation is triggered by the request of the user. This allows for changes in the user's related policy to be enforced at the time the next request is performed. In the same way, thanks to the properties

of ABE, the inclusion of new users would not require a re-encryption of the existing ciphertexts.

- **Management**: ABAC inspired standard policy languages, and centralized policy and attribute repositories simplify the management task significantly. Furthermore, the reutilization of access policies in a hierarchical manner and the attribute container capabilities provided by ABAC simplifies the administrative work needed when updating and creating policies.
- **Access revocation**: Related to the dynamic properties, the reactive nature of the decryption key allows for efficient attribute and user revocation. Every time a user requests access to a new set of data, its attributes are evaluated against the data's policy. In the case of a user attribute being revoked from a resource, instead of re-encrypting the affected data and redistributing the keying material, the next time said user attempts to access the data, the key generation process will not be able to generate a valid key with the updated attributes.
- **Implementation/deployment**: One of ABE's major issues is its complex implementation process. Expecting data owners to fully define policies for each data object is often unfeasible in industrial contexts. Recently, industry has been adopting ABAC as their standard for access control, or at the very least, it is expected for them to have an organizational role hierarchy into which access control is modeled. Those role-based policies can easily be translated into ABAC policies and, hence, simplify this proposed model implementation.

E. IDENTIFIED RESEARCH GAPS

We identify several knowledge gaps that need to be solved to efficiently implement our architecture. The most important technical challenge is how to translate ABAC policies into ABE access structures. Work has been done to integrate role hierarchy into the cryptographic access structures for ABE [18], [19], [20], [21], [22]. Furthermore, there has been work on hierarchical-ABE to efficiently enhance the expressiveness of ABE access structures. Complex ABE access structure models exist that can express and enforce hierarchical, conditional, and logical policies common in ABAC policies [20], [21]. However, the issue then lies in how complex they are to create and manage, making it unfeasible for data owners to administer. Approaches such as [22] integrate the flexibility and expressiveness of standard policy languages such as XACML with cryptographic technologies. However, this approach supports only RBAC policies and uses an RSA encryption mechanism that relies on data owners creating and managing private keys, making it more difficult to scale in an industrial environment. Hence, efficiently translating the capabilities of standard ABAC languages such as NGAC or XACML into access structures that can be cryptographically enforced efficiently is still an open challenge for research on industrial data sharing applications to solve.

Given our architecture, the translation work of access policies can be delegated to a decision point, where the ABAC access policies can be processed and transformed into cryptographically enforceable access structures. The processing inside the trusted decision point should identify the relevant complex ABAC policy and extract one or several access structures that need to be cryptographically satisfied by the user requesting the data. The automation of this transformation would allow a system to use the existing complex access structure models, overcoming the need for large management costs. Hence, research efforts should be headed toward efficient methods to identify and transform relevant access control policies into enforceable ABE access structures while minimizing the loss of features.

The proposed architecture is designed in such a way that both CP-ABE and KP-ABE are feasible. Nevertheless, work is also needed to efficiently extract attributes from users and objects to be used in the complementary ABE process, depending on which scheme is used. This processing also takes place at a trusted decision point connected to the same policy and attribute repository. Trust mechanisms are often desired features in industrial environments, where there is often no need for a fully distributed public cloud data sharing solution. The centralization of policies and attributes allows for easier management, as well as the enabling of revocation mechanisms.

VI. DISCUSSION AND FUTURE WORK

The proposed approach is designed as a mechanism to enforce attribute-based access control of encrypted data shared on the cloud, considering the properties identified in Section II. Previous related work has researched mechanisms to cryptographically enforce RBAC policies in ABE schemes, sacrificing the flexibility and expressiveness provided by the ABAC model. Furthermore, work on integrating ABAC schemes into traditional encryption mechanisms, such as RSA encryption, has been performed. However, key management and collaboration capabilities are hindered with this approach. By integrating the access policy querying capabilities of the ABAC PDP into the key generation process of ABE, it should be possible to achieve data privacy without a significant trade-off of flexibility and expressiveness in the access control.

Existing policies and attribute schemes expressed in standard policy languages should be directly translated into the proposed architecture for the enforcement of encryption of cloud stored data. Hence, for a data owner previously using an ABAC scheme to share its data, the administrative effort needed to implement the architecture proposed in this paper should be minimal. Furthermore, the proposed architecture is designed so that the key management tasks of the data owner are almost nonexistent, as those tasks are delegated to the PDP. Instead, the revocation and policy update processes are performed with changes to the attributes without the need for re-encrypting data. The separation of the data owner from the key management tasks opens the architecture for multiple data owners to collaborate inside a single data-sharing ecosystem, each controlling the way their data are shared.

These properties translates favorably to the implementation of ABE-based solutions within industrial applications. This is especially important in the reduction of management-related workload, both for the initial setup and for the maintenance. However, there are implications regarding how the attribute structure is managed in a collaborative environment that need to be solved.

In the proposed architecture, it is important to note that the key generation and decryption processes may require additional privacy mechanisms to prevent the leakage of the generated keys. This is particularly important if those processes are hosted in untrusted domains. Should decryption keys get leaked to the final user, the revocation mechanisms described in this architecture would not be effective.

For future work, a formal definition and implementation of the interaction between the PDP and the encryption and key generation algorithms is needed. This interaction includes the generation of decryption keys for the data based on its assigned attributes and, in a similar way, the creation of decryption keys for the users given their identities and associated attributes. Furthermore, an analysis should be performed to determine the implications of using either CP-ABE or KP-ABE schemes for this architecture. In addition to qualitative advantages from the industrial data sharing environment point of view, performance implications may arise as key generation processes are constantly executing as users request data. Depending on the internal PDP process, the transformation of access policies to access structures may be faster or slower than the required summarizing of attributes needed to encrypt and generate keys. Hence, quantitative performance analysis should be performed to determine the best configuration.

VII. CONCLUSION

In this paper, we analyze industrial environments and their needs for data sharing and define the following key properties for secure data sharing in that context: 1) fine-granularity, 2) expressiveness, 3) dynamic, 4) ease of management, 4) access revocation, and 5) ease of implementation. To achieve such properties, we motivate the use of ABE-enabled ABAC, an attribute-based model taking paradigms from ABAC models and architectures, and introduce attribute-based cryptographic elements to ensure data privacy in untrusted domains. In the context of the identified desired properties, we performed a state-of-the-art exploration to identify the gaps and relevant mechanisms of each individual approach, i.e., ABAC and ABE. We found that one of the main attractions of ABAC research is mechanisms to efficiently enforce expressive and fine-grained policies over data stored in either private or public cloud servers. Moreover, we presented a comparison of SoTA approaches for data sharing using ABE and identified gaps in the expressiveness and ease of implementation of such models.

Furthermore, we designed a conceptual architecture that proposes how the relevant mechanisms should work together to address the identified research gaps. This architecture benefits from the capabilities of an ABAC policy decision point by making it a central part of the encryption process. The encryption and key generation algorithm are fed attribute and policy responses from the PDP when a data consumption or data encryption request is submitted. This allows the ABE mechanisms to be controlled and managed by a central element capable of exploiting the flexibility and expressiveness of ABAC standard policy languages. Finally, we end the paper with an analysis of the knowledge gaps in the context of our architecture and discuss the deployment implications for the data owners. The main knowledge gap for this architecture is the translation of ABAC access policies to useful parameters in the encryption and decryption algorithm of ABE.

ACKNOWLEDGMENT

The authors would like to thank Rance DeLong at The Open Group for giving feedback on the article.

REFERENCES

[1] D. Wee, R. Kelly, J. Cattel, and M. Breunig, "Industry 4.0-how to navigate digitization of the manufacturing sector," *McKinsey Company*, vol. 58, no. 58, pp. 7–11, Apr. 2015.
[2] International Data Space (IDS). (2019). *International Data Space—Reference Architecture Model*. [Online]. Available: https://www.internationaldataspaces.org/wp-content/uploads/2019/03/IDS-Reference-Architecture-Model-3.0.pdf
[3] (Feb. 2019). *What is Considered Personal Data Under the Eu GDPR?*. [Online]. Available: https://gdpr.eu/eu-gdpr-personal-data/
[4] M. Copeland, J. Soh, A. Puca, M. Manning, and D. Gollob, *Microsoft Azure*. New York, NY, USA: Apress, 2015.
[5] S. Granneman, *Amazon Web Services*, AECCU Guide, Washington Univ. St. Louis, St. Louis, MO, USA, Dec. 2012.
[6] C. Alliance, "Security guidance for critical areas of focus in cloud computing V3. 0," *Cloud Secur. Alliance*, vol. 15, no. 15, pp. 1–176, Nov. 2011.
[7] L. Wei, H. Zhu, Z. Cao, X. Dong, W. Jia, Y. Chen, and A. V. Vasilakos, "Security and privacy for storage and computation in cloud computing," *Inf. Sci.*, vol. 258, pp. 371–386, Feb. 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025513003320
[8] S. Kamara and K. Lauter, "Cryptographic cloud storage," in *Proc. Int. Conf. Financial Cryptography Data Secur.* Berlin, Germany: Springer, 2010, pp. 136–149.
[9] M. Falkenthal, F. W. Baumann, G. Grünert, S. Hudert, F. Leymann, and M. Zimmermann, "Requirements and enforcement points for policies in industrial data sharing scenarios," in *Proc. 11th Symp. Summer School Service-Oriented Comput.*, 2017, pp. 1–14.
[10] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models," *Computer*, vol. 29, no. 2, pp. 38–47, 1996.
[11] D. R. Kuhn, E. J. Coyne, and T. R. Weil, "Adding attributes to role-based access control," *IEEE Comput.*, vol. 43, no. 6, pp. 79–81, Jun. 2010.
[12] V. C. Hu, D. R. Kuhn, and D. F. Ferraiolo, "Attribute-based access control," *Computer*, vol. 48, no. 2, pp. 85–88, Feb. 2015.
[13] V. C. Hu, D. Ferraiolo, R. Kuhn, A. R. Friedman, A. J. Lang, M. M. Cogdell, A. Schnitzer, K. Sandlin, R. Miller, and K. Scarfone, "Guide to attribute based access control (ABAC) definition and considerations (draft)," *NIST Special Publication*, vol. 800, no. 162, pp. 1–54, 2013.
[14] D. Huang, Q. Dong, and Y. Zhu, *Attribute-Based Encryption and Access Control*. Boca Raton, FL, USA: CRC Press, 2020.
[15] J. Weise, "Public key infrastructure overview," *Sun BluePrints OnLine*, vol. 108, pp. 1–27, Aug. 2001.
[16] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer, "OpenPGP message format," Tech. Rep. rfc4880, 2007.
[17] C.-C. Lee, P.-S. Chung, and M.-S. Hwang, "A survey on attribute-based encryption schemes of access control in cloud environments," *Int. J. Netw. Secur.*, vol. 15, no. 4, pp. 231–240, Jul. 2013.
[18] L. Zhou, V. Varadharajan, and M. Hitchens, "Achieving secure role-based access control on encrypted data in cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 12, pp. 1947–1960, Oct. 2013.

[19] Y. Zhu, D. Huang, C.-J. Hu, and X. Wang, "From RBAC to ABAC: Constructing flexible data access control for cloud storage services," *IEEE Trans. Serv. Comput.*, vol. 8, no. 4, pp. 601–616, Jul./Aug. 2015.
[20] B. Lang, R. Xu, and Y. Duan, "Self-contained data protection scheme based on CP-ABE," in *Proc. Int. Conf. E-Business Telecommun.* Berlin, Germany: Springer, 2013, pp. 306–321.
[21] B. Lang, J. Wang, and Y. Liu, "Achieving flexible and self-contained data protection in cloud computing," *IEEE Access*, vol. 5, pp. 1510–1523, 2017.
[22] G. S. Mahmood, D. J. Huang, and B. A. Jaleel, "A secure cloud computing system by using encryption and access control model," *J. Inf. Process. Syst.*, vol. 15, no. 3, pp. 538–549, 2019.
[23] Q. Huang, Y. Yang, and M. Shen, "Secure and efficient data collaboration with hierarchical attribute-based encryption in cloud computing," *Future Gener. Comput. Syst.*, vol. 72, pp. 239–249, Jul. 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X16303521
[24] Y. Wang, L. Wei, X. Tong, X. Zhao, and M. Li, "CP-ABE based access control for cloud storage," in *Information Technology and Intelligent Transportation Systems*. Berlin, Germany: Springer, 2017, pp. 463–472.
[25] P. A. Loscocco, S. D. Smalley, P. A. Muckelbauer, R. C. Taylor, S. J. Turner, and J. F. Farrell, "The inevitability of failure: The flawed assumption of security in modern computing environments," in *Proc. 21st Nat. Inf. Syst. Secur. Conf.* Princeton, NJ, USA: Citeseer, 1998, pp. 1–12.
[26] C. S. Jordan, *Guide to Understanding Discretionary Access Control in Trusted Systems*. Collingdale, PA, USA: DIANE, 1987.
[27] A. Elliott and S. Knight, "Role explosion: Acknowledging the problem," in *Software Engineering Research and Practice*. Princeton, NJ, USA: Citeseer, 2010, pp. 349–355.
[28] D. Servos and S. L. Osborn, "Current research and open problems in attribute-based access control," *ACM Comput. Surveys*, vol. 49, no. 4, pp. 1–45, Dec. 2017.
[29] S. Godik and T. Moses, "OASIS extensible access control markup language (XACML)," in *OASIS Committee Secification Cs-XACML-Specification-1.0*, OASIS OPEN, Woburn, MA, USA, 2002.
[30] D. Ferraiolo, R. Chandramouli, D. Kuhn, and V. Hu, "Extensible access control markup language (XACML) and next generation access control (NGAC)," in *Proc. ACM Int. Workshop Attribute Based Access Control*, Mar. 2016, pp. 13–24.
[31] D. F. Ferraiolo, L. Feldman, and G. A. Witte, "Exploring the next generation of access control methodologies," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep., 2016.
[32] R. Basnet, S. Mukherjee, V. M. Pagadala, and I. Ray, "An efficient implementation of next generation access control for the mobile health cloud," in *Proc. 3rd Int. Conf. Fog Mobile Edge Comput. (FMEC)*, Apr. 2018, pp. 131–138.
[33] T. T. A. Dinh, W. Wenqiang, and A. Datta, "City on the sky: Extending XACML for flexible, secure data sharing on the cloud," *J. Grid Comput.*, vol. 10, no. 1, pp. 151–172, Mar. 2012.
[34] M. Joshi, S. Mittal, K. P. Joshi, and T. Finin, "Semantically rich, oblivious access control using ABAC for secure cloud storage," in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, Jun. 2017, pp. 142–149.
[35] T. Kanwal, A. A. Jabbar, A. Anjum, S. U. Malik, A. Khan, N. Ahmad, U. Manzoor, M. N. Shahzad, and M. A. Balubaid, "Privacy-aware relationship semantics–based XACML access control model for electronic health records in hybrid cloud," *Int. J. Distrib. Sensor Netw.*, vol. 15, no. 6, pp. 1–24, 2019.
[36] X. Xiao and Y. Tao, "Anatomy: Simple and effective privacy preservation," in *Proc. 32nd Int. Conf. Very Large Data Bases*, 2006, pp. 139–150.
[37] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Advances in Cryptology–EUROCRYPT*, R. Cramer, Ed. Berlin, Germany: Springer, 2005, pp. 457–473.
[38] A. Beimel, "Secret-sharing schemes: A survey," in *Proc. Int. Conf. Coding Cryptol.* Cham, Switzerland: Springer, 2011, pp. 11–46.
[39] R. Ostrovsky, A. Sahai, and B. Waters, "Attribute-based encryption with non-monotonic access structures," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, Oct. 2007, pp. 195–203.
[40] S. Wang, J. Zhou, J. K. Liu, J. Yu, J. Chen, and W. Xie, "An efficient file hierarchy attribute-based encryption scheme in cloud computing," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 6, pp. 1265–1277, Jun. 2016.
[41] H. Deng, Q. Wu, B. Qin, J. Domingo-Ferrer, L. Zhang, J. Liu, and W. Shi, "Ciphertext-policy hierarchical attribute-based encryption with short ciphertexts," *Inf. Sci.*, vol. 275, pp. 370–384, Aug. 2014.
[42] J. Li, Q. Yu, and Y. Zhang, "Hierarchical attribute based encryption with continuous leakage-resilience," *Inf. Sci.*, vol. 484, pp. 113–134, May 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025519300684
[43] R. Bobba, H. Khurana, and M. Prabhakaran, "Attribute-sets: A practically motivated enhancement to attribute-based encryption," in *Computer Security–ESORICS*, M. Backes and P. Ning, Eds. Berlin, Germany: Springer, 2009, pp. 587–604.
[44] Z. Wan, J. Liu, and R. H. Deng, "HASBE: A hierarchical attribute-based solution for flexible and scalable access control in cloud computing," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 2, pp. 743–754, Apr. 2012.
[45] Z. Xu, B. Shen, and Z. Zhang, "Secure sharing sensitive data based on network coding and attribute-based encryption," in *Proc. Int. Conf. Smart Grid Internet Things*. Cham, Switzerland: Springer, 2020, pp. 95–105.
[46] H. Yin, J. Zhang, Y. Xiong, L. Ou, F. Li, S. Liao, and K. Li, "CP-ABSE: A ciphertext-policy attribute-based searchable encryption scheme," *IEEE Access*, vol. 7, pp. 5682–5694, 2019.
[47] S. Jingzhang, C. Chunjie, and L. Hui, "Searchable encryption scheme based on CPABE with attribute update in a cloud medical environment," in *Int. Conf. Cloud Comput. Secur.* Cham, Switzerland: Springer, 2018, pp. 265–276.
[48] W. Sun, S. Yu, W. Lou, Y. Hou, and H. Li, "Protecting your right: Attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud," in *Proc. IEEE Conf. Comput. Commun.*, May 2014, pp. 226–234.
[49] J. Zhao, D. Feng, and Z. Zhang, "Attribute-based conditional proxy re-encryption with chosen-ciphertext security," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, Dec. 2010, pp. 1–6.
[50] K. Liang and W. Susilo, "Searchable attribute-based mechanism with efficient data sharing for secure cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 9, pp. 1981–1992, Sep. 2015.
[51] Y. Zhu, D. Ma, C.-J. Hu, and D. Huang, "How to use attribute-based encryption to implement role-based access control in the cloud," in *Proc. Int. Workshop Secur. Cloud Comput.*, May 2013, pp. 33–40.

ALEX CHIQUITO was born in Mexico, in 1994. He received the B.S. degree in mechatronics engineering from the Tecnologico de Monterrey, Mexico, in 2017. He is currently pursuing the Ph.D. degree in cyber-physical systems with the Luleå University of Technology, Sweden. Prior to his Ph.D. degree, he has worked as a Data Analyst, a Robotic Process Automation (RPA) Developer, and a Solution Architect at HP Inc. His research interests include the IoT technologies for industrial cyber-security, data sharing, and cryptographic access control.

ULF BODIN received the Ph.D. degree in computer networking from the Luleå University of Technology. He is currently a Professor at the Luleå University of Technology, where he is conducting research on industrial IoT, distributed system of systems, computer communications, distributed ledgers, and applied machine learning. He has more than 15 years of experience in academia and software industry, including standardization in ETSI and other organizations.

OLOV SCHELÉN (Member, IEEE) received the Ph.D. degree in computer networking from the Luleå University of Technology. He is currently a Professor with the Luleå University of Technology and the CEO of Xarepo AB. He has more than 25 years of experience in industry and academia. His research interests include mobile and distributed systems, industrial IoT and CPS, software orchestration, computer networking, artificial intelligence, and blockchain.