©NIJEP Publications
OPEN ACCESS
ISSN: 2992-5487
NEWPORT INTERNATIONAL JOURNAL OF ENGINEERING AND PHYSICAL SCIENCES (NIJEP)
VOLUME 3 ISSUE 2 2023

Systematic Literature Review: workflow models for intra-organizational and inter-organizational communication

[1]Ezeamasobi Chibuzor, [2]H.C. Inyiama, [3]Godspower I. Akawuku

[1,2,3]Nnamdi Azikiwe Unversity, Awka (NAU).
Email:Chibuzoredith@gmail.com,hc.inyiama@unizik.edu.ng, gi.akawuku@unizik.edu.ng

ABSTRACT

Traditional access control mechanisms focus on centralized systems and implicitly assume that all resources reside in one domain. This serves as a critical limitation for inter-organizational collaboration, which is characteristically decentralized, distributed and heterogeneous. A consequence of the lack of suitable access control mechanisms for inter-organizational collaborative environments is that data owners relinquish all control over data they release. This paper emphasizes using cryptography for access control, and enforcement for addressing the access control concerns for inter-organizational environments. An essential requirement of workflows is the ability to transfer data resources from one entity to another. From a security perspective, current support for electronic data flow is inadequate for inter-organizational workflows, partly due to access control limitations for inter-organizational environments.

Keywords: Architecture model, Inter-organizational, collaborative environments

Security Policies and Access Control

Focus now shifts to information security, and in particular, security policies and access control. This section explores various policy and access control models, from abstract policy models to the traditional access control models and finally some emerging access control technologies. But to begin, first look at the concepts that are fundamental to information security [1].

Security Concepts

Confidentiality deals with the protection of sensitive information. This is typically influenced by rules stating the disclosure restrictions of the protected information. Making data incomprehensible is the most common form of confidentiality. This is achieved by using encryption. The two types of encryption techniques which exist include conventional encryption (based on symmetric encryption algorithms) and public-key encryption (based on asymmetric encryption algorithms). Both types are discussed in the following subsections [2].

OPEN ACCESS
ISSN: 2992-5487

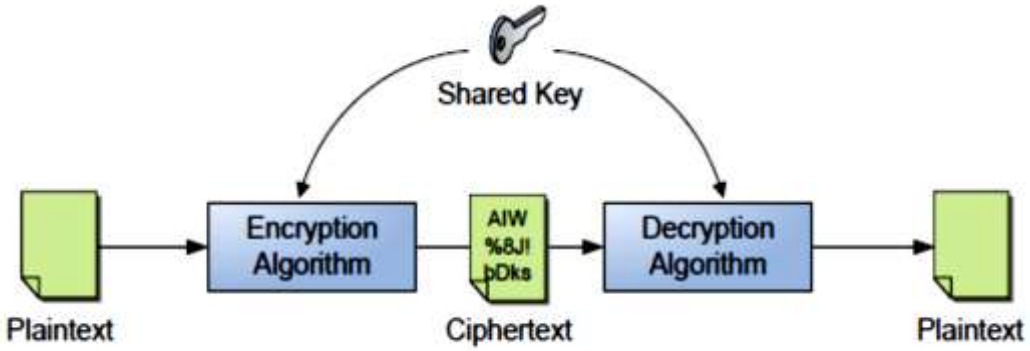

Figure 1.0: The basic model for conventional encryption

Conventional encryption techniques were, for a long time, the only known form of encryption. In fact, these are still the most widely used type today. Symmetric encryption relies on the existence of a shared secret between the message source and the destination [3].

Figure 1.0 illustrates how conventional encryption works. The plaintext (i.e., the message to be protected) and the shared key (i.e., the shared secret) are passed into the encryption algorithm to produce the cipher text (i.e., the message encrypted). The sender sends the cipher text to the recipient. Once received, the recipient passes the cipher text and the shared key into the decryption algorithm to reveal the plaintext. When using symmetric encryption, the ability to derive the plaintext from the cipher text alone must be impractical. Additionally, the security of symmetric encryption techniques must rely on the secrecy of the key – not the encryption algorithm. Knowledge of the encryption algorithm should not provide a security risk [4].

Widely accepted symmetric encryption algorithm standards today are Data Encryption Standard (DES), Triple-DES and Advanced Encryption Standard (AES). An overview of these standards can be found as highlighted by [5]. DES operates by splitting data into 64-bit blocks and encrypting them using a 56-bit cipher key. However, concerns exist over this standard. Many believe that the short key size makes it vulnerable to brute-force attacks. Furthermore, the internal structure of DES is not completely open raising concerns that backdoors may exist which enable the deciphering of messages without the key. Triple-DES was proposed to alleviate the fears of brute-force attacks against DES. With Triple-DES, two 56-bit keys are used in a three-step process. Firstly, the plaintext is encrypted with key 1, then the result is decrypted with key 2, before finally, the result of this is encrypted with key 1. This process changes the cost of discovering the encryption keys using a brute-force attack from the order of 2^{56} to 2^{112} [6].

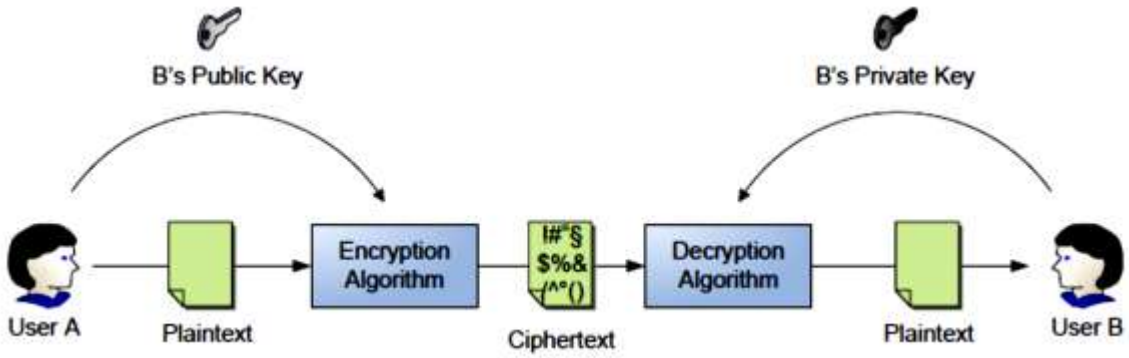

Figure 1: The basic model for public-key encryption

Although Triple-DES addresses the problem of the small key size, it performs relatively slowly when implemented in software and it is not suitable for use in limited-resource platforms. To resolve these problems, Rijndael was proposed in 1997 and this algorithm was later officially standardized as AES in 2001. AES operates fast in both hardware and software, requires little memory to operate, uses 128-bit block sizes, and supports key lengths of 128, 192 and 256 bits. Additionally, the specifications of AES are open and can be found in [7].

Public-key Encryption - A difficult issue facing conventional encryption methods is key distribution. These methods require either:

i. The communicating parties to already possess the secret key OR

ii. A key distribution centre (KDC) to be in place. But with (2), maintaining total secrecy over the communications is under threat. Key owners are forced to share keys with a KDC which could potentially become compromised.

Public-key encryption schemes tackle this issue by using two separate keys rather than a single secret key. Public-key (or asymmetric) encryption algorithms operate by using one key for encryption and the other key for decryption. An important characteristic of asymmetric encryption algorithms is that it must be computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key as illustrated by [8]. Figure 1 shows how public-key encryption works. In this example, user B owns a key pair – a public key (to be distributed to everyone) and a private key (to be kept secret). Suppose user A decides to send an encrypted message to B, the message is encrypted by A using B's public key and the resulting cipher text is sent to B. When received, B decrypts the cipher text using his private key to reveal the plaintext. Total secrecy over the communications is achieved as private keys can be generated locally and never need to be distributed to anyone.

[9], explained that the most widely accepted asymmetric encryption algorithm in existence today is the Rivest-Shamir-Adleman (RSA) scheme. RSA is a block cipher and uses mathematical functions (rather than substitution and permutations) to perform encryption/decryption functionalities. A drawback of asymmetric encryption techniques is that they are computationally slower than symmetric encryption techniques. For this reason, it is common practice for a secure communication session to initially use asymmetric encryption for key negotiation before using symmetric encryption once a secret key has been established. Integrity is concerned with protecting messages from unauthorized modifications. To achieve this, digital signatures can be used. Creating a digital signature involves taking a fingerprint of the message (otherwise known as a hash value) and then encrypting this fingerprint with the signer's private key. Digital signatures make use of public-key encryption [9].

To create the fingerprint, hash functions are used. Hash functions take the form $h=H(M)$ where h represents a fixed-length hash value, H represents the hash function and M represents a variable-length message. [10], highlighted that some important requirements of hash functions include to:

i. be able to be applied to variable-length data.
ii. be able to produce a fixed length output.
iii. be relatively easy to compute for any given message.
iv. be computationally infeasible to derive the message from the hash value (one-way property).
v. be computationally infeasible to find another message which produces the same hash value.
vi. be computationally infeasible to find for any pair (x, y), such that $H(x) = H(y)$.

Hash functions satisfying the first five requirements are known as weak hash functions. For those which also satisfy the sixth requirement (which protects against the birthday attack as explained by [11], these are known as strong hash functions.

A simple example of a hash function is one which performs a bit-by-bit exclusive-or (XOR) operation over each block of the message. Alternatively, this simple example could be enhanced by incorporating a one-bit circular shift (rotation) on the hash value after each data block is processed. More elaborate hash function algorithms include the MD5 message-digest algorithm pointed out by [12] and the Secure Hash Algorithm (SHA) illustrated by [13]. To check if message integrity has been maintained, digital signatures need to be verified at the receiver's end. This verification involves a two-step process. Firstly, the receiver decrypts the signature using the signer's public key to reveal the fingerprint of the message. Secondly, the receiver performs its own hash over the message to create a fingerprint and if the two fingerprints match, integrity has been maintained.

Authentication addresses the question of: "*Are you who you say you are?*" This principle is concerned with verifying an entity's identity claim based on evidence that no-one else can present. According to [14], authentication can be based on three types of evidence:

i. Something the user knows.
ii. Something the user has.
iii. Something the user is or does.

Passwords are a classic example of authentication based on something that the user knows. Token devices (e.g., memory cards and smart cards) are a common form of authentication based on something that the user has, whereas authentication based on biometrics (e.g., fingerprints and retina scans) fall under the category of something the user is or does. The most basic type of authentication is one-way authentication – otherwise known as simple authentication. This involves a claimant who

OPEN ACCESS
ISSN: 2992-5487

wishes to authenticate itself with a verifier by sending a proof of identity. A more advanced form of authentication is mutual authentication. This form of authentication enables two-way authentication so that the claimant can be assured that it is authenticating to the correct verifier. Authorization refers to the process of granting or denying access to resources. It is closely associated to access control which defines the models and mechanisms for applying restricted access to protected resources. The most common access control models typically consist of three main components:

i. Subjects – the entities (e.g., users) of the system.
ii. Objects – the resources (e.g., files) under protection.

Permissions – the access rights supported by the model. Common access rights include read, write, execute and own.

An important aspect of authorization is that it assumes that subjects have been properly and successfully authenticated beforehand. This enables authorization systems to enforce their policies and to make decisions based on the identities of subjects.

Non-repudiation is concerned with providing guarantees so that, for example, a sender of a message cannot deny that a message transmission never occurred when, in fact, it did. Similarly, it can be used to prevent a recipient from claiming that it never received a transmitted message when it was successfully received. Dispute resolution is a major reason for non-repudiation.

According to [14], digital signatures are the only means of achieving non-repudiation without involving a notary. Following are the different forms in which non-repudiation can take as slated by [15]

i. Non-repudiation of creator - prevents the creator from denying ever creating the data.
ii. Non-repudiation of sender - prevents the sender from denying ever sending the data.
iii. Non-repudiation of submission - proof of data transmission for the sender. Prevents the communication system or any recipients from claiming that the data was never transmitted.
iv. Non-repudiation of delivery - prevents the recipient from denying ever receiving the data. It proves to the sender that the data was delivered to the intended recipient.
v. Non-repudiation of receipt - prevents the recipient from claiming that he has never seen the data.
vi. Non-repudiation of use of service - protects a service provider from a user denying that he never used a particular service offered by the provider.

Auditing refers to the logging of system events and keeping an audit trail. It is concerned with keeping users accountable for their actions. An important characteristic of auditing is that it enhances access control. In principle, if unauthorized actions pass the access control system, then the auditing system detects it. According to [16], auditing enhances access control in the following ways:

It acts as a deterrent. Users become more reluctant to perform unauthorized activities if they know that their actions are being tracked. Log files provide evidence for investigating attempted or actual security violations. It helps with discovering security holes in the system; ensures that authorized users do not abuse or misuse their privileges. When examining the issue of trust, it is important to identify who trusts whom on what. The principle of trust suggests a relationship between (at least) two entities based on a particular action. Trust can be categorized into two types – direct trust and third-party trust. The former type describes the situation where two entities have established a trust relationship themselves. The latter refers to an implicit trust relationship between two entities since both entities trust a specific trusted third-party (the third-party determines the trustworthiness of the two entities). Third-party trust is common in public-key cryptography where a Certification Authority (CA) acts as the trusted third party.

Security Policies and Models

A security policy is simply a statement of what is allowed and what is not allowed to occur between entities in a system. Focus was on authorization policies – a type of prevention policy. Other policy types include detection policies and recovery policies.

Authorization policies can be expressed mathematically, specifying the allowed and disallowed states. Let us consider the following, where the computer system is expressed as a finite-state automaton with a set of transition functions that change state as illustrated by [17].

Let P be the system state space.

$Q \subseteq P$ defines the states in which the system is authorized to exist in.

When the system is in a state, $s \in Q$, the system is secure.

When $s \in P' \setminus Q$, the system is not secure.

OPEN ACCESS
ISSN: 2992-5487

An authorization policy characterizes *Q*. In other words, the policy distinguishes between authorized (or secure) system states and unauthorized (or non-secure) system states. A system mechanism (e.g., access control) prevents the system from entering P \ Q. A secure system is a system that starts in an authorized state and cannot enter an unauthorized state. The remaining section discusses several policy models, starting with the Chinese Wall policy model.

Chinese-Wall Policies

The Chinese-Wall security policy was proposed by [18]. to address conflict of interest issues related to consulting activities within the commercial environment. For example, the policy model can be viewed as a "*code of practice*" to prevent consultants from disclosing sensitive information (or insider knowledge) of one corporation to a competitor. Consultants can, however, freely advise corporations which are not in competition with each other, and also access general market information.

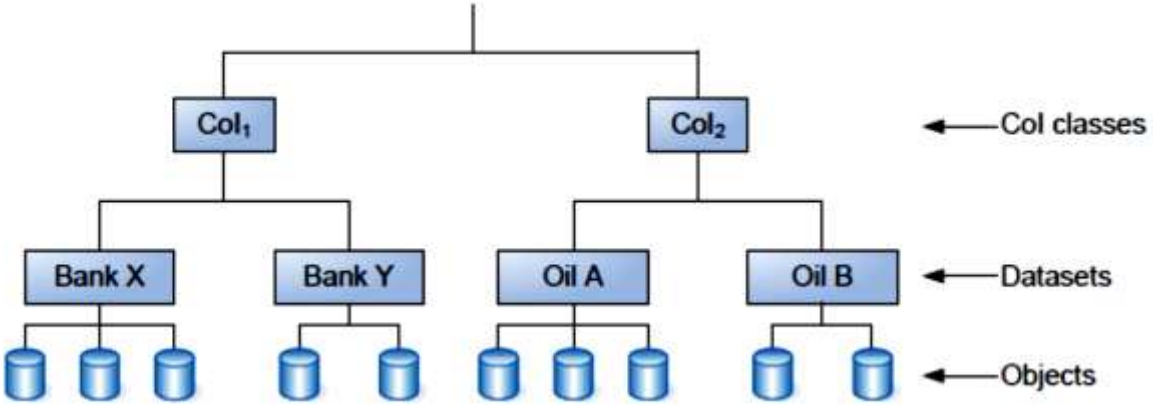

Figure 2: Chinese-Wall data hierarchy

A Chinese-Wall policy is a form of MAC. It aims to prevent the occurrence of information flows that could cause a conflict of interest as described by [19]. Initially, the user is free to access any object. But after the initial choice is made, a Chinese wall is created for that user around the dataset containing the selected object.

The model organizes all data into a hierarchy:

1. At the lowest level, there are individual items of data, concerned with a single corporation. Such data is stored as objects.
2. At the intermediate level, objects are grouped according to their respective corporation. Each group of objects is referred to as a dataset.
3. At the highest level, the datasets of those corporations in competition with one another are grouped. Each group of datasets is referred to as a conflict of interest (*CoI*) class. Each dataset belongs to only one *CoI* and each *CoI* has two or more member datasets. Hence *CoI* classes are mutually disjoint.

Let us consider the example illustrated in Figure 2 which has been adapted from [20]. If Alice has read sensitive information of Bank X, then Alice is prohibited from reading sensitive information from any other dataset (i.e., bank) within CoI_i. In other words, read access is only granted if the object requested,

(a) is in the same dataset of an object previously accessed by that user, or

(b) belongs to a *CoI* class not previously accessed by that user.

But suppose there are two users: Alice and Bob – where Alice can access Bank X and Oil A, and Bob can access Bank X and Oil B. If Alice reads sensitive information from Oil A and writes it to Bank X, then Bob can read sensitive information of Oil A; for this reason, the write rule is very restrictive. A user who has previously read objects from more than one dataset cannot write any object. As a result, the flow of information is confined to its own company dataset. The model also considers sanitized information (i.e., where information is disguised so that the corporation's identity is not discovered). For such information, access restrictions are lifted [21].

Bell-LaPadula Model

According to [22], the Bell-LaPadula (BLP) Model uses a hierarchical set of security classes to impose access restrictions. It was initially designed for use in military environments. Rather than placing constraints on the inter-relationships between objects (as is the case with Chinese-wall

OPEN ACCESS
ISSN: 2992-5487

policies), each object within the BLP model is labeled. To explain how this model works, let us consider that there exists the security classes Top Secret *(TS)*, Secret *(S)*, Confidential *(C)* and Unclassified *(U)*, such that $U \leq C \leq S \leq TS$.

Objects are assigned to these security classes to indicate their sensitivity level. An object's security label has the form *(class, {cat})* where *class* is the security class and *cat* is the category of the information. Subjects are also assigned to these security classes to indicate their clearance level. A subject's security attributes have the form *(clear, {NTK})* where *clear* is the maximum-security class to which the subject is permitted access and *NTK* is the subject's need-to-know (representing the categories to which the subject is permitted access). Figure 1.3 illustrates these security classes, showing subjects and objects allocated to these classes. Access to an object by a subject depends on the security levels associated with the two. In particular, the following principles must hold:

Read Down – a subject's clearance must dominate the security level of the object being read. In other words, subject *S* can read object *O* only if *class(O)* ≤ *class(S)*. This is known as the simple security rule. The subject's need-to-know must also include the object's category.

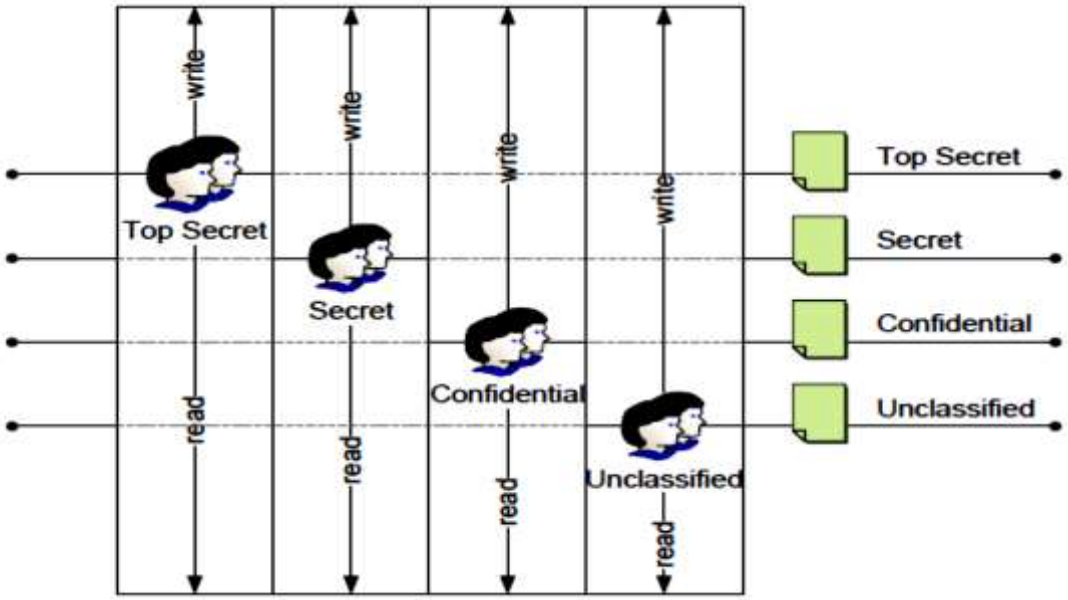

Figure 3: Bell-LaPadula model example

Write-Up: a subject's clearance must be dominated by the security level of the object being written. In other words, subject *S* can write object *O* only if *class(S)* ≤ *class(O)*. This is known as the **-property rule*. The subject's need-to-know must also include the object's category. Compliance with these principles prevents information in high-level resources (i.e., more sensitive) to flow to resources at lower levels. This is illustrated in Figure 3. In this model, information can only flow upwards or within the same security class. Although the BLP model is well-suited for upholding confidentiality, it suffers from several limitations as slated by [21].

1. The model is static. Access rights and security classes cannot be changed.
2. How to create and delete subjects and objects is not specified. This limitation can be addressed using the Clark-Wilson model as illustrated by [22].
3. The model does not deal with integrity, only confidentiality. This limitation can be addressed using the Biba model as explained by [23] and the Clark-Wilson model as illustrated by [24].
4. It contains covert channels (i.e., information flows that violate the security policy).

ORCON: Originator Controlled

According to [25], Originator Controlled (ORCON) is concerned with controlling the dissemination of objects. This policy model requires recipients to obtain approval from the object's originator in order to distribute the object to other subjects. The term originator refers to the subject responsible for the data contained in the object and for determining to whom the data can be released. This differs from the term owner, who is the subject responsible for the creation of the object and is authorized to change the Discretionary Access Control (DAC) permissions on the object as highlighted by [26]. Traditional solutions focus on enforcing ORCON policies within a closed control environment. In

OPEN ACCESS
ISSN: 2992-5487

these cases, policies are typically centrally controlled, and users behave within the scope of the policies. Figure 4 illustrates the design of a traditional ORCON solution as highlighted by [26].

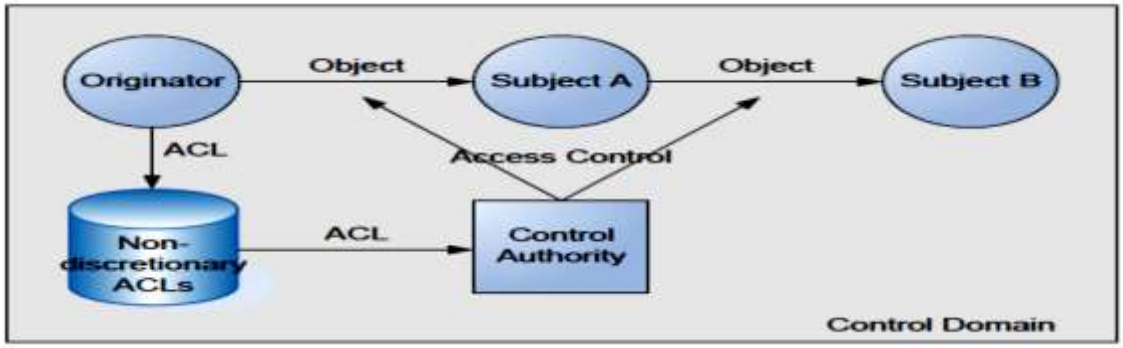

Figure 4: ORCON

In this diagram, the originator releases the object (marked with "ORCON") and makes it available to subject A in the policy settings (ACL). Should A decide to pass the object to subject B, the control authority must first check whether B is permitted access to the object. Consequently, this enforcement allows the originator to maintain control over how the object is distributed. According to [27], Usage Control (UCON) has emerged more recently as a means of controlling and managing the usage of objects. When coupled with UCON, ORCON policies can be enforced beyond the originator's local control domain. This provides a solution for controlling dissemination and re-dissemination beyond a closed control environment where a central authority is not available.

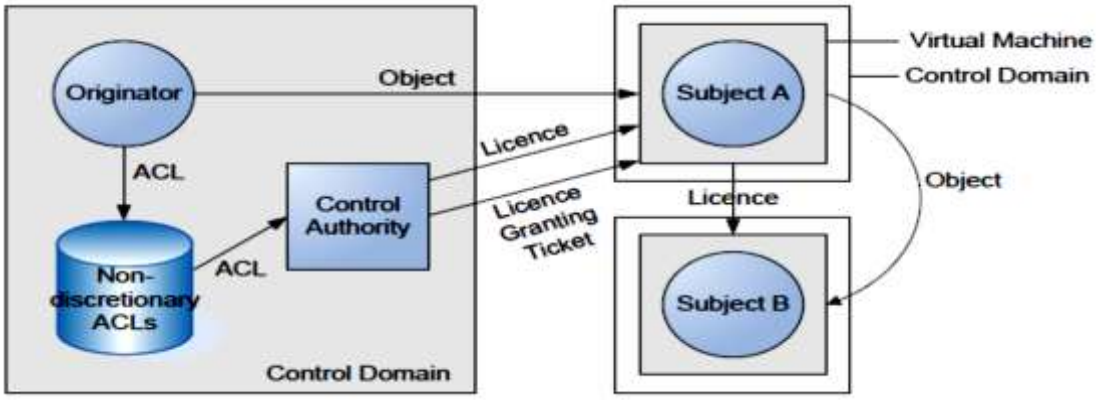

Figure 5: ORCON with UCON

The UCON model defines several components. The object is encapsulated into a cryptographically protected digital container. The encapsulated object can only be accessed from special application software or hardware (known as a virtual machine). Access rights are listed in a control set (implemented as licenses for propagating usage rights to authorized users). A license is required for the user's virtual machine to access the encapsulated object. Finally, a control center exists for storing security policies and managing access rights. Figure 1.5 shows how ORCON and UCON can be combined to enforce access control policies beyond the originator's local control domain. Subject *A* and *B* reside in separate control domains. Both subjects require a license to access the object. In this example, subject *A* is issued a license directly from the originator (via the control center) and also a license granting ticket. This ticket allows *A* to issue licenses to other authorized subjects (in this case, to subject *B*). The shaded areas (i.e., the originator's control domain and the virtual machines) indicate the extent of the originator's control over the disseminated object [28].

Digital Rights Management (DRM)

Similar to ORCON, Digital Rights Management (DRM) is also concerned with controlling the distribution and usage of objects. DRM is highly driven by media companies, who are increasingly striving to protect commercial digital content and combat digital piracy. These companies argue the need for DRM because "*without protection and management of digital rights, digital content can be easily copied, altered and distributed to a large number of recipients, which could cause revenue loss (to them)*" as illustrated by [29]. DRM systems protect high-value digital assets by protecting against

OPEN ACCESS
ISSN: 2992-5487

unauthorized access to the content, controlling the distribution of these assets and managing the content usage rights. Central to achieving this is the use of digital licenses that grant certain usage rights to the consumer. Digital content is encrypted before being distributed and licenses are purchased by the consumer to unscramble and access the content. Client-side applications (e.g., media players) are critical in DRM implementations in that they enforce protection based on these licenses. According to [28, 29, 30], many DRM models and implementations have been proposed for various industries (e.g., online music, film, print, etc.) and different platforms (e.g., personal computers, mobile phones, embedded systems, etc.). Despite this, the basic DRM process is the same and usually involves four entities: the content provider, the distributor, the clearing house and the consumer as slated by [3]. A typical DRM system is shown in Figure 1.5 as described by [8]. Each component in this system is described as follows:

1. **Content Provider**: Owns the intellectual property rights to the digital content and protects these rights.
2. **Distributor**: Receives digital content from the content provider and distributes this content to consumers. In other words, acts as the distribution channel.
3. **Consumer**: Receives digital content from the distributor and purchases licenses from the clearing house to access the content.
4. **Clearing House**: Issues digital licenses and handles payments. That is, the clearing house receives payments from the consumer, pays royalty fees to the content provider and pays distribution fees to the distributor.

There is much debate over the usefulness of DRM as stated by [11]. Some of the main arguments against DRM are:

1. No DRM standards exist. Most DRM models and technologies are proprietary.
2. Restricts innovation, research, free speech and public access to digital information.
3. Risk of data and privacy infringement.

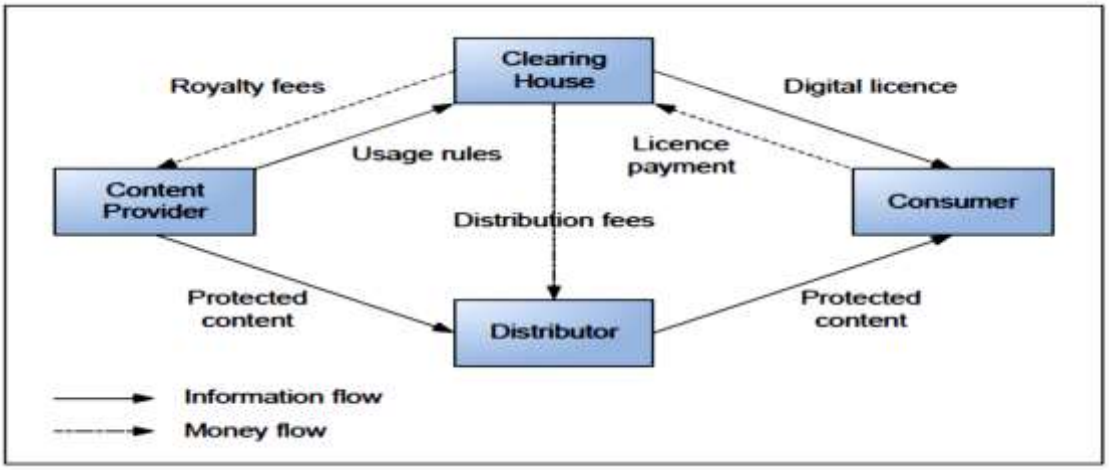

Figure 6: A typical DRM system

Despite these arguments, DRM provides the following benefits:

1. Allow consumers to acquire and use digital content legally.
2. Allow intellectual property rights to be protected.
3. Assure the authenticity and integrity of the digital content.

Information Flow Models

We have already seen examples of information flow models in the previously discussed security models – namely, Chinese-wall policies and the Bell-LaPadula model. Another security model that constrains information flow is the Biba model as illustrated by [8]. Although like the Bell-LaPadula model, the Biba model differs in that it focuses on integrity and not confidentiality. This model contains integrity levels which are analogous to the sensitivity levels used in the Bell-LaPadula model (e.g., Top Secret, Confidential, etc.). Data integrity is maintained by conforming to the following principles:

Read Up: a subject's clearance must be dominated by the integrity level of the object being read. This is known as the simple integrity property.

Write Down: a subject's clearance must dominate the integrity level of the object being written. This is known as the integrity *-property.

These principles ensure that modifications can only flow downwards or within the same integrity level, hence upholding the integrity of information in the more sensitive, higher level resources. They are opposite to the principles defined in the Bell-LaPadula model. As a result, a system cannot use both models (otherwise there would be no information flow at all).
Some recent examples of information flow models are discussed by [9]. [8], define an information flow system as a state machine model and view information flow as a sequence of meta-flows (representing the transfer of information from variable v_1 to v_2 by subject s executing primitive (atomic task) p in system-state ss). Information flow constraints are defined in terms of static security policies (which do not vary with system states) and dynamic security policies (which vary with system states). This model provides general definitions for expressing information flow properties, as well as expressing security policies and access control in terms of information flow. This contrasts with the previous models which specify precise rules and constraints for securing information flow. Other types of information flow models are more specialized. For example, [20], define an information flow model for protecting sensitive information in service-oriented architectures using web services. This model separates the underlying implementation architecture into two parts:

1. A small, functionally limited, trusted platform (for handling sensitive information)
2. A large, functionally rich, untrusted platform (that invokes the other platform when sensitive information needs to be handled). This intention behind this strategy is to ensure that access to the sensitive information is limited to only the web services components that are required to process this information. Restricting sensitive information to the trusted platform minimizes the potential for this information being exposed as a result of attackers exploiting security vulnerabilities in the system.

Traditional Access Control Models

Discretionary Access Control (DAC), Mandatory Access Control (MAC) and Role-Based Access Control (RBAC) all represent traditional approaches towards access control enforcement. Each of these models was now elaborated on.

Discretionary Access Control

According to [15], the DAC model adopts an individual resource ownership approach. It is owner-controlled in that object owners control the access permissions applied to their objects. To a large extent, authorization support in many mainstream systems is based on DAC (including UNIX operating systems, Windows operating systems and Apache Web servers).
The discretionary model functions based on user identities and authorization policies. Permissions on objects are assigned to subjects (the authorization policies specify the permissions that each subject is granted on each object). The advantage of this model is that it is highly flexible. However, [14] elaborated that due to issues based on delegation of rights and uncontrolled dissemination, the discretionary model is less suitable for corporate environments (where organizational structures exist). Access Control Lists (ACL), which specifies the access rights of each subject in the system for a particular object, is commonly used for implementing the discretionary model. Each object possesses a separate ACL. With ACLs, it is easy to
(a) determine the access rights of each subject on an object and
(b) revoke all access rights on an object.
The drawback for ACLs is that for any given subject, it is difficult to determine all of their access rights on all objects in the system. Revoking each access right for a particular subject on all objects is also difficult.
Capability lists are another DAC implementation option. Where ACLs are allocated to objects, capability lists are allocated to subjects. In other words, each subject possesses a separate capability list. It lists the objects, together with the subject's access permissions for each object.
Figure 2.0 shows an example ACL (top row) and capability list (bottom row). The ACL shows the access permissions assigned to the object *File 1* whereas the capability list shows the access permissions granted to the subject Alice.

Mandatory Access Control

According to [10], the MAC model adopts an organisational resource ownership approach. It is organization controlled in that information flows are restricted. This type of security is also known as multi-level security (MLS). Many MLS systems are based on the Bell-LaPadula.

OPEN ACCESS
ISSN: 2992-5487

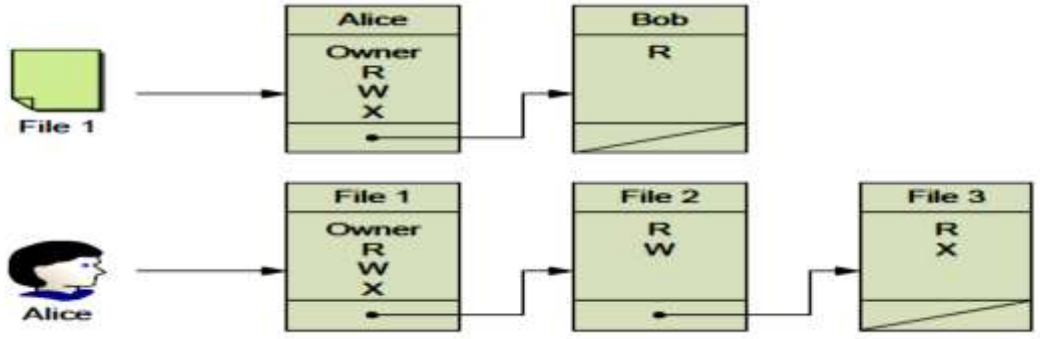

Figure 7: ACL and capability list examples

MAC enforces access control by classifying all subjects and objects in the system. Hence it is based on a security label system. Each subject is assigned a security clearance (which reflects the trustworthiness of the user not to disclose sensitive information to unauthorized users). Objects are similarly classified to reflect the sensitivity of its content. A subject is granted access if their clearance is equal to or higher than the object's classification. An example of a set of classifications (ordered from highest to lowest) is top secret, secret, confidential and unclassified. As well as the classification, security labels include different categories (representing compartments of information within the system). Unlike classifications, categories do not follow a hierarchical scheme. The purpose of categories is to enforce need-to-know rules. For example, having a top-secret security clearance should not permit the subject access to all top secret information. Categories may be based on departments, such as marketing, sales, human resources, etc. Suppose user Alice has the security clearance {*secret*, {*marketing*, *sales*}}, then access to a file labeled as {*confidential*, {*human resources*}} would be denied. This model is considered most suitable for rigid environments (such as the military), where information is owned by a central authority and where confidentially is of utmost importance. Apart from being too inflexible for many environments, this model also has the drawback that it cannot properly represent user roles.

Role-Based Access Control (RBAC)

Role-Based Access Control (RBAC) attempts to overcome the drawbacks associated with the discretionary and mandatory models as illustrated by [12, 14, 15, 17] and (Sandhu, Coyne, Feinstein and Youman, 1996). In a shift from mapping the permissions of individual subjects directly to objects, RBAC introduces the concept of roles (which represent job functions) and operates by granting permissions to roles before assigning users to these roles. Roles can reflect specific task competencies, responsibilities or even specific duties as highlighted by [20].

Key motivating factors behind RBAC's development include:

(1) simplifying management procedures and

(2) the need for specifying and enforcing enterprise-specific policies.

The introduction of this model has produced the following advantages:

1. **Ease of Administration**: allocating users to roles typically requires much less technical expertise than determining suitable access rights for each role. As user assignment to roles changes much more frequently than access permission mappings to roles (which is relatively stable), this greatly simplifies administration of the system once the RBAC framework has been established.
2. **Increased Flexibility** – apart from roles, configuration of the RBAC framework can also include role hierarchies, relationships and constraints. Additionally, policy specifications can be based on what actions, when, from where, in what order, and in some cases, under what relational circumstances as stated by [23].
3. **Delegation of Administration** – for example, global-level policies affecting the whole domain can be defined and administered centrally at the enterprise level whereas policies relating to a specific sub-domain can be enforced locally at the organizational unit level.

According to [25], RBAC (specifically RBAC96) consists of four conceptual models:

i. $RBAC_0$ contains the minimal features of a system implementing RBAC.
ii. $RBAC_1$ adds support for role hierarchies.
iii. $RBAC_2$ adds support for constraints (e.g., separation of duty).
iv. $RBAC_3$ includes all aspects of the above models.

OPEN ACCESS
ISSN: 2992-5487

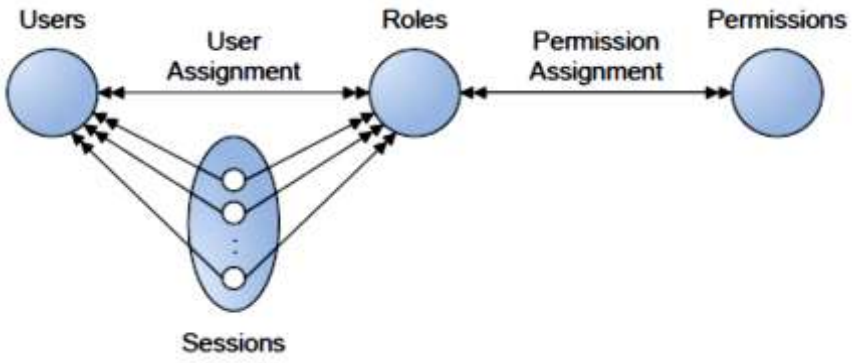

Figure 8: The Role-Based Access Control model (RBAC)

Figure 2.1 illustrates the simplest form, $RBAC_0$ (otherwise known as Core RBAC). A user can be mapped to multiple roles and a role can have multiple users mapped to it. Similarly, permission can be mapped to multiple roles and a role can contain multiple permissions. There is also the notion of sessions. A session represents an instance of a user connected to the system and defines the subset of activated roles. Different sessions for the same user can be active.

RBAC suits multi-user, multi-application systems and promotes two important features – least privilege and separation of duty. Least privilege describes the administrative practice of selectively assigning permissions to users. The objective is that the user should not be given more permissions than necessary to perform their job function.

Separation of duty (SoD), as defined by Simon and Zurko (1997), is a form of constraint, designed to prevent users from exceeding a reasonable level of authority and to prevent failures caused as a result of collusion amongst individuals. The two main categories are static SoD and dynamic SoD as illustrated by Fumy and Sauerbrey (2006):

1. **Static SoD**: Places constraints on the assignment of roles to avoid conflicts of interest.
2. **Dynamic SoD**: like static separation but based on time or other dynamic factors.

Implementing Access Control

Access control implementations typically include a reference monitor as described by [8] – a layer of functionality that safeguards access to protected objects. Reference monitors play the crucial role of validating access requests (i.e., to verify a subject's right to access a protected object every time the object is referenced).

According to [11], the primary building blocks of access control implementations are:

i. **Policy Enforcement Points (PEP)**: intercept access requests, send authorization decision requests to PDPs and enforce authorization decisions. The PEP is representative of a reference monitor.
ii. **Policy Decision Points (PDP)**: accept authorization decision requests from PEPs and make authorisation decisions based on the authorization policies provided by PMAs.
iii. **Policy Management Authorities (PMA)**: create, supply and maintain authorization policies.

Access control implementations can be compared based on the level of policy expressiveness that they support. Common forms of authorization as described by Fumy and Sauerbrey (2006) include:

i. **Identity-centric Authorization** – based on subject and object identifiers. This is the simplest form of authorization. For example, Alice may access file X.
ii. **Category-aware Authorization** – extends on identity-centric by supporting structured sets of subjects and objects (independent of one another). This involves subject attributes (e.g., group memberships, role assignments) and object attributes (e.g., object classifications, object properties, etc.). For example, Developers may access files within directory dev.
iii. **Relationship-aware Authorization** - extends on category-aware by supporting jointly structured subjects and objects. This provides the ability to express conditional relationships and to evaluate them at runtime. For example, Developers may access files within directory *dev/region3* if the user is based in the same region.

Emerging Access Control Technologies

A limitation with the traditional access control models is that they do not fully satisfy the requirements for distributed access management (needed for inter-organizational scenarios). This is because these models mainly focus on intra- organizational environments. They implicitly assume that identity and resource providers reside in one organization.

Several technologies are now emerging that focus on federation. In other words, they aim to allow organizations to extend their existing systems to accommodate collaboration with partner organizations. This section looks at some of these emerging access control technologies.

Security Assertion Markup Language (SAML)

The Security Assertion Markup Language (SAML) is an XML-based framework for exchanging security-related information. Developed by an Organization for the Advancement of Structured Information Standards [14] technical committee, the current specification release (at the time of writing) is version 2.0 published as an [14] standard. This release improves on SAML version 1.3 by [17] and includes Liberty Alliance contributions (2001 – 2008).

This framework plays a significant role in providing federated identity management systems and integration amongst heterogeneous environments. It is designed for cross-domain authorization services and single sign-on. Of all the components in the SAML specification suite, assertions are fundamental. An assertion is simply a statement (or "declaration of fact") based on a subject and is issued by an issuing authority. There are three types of assertions in SAML:

i. **Authentication Assertions**: An issuing authority asserts that subject *S* was authenticated by means *M* at time *T*. These assertions are essentially proof of a past authentication event and are targeted towards single sign-on.
ii. **Attribute Assertions**: An issuing authority asserts that subject *S* is associated with attributes *A*, *B*, etc. with values "a", "b", etc. Aimed towards authorisation services and distributed transactions, these assertions typically contain information contained in directories (e.g., LDAP repositories).
iii. **Authorization Decision Assertions** – An issuing authority decides whether to grant the request by subject *S* for access type *A* to resource *R* given evidence *E*. Again, these assertions are used by authorization services and distributed transactions.

Additional components of the SAML specification suite include the SAML protocols, bindings and profiles. SAML protocols define the process of how assertions are requested (e.g., the request and response messages involved). Bindings refer to how the SAML protocol messages are transmitted across lower-level application protocols such as HTTP and SOAP. Profiles define how SAML protocols and bindings (as well as assertions) can be combined to solve specific business problems.

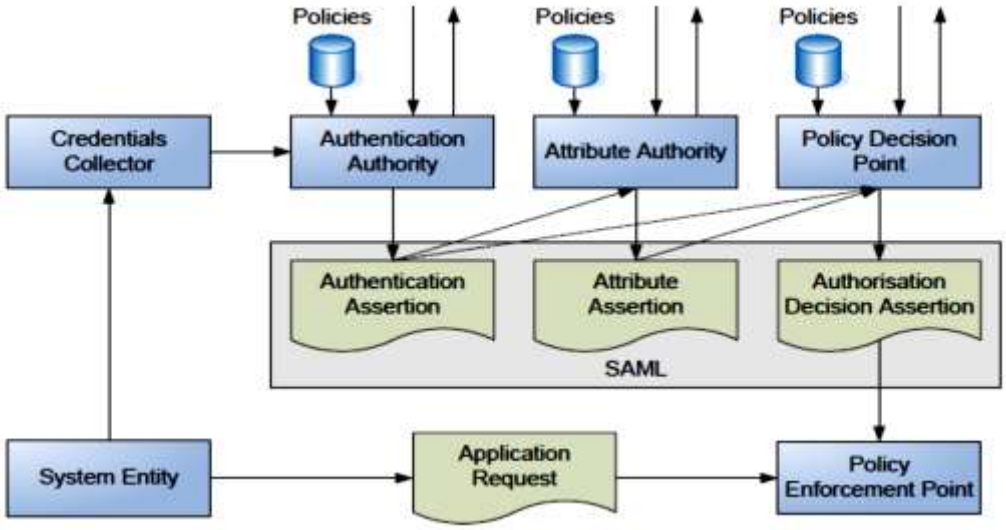

Figure 9: The SAML producer-consumer model: Fumy, W. and Sauerbrey, J. (2006)

The main properties of SAML as narrated by [21] include:

i. SAML authentication assertions do not constrain the initial authentication method. They simply contain information on the method as well as metadata on the initial authentication context (e.g., length of authentication keys).
ii. SAML does not specify attribute vocabularies. It is generic in terms of the attributes that it can carry.
iii. SAML assertions can be passed to multiple relying parties. They do not need to be scoped to dedicated targets.

OPEN ACCESS
ISSN: 2992-5487

iv. SAML relies on XML Signature illustrated by XML Signature Syntax and Processing. (2002) and XML Encryption discussed in XML Encryption Syntax and Processing. (2002) for object protection.

Figure 9 displays the SAML producer-consumer model. This model illustrates the flow of events of how SAML assertions are produced and consumed.

The Credentials Collector collates the System Entity's credentials and passes them to the Authentication Authority. In this model, the System Entity is the subject. The Authentication Authority assesses these credentials in accordance with its policies and then issues an authentication Assertion. From here, one of two paths can be followed. The Authentication Assertion can be passed directly to the Policy Decision Point (PDP) or the assertion can be passed to the Attribute Authority. As before, the Attribute Authority receives the Authentication Assertion, assesses it according to policy and issues an Attribute Assertion which can then be passed onto the PDP. When the PDP receives these assertions, it also assesses them according to its own policies and issues an Authorization Decision Assertion which is then forwarded to the Policy Enforcement Point (PEP). Meanwhile, the System Entity has sent a request to the PEP to access a particular service. Using the information in the Authorization Decision Assertion, the PEP can either permit or deny the request.

Extensible Access Control Markup Language (XACML)

The Extensible Access Control Markup Language (XACML) is an XML-based language for access control. Similar to SAML, this technology is also developed by an OASIS (1993 – 2008) technical committee. The current specification release (at the time of writing) is version 2.0 defined by eXtensible Access Control Markup Language (XACML) (2005) (published as an OASIS standard). The objectives behind XACML are:

(1) forming a standardized approach for access control and

(2) the ability to provide finer granularity for making access control decisions.

XACML defines a:

i. Syntax to express authorization policies (in other words, an authorization policy language).
ii. Syntax to express authorization decision requests and responses.

XACML can make access control decisions based on dynamic information. Additionally, the XACML specification describes an architecture for access control enforcement. This architecture is illustrated. An authorization request sent by the subject (access requester) is received by the Policy Enforcement Point (PEP). The PEP is responsible for accepting requests and for enforcing access control decisions. To determine how to handle this request, the PEP creates an XACML request and sends this to the Policy Decision Point (PDP) so that a decision can be made regarding whether to permit or deny the requested action. The PDP retrieves the applicable policies from the Policy Access Point (PAP) – the data store for the access control policies. Additionally, the PDP may also request attributes from the Policy Information Point (PIP) so that the policies can be evaluated.

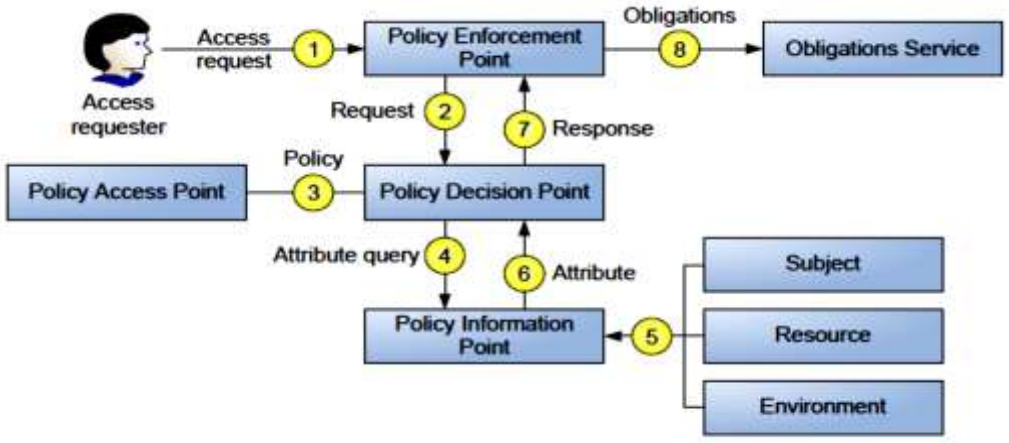

Figure 10.: The XACML architecture: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml

The types of attributes provided by the PIP include subject attributes, resource attributes and environment attributes. This component provides XACML with access to dynamic statistics for making access control decisions. Once the applicable policies have been evaluated, the PDP sends its authorization decision to the PEP who then enforces this decision. The PDP can also state obligations that need to be fulfilled by the PEP before access is granted.

OPEN ACCESS
ISSN: 2992-5487

Policies in XACML consist of several components: target, rules, rule-combining algorithm and obligations.

i. **Target**: The target determines the applicability of the policy for the authorization request. Determining this is achieved by comparing the attributes of the target (i.e., subject, resource, action and environment) against the equivalent attributes in the request message. Should these values match; the policy is then declared to be relevant for the request. A policy contains only one target.

ii. **Rules**: Policies can contain multiple rules and each rule consists of the following elements: a condition, an effect and a target. When the condition is evaluated, it returns true, false or indeterminate. If the condition returns true, the rule returns the value of the effect element (i.e., permit or deny). If the condition returns false, the rule returns not applicable and if the condition returns indeterminate, the rule also returns indeterminate. The target element of a rule is comparable to the target element of a policy – i.e., is the rule relevant for the request?

iii. **The Rule-combining algorithms**: As policies can contain multiple rules, the rule-combining algorithm is responsible for resolving conflicting rule results so that a single authorization decision can be arrived at. Examples of these algorithms include deny-overrides, ordered-deny-overrides, permit-overrides, ordered-permit-overrides and first-applicable.

iv. **Obligations**: Obligations are actions which must be fulfilled by the PEP in conjunction with executing the authorization decision. This provides XACML with finer granularity access control.

Web Services Security

According to the Security in a Web Services World (2002), the WS-Security roadmap outlines several emerging specifications focusing on identity and access management for Web services (WS) technologies. A partial overview of the Web services security framework is shown in Figure 10 below.

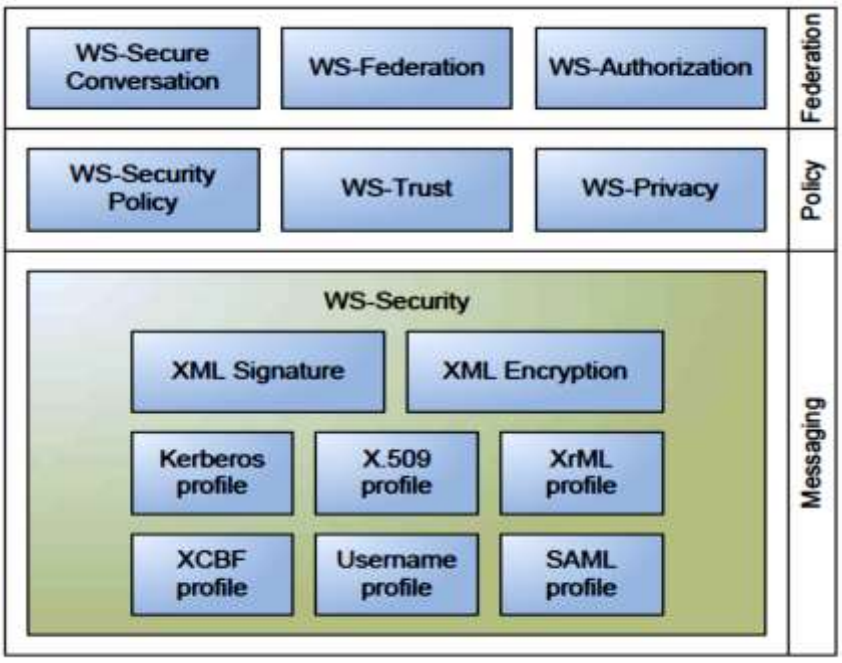

Figure 11: The Web Services Security framework http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss

The WS-Security specification addressed by WS-Security (2004) focuses on message integrity and message confidentiality. More precisely, it describes how signatures and encryption headers are attached to SOAP messages as illustrated in [30]. Additionally, it defines various profiles for attaching security credentials to messages (including binary security tokens such as X.509 certificates and Kerberos tickets).

The level above WS-Security is the Web services policy layer. This layer consists of the following specifications:

i. **WS-Security Policy:** defined by WS-Security [21] – for expressing security-specific requirements and capabilities in WSPolicy as illustrated by Web Services Policy (2007). WS-Policy is a meta-language for specifying service characteristics and supported features so that service providers and consumers can coordinate their preferences (e.g., required security tokens, supported encryption algorithms, privacy rules).
ii. **WS-Trust defined** by [24] – defines a framework of trust models for establishing and managing the direct or brokered trust relationships. Also allows for security tokens to be exchanged across different trust domains.
iii. **WS-Privacy**: defined by Security in a Web Services World (2002) – defines a model for specifying the privacy preferences of service providers and consumers. At the time of writing, this specification has not been released. The next level up is the Web services federation layer. This layer consists of the following specifications:
iv. **WS-Secure Conversation**: defined by WS-Secure Conversation (2007) – describes how message exchanges are authenticated and managed. It covers security context exchange and establishing and deriving session keys.
v. **WS-Federation**: defined by Web Services Federation Language (2006) – defines a federation model for the federation of identity, authentication and authorization information across security domains.
vi. **WS-Authorization** defined by Security in a Web Services World (2002) – for managing authorization data and authorization policies. At the time of writing, this specification has not been released.

Development on Web services security is currently still in progress. The completed components to date tend to be mainly directed at authentication and secure messaging. Despite this, the framework can be augmented with custom authorization models to provide access control.

ACCESS CONTROL FOR INTER-ORGANISATIONAL WORKFLOWS

Applying access control to inter-organizational workflows requires the design of an extended model. Inter-organizational workflows span across multiple organizations, presenting several challenges for access control enforcement. Recognizing the challenges, [23] define the following access control requirements for inter-organizational workflows:

i. **Separation of Application-level and Organization-level Security**: Changes to workflow participants (e.g., where a new organization replaces an existing organization in the workflow) should not require other organization involved to restructure their security infrastructures in order to uphold the inter-organizational workflow. Hence inter-organization workflows (i.e., the application-level security infrastructure) need to be insulated from organization level changes to ensure that workflows can continuously operate without changing the workflow specifications.
ii. **Fine-grained Access Control**: Inter-organizational workflows can contain many workflow tasks, with some tasks existing as threads within a process. Workflow tasks provide a user's working context. Access control as far as the process level is not sufficient. A finer grained access control on the task level is needed to apply access control on the user's working context.
iii. **Support for Dynamic Constraints**: Dynamic constraints may be based on the users of a specific task (e.g., separation of duty constraints). Since inter-organizational workflows may consist of several autonomous workflows, a framework for workflow history sharing between participating workflows is required.

[25] focus on this third requirement by considering separation of duty constraints that span multiple instances of a workflow. Their work provides a formalism that includes predicates over temporal and workflow parameters so that "*inter-instance*" constraints can be specified as conditions on time and workflow history. In a separate but related work, [25] investigate access control for inter-organizational resource sharing and propose an approach that exploits the "semantics associated with the user and object attributes associated with a specific role and its permissions". An external user is granted access to an object if he/she possesses at least the required set of attributes derived through the proposed process. This approach targets dynamic environments, characterized by entities joining or leaving the collaboration in an ad-hoc manner. [21] lists several design goals for a distributed access control processor applicable to inter-organizational workflows. Such goals include extensibility, ease of integration, usability, performance and scalability.

Many security architecture proposals for inter-organizational workflows base their designs on web services as illustrated by [21]. This highlights the significance of web services as an enabling technology for such workflows. Although many web services technologies are open standards, such

tightly coupled designs are too limiting. A broader model is needed that serves the access control requirements of inter-organizational workflows implemented using alternative enabling technologies. Currently many reference implementations of inter-organization workflows and data-interaction happen through secure web service implementation. Many new technologies for web services are emerging today. For instance, financial institutions allow for inter-organization workflows through loan applications, account cross-reference, funds-transfer, account-lookup and balance inquiry.
Some popular web service protocols used today include the following:

i. **SOAP (Simple Object Access Protocol):** One of the foremost and widely used web service protocols. However, its popularity is fast declining due to security issues, flexibility and emerging newer technologies.
ii. **WSDL (Web Service Definition Language):** a platform independent set of specification for interoperability of disparate system.
iii. **MTOM (Message Transmission Optimization Mechanism):** this is a method of efficiently sending binary data to and from Web services. MTOM is usually used with the XOP (XML-binary Optimized Packaging).
iv. **JSON (JavaScript Object Notation):** is an open standard that uses human-readable text to transmit data objects consisting of attribute–value pairs. It is used primarily to transmit data between a server and web application, as an alternative to XML.
v. **REST (Representational State Transfer):** provides a simple way to create, read, update or delete information on a server using simple HTTP calls. It is an alternative to more complex mechanisms like SOAP, CORBA and RPC. A REST call is simply an HTTP request to the server.

SUMMARY

Current research into access control for inter-organizational workflows and cryptographic access control have encompassed many forms, a few of which have been presented in this literature review. However, these works have predominately focused on the theoretical side and less on the practical aspects. Hence on a theoretical level, combining inter-organizational workflow, security policies and access control is well understood. As much as possible, some much effort has been put in to ensure that the practical implementations of the reference models are well documented where necessary. Also, in line with standard research paper documentation, the use of the appendix section to document practical implementations has been adopted. This implies that sample project implementation codes, procedures and methods have been documented in the appendix of this thesis.

REFERENCES

1. Bell, D. E., and LaPadula, L. J. (1976). *Secure Computer Systems: Unified Exposition and MULTICS Interpretation.* Technical Report 75-306, The MITRE Corporation.
2. Bertino, E., Castano, S., and Ferrari, E. (2001). *Securing XML Documents with Author-X.* IEEE Internet Computing, Vol. 5(3), pp. 21–31.
3. Biba, K. J. (1977). *Integrity Considerations for Secure Computer Systems.* Technical Report 76-372, The MITRE Corporation.
4. Bishop, M. (2003). *Computer Security: Art and Science.* Addison-Wesley.
5. Bracher S., (2009). *Secure Information Flow for Inter-organisational Collaborative Environments*, pp. 50-120, Brisbane, Australia.
6. Bracher, S. and Krishnan, P. (2008). *Implementing Secure Document Circulation: A Prototype.* If08909n SAC '08: Proceedings of the 23rd Annual ACM Symposium on Applied Computing, pp. 1452–1456, Fortaleza, Brazil. ACM ISBN 1-59593-753-7
7. Bracher, S. and Krishnan, P. (2012). *Supporting Secure Information Flow: An Engineering Approach.* International Journal of e-Collaboration (IJeC), Volume 8, Issue 1, pp. 1-19.
8. Brewer, D., and Nash, M. (1989). *The Chinese Wall Security Policy.* In IEEE Symposium on Security and Privacy, pp. 206–214.
9. **,,.,G…;/i?Uff y vuvff**
10. Crampton, J., Martin, K., and Wild, P. (2006). *On Key Assignment for Hierarchical Access Control.* In CSFW '06: Proceedings of the 19th IEEE workshop on Computer Security Foundations, pp. 98–111, Washington, DC, USA. IEEE Computer Society.
11. Di Vimercati, S.D.C., Foresti,9 S., Jajodia, S., Paraboschi, S. and Samarati, P. (2007). *A Data Outsourcing Architecture Combining Cryptography and Access Control.* In CSAW '07: Proceedings of the 2007 ACM workshop on Computer Security Architecture, pp. 63–69, New York, NY, USA, ACM Press.

12. Miklau, G., and Suciu, D. (2003). *Controlling Access to Published Data Using Cryptography*. In VLDB'03: Proceedings of the 29th International Conference on Very Large Data Bases, pp. 898–909. VLDB Endowment.
13. Koshutanski, H. and Massacci, F. (2003). *An Access Control Framework for Business Processes for Web Services*. In XMLSEC '03: Proceedings of the 2003 ACM Workshop on XML Security, pp. 15–24, New York, NY, USA. ACM Press.
14. Kraft, R. (2002). *Designing a Distributed Access Control Processor for Network Services on the Web*. In XMLSEC '02: Proceedings of the 2002 ACM Workshop on XML Security, pp. 36–52, New York, NY, USA. ACM Press.
15. Harrington, A. and Jensen, C. (2003). *Cryptographic Access Control in a Distributed File System*. In SACMAT '03: Proceedings of the 8th ACM Symposium on Access Control Models and Technologies, pp. 158–165, New York, NY, USA,. ACM Press.
16. Miklau, G., and Suciu, D. (2003). *Controlling Access to Published Data Using Cryptography*. In VLDB'03: Proceedings of the 29th International Conference on Very Large Data Bases, pp. 898–909. VLDB Endowment.
17. Kwok, S. H. (2002). *Digital Rights Management for the Online Music Business*. SIGecom Exch., 3(3), 0pp. 17–24.
18. Stallings, W. (1995). *Network and Internetwork Security: Principles and Practice*. Prentice Hall.
19. Stallings, W. (2006). *Cryptography and Network Security: Principles and Practice*. Prentice Hall.
20. Stinson, D. (1995). *Cryptography: Theory and Practice*. CRC.
21. Stohr, E., and Zhao, J. L. (2001). *Workflow Automation: Overview and Research Issues*. Information Systems Frontiers, 3(3), pp. 281–296.
22. Summers, R. (1997). *Secure Computing: Threats and Safeguards*. McGraw-Hill.
23. Riehle, D., and Z¨ullighoven, H. (1996). *Understanding and Using Patterns in Software Development*. Theory and Practice of Object Systems, 2(1), pp. 3–13.0
24. Rivest, R. (1992). *RFC 1321 – The MD5 Message-Digest Algorithm*.http://www.faqs.org/rfcs/rfc1321.html.
25. Rivest, R., Shamir, A., and Adleman, L. (1978). *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*. Commun. ACM, 21(2), pp. 120–126.
26. Rossi, D. (2004). *Orchestrating Document-based Workflows with X-Folders*. In SAC '04: Proceedings of the 2004 ACM Symposium on Applied Computing, pp. 503–507, New York, NY, USA. ACM Press.
27. Rossignoli, C9f09. and Ricciardi, F., (2014) Inter-Organizational Relationships: Towards a Dynamic Model for Understanding Business Network Performance. Springer.
28. Sandhu, R. (1996). *Rationale for the RBAC96 Family of Access Control Models*. In RBAC '95: Proceeding of the 1st ACM Workshop on Role-based Access Control, pp. 1–8, New York, NY, USA. ACM Press.
29. Sandhu, R., Coyne, E., Feinstein, H., and Youman, C. (1996). *Role-Based Access Control Models*. IEEE Computer, 29(2), pp. 38–47.
30. Sandhu, R., and Samarati, P. (1994). *Access Control: Principles and Practice*. IEEE Communications Magazine, 32(9), pp. 40–48.

Ezeamasobi Chibuzor, H.C. Inyiama and Godspower I. Akawuku (2023). (2023). Systematic Literature Review: workflow models for intra-organizational and inter-organizational communication. ***NEWPORT INTERNATIONAL JOURNAL OF ENGINEERING AND PHYSICAL SCIENCES (NIJEP)* 3(2):76-92.**