Journal of Computer Science

Original Research Paper

A Federative Access Control in a Cloud Environment with a Publish/Subscribe Protocol

[1]Abdelali Saidi, [2]Khalid Aissaoui and [2]Abdesamad Mektoubi

[1]Department of Mathematics and Computer Science, High School of Technology, El Jadida, Morocco
[2]Department of Computer Science, High School of Technology, Casablanca, Morocco

Article history
Received: 01-06-2022
Revised: 15-07-2022
Accepted: 28-09-2022

Corresponding Author:
Abdelali Saidi
Department of Mathematics and Computer Science, High School of Technology, El Jadida, Morocco
Email: saidi.a@ucd.ac.ma

Abstract: Cloud computing is a new concept that has interesting advantages; high computing performances provided as needed, much lower cost than in-house infrastructures, and better reliability and scalability. However, with some drawbacks that are impeding its adoption, security is so far the most alarming concern for companies or organizations. Access control policies, for example, are the most challenging issue that has been tackled these last years; to ensure that the right user is accessing the right resources in such a distributed, virtual and scalable environment handled by a third party, to manage an access request to a shared resource from heterogeneous entities following different policies. In this study, the authors aim to overcome this issue by implementing an access control architecture enabling access to the shared data for collaborating organizations. They propose a model supplied with an ontology database describing the whole environment to control access to a publish/subscribe messaging protocol; which capitalizes on the advantages provided by innovative techniques such as Semantic Web technologies and the publish/subscribe protocol. Semantic Web technologies provide dynamism and scalability for this model thanks to its SWRL inference engine while the publish/subscribe protocol, in this case, MQTT, which is a light-coupling protocol simplifies the traffic between the different actors involved.

Keywords: Publish/Subscribe, Cloud Computing, Privacy, ABAC, XACML, Web Ontology, Access Control, PKI Certificate, Federative Policy, Cryptography

Introduction

Cloud computing is considered one of the most interesting and growing concepts in Information Technology (IT) these last few years. The NIST (Mell and Grance, 2011) defined cloud computing as "a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction". Cloud computing offers many advantages that encourage companies and organizations to embrace it; leasing resources from a virtual and unlimited pool, paying only for the resources used (a pay-as-you-go billing) and reducing considerably costs (such as new devices, more techs, and engineers, maintenance, etc.), reaching data from anywhere at any time using a simple internet connection. Those are the obvious benefits for Cloud services consumers.

Cloud environments have particular threats and issues specific to virtualized and multi-tenant platforms, which need to be addressed with proper methodologies. The trickiest issue is related to data security, which requires more attention. Security is a major concern for organizations and businesses that are interested in cloud services. Data owners and cloud users need to be sure that the system is tightly protected; their sensitive data and the services provided need to be secure.

In a cloud environment, sensitive data is stored on shared remote servers managed by a Cloud Service Provider (CSP) who has to guarantee the confidentiality property (Wang *et al*., 2009). Besides, users' requests also represent a source of information that can be exploited by intruders (the number of requests, their types, dates, etc.) and need to be protected as well. Authentication, as well, is an important step that allows the provider or the IT manager to identify users to enable or not access the system. Furthermore, to guarantee data integrity, the owner has to protect files from any modification or

deletion that could happen (accidentally or deliberately). Finally, users who need access to stored data in a cloud environment have different profiles and roles; they are employees, customers, or even suppliers. Hence, the system requires an access control model to ensure that the end user is allowed to perform a particular action on a specific resource. The latter issue is even more concerning because the information in such environments may be shared among different entities with various levels of sensitivity. Additionally, virtualization and pooling, which are the main techniques used by providers, bring their concerns, like data leakage (Lombardi and Di Pietro, 2011).

In this article, the authors will provide an overview of the access control models found in the literature. They will propose an approach to improve the access control system regarding the specificities of the Cloud environment. First, the authors provide an overview of the traditional access control models and those specially used for cloud platforms. Then, they present a summary of semantic access control models. The next section sets out the proposed solution and the implementation is detailed. Finally, a conclusion discusses the main features developed within this study.

Access Control Requirements for Cloud

To assess admission to services and resources by a user, an IT manager should set up an access control system which is a collection of modules and techniques that evaluate an access request using a set of policies based on permissions and restrictions (Wang *et al*., 2008). The main goal is to determine whether a user or group of users can perform specific actions on a particular resource. Many models, technologies, and architectures are used to implement access control (Park and Sandhu, 2002). In this section, the researchers first introduce some conventional models, then they expose the cloud environment specifications for a tailored Access Control (AC) model.

Security policy administrators mostly use three main models (Ubale Swapnaja *et al*., 2014). Discretionary Access Control (DAC) is a widely used model in which the owner of the information can grant or deny access to his data using an access matrix (Downs *et al*., 1985). In turn, Mandatory Access Control (MAC) implies that the creator of data does not manage access rights. A system administrator is, instead, responsible for defining and restricting the privileges over the objects (Jiang *et al*., 2004). On the other hand, the Role Based Access Control (RBAC) is more fitted to structured systems and hierarchical organizations. The role of a user is the basis of the access policy (Sandhu *et al*., 1996). In this latter model, a set of rights and permissions is assigned to a particular role in the entity regardless of the identity of the end user. Thereby, the policy does not apply directly to users or groups of users.

Cloud computing is, as described above, a shared virtual environment that has its requirements regarding access control and identity management. Many characteristics should be taken into consideration to strengthen the system security while maintaining the efficient functioning of a cloud environment solution. The list below summarizes some important requirements:

1. Dynamism and scalability: Basically, the number of users in a cloud solution is significant and permanently changing. Besides, the system is capable of handling a growing amount of work and data along with creating or deleting VMs that respond to real-time demand. The implemented AC model has to address this issue and ensure that the system performances are not deteriorating (Wang, 2011)
2. Complexity and response time: Customers are expecting a given level of QoS that the providers guarantee. However, updating, handling access requests, and applying policies are overwhelming operations, especially in a virtual and distributed environment. The computational complexity in an AC system becomes even more cumbersome given the high number of end users and the huge number of requests (Hu *et al*., 2006)
3. Heterogeneity: Architectures and technologies employed in the Cloud are differing. This may yield tremendous challenges when dealing with diverse policies and mechanisms of access control. Most models are not applicable in the case of heterogeneous systems, which requires a semantic-based approach to cope with such configurations, all the more so in a large and dynamic environment (Crago *et al*., 2011)
4. Interoperability: Adopting a virtual and scalable environment such as the cloud becomes more interesting as a customer can choose between several providers' solutions consistent with his needs. Besides, those providers may collaborate and eventually share data for better achievement. Nevertheless, this valuable advantage raises many questions about conflicts between access control models or policies, which may impede the efficiency of any integration or contributing resources (Patil *et al*., 2007)
5. Resource pooling: Sharing the same physical resource brings the multi-tenancy issue to the table. Indeed, in the cloud environment, resource pooling consists of serving multiple customers or tenants with scalable services on virtual platforms in a transparent way regarding the end user (Almutairi *et al*., 2011). However, there is a risk of interference and access policy flaws among tenants, which can be extremely damaging

6. Policy management: The access control model put in place should flexibly facilitate a policy management system. The system implemented should be capable of updating, creating, or deleting policies, roles, or even user attributes. Furthermore, it is interesting to be able to resolve the conflict between different policies and rules or combine them to retrieve a more appropriate decision. Besides, there is an increasing interest to settle collaborative cloud solutions among different organizations. In this case, considering that those organizations are likely to be heterogeneous and to hold different structures, the model proposed has to ensure a sort of synergy between all the policies and particularities to deliver proper responses to access requests (Hamlen *et al*., 2010)

Several access control models are used to secure access to data in in-home or outsourced environments. Only a few of them have been dedicated to cloud computing (Meghanathan, 2013). Consequently, such particular requirements need new approaches freed from the constraints encountered in conventional models. In the next section, an overview of the access control models based on a semantic description of the entities involved in the process will be given.

Materials and Methods

To implement the solution, the authors have conceived a platform bringing together several organizations willing to share sensitive data. These organizations may each have a different access control policy, but wish to collaborate on common projects. To achieve this collaboration successfully, there is a need to federate these different policies. A Trusted Entity (TE) handles the access control management system as well as the distribution of certificates and keys to supply the necessary cryptographic protocols. Figure 1 shows the main architecture.

For instance, when a user from organization B needs to access sensitive data created by another user from organization A, he must submit his request to the TE. To grant or revoke access, the TE will rely on the access policies located within its servers.

The Publish/Subscribe Protocol

In addition to the confidential nature of the exchanges, the flow of information between these different elements needs to be fluid and light. With this in mind, the authors have chosen to implement the communication with a publish/subscribe protocol (Fig. 2) (Fidler *et al*., 2005). It describes the exchanges without prior knowledge of the different actors involved. Unlike traditional models that require IP addresses, "Pub/Sub" avoids any coupling between actors. This offers more dynamism, scalability, and flexibility. There are three essential components in a "Pub/Sub" model: The publisher who sends messages, a server (namely the Broker) that handles and switches messages, and finally the subscriber, which remains constant and exclusively listens to the messages related to him. The publisher tags its messages with a label representing a specific topic that the server distributes to all users subscribed to this topic.

Previous work was done to ensure a key distribution model with the publish/subscribe protocol (Mektoubi *et al*., 2017). In this case, the "Pub/Sub" protocol is used to ensure that data and cryptographic entities are fluidly shared among users or subscribers without prior knowledge of the network.

The XACML Standard

The trusted entity that will oversee access control requires a mechanism that can guarantee scalability and dynamism. This will be provided by XACML (eXtensible Access Control Markup Language), which is an XML-based policy language that can implement access control models, such as ABAC in this case. It defines a control policy language based on attributes, architecture, and processing model describing how to evaluate access requests according to policies' rules. This is done using an XML schema that shows authorization policies and describes the rules for accessing resources (Standard, 2013).

The XACML architecture is comprised of four main elements depicted in Fig. 3:

- Policy Administration Point (PAP): Manages access authorization policies
- Policy Decision Point (PDP): Evaluates access requests against authorization policies before issuing access decisions
- Policy Enforcement Point (PEP): Intercepts user's access request to a resource, makes a decision request to the PDP to obtain the access decision
- Policy Information Point (PIP): Acts as a source of attribute values (resource, subject, and environment)

The Semantic Web Technologies

The Semantic Web is a domain that aims to improve the rendering of Internet technologies. This concept improves the network of hyperlinks between classical web pages through a network of structured data links by recognizing their meanings, allowing computing agents to access different data sources on the web more intelligently and, in doing so, accomplish more precise tasks for users. To fulfill its goal, the Semantic Web involves a set of technologies that enable the collection, structuring, and retrieval of data:

- Ontology: Addresses the issue of knowledge representation of information. The creation of an ontology is done through the OWL language which allows for the definition of more or less complex associations of resources and the properties of their

classes. The OWL language is based on the RDF graph model, which is used to formally describe Web resources and their metadata

- SPARQL: This is a query language that queries RDF files to extract machine-readable metadata. It is for the Semantic Web what SQL is for databases
- SWRL: (Semantic Web Rule Language) is a language that allows the use of inference rules in ontologies to deduce new associations or relationships which could then be integrated into the initial ontology

These languages will allow us to describe the system's components flexibly, extract the attributes evaluated at the access policy level, and, in a second step, compute by inference new relationships to feed the control system.

Related Works

A Cloud federation is a collaboration of organizations sharing data hosted on their private cloud infrastructures. Its adoption is impeded by some concerns, especially between competitive organizations. Blockchain technology is widely used in recent works to allow access to privacy-enhanced data (Ra *et al*., 2021; Zhang *et al*., 2019; Wang *et al*., 2018; Sabzmakan and Mirtaheri, 2021). Ra *et al*. (2021) proposes a labeled data access system with cloud outsourcing and a keyword search protocol with data linking token. Zhang *et al*. (2019; Wang *et al*., 2018) present a secure scheme against keyword guessing attacks, where users encrypt keywords using a dedicated key server. Sabzmakan and Mirtaheri (2021) propose a distributed role-based access control model to enable the management of resources and the parties' access securely (Sabzmakan and Mirtaheri, 2021); what is interesting here is that the roles are determined according to the organization's contribution' collaborative project.

In semantic web technologies, data are defined, linked, and put on the network in a form that computers can easily understand and process with more efficiency. The semantic web includes machine-readable files, in addition to the conventional web structure. It aims at automation, integration, and reuse of data among different web applications (Berners-Lee and Hendler, 2001). This opens new possibilities for cloud-like environments. Rezgui *et al*. (2017) propose an ontological framework to provide a unifying semantic foundation for describing competencies and their related details within the contexts of technology-enhanced competency-based learning and training.

In cloud computing, semantic web technologies focused on different aspects as mentioned in (Brabra *et al*., 2016). Security, as well, was addressed in some works to reveal how the implementation of those techniques can improve it (Ait Idar *et al*., 2018). Many works, interested in adapting access control models to the cloud's requirements, reveal that flaws are likely to occur when conventional solutions (e.g., MAC, DAC, RBAC, etc.) are used. This is due, in part, to a lack of considering interrelationships among access control entities.

To overcome this issue, Auxilia and Raja proposed a Semantic-Based Access Control model (Auxilia and Raja, 2016; 2012). The SBAC model includes three modules: The Ontology Base, which is a collection of ontologies (Subject Ontology, Object Ontology, and Action Ontology), designed with an OWL ontology language, Authorization Base which is a collection of authorization rules made with SWRL language and operations that are executed on Authorization Base to grant or revoke access to objects. Furthermore, an inference engine can automate the decision. The Cloud Service Provider (CSP) is involved in the process since he sends an access request to the access system and returns the decision to users. XACML is widely used to implement policy rules in cloud computing due to its compatibility with heterogonous platforms. Sun *et al*. (2012), for their part, extended the RBAC model by using semantic web technologies. They introduce the Semantic Access Control model (SAC) which authenticates users on an ontologies basis. Those ontologies include subjects, objects, actions, and attributes-. They used purpose, condition, and right in the usual 3-tuple rule (subject, object, and action) and implemented it with XACML and SPL language. Choi *et al*. (2014) proposed an Ontology-based Access Control Model (Onto-ACM) that contains two main components: The Context analysis engine that gathers context-aware information and sends requests. The Access Control Module, meanwhile, responds to these requests and delivers decisions based on specified policies and uses, when appropriate, the Jena inference engine. Hu *et al*. (2006) used XACML to propose an ontology-based Semantic Access Control Policy Language (SACPL) (Hu *et al*., 2009). In this approach, subject, object, action and attribute items are annotated with semantic information using the Access Control Oriented Ontology System (ACOOS), which is modeled through the OWL-DL language. They aimed to address the interoperability issue between distributed ACPs in cloud computing environments. Similar work is described in (Priebe *et al*., 2006) where XACML architecture is extended with the use of Semantic Web techniques. The procedure is based on OWL language to create ontologies, SWRL language to define inference rules, SPARQL language to define queries, and RDF to describe the resources. Sifou *et al*. (2020) try to prevent all unauthorized access to remote resources using XACML; the implementation of the proposed solution is based on the abac model and the access decision is based on

attributes like user identity, resource, environment, and a set of policies. Abd El-Aziz (2019) uses the XACML framework to protect encrypted metadata from unauthorized users. Their approach controls access by generating a security fingerprint for user authentication. Alruwaili *et al*. (2021) propose a model that controls access to encrypted data by identifying whether the user who's asking for data, is authorized or not dependent on his/her attributes stored in the XACML policy.

Nowadays, IT solutions are increasingly embedding collaborative applications (e.g., remote conferencing, document sharing, groupware, etc.). Largely, this involves entities with heterogeneous structures and even different policies, which is most likely to occur in a cloud environment. This requires stronger access control and raises serious trust concerns.

This contribution aims at implementing an access control system spanning many heterogeneous entities. The approach will include an Attribute Based Access Control (ABAC) model supplied with ontological bases. Besides, the researchers will add a publish/subscribe protocol, as a light messaging pattern, to provide greater scalability and dynamic topology. The next section describes the solution in detail.

The Proposed Approach

In concrete terms, there are three main actors:

- The Trusted Entity (TE): Will manage the cryptographic part. It provides and stores the necessary certificates and keys, hosts predefined access policies, and evaluates access requests. Consequently, it delivers the decision to the requesters
- Organization A (Org_A): This is an independent functional entity with its access policies and occasionally collaborates with other organizations on joint projects, potentially sharing sensitive data. It has its in-house organizational structure and manages a certain number of users, including U_{A1}, who will participate in this scenario
- Organization B (Org_B): As well as organization A, the user U_{B1} represents Org_B in the interactions. The latter will exchange messages with the two other actors which are *TE* and U_{A1}

This system is set up with an initial configuration to secure the first communications. In addition to its private key, the TE holds public key certificates to authenticate all the participants as well as topics. The aim is, of course, to be able to grant the topic's decryption key to authorized partners. Figure 4 shows initial arrangements, considering that the user U_{A1} will initiate the Topic that contains sensitive data, which is noted as T1.

To better help understand the algorithms below, Table 1 describes several notations used.

Table 1: Notations

Notation	Description
TE	Trusted Entity
Org_A	Organization A
Org_B	Organization B
U_{A1}	User 1 in the organization A
U_{B1}	User 1 in the organization B
U	Any user
C_i	i certificate
PU_i	i public key
PR_i	i private key
TT	Target Topic
DT	Demand Topic
MT	Message Type
RTx	Response Topic number x
AT	Authorization Topic
EK	Encryption
Sig	Digital Signature

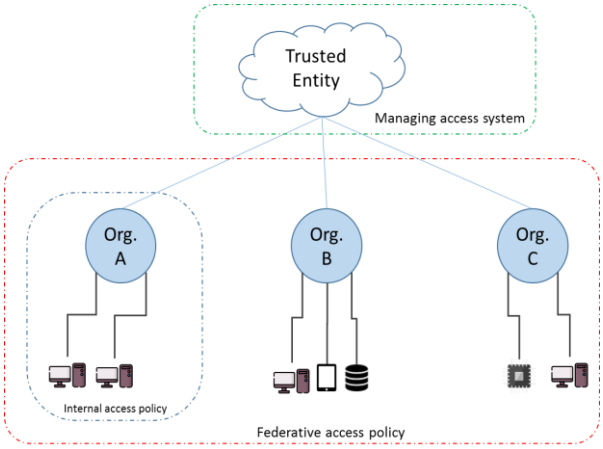

Fig. 1: SEQ Figure*ARABIC 1

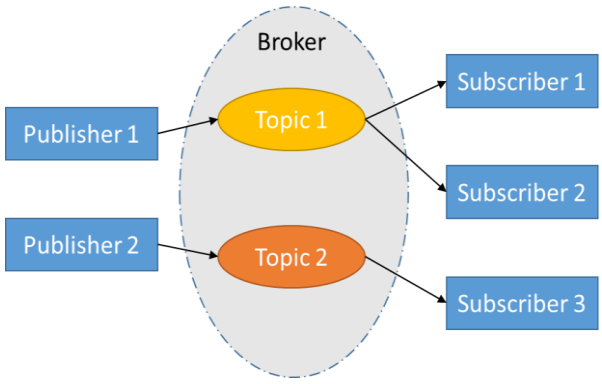

Fig. 2: SEQ Figure*ARABIC 2

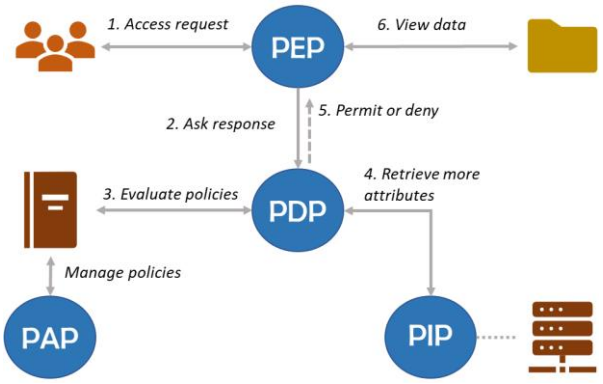

Fig. 3: SEQ Figure*ARABIC 3

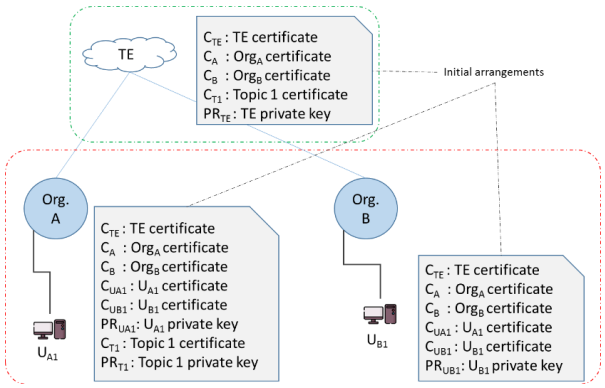

Fig. 4: SEQ Figure*ARABIC 4. Initial

The user U_{A1} sends sensitive data on the appropriate topic *TT*. This data will be encrypted with the *TT* secret key PR_T. Users interested in accessing the protected content of these messages must obtain the key PR_T in compliance with the rules and policies predefined and stored within the *TE*.

The recovery process is performed by running the six algorithms below. The researchers use both cryptographic protocols (encryption, decryption, and signature) as well as the publish/subscribe protocol to ensure lightweight, secure and dynamic communication.

To better describe the model, the researchers use a message–arrow formalism. For instance, sending a message *M* to Topic *T* by a user U_1 using a Broker *B* and considering a user U_2 as a subscriber, will be modeled by the following:

$$U_1 \rightarrow B : T, M \; B \rightarrow U_2 : T, M$$

More simply, by skipping the exchanges with the broker *B*:

$$U_1 \rightarrow^T U_2 : M$$

Each message sent consists of two parts: The first part is a header describing the type of the message; the second part contains the message body.

Retrieving the Topic's Certificate

One who is interested in a particular topic need first to obtain the topic's digital certificate. This will allow him to authenticate the topic. For instance, the user U_{B1} publishes a request on the topic *DT*:

$$U_{B1} \rightarrow^{DT} U : MT, TT, RT1$$

DT: The topic on which all requests are sent. By default, all users are subscribed to *DT*
MT: The message is a tag describing the purpose of the message
*RT*1: The topic on which to answer

The algorithm "*TT* Certificate request", which is described below, explains how to request the topic's certificate.

Algorithm: TT Certificate request

```
Input: DT, TT output:
BEGIN
    MT ← getTypeMessageCertificateRequest()
    RT1 ← generateRandomTopic()
    subscribe(RT1)
    M ← {M, TT, RT1}
    publish(DT, M)
END
```

M: The message sent on the topic *DT*.
MT: "certificate request" in this step.
subscribe (RT1): U_{B1} needs to prior subscribe to *RT*1 to get the response.
publish (DT, M): sends the message *M* on the topic *DT*.

Any user with the *TT*'s certificate can respond to this request on topic *RT*1. Since the certificate is a public document, this step requires neither the intervention of the TE nor the use of the access policy.

Sending TT Certificate

The second algorithm "*TT* Certificate response" allows any user to respond to the first request by publishing the *TT* certificate. Only users holding the certificate can send it.

$$U \rightarrow^{RT1} U_{B1} : MT, TT, C_T$$

C_T : The TT certificate

Algorithm TT Certificate response

```
Input: MT, TT, RT1 output:
BEGIN
    isCertExist ← verifyCertificateTopicExist(TT)
```

```
        IF isCertExist == true THEN
                C_T ← getTopicCertificate(TT)
        MT←getTypeMessageCertificateResponse()
                M ← {MT, TT, C_T}
                publish(RT1, M)
         END
END
```

M : The message sent on the topic *RT*1
MT : "Certificate response" in this step
publish (*RT*1, *M*) : Sends the message *M* on the topic *RT*1

If a user has the certificate of the topic *TT*, he can send it on the topic *RT*1. From that point on, the user $U_{B1,}$ can retrieve the TT certificate on the topic *RT*1, since he is a subscriber to *RT*1.

Authorization Request

In the third step, namely the "*TT* authorization request" algorithm, U_{B1} will be able to request the access right to the content of the topic *TT*. To this aim, he sends the demand to the topic *AT* of which the TE is the only subscriber.

$$U_{B1} \rightarrow^{AT} TE : MT, EK\left(\left\{TT, C_{UB1}, RT2\right\}, PU_{TE}\right)$$

C_{UB1} : U_{B1}'s certificate
EK : the encryption algorithm
RT2 : the topic to be answered on
PU_{TE} : TE's public key

Algorithm TT authorization request

```
Input: MT, TT, C_T output:
BEGIN
        PU_TE ← getPublicKeyTE()
        RT2 ← generateRandomTopic()
        C_UB1 ← getUserCertificat()
        EM ← encrypt({TT, C_UB1, RT2}, PU_TE)
        subscribe(RT2)
        MT ← getTypeMessageAuthorizationReq()
        M ← {MT, EM}
        Publish(AT, M)
END
```

EM : Results from the encryption of TT, C_{UB1} and RT2 with the key PU_{TE}.
MT : "Authorization request" in this step.

For the first time, the encryption is integrated with TE's public key (PU_{TE}) to ensure the confidentiality of the message. Only the TE should process this request and therefore decrypt the message.

Responding to Request

The fourth algorithm, "Response to authorization request", includes checking the rules of the access policy, by questioning the XACML scheme to grant or not access to the content of the TT topic:

$$TE \rightarrow^{RT2} U_{B1} : MT, EK$$
$$\left(sign\left(\left\{EK\left(Resp, PU_T\right), TT, C_{UB1}\right\}, PR_{TE}\right), PU_{UB1}\right)$$

Sign : Signature algorithm
Resp : Response (allow or deny)
PU_T : TT public key
PR_{TE} : TE private key
PU_{UB1} : U_{B1} public key

Algorithm Response to the authorization request

```
Input: MT, TT, C_UB1, RT2 output:
       BEGIN
       PR_TE ← getPrivateKeyTE()
       {TT, C_UB1, RT2} ← decrypt({TT, C_UB1, RT2},
       PR_TE)
       isUserOk ← verifyUser(C_UB1, TT)
       IF isUserOk == true THEN
       Resp ← allow
       ELSE
       Resp ← deny
       END
       PU_T ← getPublicKeyOfTopic(TT)
       RespEn ← encrypt(Resp, PU_T)
       PU_TE ← getPublicKeyTE()
       SigM ← sign({RespEn, TT, C_UB1}, PR_TE)
       PU_UB1 ← getPublicKeyFromCertificat(C_UB1)
       MEn ← encrypt(SigM, PU_UB1)
       MT ← getTypeMessageAuthorizationResponse()
       M ← {MT, MEn}
       publish(RT2, M)
END
```

RespEn : Response in encrypted form
SigM : Signature of the response, TT and CU_{B1} using PR_{TE}
Men : SigM encrypted with PU_{UB1}
MT : "Authorization response" in this step
verifyUser(C_{UB1}, TT) : Interrogates the access policy

Several layers of encryption and signature are used to guarantee the confidentiality and authenticity of these exchanges. Specifically, the encryption of the response Resp by PU_T is intended to restrict its disclosure to users who actually hold the secret key of *TT* and can therefore process the message. The signature layer proves the origin of the response. Finally, the last encryption is done with the public key of U_{B1}, since he is the requester, for protection purposes. Of course, the response depends on the access policy implemented in XACML, which will be described below in more detail.

Requesting the Secret Key

Once U_{B1} has received the encrypted version of the response, he will forward it to the concerned users to request the secret key of the topic *TT* as detailed in the algorithm: "Transmitting the response".

$$U_{B1} \rightarrow^{DT} U : MT, EK\left(\{SRS, RT3\}, PU_T\right)$$

SRS : The response in an encrypted form
RT3 : U_{B1} specifies the topic to be answered on, here RT3

Algorithm: Transmitting the response

Input: MT, Resp output:
BEGIN
PR$_{UB1}$ ← getUserPrivateKey()
RS ← decrypt(EK(sign({EK(Resp, PU$_T$), TT, C$_{UB1}$}, PR$_{TE}$), PU$_{UB1}$), PR$_{UB1}$)
SRS ← sign(RS, PR$_{UB1}$)
RT3 ← generateRandomTopic()
subscribe(RT3)
EM ← encrypt({SRS, RT3}, PU$_T$)
MT ← getTypeMessagePrivateKeyRequest()
M ← {MT, EM}
publish(DT, M)
END

PR_{UB1} : U_{B1} private key
RS : Output from decryption with PR_{UB1}
SRS : Output from signature with PR_{UB1}
EM : Output from encryption with PU_T
MT : "Private key request" in this step

At this point, the requester U_{B1} decrypts the previously received response before signing the package with his private key to authenticate himself. Ultimately, he publishes the whole to the topic DT to obtain PR_T and disclose the content on the to Ypic TT.

Getting the Private Key

Finally, the algorithm "Publishing *TT* private key" explains how a user in possession of the secret key will be able to publish the last message. Of course, this operation is run when it is per the access rules as formulated in *TE* repositories:

$$U \rightarrow^{RT3} U_{B1} : MT, EK\left(PR_T, PU_{UBI}\right)$$

PR_T : Private key of the target topic *TT*
U : A user holding PR_T

Algorithm: Publishing TT private key

Input: MT, EK({SRS, RT3}, PU$_T$) output:
BEGIN
PR$_T$ ← getSecretKeyOfTopic(TT)
DM ← decrypt(EK({SRS, RT3}, PU$_T$), PR$_T$)
isSignedByClient ← verifySign(SRS, C$_{UB1}$)
IF isSigneByClient == true THEN
isSigneByTE ← verifySign(RS, C$_{TE}$)
IF isSigneByTE == true THEN
TEResponse←getResponse(RS,PR$_T$)
IF TEResponse == allow THEN
EM←encrypt(PR$_T$, PU$_{UB1}$)
MT←getTypeMessageRes
ponseKeyRequest()
M←{MT, EM}
publish(RT3, M)
END IF
END IF
END IF
END

getResponse(RS, PR$_T$): Verify the access right.
MT: "Private key sending" in this step.

Through the cryptographic protocols used, all measures are taken to ensure the authenticity of the parties involved and the exchanged data. Finally, since the user U_{B1} is a subscriber to the topic RT3, he will be able to access the PR_T key.

As discussed above, the program only returns the response after questioning the access policy implemented with XACML, to either grant or deny access and therefore send or not send the TT's private key PR_T. For instance, Fig. 5 shows some features of the user access policy.

As explained, the XACML's PIP module allows extracting the necessary attributes to evaluate access requests. The source of these attributes is the ontology created separately. The ontology describes all the elements' attributes and relationships within the environment in a flexible way. As the activity evolves, the use of the SWRL language makes it through inference to establish new relationships and new attributes. Consequently, this ability is most relevant to the context.

The following section gives an insight into the practical aspect. The platform deployed with algorithms is described and the results are outlined.

Implementation

To implement the solution, the authors design the architecture shown in Fig. 6.

The user U_{A1} (a Raspberry module) collects the measurements transmitted by a Wireless Sensor Network (WSN) and publishes them on the network in an encrypted way. On the other hand, the user U_{B1} is interested in subscribing to the topics that he considers important and wishes to retrieve their decryption keys.

The designed algorithms were developed with Java. Figures 7 and 8 show the class diagrams used in the application:

- Reques Traitement: The main class of the program, is responsible for sending requests as well as processing the responses received
- OnMessageArrivedImpl: Determines the nature of the message and executes the correct java method within the class *Reques Traitement*
- Mqtt Publish Subscribe Impl: Implements the *Publish-Subscribe* interface for the Message Queuing Telemetry Transport (*MQTT*) protocol

Figure 8 shows the diagram modeling the classes:

- Request PK implements the public key request or otherwise the topic certificate
- Response Request PK contains the response to the previous request
- Request Autorisation SK processes the sending of the authorization request
- Response Request Autorisation contains the response to the request for authorization and questions the security policy
- Request SK*:* Implements the private key request of the required topic
- Response Request PK*:* Creates the message containing the private key of the topic
- Message: Instantiates the various types of messages exchanged and handles the publish/subscribe part

The authors have installed an MQTT client to view the different exchanges conveniently. For this purpose, they have chosen "*MQTT Box*" which is a client with a user-friendly and ergonomic interface. It allows the user to subscribe and/or publish on topics.

Figure 9 shows the interface used to publish a message by choosing the topic concerned. Figure 10, on the other hand, represents a topic to which the MQTT client is subscribed. The details of the messages published on these topics are also described in it.

To facilitate the management of the publish/subscribe protocol; The authors have created administration topics as follows:

- Control topics: Send commands to perform specific actions based on the progress in the protocol exchanges
- Log topics: Track the history of events; allow monitoring the traffic

Displaying log topics allows us, among other things, to verify that sensitive messages are indeed encrypted and only disclosed to authorized users according to the access policy.

Additionally, the "Key Store Explorer" is a graphical tool used to manage the key store (i.e., a repository for the cryptographic entities used). It allows the visualization of the certificates and the keys held by the different actors at a given time. In this case, the Key Store Explorer allows verification that an authorized user has retrieved the private key of the requested topic. For example, Fig. 11 shows the cryptographic elements held by the TE, namely its certificate, its private key, and the respective certificates of the organizations that are part of the federated system (Org A and Org B).

As the exchanges evolve, the progress of each actor's *Key Store* content can be followed. Therefore, it is possible to check that the access control system is working properly. For instance, if the requesting user fulfills the necessary conditions as stipulated in the access policy, he will receive the secret key of the topic PR_T, which must appear, in his *Keystore* as shown in Fig. 12.

```
<Subject>
    <SubjectMatch MatchId="...:rfc822Name-match">
      <AttributeValue DataType="...#string">user2</AttributeValue>
      <SubjectAttributeDesignator AttributeId="...:subject:subject-id"
                                  DataType="...:rfc822Name"/>
    </SubjectMatch>
 </Subject>
```

```
<Resource>
  <ResourceMatch MatchId="...:rfc822Name-match">
    <AttributeValue DataType="...#string">topic1</AttributeValue>
    <ResourceAttributeDesignator AttributeId="...:resource:resource-id"
                                 DataType="...#string"/>
  </ResourceMatch>
</Resource>
```

```
<Rule RuleId="Topic1AccessRule" Effect="Permit">
    <Condition FunctionId="...:function:string-equal">
        <Apply FunctionId="...:function:string-one-and-only">
            <SubjectAttributeDesignator AttributeId="organisation"
                                        DataType="...#string"/>
        </Apply>
        <AttributeValue DataType="...#string">OrganisationA</AttributeValue>
    </Condition>
</Rule>
```

Fig. 5: Access policy XML

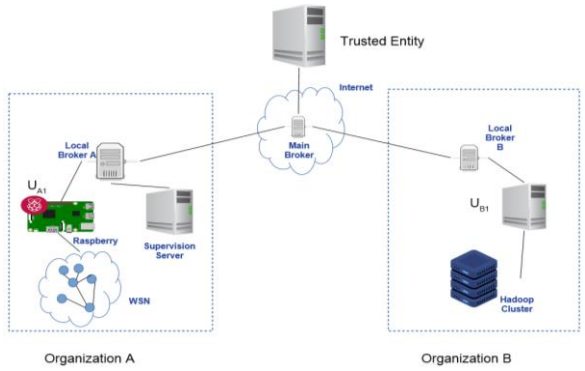

Fig. 6: the main architecture

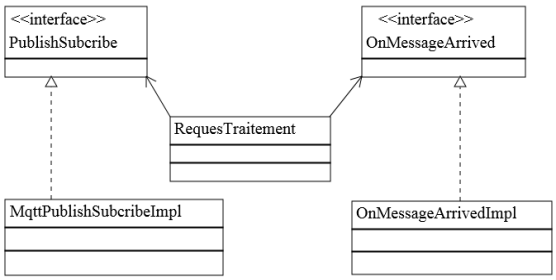

Fig. 7: Class DIAGRAM Part 1

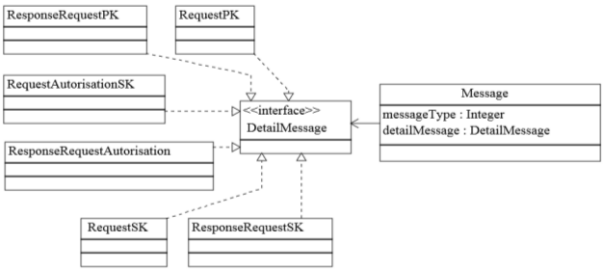

Fig. 8: Class diagram part 2

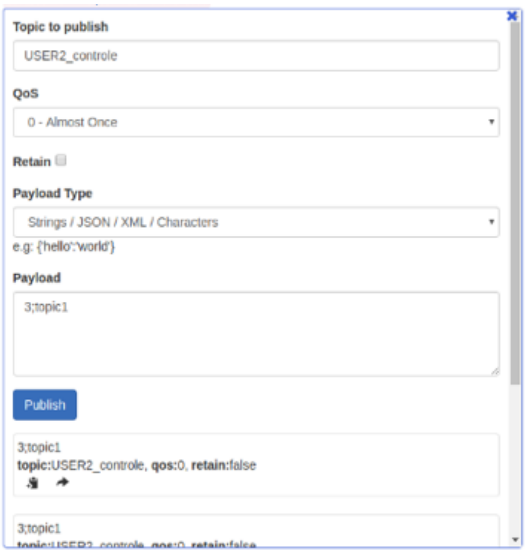

Fig. 9: Publishing interface

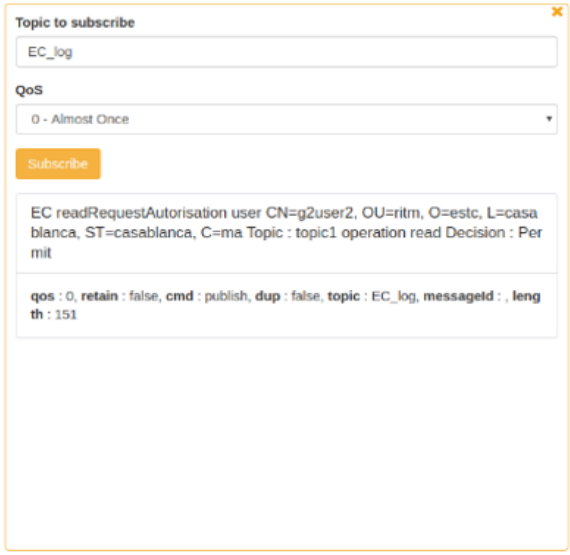

Fig. 10: Subscribing interface

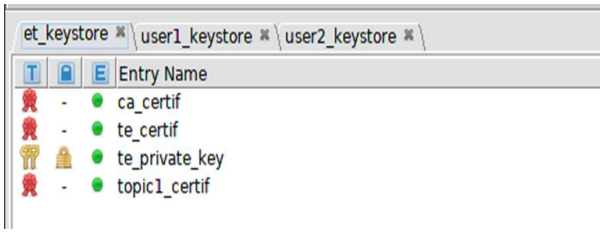

Fig. 11: TE (trusted entity) key store

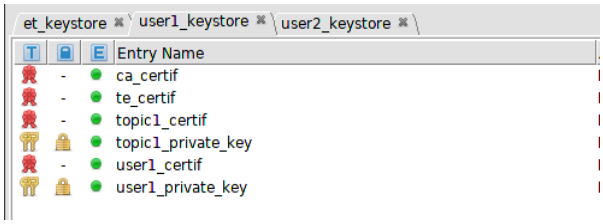

Fig. 12: The requester's keystore

Conclusion

In this article, the researchers proposed an access control solution that focuses on the collaborative aspect that is often found in Cloud services. To do so, they have implemented a federative approach based on the ABAC model. However, although the implementation of the model has proved the practicability of the solution, a kind of "proof of concept", there are still some points that need to be addressed.

First, the dynamic aspect of the approach has not been further developed since the inference engine is not used. It would allow new relationships between the users and thus enrich the access policy. To set up this part and be able to test it properly, it would be necessary to create an

ontology that describes a larger context with higher granularity and a wide range of attributes and relationships. This is part of future improvement work. Another challenge, which seems interesting, is to measure the resilience of the publish/subscribe protocol to overloaded traffic, which may be the case in cloud environments and more specifically Big Data applications. The protocol has been designed in such a way that any user with the desired information can respond to a request from a subscriber. This risks considerably overloading the traffic and needs to be addressed.

In this article, the authors have addressed the issue of access control in the cloud environment. They were interested in sensitive data shared between several heterogeneous organizations. In this regard, they have designed a federated architecture allowing the secure distribution of encryption keys used to protect messages. In addition, the creation of an ontology has enabled defining the contours and relationships that govern all the actors in the system. Moreover, the publish/subscribe protocol facilitates the exchange of messages. However, improvements are needed to refine the solution and improve its performance.

Acknowledgment

This study has been supported by its own authors.

Author's Contributions

Abdelali Saidi and Abdesamad Mektoubi: Critical revision of the article final approval of the version to be published.

Khalid Aissaoui: Drafting the article critical revision of the article Final approval of the version to be published.

Ethics

The authors confirm that this manuscript has not been published elsewhere and that no ethical issues are involved.

References

Abd El-Aziz, A. A. (2019). An extended data protection model based on cipher-text-policy attribute-based encryption model and an XACML framework in cloud computing. *Int. J. Adv. Sci. Technol.*, *28*(16), 1021-1033.

Ait Idar, H., Aissaoui, K., Hicham, B., & Filali Hilali, R. (2018). A Survey on Semantic Access Control in Cloud Computing. *Smart Application and Data Analysis for Smart Cities (SADASC'18)*. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3185338

Almutairi, A., Sarfraz, M., Basalamah, S., Aref, W., & Ghafoor, A. (2011). A distributed access control architecture for cloud computing. *IEEE Software*, *29*(2), 36-44. https://ieeexplore.ieee.org/abstract/document/6095492

Alruwaili, A., El-Aziz, A. A., & Hamdi, H. (2021). An enhanced access control model to encrypted data based on an xacml framework in cloud environment. *Journal of Theoretical and Applied Information Technology*, *99*(8).

Auxilia, M., & Raja, K. (2012, December). A semantic-based access control for ensuring data security in cloud computing. In *2012 International Conference on Radar, Communication and Computing (ICRCC)* (pp. 171-175). IEEE. https://ieeexplore.ieee.org/abstract/document/6450570

Auxilia, M., & Raja, K. (2016). Ontology centric access control mechanism for enabling data protection in cloud. *Indian Journal of Science and Technology*, *9*(23), 1-7.

Berners-Lee, T., & Hendler, J. (2001). Publishing on the semantic web. *Nature*, *410*(6832), 1023-1024. https://www.nature.com/articles/35074206

Brabra, H., Mtibaa, A., Sliman, L., Gaaloul, W., & Gargouri, F. (2016, June). Semantic web technologies in cloud computing: A systematic literature review. In 2016 IEEE International Conference on Services Computing (SCC) (pp. 744-751). IEEE. https://ieeexplore.ieee.org/abstract/document/7557522

Choi, C., Choi, J., & Kim, P. (2014). Ontology-based access control model for security policy reasoning in cloud computing. *The Journal of Supercomputing*, *67*(3), 711-722. https://link.springer.com/article/10.1007/s11227-013-0980-1

Crago, S., Dunn, K., Eads, P., Hochstein, L., Kang, D. I., Kang, M., ... & Walters, J. P. (2011, September). Heterogeneous cloud computing. In *2011 IEEE International Conference on Cluster Computing* (pp. 378-385). IEEE. https://ieeexplore.ieee.org/abstract/document/6061155

Downs, D. D., Rub, J. R., Kung, K. C., & Jordan, C. S. (1985, April). Issues in discretionary access control. In *1985 IEEE Symposium on Security and Privacy* (pp. 208-208). IEEE. https://ieeexplore.ieee.org/abstract/document/6234824

Fidler, E., Jacobsen, H. A., Li, G., & Mankovski, S. (2005). Publish/subscribe system. *Feature Interactions in Telecommunications and Software Systems VIII*, 12. ISBN-10: 9781586035242.

Hamlen, K., Kantarcioglu, M., Khan, L., & Thuraisingham, B. (2010). Security issues for cloud computing. *International Journal of Information Security and Privacy (IJISP)*, *4*(2), 36-48. https://www.igi-global.com/article/security-issues-cloud-computing/46102

Hu, L., Ying, S., Jia, X., & Zhao, K. (2009, December). Towards an approach of semantic access control for cloud computing. In *IEEE International Conference on Cloud Computing* (pp. 145-156). Springer, Berlin, Heidelberg. https://link.springer.com/chapter/10.1007/978-3-642-10665-1_13

Hu, V. C., Kuhn, D. R., & Ferraiolo, D. F. (2006, June). The computational complexity of enforceability validation for generic access control rules. In *IEEE International Conference on Sensor Networks, Ubiquitous and Trustworthy Computing (SUTC'06)* (Vol. 1, pp. 7-pp). IEEE. https://ieeexplore.ieee.org/abstract/document/1636184

Sifou, F., AlShahwan, F., Hammoud, A., Marwan, M., & Hammouch, A. (2020). Implementing Policy Rules in Attributes Based Access Control with XACML within Cloud-Enabled IoT Environment. *J. Commun.*, *15*(1), 107-112.

Jiang, Y., Lin, C., Yin, H., & Tan, Z. (2004, October). Security analysis of mandatory access control model. In *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583)* (Vol. 6, pp. 5013-5018). IEEE. https://ieeexplore.ieee.org/abstract/document/1400987

Lombardi, F., & Di Pietro, R. (2011). Secure virtualization for cloud computing. *Journal of Network and Computer Applications*, *34*(4), 1113-1122. https://doi.org/10.1016/j.jnca.2010.06.008

Meghanathan, N. (2013). Review of access control models for cloud computing. *Computer Science & Information Science*, *3*(1), 77-85.

Mektoubi, A., Malass, O., Belhadaoui, H., & Rifi, M. (2017). New approach for keys distribution througth publish/subscribe protocols. *International Journal of Computer Science and Information Security (IJCSIS)*, *15*(8).

Mell, P., & Grance, T. (2011). The NIST definition of cloud computing. http://faculty.winthrop.edu/domanm/csci411/Handouts/NIST.pdf

Park, J., & Sandhu, R. (2002, June). Towards usage control models: Beyond traditional access control. In *Proceedings of the seventh ACM symposium on Access Control Models and Technologies* (pp. 57-64). https://doi.org/10.1145/507711.507722

Patil, V., Mei, A., & Mancini, L. V. (2007, March). Addressing interoperability issues in access control models. In *Proceedings of the 2nd ACM symposium on Information, computer and communications security* (pp. 389-391). https://doi.org/10.1145/1229285.1229337

Priebe, T., Dobmeier, W., & Kamprath, N. (2006, April). Supporting attribute-based access control with ontologies. In *First International conference on availability, reliability and security (ARES'06)* (pp. 8-pp). IEEE. https://ieeexplore.ieee.org/abstract/document/1625344

Ra, G., Kim, D., Seo, D., & Lee, I. (2021). A federated framework for fine-grained cloud access control for intelligent big data analytic by service providers. *IEEE Access*, *9*, 47084-47095. https://ieeexplore.ieee.org/abstract/document/9382975

Rezgui, K., Mhiri, H., & Ghédira, K. (2017, October). An ontology-based multi-level semantic representation model for learning objects annotation. In *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)* (pp. 1391-1398). IEEE. https://ieeexplore.ieee.org/abstract/document/8308452

Sabzmakan, A., & Mirtaheri, S. L. (2021, March). An Improved Distributed Access Control Model in Cloud Computing by Blockchain. In *2021 26th International Computer Conference, Computer Society of Iran (CSICC)* (pp. 1-4). IEEE. https://ieeexplore.ieee.org/abstract/document/9420586

Sandhu, R. S., Coyne, E. J., Feinstein, H. L., & Youman, C. E. (1996). Role-based access control models. *Computer*, *29*(2), 38-47. https://ieeexplore.ieee.org/abstract/document/485845

Standard, O. A. S. I. S. (2013). extensible access control markup language (xacml) version 3.0. 2011-09-24]. http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf

Sun, L., Wang, H., Yong, J., & Wu, G. (2012, May). Semantic access control for cloud computing based on e-Healthcare. In *Proceedings of the 2012 IEEE 16th international conference on computer supported cooperative work in design (CSCWD)* (pp. 512-518). IEEE. https://ieeexplore.ieee.org/abstract/document/6221866

Ubale Swapnaja, A., Modani Dattatray, G., & Apte Sulabha, S. (2014). Analysis of dac mac rbac access control based models for security. *International Journal of Computer Applications*, *104*(5), 6-13. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.800.5413&rep=rep1&type=pdf

Wang, C., Wang, Q., Ren, K., Lou, W. (2009). Ensuring data storage security in cloud computing. In: 2009 17th International Workshop on Quality of Service, IEEE. pp, 1e9. http://dx.doi.org/10.1109/IWQoS.2009.5201385

Wang, H., Zhang, Y., & Cao, J. (2008). Access control management for ubiquitous computing. *Future Generation Computer Systems*, *24*(8), 870-878. https://doi.org/10.1016/j.future.2007.07.011

Wang, S., Zhang, Y., & Zhang, Y. (2018). A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems. *IEEE Access*, *6*, 38437-38450. https://ieexplore.ieee.org/abstract/document/8400511

Wang, Z. (2011, October). Security and privacy issues within the Cloud Computing. In *2011 International Conference on Computational and Information Sciences* (pp. 175-178). IEEE. https://ieeexplore.ieee.org/abstract/document/6086163

Zhang, Y., Xu, C., Ni, J., Li, H., & Shen, X. S. (2019). Blockchain-assisted public-key encryption with keyword search against keyword guessing attacks for cloud storage. *IEEE Transactions on Cloud Computing*, *9*(4), 1335-1348. https://ieeexplore.ieee.org/abstract/document/8737775