Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000. 10.1109/ACCESS.2017.DOI

[1]Faculty of Information Technology, Monash University, Melbourne, Australia (e-mail: mrah0018@student.monash.edu and {ahmad.salehishahraki and carsten.rudolph}@monash.edu)

Corresponding author: M Ridwanur Rahman (e-mail: mrah0018@student.monash.edu).

Attribute-Based Access Control (ABAC), Multi-Domain Environments, Policy Information Point (PIP).

=-15pt

Decentralized Policy Information Points for Multi-Domain Environments

M RIDWANUR RAHMAN1, AHMAD SALEHI S.1*MEMBER, IEEE*, AND CARSTEN RUDOLPH1

***Abstract*—Access control models have been developed to control unauthorized access to sensitive resources. This control of access is important because in the case of an organization, not all the data that they possess are meant to be accessed by everyone. Also, there is now a need for collaborative resource sharing between multiple organizations over open environments like the internet. Although there are multiple access control models that are being widely used, these models are providing access control within a closed environment i.e. within the organization using it. These have restricted capabilities in providing access control in open environments. Attribute-Based Access Control (ABAC) has emerged as a powerful access control model to bring fine-grained authorization to organizations which possess sensitive data and resources and want to collaborate over open environments. In an ABAC system, access to resources that an organization contains can be controlled by setting policies on attributes of the users. These policies are essentially conditions that need to be satisfied by the requester in order to gain access to the resource. In this paper, we provide an introduction to ABAC and by carrying forward the architecture of ABAC, we propose a Decentralized Policy Information Point (PIP) model. Our model proposes the decentralization of PIP, which is an entity of the ABAC model that allows the storage of attributes, enforces fine-grained access control for controlling the access of sensitive resources over multiple-domains and makes use of the concept of a cryptographic primitive called Attribute Based Signature (ABS) to keep the identities of the users involved, private. Our model can be used for collaborative resource sharing over the internet. The evaluation of our model and some comprehensive, real-world uses are also discussed to reflect the application of the proposed decentralized PIP model.**

I. Introduction

Nowadays, data is everywhere. Just to name a few, these data are being produced from users, from hospitals, sensors and machines. These data are being collected by businesses and academia from a wide variety of sources and at a high velocity. With this exponential rise of data, the need for collaborative relationships and the sharing of resources has become a necessity as the only way to leverage data is by securely sharing and leveraging it across open environments like the internet where most interactions occur between entities that have no pre-existing relationship [1]. The internet has made it possible for collaboration over not only an organization but across multiple different organizations over different geographic zones [2], [3]. This gives rise to what is called 'open environments' where different organizations from different parts of the world want to collaborate and share their resources. In other words, an open environment means multiple domains with different security and privacy requirements can collaborate and share resources. Data Owners (DO) like Businesses, Industries or Academia who possess data, operate their own administrative domains, called security domains, where they control access to their own data and resources and work independently of other domains but want to collaborate and share data [4]. This collaboration and sharing of resources is important as the data that the different domains possess can be used for new findings or to continue studies.

As the adoption of cloud computing has seen an exponential rise in recent years [5], DOs have been increasingly outsourcing their data to the cloud. This is due to the rapid elasticity, minimal investment, ease of use, reliability that the cloud provides for the benefits of resource sharing. Apart from that, cloud services provide security like Identity Management and Encryption and also provide the services of replication of data to provide data backup when services are down or data is lost [6]. The cloud has made data accessible as it provides location independent resource pooling [7]. However, the cloud works on the idea that the parties accessing the resource are from trusted domains. But the DO and the data requester might not be in the same trusted domain [8], [9]. This creates limitations in access control (AC) which is a major security vulnerability.

AC is where a user should have access to a resource in one context and not have the access in another [10], [11]. This is a very important requirement when sharing data or resources because unauthorized access of sensitive data or resource is risky [12], [13]. Therefore, the enforcement of strong AC over open environments is essential for resource sharing.

The traditional models that are currently in use are DAC (Discretionary Access Control) [14], MAC (Mandatory Access Control) [14] and RBAC (Role Based Access Control) [14]. These AC models are centralized and focus on the protection of data in an environment where the details of the users are already known. This means, the details or attributes of the users are already stored in their domain. This is called a 'closed environment' [15]. It is difficult to apply the traditional access control models to open environments as they are not very dynamic and flexible which we believe is limiting the sharing of resources across different domains. Also, MAC and DAC have high implementation costs and RBAC suffers from 'role explosion' if too many roles exist in the organization that is using it [10], [16], [17].

To overcome the limitations of RBAC, ABAC was identified with the idea that access control can be established based on present attributes of object, subject, action and environment of users [18]. ABAC has been called the next generation AC model as it provides AC with more efficient management and fine-grained control [19]. Key advantages of ABAC include: using the attributes of the users to provide fine-grained authorization over the resources they can access. In this way, traditional access control models can be implemented in an ABAC system. Also, ABAC can be used to eliminate the

closed world assumption of traditional access control models. This is achieved through the use of attributes in an ABAC model instead of identity which is the case of traditional access control models. Using attributes eliminates the need for the domains to know other domains, as this is a basic foundation of an open environment. ABAC in multi-domain environments require the secure exchange and cross-domain management of attributes [20], [21].

The main problem with current AC models is that they are limited in open environments. A solution to the problem of AC in open environments is to exchange attributes [22]. The attributes of the users are queried from the database by the Policy Information Point (PIP) of an ABAC system. This forms the basis of the research. Different domains would have their own PIPs and would query specific attributes when needed. In a multi domain environment, different domains have different security configurations. However, the domains need to have a system to query their attributes to be granted permissions to access resources in other domains. This is the main advantage of a decentralized PIP. Domains would have their own security configurations but the feature of querying the attribute remains the same. However, this is another problem. Privacy is required otherwise users can be identified based on their attributes. To keep the attributes secure, cryptographic primitives must be applied. Attribute Based Signature (ABS) was proposed in 2010 by Maji, Prabhakaran and Rosulek [23]. ABS is a cryptographic primitive where, a user, who possesses attributes, can sign a message using a key based on the predicate that is satisfied by their attributes. This signature only reveals that a single user who possess a set of attributes that satisfy a predicate has attested to the message. The signature keeps the attributes of the user secure. So, the identity of the user stays unknown. Therefore, we believe that an ABS scheme can be added to an ABAC model to keep the privacy of the users between multiple domains. This paper is not concerned with the cryptographic processes involved in signature generation. We assume that the appropriate signature generation and verification process works correctly. Which is why for our model, we used an open source implementation of ABS [24].

While many works have explored the applications of ABAC for existing AC problems [25], [26] [27], few have looked into the secure transfer of the attributes in multiple domains [28]. In this paper, we propose a general approach to the AC of resources by the enforcement of ABAC and apply the concept of ABS to implement security for the transfer of attributes between domains. The main advantage of this model is the decentralization of PIP which allows the ABAC model to run within different security configurations as set by the domain. The goal of this research is to show proof of a fine-grained AC model that will extend the functionalities of secure resource sharing in open environments where the details of the users are not known. We entitle the main contributions of this paper as follows:

- We propose an idea for a Distributed PIP which stores the attributes of the users of the domain.
- We carry forward the architecture of ABAC and focus on the development of a Decentralized Policy Information Point for ABAC. This can be used to provide fine-grained authorization to share data over the internet as it makes use of attributes.
- We use the concept of ABS as a form of cryptographic scheme to ensure both identity-anonymity and attribute-anonymity which would guarantee privacy for the user.

The rest of the paper is organised as follows. Section II provides the background of the access control models that are in use. Section III talks about related work done in the field of ABAC and ABS and provides a comparison table that compares different solutions proposed in the text. Section IV defines our proposed Decentralized PIP model. Section V shows the configurations and frameworks we used to implement our model. Section VI shows the evaluation of our model. In Section VII, we talk about where our model can be applied to be used. In Section VIII, we talk about the open issues and challenges of our model. In Section IX, we conclude the paper with a summary and future work.

II. BACKGROUND

TABLE I
LIST OF ABBREVIATIONS USED IN THIS PAPER AND THEIR ACRONYMS.

ABAC	Attribute Based Access Control
ABE	Attribute Based Encryption
ABS	Attribute Based Signature
AC	Access Control
ACL	Access Control List
API	Application Programming Interface
APK	Attribute Public Key
ASK	Attribute Signing Key
CP-ABE	Ciphertext-Policy Attribute Based Encryption
DAC	Discretionary Access Control
DO	Data Owner
EHR	Electronic Health Record
HTTP	Hypertext Transfer Protocol
IOT	Internet of Things
IIoT	Industrial Internet of Things
KP-ABE	Key-Policy Attribute Based Encryption
JSON	Javascript Object Notation
MAC	Mandatory Access Control
NIST	National Institute of Standards and Technology
OASIS	Organization for the Advancement of Structured Information Standards
PEP	Policy Enforcement Point
PDP	Policy Decision Point
PAP	Policy Administration Point
PIP	Policy Information Point
RBAC	Role Based Access Control
REST	Representational State Transfer
RFID	Radio-frequency identification
SKA	Signing Key
TPK	Trustee Public Key
UML	Unified Modeling Language
XACML	eXtensible Access Control Markup Language
XML	Extensible Markup Language

This section explains how the traditional AC models like ACL, MAC, DAC and RBAC work and what their disadvantages are [14]. Then it thoroughly explores the concept and architecture of ABAC. A list of abbreviations used in this paper and their acronyms are depicted in Table 1.

A. Traditional Access Control Models

The ability to perform actions such as create, read, update or delete on a resource is called access. AC limits these actions that a user can perform on a resource [29]. AC is usually controlled by an administrator who grants different permissions based on the actions to different users.

1) Access Control List (ACL): An ACL is a table that tells the operating system of a computer which access rights (read or write) a particular user has to a system object (file or directory). ACL is mostly used for applying security at the individual user level in an operating system or hardware like routers and cannot be used for business applications like web services.

2) Mandatory Access Control (MAC): MAC provides the strictest levels of control. They are primarily used by the government. MAC makes use of security policies to make access decisions where the policies are defined by the system administrator [30]. Users cannot change the access control settings. They are relatively difficult to use, have high-implementation costs and they are not dynamic enough to be used extensively with multiple policies or policy sets [31].

3) Discretionary Access Control (DAC): DAC supports more fine-grained access control for resources than MAC [31]. The central idea of DAC is that the owner of an object, who is usually the creator, has discretionary authority over who else can access a specific object [32]. The biggest vulnerability of DAC is that it can be attacked by the trojan horse malware [33]. Standard UNIX and Windows operating systems use DAC for their file systems.

4) Role-Based Access Control (RBAC): RBAC is one of the most widely used AC models [34]. In an RBAC system, an administrator adds different roles and sets up access permissions to those roles. Users are then made members of these roles. This is usually decided according to their responsibilities and qualifications. These roles can be modified or reversed very easily [17], [35].

The main limitations of RBAC are ”role expansion” and ”role-permission explosion” [10]. These occur when the industry grows and administrators may be overloaded with the maintenance of hundreds or thousands of roles across several applications. This can significantly affect the whole system [16]. If the number of roles and permissions increase significantly then the whole application might even become too heavy for operation. This reduces the scalability of the application and might even be needed to change which increases cost. Thus, the effects of using RBAC might be undesirable for too large and complex systems [36].

Apart from role-explosion, another limitation of an RBAC model is, the user's details must be known to the domain. This means, the domain administrator cannot supply a role to a user whose details are not known by the system. This means, RBAC cannot be used for systems that operate in open environments where the details of the user are not known by the system.

In conclusion, the limitations of the currently available access control models, like individual level security, role-explosion, reduced dynamicity have motivated the necessity for ABAC.

B. Attribute-Based Access Control (ABAC)

According to the National Institute of Standards and Technology (NIST), ABAC is an access control method where access permissions to resources are granted or denied by policies which are based on assigned attributes containing values of the subject, object, environment conditions and the names of resources [37]. The basic idea of ABAC is simple: if the entity who wants to access a resource, possess the specific attributes related to the resource as set by the administrator, then the access is granted, otherwise it is denied. An ABAC system can be implemented in technologies like web applications, IOT (Internet of things) devices and data science tools (Hadoop) [38], [19], [39]. Since AC is a major limitation in web applications that cater to open environments, we will be applying ABAC on a web application. More specifically, it will be implemented on the authorization layer of a web application which possess sensitive resources. This means, let's say an API (Application Programming Interface) of a web application has sensitive data. We want to share these data to authorized users over open environments. The administrator of the application will write a policy to protect the resource. For authorization, the requester would need to provide his attributes. The attributes are then sent to the ABAC system which decides whether to permit or deny the access request by checking if the attributes of the requester are reflected on the policy of the resource.

The access request has to contain attributes. An attribute is a statement about a user. For example, an attribute can be name, age, gender (male or female), state of residence of an individual [40]. These attributes need to be contained in a specific format based on categories. The attributes can be classified into the categories: Subject, Action, Resource and Environment. Subject category can hold attributes like username of the user, role, email address and age. Action category can hold what action the user wants to perform on the resource which can be create, read, update or delete. Resource category can hold the name of the resource they want to access, name of folder or path of the file. Environment category can hold the time of day or location.

Access policies or rules can be written using these attribute values. In an access policy, the administrator would set up different attributes based on the categories. These policies are usually written using a policy language. Out of which XACML (eXtensible Access Control Markup Language) is the best known and is mentioned in the NIST publication on ABAC [41], [37].

XACML is an XML-based language for access control that has been standardized by the Technical Committee of the OASIS consortium [42]. It describes both access control policy language and request/response language. The policy language is used to express access control policies. The request/response language expresses whether a particular access should be permitted or denied. XACML has a rich data typing model and allows stating complex conditions. The level of access that an administrator wants to restrict can be high or low depending on how the XACML policy is written [25].

```
<Policy PolicyId = "Policy1" rule-
```

```
    combining-algorithm="permit-overrides"
    >
 <Target>
    :Attr-Categ     :AttrID      :Attr Val
     :subject        :Role        :doctor
     :resource       :res-id      :doc1
     :action         :ac-id       :read
     :env            :city        :Melbourne
 </Target>

 <Rule RuleId="Rule1" Effect="Permit">
  <Condition>
      :Attr-Categ   :AttrID   :Attr Val
       :subject      :Name     :Alice
       :env          :city     :Melbourne
  </Condition>
 </Rule>

 <Rule RuleId="Rule2" Effect="Deny">
  <Condition>
      :Attr-Categ   :AttrID   :Attr Val
       :subject      :Role     :intern
       :env          :city     :city
  </Condition>
 </Rule>
</Policy>
```

An example of a XACML policy is given above.

Here, instead of exact XACML syntax, we used pseudo code for readability. Policy 1 applies to "All read access to doc1 by doctors from Melbourne" (target). For this policy, rule 1 would permit if Alice from Melbourne is trying to access but will deny if any interns from Melbourne are trying to access.

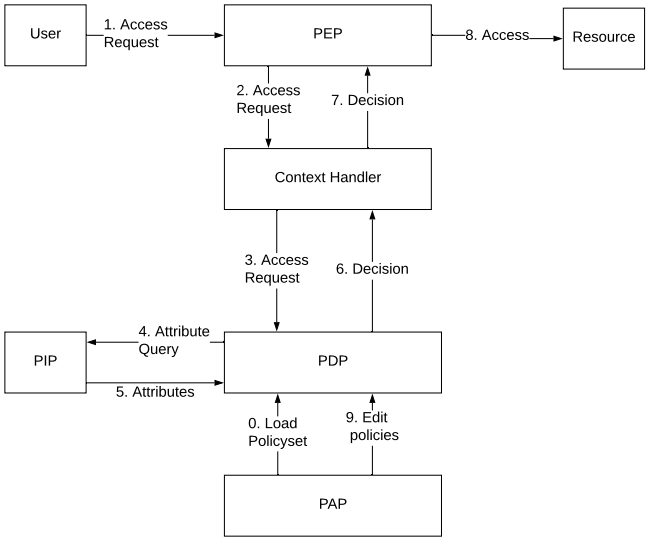

Fig. 1. ABAC Architecture as standardised by NIST [37].

An ABAC system constitutes of multiple domain entities. Each domain entity performs a specific function. They are presented in Figure 1.

The PEP controls the access to the resource and enforces the response of the PDP. It intercepts the access request and forwards it to the PDP. The PDP makes the access decision based on the stored policies in the PAP. It sometimes might need additional attributes which it can get from the PIP. The PIP stores the additional attributes which might be required by the PDP to make decisions.

In a web application, the access request containing the attributes can be transferred via RESTful APIs within a request-response HTTP protocol. The data format could be JSON or XML. However, upon interception by the PEP, the request is transformed to XACML format. The administrator would write the policies for the resources in XACML format too.

The process of ABAC (Figure 1) is explained as follows:

1) A request comes to the PEP. For the case of simplicity, this request contains attributes and wants to access a specific resource. PEP intercepts the request and sends it to the PDP via the context handler.
2) The PDP receives the request. Sometimes, it would not have enough attributes needed to make the decision. In which case, it asks the PIP for the missing attributes.
3) The PIP is mainly responsible for storing the attributes. Upon a query request, it searches the local database and responds with the attributes.
4) The PDP, with all the attributes now present, will check the policy stored in the PAP for that specific resource. It then will check if the attributes are reflected in the policies of the resources. Then it will generate an access/deny response which it passes to the PEP [43].
5) The PEP checks the response. If Accept or True (depending upon how the ABS sends a positive response), it lets the user access the resource; otherwise, it denies the request (Deny/False).

C. Attribute-Based Signature (ABS)

ABS is a security primitive that lets a user who possesses a set of attributes, sign a message by generating a predicate that is satisfied by his attributes [23]. This signature shows that a single user with a set of attributes that satisfied the predicate has attested to the message. It hides the attributes and all other information that can identify the user. It is suitable to implement ABS in applications that require both data authentication and the preservation of privacy [44]. To generate ABS, as explained in [23],

- ABS.TSetup: This is run by the signature trustee. It generates a trustee public key (TPK).
- ABS.ASetup: This is run by the attribute authority. It generates a key pair, attribute public key (APK) and attribute signing key (ASK).
- ABS.AttrGen: Takes the ASK as input and the attributes, this will generate the signing key (SKA).
- ABS.Sign: Takes the TPK, APK, SKA, message and predicate as input. A predicate is a combination of all the attributes. For simplicity, in our prototype, we used only the conjunction of the attributes. An ABS scheme in this process, by taking TPK, APK, SKA, message and the predicate as input, outputs a signature.

- Verify: Takes the TPK, APK, message and predicate as input. Outputs the value True/False.

III. Related Work

This section talks about the researches that used ABAC. We start by talking about Trust Management as it has certain similarities to ABAC. Then the section explores the research conducted on ABAC in multi-domains. Then it explores the implementations and applications of ABAC in multi-domains. Then, it talks about researches conducted for ABS. Table 2 shows the comparison between the proposals that are discussed here. Some of the works provide different levels of fine-grained authorization. However, none of the implementations provide a decentralized PIP.

A. Trust Management

Chapin et al. [4] defined Trust Management as an 'authorization procedure' where access to resources is granted based on multiple conditions. The authors talked about how the recent advances in distributed computing require a lot of system security that are being unmatched by traditional authorization mechanisms like RBAC. This can be matched by Trust Management systems as they enforce authorization policies. In another work by Blaze et al. they described Trust Management as something that expresses privileges and restrictions using a programming language [45]. Different automated Trust Management systems were suggested [46], [47]. These systems made use of attribute values which were described in name/value pairs. These values were signed using the issuer's private key and verified using the issuer's public key. This process is called a Digital Credential. Their approach for each domain in a multi-domain environment was to disclose each credential whose access policy satisfied the other domain's access policy. This process is very slow and adds overhead. Also it would be very difficult to scale and integrate with protocols and standards that are already in use. Most of the works on Trust Management were conducted for legacy systems in operating systems. Nowadays, industries use web services that transfer data over the internet in open environments so trust management would not be suitable [48]. However, it's concept is in one way quite similar to ABAC. In that, both use a language to write policies which checks for attributes and both have been intended to be used for distributed computing. This shows that the idea of using attributes for AC has been around for quite some time.

B. ABAC in open environments

The concept of ABAC has been found to be suitable for open environments like the internet where resource sharing is important and AC, based on identity, is not enough. This is where the use of attributes are more suitable [49]. In recent years, a lot of work has been conducted on ABAC [25], [27], [50]. These works list the deficiencies of the traditional access control models for web services like MAC, DAC and RBAC as insufficient in dynamicity, unscalable in regards to the growing business and limited to single domains. RBAC suffers from 'role-explosion' and 'role-expansion'. This is a highly undesirable situation where the growth of a company increases the roles exponentially which hampers their system. In this situation, the system becomes overloaded and in the worst case, might need to be changed which is not cost effective [16], [51], [52]. Most of these works cite ABAC as being the appropriate model for open environments because it is more semantically expressive, scalable and flexible.

Seol, K. el al., implemented ABAC for sharing sensitive Electronic Health Records (EHR) between hospitals in the cloud [25]. EHRs are designed to allow interoperability between multiple hospitals, laboraties, pharmacies and universities [53]. So, it is very important to be able to share EHRs. However, since EHR data contain sensitive and confidential information, their access needs to be controlled so that only authorized doctors can access them. The authors mentioned the inefficiency of RBAC as being inflexible and proposed an ABAC model for AC, where the security policies were written in XACML. In their model, they used an open source ABAC model developed by WSO2 and added cryptographic schemes like XML encryption and XML digital signature for encrypting their data so it cannot be seen by unwanted users. WSO2 is a company that has built a complete production ready ABAC system. They have an open-source version as well.

Although the authors' work is interesting as they provide fine-grained access control and security over multi-domains, it is unclear what attributes the first XACML request would contain. Different domains would contain numerous resources which need various attributes. Some might need a combination of ten attribute while some resource which are not very important can have 2 attributes in its policy. This is a clear gap in an ABAC system as not knowing the number of attributes to keep in the first request, one domain might send more or fewer attributes than is needed. This would add an overhead as sending too many attributes slow down the web service while sending fewer attributes would result in either deny response. Which would increase the number of requests-responses. The privacy issue of exchanging attributes between domains was not mentioned in their work.

Smari et al. proposed a crisis management system using ABAC in a distributed environment where crowd sourcing was used to gather data [54]. Their work shows how unlike traditional AC models, ABAC is suitable to be implemented in a collaboration software that uses attributes of users to give permissions to access certain services in times of emergencies. For security, the authors implemented trust management. The level of security the authors provided for their model is sufficient as their system is only useful during times of emergencies and does not work with sensitive data.

Utilizing ABAC for access control rather than the other access control methods seem favourable, however, it has a drawback as mentioned by Sandhu et al [55]. In ABAC, if there are numerous resources, each resource can have numerous attributes thus making the policies extremely complex. As written by Sandhu, this can result in an explosion of attributes. Having a large set of attributes makes it difficult for the administrative adoption of ABAC. According to their work, it would get difficult to assign attributes to

users and resources and policies might become too large and complex and will be difficult to modify. As a result, the authors proposed the concept of Attribute Transformation which includes Attribute Reduction and Attribute Expansion. In their work, they provided examples of both in predicate calculus. The proposed Attribute Transformation appears to be implementable however, it would be needed at a later part of the development when the number of attributes become too large.

A considerable amount of work has been done on combining ABAC and RBAC to gather the best features of both. Sandhu et al. suggested that ABAC users possess attributes however, little research has been done on the administration of the user attributes [50]. This is an issue of ABAC. Their paper proposes a framework to administer the user attributes of an ABAC system by using RBAC. The authors studied a popular administrative role based access control model called ARBAC97 as it is easy to operate and has sufficient documentation to provide a formal model for administering user attribute assignment in. The authors discussed that once the permissions for the roles are specified, it is easy to allocate and de-allocate users from administration however there are limitations since this is an RBAC system, there are chances of role explosion. Their system is however not clearly explained enough to be implemented and there might not even be a need to implement RBAC since it is possible to keep a limited number of people in charge of user attribute administration.

C. Implementations of ABAC in multi-domains

ABAC implementations using cryptographic approaches have been studied and proposed in different ways [56], [57]. This cryptographic approach is called Attribute Based Encryption (ABE). ABE is a scheme which utilizes a user's identity as attribute, and a set of attributes to encrypt and decrypt data [58]. ABE has two categories: the key-policy attribute based encryption (KP-ABE) and ciphertext-policy attribute based encryption (CP-ABE). KP-ABE uses the user's set of attributes to descibe the encrypted data and says that the access policy is attached to the user's private key. If a set of attributes of the private key satisfy the access policy, the user can decrypt the data. Otherwise, the user cannot get the data. CP-ABE implies that the access policy is associated with the encrypted data. It uses the user's set of attributes to describe the user's private key. If a set of attributes of the encrypted data satisfy the policy, the user will decrypt the data. Otherwise the user will not be able to decrypt. Although ABE schemes have strong security features, the models make use of a central authority to generate the keys needed to encrypt the data. The biggest lacking for ABE is that the central authority runs the risk of being compromised. Also, this model is not suitable for multi-domains. Another issue with ABE is that the policy needs to be decided at the time of encryption. It is very static which makes the policy management very difficult.

D. Applications of ABAC in multi-domains

The applications of ABAC are not only limited to web services. It can be applied in several other places. In the works of Figueroa et al., they implemented an ABAC method for RFID systems [59]. Their work is Internet of Things (IOT) based where the IOT devices needed fine-grained access permissions to access a Blockchain backend. Essentially, the ABAC method was implemented to grant or deny access to IOT devices. In the works of Zhao et al., they applied ABAC to a multi-domain grid computing environment [60]. They argued that implementing ABAC is the most optimized choice since grid computing runs on multi-domain and many different computers are involved. The authors chose ABAC for grid computing because according to them, grid computing runs in a multi-domain environment which has dynamically changing users, resources and services in different security domains. Colombo et al. has performed significant work in unifying ABAC with NoSQL databases [19]. Their paper targeted the development of a general ABAC framework for NoSQL databases like MongoDB. These works show the dynamicity and flexibility of ABAC and how it can be used for different purposes.

E. Applications of ABS in multi-domains

There are multiple signature schemes like Group Signatures, Ring Signatures [23]. Group signature require a group manager to create groups. The main limitation is if an organization is using group signatures, some members might not have been added. Ring signatures allow the user to choose any other signer including himself, and sign any message either by using his own secret key and the others' public key. This process does not require their approval or assistance so any user can use any other signer which is not secure [61].

In an ABS scheme, the signer signs a message with a claim-predicate generated from his attributes that satisfy a predicate [62]. The signature only reveals that the message was endorsed by one individual signer whose attributes satisfy a specific predicate [63]. This is completely different from ring and group signatures as they reveal a list of possible signers.

ABS has been implemented to provide security for distributed open environments like blockchain and IOT, [64], [44]. A few works have been done on multi-authority ABS as well [65], [63]. These works on multi-authority state that having a central authority to manage other authorities from where different secret keys are queried, has a major security issue as the central authority can be compromised. This is the main issue with ABE which makes use of central authority.

From these works, it can be clearly seen that ABAC plays a much better role in practice as compared to traditional AC models in decentralized and open environments. This is our reason for choosing ABAC. However, since ABAC works with exchanging attributes, security is an issue. ABS is suitable to eliminate this issue since it keeps the user's identifying information (attributes) secure and only reveals the user who satisfies the AC policy wants to access the resource. Apart from that if ABS is applied, users cannot lie about their attributes as they are signed by their signature key which is generated by an attribute authority. Also, in ABS, a verifier can create a predicate and no attributes are revealed to others. Also, even if the signatures do not match, the verifier does

not learn the attributes. Hence, attributes remain completely hidden and fine-grained access control is imposed.

IV. Proposed Model

This section talks about our proposed Decentralised PIP model and explains the fundamental concepts behind it.

A. System Model

In this section, we describe how our decentralized PIP model works. A scenario for our work would be: there are multiple different domains with resources. Each domain will be able to view the names of the resources that the other domains possess. However, since some resources are sensitive, the administrator of the domain can set access control policies to control the access of those resources. These policies are conditions or rules. These conditions need to be matched by the attributes of the users of other domains in order for them to gain access. Although our proposed model can be used in multi domains, for conveying the functionality of the model, our scenario is based on a peer-to-peer environment where the domains both know each other.

Requester Alice from domain 2 wants to view a resource in domain 1 where access control policies are set by Bob. So, Alice sends an access request to domain 1 containing the unique id of the resource she wants to access. Domain 1 checks the policy of the resource and responds with the names of the attributes needed to satisfy the policy, in the form of a predicate statement. A predicate statement would represent the names of the attributes that Alice needs to satisfy with values from her domain. Alice's domain will be able to see the names of attributes that are needed and lets Alice know if her available attributes can satisfy the predicate statement. Then, using only the attributes that are required in the predicate statement, Alice will be able to generate her own claim-predicate. Her claim-predicate contains the values of attributes that satisfy the predicate from domain 1. Using this claim-predicate statement, she will sign a message. This message is transferred to domain 1 where it is verified using another predicate statement generated from the values of the attributes in the policy. If the verification is true, then Alice will be able to view the contents of the resource, otherwise, she will not be granted access. Other domains would have to follow the same procedure in order to gain access to resources. This process is depicted in Figure 2.

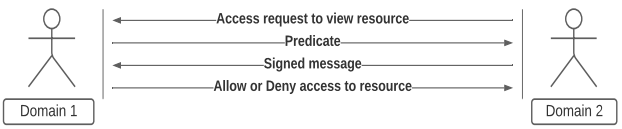

Fig. 2. Interaction between two domains in the proposed Decentralized PIP model.

B. Proposed Methodology

The fundamental steps of our proposed model are listed here:

- Administrator will write policies for resources. This policy will be written in a name-value pair.
- Requester from a different domain, who wants to view the resource, will get a predicate which would contain the names of attributes needed to view the resource. This predicate is generated from the attribute names in the policy of that resource as set by the administrator.
- Requester would collect the attributes needed to satisfy the predicate from their PIP. This essentially forms the base of our Decentralized PIP model. The PIPs are used to query attributes. In a multi-domain environment, different domains reside. So for collaboration, these domains would query their PIPs for attributes. All these PIPs are not controlled by a single domain rather they are controlled by their own domain.
  Using the attributes that are returned from the PIP, the requester would generate their claim-predicate statement. Then sign a message with the claim-predicate statement. This signing is accomplished with the open source ABS.
- The message will be sent to the domain that contains the resource using HTTP methods.
- The domain will verify the signature and permit or deny the access to the resource. This is also accomplished with the open source ABS.

1) Predicate Management: Bob is an administrator of domain 1. He enters a new resource called 'Resource 1'. Bob has the option to either make it accessible for all domains or he can protect it by writing a policy. In a policy, Bob would have to enter attributes according to category Subject, Action, Resource and Environment. The Resource category will hold the identifier of the resource which could be the name/unique id/file path. For each attribute, he would need to enter the name of the attribute as well as the attribute value. This means, each resource will have a name-value pair of attributes where the name is the attribute name and the value is the attribute value. Lets say resource 1 can only be read by Alice who is a cardiologist from the Box Hill hospital in Melbourne. Alice's subject will be Alice, cardiologist and Box Hill hospital. Her environment will be Melbourne. So, to enter this policy, Bob would need to enter names of each of the attributes. This means, attribute value Alice will have the attribute name 'first name', cardiologist attribute value will have the attribute name 'position' or 'doctor_type', 'Box Hill' attribute name would have the attribute value 'hospital' and Melbourne attribute name will have the attribute value 'city'. The structure is shown here:

```
first_name: Alice
position: cardiologist
hospital: Box Hill
city: Melbourne
```

Different domains would design their databases differently where the variable names for attributes might be different. For example, the database field 'first_name' in one domain might be named 'fname' in another domain. This is why the name-value pair of the attribute is important as this would enable a way to depict a database field in one domain that might be named differently on a different domain. Although the name-

TABLE II
COMPARISON OF EXISTING WORKS.

Citation	Proposed model	Strength	Weakness	Privacy-preservation method	Operating Environment	Fine-grained AC	Research Type	PIP Distribution
17	Cloud-based EHR model	Privacy	Difficult to implement. It might be prone to attacks. Unsuitable for open environments.	XML encryption and XML Digital Signature	Cloud-based	✓	Implementation	X
19	PASH	Supports a large number of attributes. It reveals only attribute names while attribute values are hidden	Uses ABE, so a central authority might be needed for which is prone to attacks	CP-ABE and partially hidden access policies	Cloud-based	✓	Implementation	X
39	MD-ABAC	Makes up shortcomings of RBAC	Security	-	Cloud-based	✓	Implementation	X
40	ARBAC97	Extends RBAC	Cannot use a large number of attributes (RBAC). Cannot express complex policies.	-	Cloud-based	X	Implementation	X
43	Extended ABAC with trust and privacy	Scalability	Requires central authority which is prone to attacks	ABE	Cloud-based	✓	Implementation	X
45	PMDAC-ABSC	More efficient cryptography	Requires central authority	ABE and ABS	Cloud-based	✓	Implementation	X
34	SA-ABS for IIoT	Ensures data authentication and protects privacy.	Lots of overhead for calculations.	ABS	Cloud-based	-	Implementation	X
44	Attribute Transformation	Reduction of attributes hence modular policy design	-	-	-	X	Proposal	X
46	Hybrid ABE	-	Requires central authority	CP-ABE and KP-ABE	Cloud-based	✓	Implementation	X
47	ABAC in RFID	Robust and scalable	Difficult to implement	-	RFID	✓	Implementation	X
48	ABAC in Grid Computing	Secure and Robust	Sensitive to malicious attacks	Uses Trust Engine to manage trust between nodes in grid	Grid Computing	✓	Implementation	X
49	Decentralized ABS for Healthcare Blockchain	Efficient verification	Difficult to implement	Cryptography	Blockchain	✓	Implementation	X

Citation	Proposed model	Strength	Weakness	Privacy-preservation method	Operating Environment	Fine-grained AC	Research Type	PIP Distribution
15	Attribute-Based Signatures	Users cannot collude their attributes	Costly computation calculations	Signature Generation	Cloud-based	✓	Implementation	X
50	Decentralized ABS scheme for healthcare blockchain	Fully decentralized	Blockchain is fairly new.	ABS	Blockchain	✓	Implementation	X
51	ABS in Blockchain	Multiple authorities for generating keys.	Blockchain is fairly new.	ABS	Cloud-based	✓	Implementation	X
52	ePass	Enhanced performance with less computational costs.	Does not support multi-authority.	ABS	Cloud-based	✓	Implementation	X

value pair is useful, there needs to be a mapping process for the names in other domains.

2) Accessing Resource: With the predicate statement provided from domain 1, Alice's PEP will query her PIP to check if the required attributes exist or not. For our scenario, we are assuming she has the specific attributes required to satisfy the predicate and the PIP returns the specific attributes to the PEP. Then, using the set of attributes, her PEP will generate a claim-predicate statement which satisfy the predicate statement from domain 1. An example of her claim-predicate statement is: ALICE AND CARDIOLOGIST AND BOX HILL AND MELBOURNE. Using this claim-predicate, her PEP will sign a message. This message endorses that Alice has the specific attributes needed to satisfy the predicate from domain 1. This message will be serialized and transferred to domain 1. Domain 1's PEP will verify the signature based on the attribute values set on the policy. The verifier outputs value True or False. She will be able to access the resource if the verifier returns True otherwise she will not.

The steps are outlined here and the complete UML Sequence diagram is show in Figure 3:

- Each domain will have their own ABAC system consisting of the basic domain entities. Each domain entity is a separate decoupled microservice running on their own web server. The PEP only allows the names of the resources be available to the other domains. The contents are hidden as set by the administrator. The PEP also intercepts any request to the database which has the resources saved.
- The administrator write policies for the resources.
- Users from Domain 2 will be able to view the resource names and select a specific resource they want to view. This is provided by the PEP of domain 1. Domain 2 will also be able to send an access request to domain 1 to view the resource.
- The PEP in domain 1 would receive the request. It will be able to query the PAP to find the policy for the specific resource. After finding the specific policy, the PEP will turn the policy into a claim-predicate and send it to domain 2.
- The PEP of domain 2 would receive the claim-predicate. Then the PEP would need to know if domain 2 can satisfy it. So, the PEP would query the PIP to know if the attributes exist. This forms the basis of of the decentralized PIP. Each domain will have their own PIP and will be able to query the attributes of the user each time any user of the domain wants to access resources in other domains.
- The PEP will receive the attributes. Using these attributes, it will generate a predicate statement that satisfies the claim-predicate. Then it will sign a message with the predicate.
- The PEP of domain 2 will send the signature to domain 1 for verification.
- The PEP of domain 1 receives the signature and verifies it to give a True or False response.

C. Security Goal

We consider each domain to have an administrator who writes policies and adds resources. Each domain would have users as well. In a general sense, each user should be able to add resources and policies, however, for our implementation we only consider the administrator to be able to do that.

For our model, we assume the users do not collude with their attributes. This means multiple users should not be able to pool together their attributes. Users can only use the attributes they themselves own. This is maintained by keeping the attributes stored in a database and using the PIP to query their attributes. We also assume that all the domain entities function properly.

V. Implementation

We used Django (v3.1) which is a python web framework to implement the web applications [66]. Django web framework is a fast, easy to use and provides numerous features for fast prototype development like a HTML template engine that

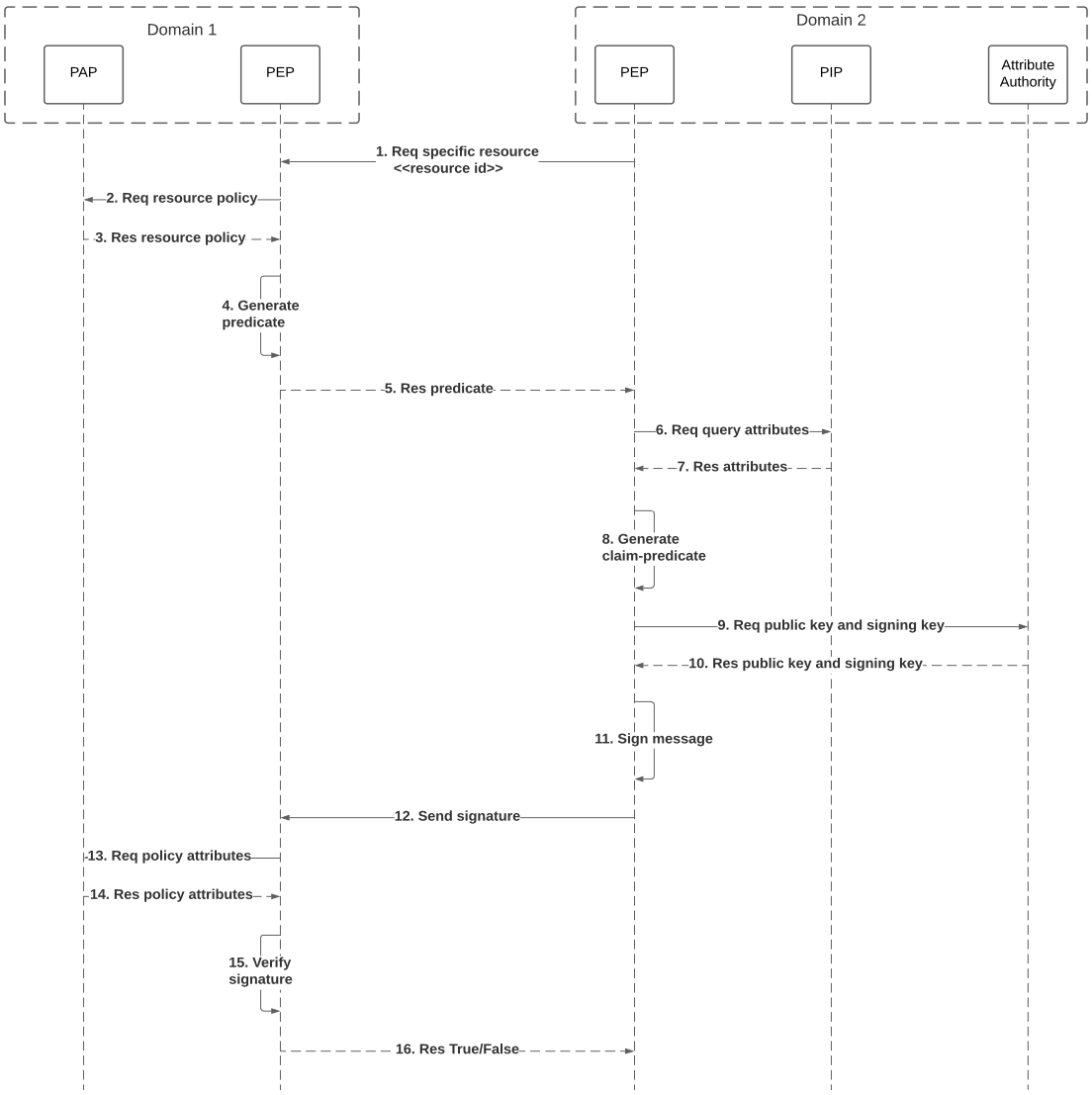

Fig. 3. The UML (Unified Modeling Language) Sequence Diagram of the Proposed Decentralized PIP model.

provides features for fast frontend development for the user. Django supports authentication, development of RESTful APIs and easily connects to databases.

Each domain entity of the ABAC system, PEP, PDP, PAP and PIP, was implemented following microservice software architecture. Meaning, each domain entity is a separate web application running on their own server. This represents the actual architecture of the ABAC system since each part is represented as a separate domain entity (Figure 1). Instead of having one monolithic architecture, a microservice architecture allows a complete system to be divided into multiple, smaller, individual and independent services. Each domain entity runs as an autonomous process and communicate with other domain entities through web services. Web services are software that are available over the internet and uses standard messaging. This messaging is provided by RESTful APIs over open standards like HTTP protocol. RESTful APIs provide http methods like GET, POST, UPDATE and DELETE. Web services use these methods to make their communications clear. In our model, RESTful APIs are used over HTTP protocol to send or receive policies and messages.

In our model, the PAP provides the functionality for users to add resources and policies. The user interface of the PAP was implemented using the Django template engine and basic HTML, CSS. Jquery was used to make the pages more user friendly and intuitive [67]. Policies can be only set by the administrator of the domains. This means, that our model needed a user login-logout system. In-built User Authentication of the Django web framework was used to handle the administrator authentication [68].

We used an open source implementation of the ABS in our prototype [24]. It uses an open-source framework called Charm-Crypto, which is written in Python, to implement the ABS algorithms [69]. Charm-Crypto is a framework for developing advanced cryptosystems and it supports the development of new protocols. It is freely available to researchers and has been used in many researches [70].

VI. Evaluation

This section evaluates the implementation of the proposed Decentralized PIP model with the open source ABS. We are interested to see how quickly a user can access a resource. This is the because in typical use, a user would like to access a resource as soon as possible in multi-domain environments. Large wait times are undesirable when trying to access resources. In typical use, the user will be presented with the contents of the resource that they requested. But there would be a lot of processes going on in the back-end, which is hidden from the user. This evaluation section was conducted on a Linux machine with Intel Core i7-10710U 12 core CPU running at 1.10GHz, 16GB RAM and Python 3.6.9 was used.

In our model, domain 1 and domain 2 are two separate domains each running with their own security configurations in their own web server. This simulates the multi-domain environment. We generated the multi-domain environment by running two different applications in two different ports. Each of the domains possess their own PIP. The function of each PIP is to query the attributes. This is to make sure that the users are not able to pool their attributes together. Each user is charge of their own attributes. We imagine domain 1 possessing resources which means, domain 1 would be in charge of verifying signature to permit or deny access to their resource and for the administrator to enter policies. User from Domain 2 would sign messages based on claim-predicate. This signing process is very resource-intensive since the attribute authority of domain 2 would have to generate multiple keys and then the domain would sign a message using those keys. We list two methods to access resources:

- By generating new private and public keys and signature each time we want to access a resource.
- By generating public key, private key, signing key and signature once and storing according to their predicate in a database. Then querying the specific keys and signature when predicates match.

Here we evaluate our model to see which method is more efficient by finding a relation between the number of attributes and the time taken to get a response. The more efficient method would allow the access to the resource in less time.

In the graphs provided in this section, the 'number of attributes' mean the attributes that are being used to generate the key and signature which is depicted in the x-axis of the graphs. The y-axis of the graphs show the get taken to get the response in seconds.

The bar graph in Figure 4 represents the time taken to verify signature based on the number of attributes. This graph has been generated based on the data from domain 1. Since we ran the prototype using two methods, there are duplicate number of attributes. The graph shows a linear relation between the time and the number of attributes. As the number of attributes increase, the time it takes to verify increases. This happens because the more the number of attributes, the more number of calculations need to be performed by the domain to verify.

The bar graphs in Figure 5 and 6 represent the time taken to get a response where the keys were generated each time and the time taken to get response where the keys were stored beforehand, respectively. This test was performed to see which method is quicker. They both show a linear relation between the time in seconds and the number of attributes. However,

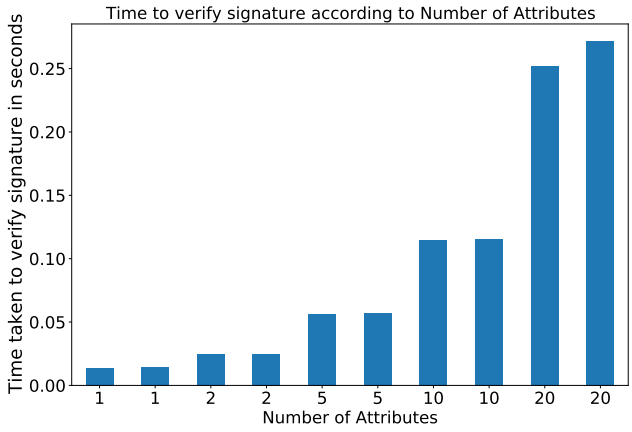

Fig. 4. Time to verify signature.

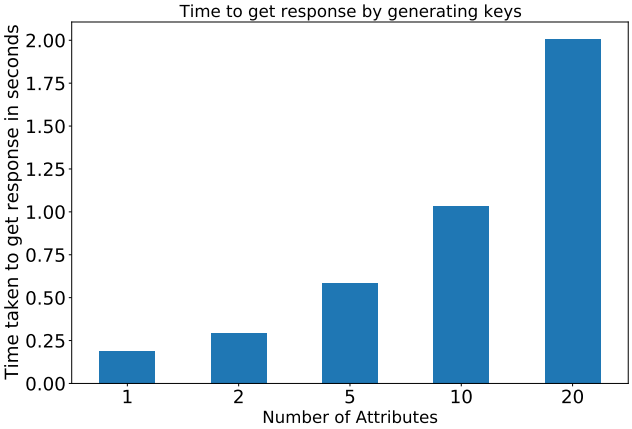

Fig. 5. Overall time taken to get response by generating keys.

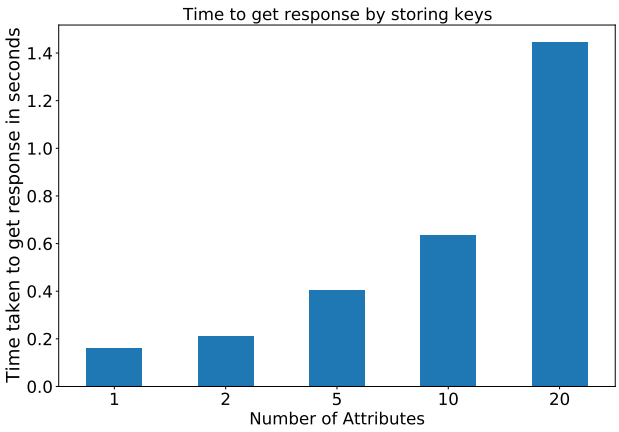

Fig. 6. Overall time taken to get response by storing keys.

the time taken for each set of attribute values in stored key is less than the time taken by generating new keys each time. We can say that storing the keys before might be a viable option to achieve more efficiency.

The keys in our model were stored in a Mongo database collection. Mongo database is a no-sql database program which is said to be faster than a relational database like PostgreSQL [71]. Although it is fast, there might be performance bottlenecks in the case of querying through millions of records when the database gets heavy. This would add more time during the querying process which would affect the overall time. Also, keeping keys stored in database might lead to attacks on the database where a third-party would be able to copy the keys and signature and might pose threats. Apart from this, we have not taken into account the geographic locations of the domains. The more geographic distance there is between two domains, the more time it would take for them to interact. Speeds might vary due to domains being separated across large geographic locations as this introduces latency. Also, the configurations of the servers might affect the speed of key generation too. As key generation requires mathematical computations which is resource consuming. So, a server with low specifications would take longer to generate keys based on the same attributes whereas a server with high specification would take less time.

In conclusion, although generating keys each time has proved to be slower, it is the better option. As generating new keys each time eliminates the need of storing in a database.

VII. Applications of the proposed model

ABAC, because of the utilization of attributes, can describe any AC scenario no matter how complex the situation gets. One of the biggest advantages of our decentralized PIP model is its ability to act like traditional AC models if needed. If there is an API and we want to provide access based on roles, we can start with two attributes: Role and resource. Role attribute will hold the name of the role and resource attribute will hold the name of the API.

Our model can be treated as an externalized authorization application. This mean, our model can protect APIs containing sensitive resources with its fine-grained authorization. It can exist outside the complete technology stack that is being used in the business. Users from other domains who want to access the APIs would have to go through the ABAC model first before getting access to the API. This system can be used in healthcare. Hospitals protect access to sensitive EHRs. Hospitals need to share EHRs to various other hospitals, laboratories, pharmacies and universities. But due to the sensitive nature of EHRs, it is important for the hospitals to control the access so that unauthorized users do not access them. Using our ABAC model, fine-grained authorization can be provided to EHRs.

Since the internet is an open environment, numerous different resources are available in numerous different domains. In the traditional system, to access a resource, a user would have to sign up with the domain by providing their details then remember the username and password every time they need to access the resources from that domain. Although this system works well, users might lose their credentials or even their credentials might get stolen from the domain. Decentralized PIP model might be useful in this situation since to access a domain's resource, a user does not need to sign up. The user's domain will communicate with the domain and allow or reject access of the user to the resource.

VIII. Open Issues and Challenges

The AC of data is a very important issue. This is because, collaboration is undeniably important and granting all users access to all resource is risky. This section highlights some of the most important challenges in deploying and utilizing the decentralized PIP model.

- Attribute Name Mapping: For implementation of the prototype, we used attributes in a name-value pair for each resource. So there needs to be a mapping for the attribute names in the domain that wants to access the resource through which the domain that contains the resource can get the correct attributes. This is an issue since different domains have different names for their attributes.
- Name-value pair policy provides less features as compared to XACML. So, there needs to be a way to extract a predicate statement from a XACML policy. This will ensure complex policies to be used as a predicate.
- The transfer of a predicate statement from a resource-containing domain might be open to attacks. This is a security threat. There needs to be a way to encrypt the predicate so as to open be accessible by the domain that sent the access request to view a resource.
- A major challenge of multi-domains is collusion attacks. A collusion attack is when multiple authorities collude and combine attributes of different users. There needs to be a way to implement trust among the authorities.

IX. Conclusion and future work

In this paper, we proposed a Decentralized PIP model that makes use of attributes of users to grant or deny access to data. The proposed model provides more flexible and fine-grained AC than existing AC systems like RBAC and eliminates the risk of exposing private user information by implementing the concept of ABS in which a signer signs message with his attributes and the verifier can only know whether the signer owns attributes satisfying his policy. The implementation of a prototype demonstrated the feasibility of the proposed model.

Currently, our model only allows writing attribute names and values and does not support ranges of values when signing new messages. In the future, we will add more functionalities in signing messages. We will also expand the implementation of the prototype to implement a more refined system and deploy it on the cloud to make performance evaluations. Also we will make evaluations with other attribute based security schemes.

References

[1] Ting Yu, Marianne Winslett, and Kent E Seamons. Interoperable strategies in automated trust negotiation. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 146–155, 2001.

[2] Surekha Kaul. Information resource sharing models in developing countries: a network emerging from the world bank supported environmental management capacity building project. *Inspel*, 35(1):9–26, 2001.

[3] Fara Jamal, Mohd Taufik Abdullah, Zurina Mohd Hanapi, and Azizol Abdullah. Reliable access control for mobile cloud computing (mcc) with cache-aware scheduling. *IEEE Access*, 7:165155–165165, 2019.

[4] Peter C Chapin, Christian Skalka, and X Sean Wang. Authorization in trust management: Features and foundations. *ACM Computing Surveys (CSUR)*, 40(3):1–48, 2008.

[5] Mohammed A Aleisa, Abdullah Abuhussein, and Frederick T Sheldon. Access control in fog computing: Challenges and research agenda. *IEEE Access*, 8:83986–83999, 2020.

[6] Michele De Donno, Koen Tange, and Nicola Dragoni. Foundations and evolution of modern computing paradigms: Cloud, iot, edge, and fog. *Ieee Access*, 7:150936–150948, 2019.

[7] Cong Wang, Qian Wang, Kui Ren, and Wenjing Lou. Privacy-preserving public auditing for data storage security in cloud computing. In *2010 proceedings ieee infocom*, pages 1–9. Ieee, 2010.

[8] Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou. Achieving secure, scalable, and fine-grained data access control in cloud computing. In *2010 Proceedings IEEE INFOCOM*, pages 1–9. Ieee, 2010.

[9] Qinglin Qi and Fei Tao. A smart manufacturing service system based on edge computing, fog computing, and cloud computing. *IEEE Access*, 7:86769–86777, 2019.

[10] Mumina Uddin, Shareeful Islam, and Ameer Al-Nemrat. A dynamic access control model using authorising workflow and task-role-based access control. *IEEE Access*, 7:166676–166689, 2019.

[11] Chi-Wei Liu, Wei-Fu Hsien, Chou Chen Yang, and Min-Shiang Hwang. A survey of attribute-based access control with user revocation in cloud data storage. *IJ Network Security*, 18(5):900–916, 2016.

[12] Abdulrahman Almutairi, Muhammad Sarfraz, Saleh Basalamah, Walid Aref, and Arif Ghafoor. A distributed access control architecture for cloud computing. *IEEE software*, 29(2):36–44, 2011.

[13] Sheng Ding, Jin Cao, Chen Li, Kai Fan, and Hui Li. A novel attribute-based access control scheme using blockchain for iot. *IEEE Access*, 7:38431–38441, 2019.

[14] N Geetha and MS Anbarasi. Role and attribute based access control model for web service composition in cloud environment. In *2017 International Conference on Computational Intelligence in Data Science (ICCIDS)*, pages 1–4. IEEE, 2017.

[15] Stephen Weeks. Understanding trust management systems. In *Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001*, pages 94–105. IEEE, 2000.

[16] Aaron Elliott and Scott Knight. Role explosion: Acknowledging the problem. In *Software Engineering Research and Practice*, pages 349–355, 2010.

[17] Qiang Liu, Hao Zhang, Jiafu Wan, and Xin Chen. An access control model for resource sharing based on the role-based access control intended for multi-domain manufacturing internet of things. *IEEE Access*, 5:7001–7011, 2017.

[18] Canh Ngo, Yuri Demchenko, and Cees de Laat. Multi-tenant attribute-based access control for cloud infrastructure services. *Journal of information security and applications*, 27:65–84, 2016.

[19] Pietro Colombo and Elena Ferrari. Towards a unifying attribute based access control approach for nosql datastores. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 709–720. IEEE, 2017.

[20] Ahmad Salehi, Carsten Rudolph, and Marthie Grobler. A dynamic cross-domain access control model for collaborative healthcare application. In *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 643–648. IEEE, 2019.

[21] Ahmad Salehi Shahraki, Carsten Rudolph, and Marthie Grobler. A dynamic access control policy model for sharing of healthcare data in multiple domains. In *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 618–625. IEEE, 2019.

[22] Mikel Uriarte, Jasone Astorga, Eduardo Jacob, Maider Huarte, and Manuel Carnerero. Expressive policy-based access control for resource-constrained devices. *IEEE Access*, 6:15–46, 2017.

[23] Hemanta K Maji, Manoj Prabhakaran, and Mike Rosulek. Attribute-based signatures. In *Cryptographers' track at the RSA conference*, pages 376–392. Springer, 2011.

[24] Mamietti. Abs. https://github.com/Mamietti/ABS, 2020.

[25] Kwangsoo Seol, Young-Gab Kim, Euijong Lee, Young-Duk Seo, and Doo-Kwon Baik. Privacy-preserving attribute-based access control model for xml-based electronic health record system. *IEEE Access*, 6:9114–9128, 2018.

[26] Jianghong Wei, Wenfen Liu, and Xuexian Hu. Secure and efficient attribute-based access control for multiauthority cloud storage. *IEEE Systems Journal*, 12(2):1731–1742, 2016.

[27] Yinghui Zhang, Dong Zheng, and Robert H Deng. Security and privacy in smart health: Efficient policy-hiding attribute-based access control. *IEEE Internet of Things Journal*, 5(3):2130–2145, 2018.

[28] Hai-bo Shen and Fan Hong. An attribute-based access control model for web services. In *2006 Seventh International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'06)*, pages 74–79. IEEE, 2006.

[29] Younis A Younis, Kashif Kifayat, and Madjid Merabti. A novel evaluation criteria to cloud based access control models. In *2015 11th International Conference on Innovations in Information Technology (IIT)*, pages 68–73. IEEE, 2015.

[30] Gaozu Wang, Weihuai Li, and Wenbin Li. Research on validity evaluation of mandatory access control policy under lsm framework. In *2010 International Conference on Computational Intelligence and Security*, pages 306–309. IEEE, 2010.

[31] Ryan Ausanka-Crues. Methods for access control: advances and limitations. *Harvey Mudd College*, 301:20, 2001.

[32] Ravi Sandhu and Qamar Munawer. How to do discretionary access control using roles. In *Proceedings of the third ACM workshop on Role-based access control*, pages 47–54, 1998.

[33] Deborah D Downs, Jerzy R Rub, Kenneth C Kung, and Carole S Jordan. Issues in discretionary access control. In *1985 IEEE Symposium on Security and Privacy*, pages 208–208. IEEE, 1985.

[34] MAC Dekker, Jason Crampton, and Sandro Etalle. Rbac administration in distributed systems. In *Proceedings of the 13th ACM symposium on Access control models and technologies*, pages 93–102, 2008.

[35] David Ferraiolo, Janet Cugini, and D Richard Kuhn. Role-based access control (rbac): Features and motivations. In *Proceedings of 11th annual computer security application conference*, pages 241–48, 1995.

[36] Rubina Ghazal, Ahmad Kamran Malik, Nauman Qadeer, Basit Raza, Ahmad Raza Shahid, and Hani Alquhayz. Intelligent role-based access control model and framework using semantic business roles in multi-domain environments. *IEEE Access*, 8:12253–12267, 2020.

[37] Vincent C Hu, David Ferraiolo, Rick Kuhn, Arthur R Friedman, Alan J Lang, Margaret M Cogdell, Adam Schnitzer, Kenneth Sandlin, Robert Miller, Karen Scarfone, et al. Guide to attribute based access control (abac) definition and considerations (draft). *NIST special publication*, 800(162), 2013.

[38] Maanak Gupta, Farhan Patwa, and Ravi Sandhu. An attribute-based access control model for secure big data processing in hadoop ecosystem. In *Proceedings of the Third ACM Workshop on Attribute-Based Access Control*, pages 13–24, 2018.

[39] Ali Nauman, Yazdan Ahmad Qadri, Muhammad Amjad, Yousaf Bin Zikria, Muhammad Khalil Afzal, and Sung Won Kim. Multimedia internet of things: A comprehensive survey. *IEEE Access*, 8:8202–8250, 2020.

[40] Keith Frikken, Mikhail Atallah, and Jiangtao Li. Attribute-based access control with hidden policies and hidden credentials. *IEEE Transactions on Computers*, 55(10):1259–1270, 2006.

[41] Jason Crampton and Charles Morisset. Ptacl: A language for attribute-based access control in open systems. In *International Conference on Principles of Security and Trust*, pages 390–409. Springer, 2012.

[42] OASIS Standard. extensible access control markup language (xacml) version 3.0, 2013.

[43] Ed Coyne and Timothy R Weil. Abac and rbac: scalable, flexible, and auditable access management. *IT Professional*, (3):14–16, 2013.

[44] Hu Xiong, Yangyang Bao, Xuyun Nie, and Yakubu Issifu Asoor. Server-aided attribute-based signature supporting expressive access structures for industrial internet of things. *IEEE Transactions on Industrial Informatics*, 16(2):1013–1023, 2019.

[45] Matt Blaze, Joan Feigenbaum, John Ioannidis, and Angelos D Keromytis. The role of trust management in distributed systems security. In *Secure Internet Programming*, pages 185–210. Springer, 1999.

[46] Marianne Winslett, Ting Yu, Kent E Seamons, Adam Hess, Jared Jacobson, Ryan Jarvis, Bryan Smith, and Lina Yu. Negotiating trust in the web. *IEEE Internet Computing*, 6(6):30–37, 2002.

[47] Ting Yu and Marianne Winslett. A unified scheme for resource protection in automated trust negotiation. In *2003 Symposium on Security and Privacy, 2003.*, pages 110–122. IEEE, 2003.

[48] Zhengping Wu and Alfred C Weaver. Dynamic trust establishment with privacy protection for web services. In *IEEE International Conference on Web Services (ICWS'05)*. IEEE, 2005.

[49] Lanjing Wang and Baoyi Wang. Attribute-based access control model for web services in multi-domain environment. In *2010 International Conference on Management and Service Science*, pages 1–4. IEEE, 2010.

[50] Xin Jin, Ram Krishnan, and Ravi Sandhu. A role-based administration model for attributes. In *Proceedings of the First International Workshop on Secure and Resilient Architectures and Systems*, pages 7–12, 2012.

[51] D Richard Kuhn, Edward J Coyne, and Timothy R Weil. Adding attributes to role-based access control. *Computer*, 43(6):79–81, 2010.

[52] Aaron Elliott and Scott Knight. Towards managed role explosion. In *Proceedings of the 2015 New Security Paradigms Workshop*, pages 100–111, 2015.

[53] Ahmed Raza Rajput, Qianmu Li, Milad Taleby Ahvanooey, and Isma Masood. Eacms: emergency access control management system for personal health record based on blockchain. *IEEE Access*, 7:84304–84317, 2019.

[54] Waleed W Smari, Patrice Clemente, and Jean-Francois Lalande. An extended attribute based access control model with trust and privacy: Application to a collaborative crisis management system. *Future Generation Computer Systems*, 31:147–168, 2014.

[55] Prosunjit Biswas, Ravi Sandhu, and Ram Krishnan. Attribute transformation for attribute-based access control. In *Proceedings of the 2nd ACM Workshop on Attribute-Based Access Control*, pages 1–8, 2017.

[56] Qian Xu, Chengxiang Tan, Zhijie Fan, Wenye Zhu, Ya Xiao, and Fujia Cheng. Secure multi-authority data access control scheme in cloud storage system based on attribute-based signcryption. *IEEE Access*, 6:34051–34074, 2018.

[57] Yogita S Gunjal, Mahesh S Gunjal, and Avinash R Tambe. Hybrid attribute based encryption and customizable authorization in cloud computing. In *2018 International Conference on Advances in Communication and Computing Technology (ICACCT)*, pages 187–190. IEEE, 2018.

[58] Jialu Hao, Jian Liu, Huimei Wang, Lingshuang Liu, Ming Xian, and Xuemin Shen. Efficient attribute-based access control with authorized search in cloud storage. *IEEE Access*, 7:182772–182783, 2019.

[59] Santiago Figueroa, Javier Añorga, Saioa Arrizabalaga, Iñigo Irigoyen, and Mario Monterde. An attribute-based access control using chaincode in rfid systems. In *2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–5. IEEE, 2019.

[60] Tiezhu Zhao and Shoubin Dong. A trust aware grid access control architecture based on abac. In *2010 IEEE Fifth International Conference on Networking, Architecture, and Storage*, pages 109–115. IEEE, 2010.

[61] Ronald L Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 552–565. Springer, 2001.

[62] You Sun, Rui Zhang, Xin Wang, Kaiqiang Gao, and Ling Liu. A decentralizing attribute-based signature for healthcare blockchain. In *2018 27th International conference on computer communication and networks (ICCCN)*, pages 1–9. IEEE, 2018.

[63] Rui Guo, Huixian Shi, Qinglan Zhao, and Dong Zheng. Secure attribute-based signature scheme with multiple authorities for blockchain in electronic health records systems. *IEEE access*, 6:11676–11686, 2018.

[64] Jinshu Su, Dan Cao, Baokang Zhao, Xiaofeng Wang, and Ilsun You. epass: An expressive attribute-based signature scheme with privacy and an unforgeability guarantee for the internet of things. *Future Generation Computer Systems*, 33:11–18, 2014.

[65] Jiameng Sun, Ye Su, Jing Qin, Jiankun Hu, and Jixin Ma. Outsourced decentralized multi-authority attribute based signature and its application in iot. *IEEE Transactions on Cloud Computing*, 2019.

[66] Django Software Foundation. Django. https://www.djangoproject.com/start/overview/, 2020.

[67] jQuery. jquery. https://jquery.com/, 2020.

[68] Django Software Foundation. Django authentication system. https://docs.djangoproject.com/en/3.1/topics/auth/default/, 2020.

[69] Joseph A. Akinyele, Christina Garman, Ian Miers, Matthew W. Pagano, Michael Rushanan, Matthew Green, and Aviel D. Rubin. Charm: a framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering*, 3(2):111–128, 2013.

[70] Joseph A Akinyele, Christina Garman, Ian Miers, Matthew W Pagano, Michael Rushanan, Matthew Green, and Aviel D Rubin. Charm: a framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering*, 3(2):111–128, 2013.

[71] Min-Gyue Jung, Seon-A Youn, Jayon Bae, and Yong-Lak Choi. A study on data input and output performance comparison of mongodb and postgresql in the big data environment. In *2015 8th International Conference on Database Theory and Application (DTA)*, pages 14–17. IEEE, 2015.

M Ridwanur Rahman was born in Dhaka, Bangladesh in 1992. He received B.S degree in computer science from North South University in Dhaka, Bangladesh in 2016 and his M.S degree in Information Technology from Monash University, Australia in 2020.

His current research interests include big data, network security, access control models and software engineering.

Ahmad Salehi Shahraki (S'13 and M'17) received his MSc degree in Information Security from the Faculty of Computing, UTM in 2013 and M.Phil. degree in Information Security from the Science and Engineering Faculty, QUT, Brisbane, Australia in 2017. He received his PhD from Monash University, Melbourne, Australia and the DSS Group at CSIRO, Data61 in Melbourne, Australia in 2020. He held an associate lecturer position in 2013 and also a RA from the UM in 2013. Ahmad currently holds a position as researcher at Cybersecurity Lab and Blockchain Technology Centre in the Faculty of Information Technology at Monash University. His research interests include information security in wireless technologies, Access Control, Healthcare and cryptography.

Carsten Rudolph is an Associate Professor in the Faculty of IT at Monash University where he head of the Department of Software Systems and Cybersecurity (SSC). Further, he is the Deputy Director of Blockchain Technology Centre at Monash University and Director of the Oceania Cyber Security Centre (OCSC) in Melbourne, Australia. His research concentrates on information security, formal methods, security engineering and cryptographic protocols. Results of his research work have been applied in areas such as critical infrastructures, industry control systems, or certified systems. He received his PhD from QUT in 2001. Before joining Monash University, he was Head of the Research Group on Trust and Compliance at the Fraunhofer Institute for Secure IT, Darmstadt, Germany and supported Huawei in setting up a Trusted Computing Research Lab in Germany.