Cryptographic Access Control in support of Object Level Protection

Sander Oudkerk
Agent Sierra Consultancy Services
Amsterdam, Netherlands
sander.oudkerk@agentsierra.nl

Konrad Wrona
NATO Communications and Information Agency
The Hague, Netherlands
konrad.wrona@ncia.nato.int

***Abstract*—This paper presents the results of a high-level investigation into the application of cryptographic access control in support of Object Level Protection (OLP). The investigation focused on the concept of attribute-based encryption (ABE) applied to an architecture in which OLP is realized through Content-based Protection and Release (CPR). The choice of ABE is motivated by a natural correspondence that exists between ABE and CPR. We explain this correspondence and present a detailed step-by-step description of how the concept of ABE can be applied to a number of basic use cases for information sharing in a CPR-enabled environment. The paper assumes an appropriate ABE scheme and does not contain details on cryptographic operations.**

Keywords—Object Level Protection; cryptographic access control; attribute-based encryption; information release.

I. INTRODUCTION

The concept of Object Level Protection (OLP) [1] was developed in support of the North Atlantic Treaty Organization (NATO) Network Enabled Capability (NNEC) [2]. In the OLP information architecture, protection is applied to individual data objects (or portions thereof) instead of to a collection of data objects and systems. With NATO information and network resources, which are typically separated into security domains (e.g. NATO Secret and Mission Secret network enclaves), protection is applied to a collection of data objects; this hampers information sharing because all data objects are subject to the same security and access control policy regardless of the nature of the data object (e.g. consider a NATO Unclassified document in a NATO Secret domain). The same principle applies to the practice of applying a security marking to a document as a whole, instead of (also) at the paragraph level. Furthermore, metadata are bound to data objects, and the metadata are used by protection mechanisms to determine the protection requirements for a data object. A common example is the use of Communications and Information System (CIS) security metadata, but in general any type of metadata can be used.

In an OLP architecture, a protection mechanism derives the protection requirements for a data object from the protection policy, the data object's metadata and information on the environment. This is referred to as the *OLP Model*. The protection mechanism can be of different types, e.g. an information release mechanism, an encryption function or a system access control mechanism. The information on the environment that is input to the derivation of protection requirements is specific to the type of protection mechanism, and may take a further different form depending on the access control use case. Examples of information on the environment are: current network risk signature, the set of terminal capabilities, user clearance and a system high security classification. Figure 1 illustrates the OLP Model based on CIS security metadata (CSM). The data object and metadata are assumed to be bound by means of a binding mechanism [1].

A. OLP based on NL and MAXLG

The NATO Communications and Information Agency has developed a cross-domain information exchange solution implementing the OLP Model as a part of the Medium Assurance XML-labelling Guard (MAXLG) [3]. The solution is based on the use of NATO labelling (NL) [1] – the specification of which includes the definition of both a label format and a binding mechanism – and the mediation of access requests for data objects by a cross-domain guard that enforces a release policy.

In the cross-domain information exchange solution based on NL and MAXLG the OLP protection mechanism is provided by the MAXLG. In this use case the act of protection is making the decision whether or not to release information from one (security) domain to the other (security) domain. The data object is assumed to be formatted in XML and CIS security metadata are bound to the resources comprising the XML document. The binding is realized through the NL binding mechanism. The OLP protection policy is formulated as a *release policy*, detailing the dissemination control requirements for different types of CSM (e.g. information cannot be released to the International Security Assistance Force (ISAF) domain unless it carries the category *Releasable*

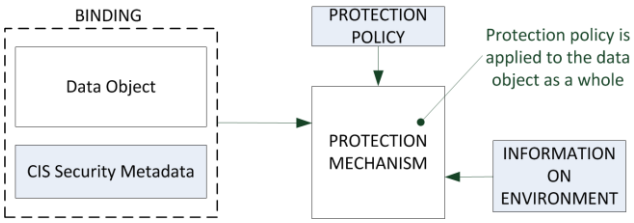

Figure 1. OLP Model: a protection mechanism derives the protection requirements for a data object from the protection policy, the data object's metadata and the information on the environment

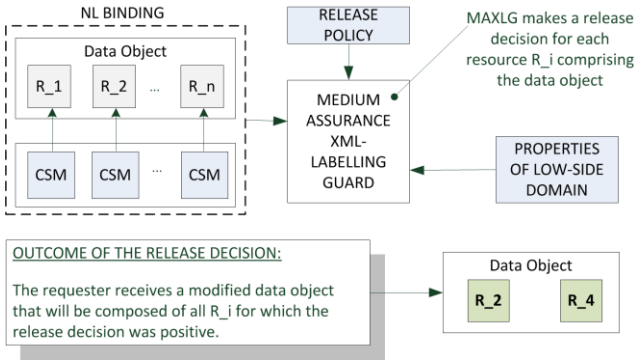

Figure 2. For each *R_i* the MAXLG bases its release decision on the CSM bound to the *R_i*, the release policy and the release level; the released data object is composed of those *R_i* for which the release decision is positive (the figure shows *R_2* and *R_4* as an example)

to ISAF). The information on the environment pertains to the (low-side) security domain to which information is released or from which information is imported, and is composed of the classification of the domain (e.g. *NATO Unclassified*) and the composition of the coalition that is active in that domain (e.g. *Sweden, Finland, ISAF*). The result of the enforcement of the release policy is the release of a data object that is composed of those resources for which the release decision was positive (e.g. in the case of a low-side security domain that operates at the classification *NATO Unclassified* and for which the active coalition includes *Sweden*, only the resources that carry the marking *NATO Unclassified Releasable to Sweden* will be released). Figure 2 shows how the model from Figure 1 is applied in the cross-domain information exchange solution based on NL and MAXLG.

II. Content-based Protection and Release

The *Content-based Protection and Release (CPR) Model* is based on Attribute-based Access Control (ABAC) and assumes that a CPR Enforcement Separation Service (CPRESS) mediates access requests to a data object. In CPR the enforcement of a protection policy is attribute-based because the protection policy is expressed in terms of attributes of the requesting user and the terminal (or system) from which the request is made.

The OLP protection mechanism is provided by the CPRESS. In the default use case of CPR the act of protection is making the decision whether or not to release information to a (combination of) user and terminal. The data object is assumed to contain information to which one can assign content properties. In CPR a data object is therefore also called an *information object*. The metadata bound to the information object consist of a set of content properties. The protection policy is called the *CPR policy*, which consists of a *release policy* (related to users) and a *protection policy* (related to terminals and the environment). The CPRESS derives the protection requirements and release conditions for an information object by correlating the CPR policy with the content properties. Both protection requirements and release conditions are expressed in terms of user and terminal attributes. The information on the environment is in the form of information on the requester of the information object, i.e. user and terminal attributes. The CPRESS makes a release decision by matching the user and terminal attributes against the release conditions and protection requirements respectively. Note that it is possible to include additional information on the environment, such as the current network risk signature, in the release decision. Figure 3 shows the elements that together comprise the CPR Model.

III. Cryptographic Access Control in a CPR Architecture

A. Motivation

The OLP and CPR Models described above do not prescribe the use of Cryptographic Access Control (CAC) in order to realize access control on a data object (or portions thereof). In case of enforcement of access control on data (or information) objects composed of a number of resources, both MAXLG and CPRESS generate a permitted view composed of only those resources to which the user (or combination of user and terminal) has access. The data objects are assumed to be unencrypted and any confidentiality protection required during transit is assumed to be provided by the network. The enforcement of access control through CAC can however complement the OLP and CPR architectures. First of all, when the data objects that are protected by the CPRESS are also encrypted, a bypass of the CPRESS does not immediately lead to unauthorized disclosure of the information represented by the data objects. This has the advantage that the assurance requirements for the CPRESS with respect to its separation functionality may be relaxed. Further, in distributed systems architectures it may not be possible for all systems and information domains (hosting data objects) to be protected by a trusted CPRESS, or to have absolute control over the dissemination of data objects. In such environments CAC on data objects may offer the required protection. Also, in certain environments the encryption of single data objects may remove the need for encryption at the network layer, or encryption of storage devices. Finally, instead of removing portions of a data object that cannot be released to a user or terminal (i.e. generating a permitted view), such portions can be released in encrypted form so that the data object can be kept intact. This has advantages in the following areas:

1. *Information management*: There is a direct mapping between released data objects and the originally requested data object in the sense that the (partially) encrypted data object is an instance of the original data object. This is not the case when sanitizing data objects because by doing so one introduces a new data object in the information system.

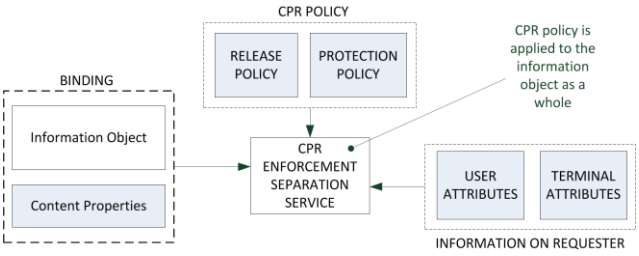

Figure 3. CPR Model and role of the CPRESS

2. *Efficiency and scalability*: Instead of mediating an access request for individual users, it is possible to distribute a data object to a group of users (in which each of the group members has a different set of attributes). Group members then decrypt the portions of the data object that they are authorized to access.

3. *Data manipulation*: Users who are not authorized to view a data object may in fact be authorized to execute specific operations on (portions of) the data object. This becomes possible by applying homomorphic encryption; see for example [4].

CAC in support of OLP can realize several protection goals related to both confidentiality and integrity protection. In this paper CAC is considered for confidentiality protection in a CPR architecture.

B. *Granular Application of the CPR Model with CAC*

Regardless of the use case or information exchange requirements that must be supported, the concept of using CAC in a CPR architecture remains the same:

1. An information object is encrypted based on the user and terminal attributes that would be required to access the information object in the default use case of the CPR Model (as depicted in Figure 3). By encrypting the information object a default denial of access is realized for all users and terminals that do not possess the required set of attributes.

2. Access to an information object is provided by making available the required key material for decryption to the user and terminal that possess the required attributes.

Note that in the default use case of the CPR Model a user and terminal request access to the information object itself, but in the case of CAC they request access to the key material that is required for decryption. Also, in the case of CAC the CPR policy is not evaluated as part of the processing of the access request, but instead to determine the requirements for encryption, and to identify (and potentially create) the associated key material.

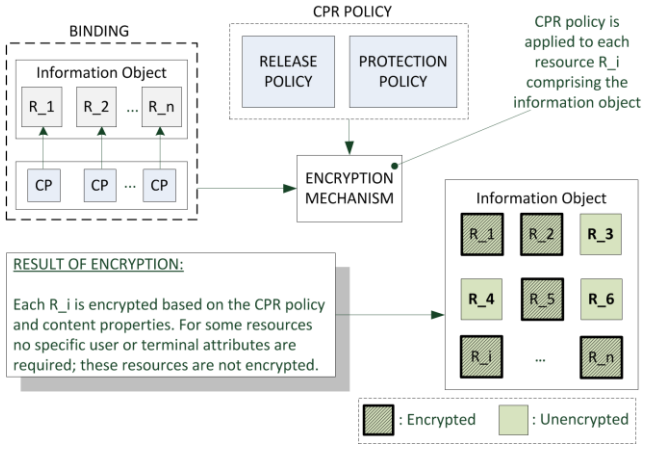

Figure 4. Each resource R_i is encrypted individually based on its content properties and the CPR policy; when the CPR policy does not require specific user or terminal attributes to access a resource, the resource is not encrypted

In case an information object is composed of multiple resources, each resource is individually encrypted based on its content properties. When two resources have different sets of content properties, each resource will be encrypted with different key material (because different sets of content properties may require different sets of user and terminal attributes). When the CPR policy requires no specific user and/or terminal attributes to access a resource, the resource is not encrypted.

The granular application of the CPR Model with CAC is illustrated in Figure 4. Note that the figure does not show the use of encryption keys, however each resource is encrypted with a key that is associated with a set of user and terminal attributes (which is determined by the content properties and the CPR policy).

Note that in Figure 4 the binding of the content properties to the information object is not shown for the resulting (partially) encrypted information object. Whether or not the content properties must remain available after encryption depends on the use case, and one may also choose to encrypt the content properties themselves. In the case that content properties or (resources in) the information object have been encrypted, a cryptographic binding will have to be regenerated as the original digital signature is no longer valid. In this paper it is assumed that the signing of information objects is based on the use of attribute-based (digital) signatures.

Figure 5 illustrates the decryption of the (partially) encrypted information object from Figure 4. A user is able to decrypt a resource if the user and terminal attributes provide access to the key material required for decryption. The figure does not show the use of decryption keys, however it illustrates that the decryption of a resource depends on the user and terminal attributes of the requester. The result of the decryption process is an information object composed of resources of which only those for which user and terminal are not authorized remain encrypted. The binding of the content properties to the information object is not shown for reasons mentioned in the paragraph above.

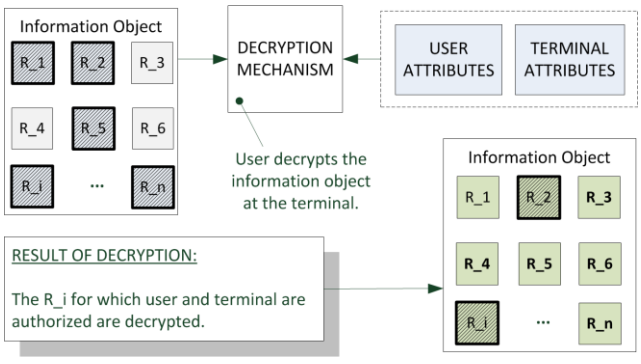

Figure 5. If user and terminal are authorized (based on their attributes) to access a resource R_i, the required decryption key is made available and the resource is decrypted; resources for which user and terminal are not authorized remain encrypted

IV. TARGETED INFORMATION AND NETWORK ARCHITECTURE

CAC can be applied in a CPR architecture in different ways depending on use case, information exchange scenario and key management architecture. The main components of the assumed high-level information and network architecture are shown in Figure 6.

All information objects are stored via *cloud storage* and can be uniquely identified. The cloud is managed by a NATO information management authority.

The CPRESS mediates all requests for information objects in the cloud, as well as the upload or push of information objects. The CPRESS is managed by NATO.

The networking infrastructure is assumed to be based on the concept of Protected Core Networking (PCN) [5]. In particular this implies that it is possible to determine the level of confidentiality protection offered by any network path in the Protected Core Network. In this paper it is assumed that the Protected Core Network can enforce the requirements that communication between terminals is possible only via the CPRESS and that users can request, receive, view and process information objects only on terminals that are connected to the Protected Core Network.

An information object can be stored in the cloud. When stored in the cloud it is encrypted based on the CPR policy. An information object can be also in transit in the Protected Core Network. Although the path through the network can be selected based on required confidentiality protection, while in transit an object remains encrypted at the object level (based on the CPR policy). This accommodates potential unavailability of a sufficiently protected network path. Alternative approaches that base the confidentiality protection of an information object on the combination of encryption at the object level and the protection offered by the Protected Core Network are not considered in this paper. Finally, an information object can reside at a user's terminal. Here, three states are distinguished:

1. The information object is being created by the user and it is assumed that the user assigns only content properties for which the user and terminal are authorized, hence the object can reside at the terminal.
2. The information object is in encrypted state (based on the CPR policy). In this state the object can reside at any terminal.
3. The information object, or a part thereof, is being viewed or otherwise processed by the user, which requires the object to be in a (partially) decrypted state. In this case the object resides at a terminal that is authorized to decrypt the object and handle the object in its unencrypted state.

Note that when an information object is moved between terminals by a user through the use of removable media, the information object is assumed to be encrypted.

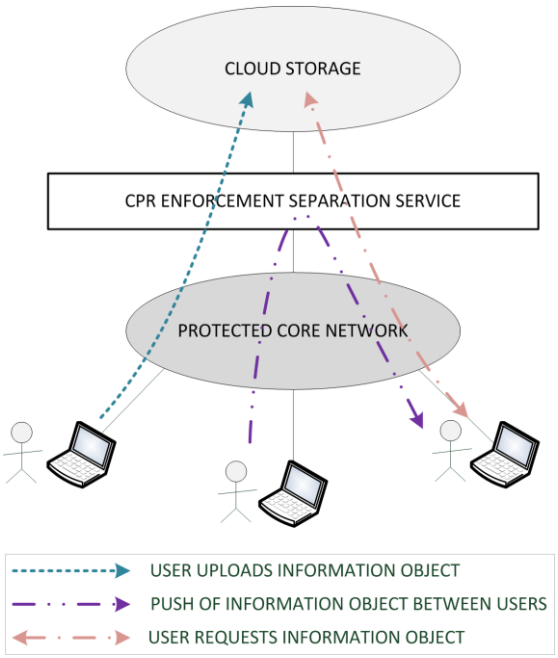

Figure 6. Information flow for the basic information exchange use cases *information object upload*, *information object push* and *information object request*

A. Assumptions

In our CPR Model we assume that users and terminals are authenticated, i.e. both user and terminal are associated with a set of attributes and this association is verifiable by the CPRESS. Sessions between the CPRESS and users and terminals are stateful in the sense that the CPRESS knows the attributes of user and terminal to which an information object is released within a given authenticated session. The CPRESS keeps an information object upload and release record in which for each information object it records the user attributes of the user who uploaded or requested the information object. Users and terminals belong to a federation of systems for which a federated identity management capability has been established (which allows for the mutual recognition of user and terminal attributes). An information object is created by a user at a terminal connected to the Protected Core Network. Upon creation of the object, the user assigns content properties and binds the content properties to the object. This requires interaction with the NATO Metadata Binding Service (NMBS). Encryption of the information object is then executed by the user at the terminal, after which the object is uploaded to the cloud. The user is trusted to select the correct content properties. The (private) key material that the terminal requires to decrypt information objects is stored securely at the terminal, for example in a Trusted Platform Module (TPM) [6]. This key material can be accessed only by trusted processes and users, and it is assumed that only authorized users can initiate decryption or encryption of information objects on a given terminal. The process that determines whether or not a user is authorized is not within the scope of this paper. A terminal securely wipes symmetric-key material after use (see Section V.C for a description of how symmetric-key encryption is used). If a terminal is able to host an information

object with a given set of content properties, the terminal also has the capability to encrypt or decrypt such an information object (according to the CPR policy). In addition to upload by users it is possible for information objects to be imported into the cloud. If information objects are imported from a federation partner network (in the federation of systems), it is assumed that the information object has already been encrypted by the federation partner based on the CPR policy that is in effect in the partner network.

As it is not known in advance to which threats an information object will be exposed with respect to its confidentiality and integrity, either in transit or when stored at a terminal, it is assumed that the strength of the encryption is such that it sufficiently mitigates the risk resulting from such threats. An alternative approach would be to base the cryptographic configuration (including the assurance of the implementation of the cryptographic mechanisms used, and cryptographic parameters such as key length and type of algorithm) on the expected risk (which is determined through risk assessment) for a particular use case. The drawback of such a solution is that if the use case changes then either the information object will have to be re-encrypted based on the risk associated with the new use case, or the CPRESS must not mediate the information object in the new use case. The decision to mediate or not requires the CPRESS to be able to determine the risk associated with the new use case. Determining the cryptographic configuration based on the risk associated with a particular use case requires an extension of the CPR protection policy to include cryptographic requirements. This is not within the scope of this paper.

B. *Information Exchange Use Cases*

Three basic information exchange use cases are distinguished for which the realization of CAC is investigated in Section VII. In an *information object upload* a user uploads an information object to the cloud after creation or modification. Before upload the information object is encrypted by the user. In an *information object push* a user pushes an information object to another user. In an *information object request* a user requests an information object from the cloud. In all three use cases the CPRESS mediates the information flow. The information flow for each use case is illustrated in Figure 6.

V. ATTRIBUTE-BASED ENCRYPTION

Attribute-based encryption (ABE) [7] can be viewed as a generalization of identity-based encryption (IBE) [8]. Both systems are based on asymmetric encryption, and involve a central key generation/distribution authority. In IBE, encryption of an information object is based on the identity of the intended receiver of the information object. The identity of the receiver can be viewed as a single attribute, possession of which the receiver must prove in order to decrypt the information object. In ABE, encryption of an information object is based on a set of attributes, with the understanding that only users who can prove possession of those attributes are authorized to decrypt the information object.

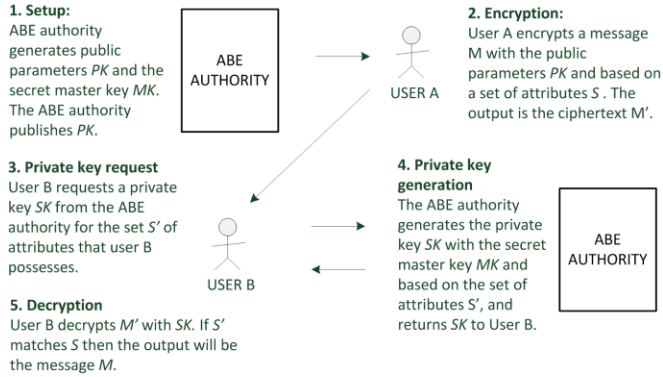

Figure 7. Simplified description of the concept of ABE; the exact setup process and the process of matching attributes depends on whether CP-ABE or KP-ABE is chosen

A. *General Principle of Attribute-based Encryption*

ABE uses asymmetric encryption meaning that different keys are used for encryption and decryption. Encryption of an information object is based on a set of attributes and takes place with a public key belonging to a central key generation/distribution authority, called the ABE authority. The public key is also referred to as the 'public parameters' (*PK*). A user who wants to encrypt an information object *M* based on a set of attributes *S* will input *PK*, *S* and *M* into the ABE encryption algorithm that is used. Note that the set of attributes can vary from a list of attributes to a Boolean formula on attributes, and even an access matrix [7] [9]. The latter two can be used in a Ciphertext-Policy ABE (CP-ABE) system.

Each user is able to request a private decryption key (*SK*) from the ABE authority. The private key is uniquely associated with a set of attributes for which the user is authorized (if two users are authorized for the exact same set of attributes, then they will in principle receive identical private keys). The private user keys are generated from a secret master key (*MK*) that is in the possession of the ABE authority. Here too, the set of attributes with which the private key is associated can range from a list of attributes to an access policy (which expresses to which (combinations of) attributes the private key provides access). The access policy can be used in a Key-Policy ABE (KP-ABE) system.

Figure 7 presents a simplified description of the concept of ABE. Specific details will depend on whether CP-ABE [9] or KP-ABE [7] is chosen. In the remainder of this paper, the roles of users *A* and *B* in Figure 7 are referred to as *encryptor* and *recipient* respectively. Note that in CPR the terminal can also fulfil these roles.

B. *Ciphertext-Policy ABE*

When encrypting an information object *M* in CP-ABE [9] [10], the encryptor (or information management authority) specifies an access policy *p* on the recipient's attributes that determines the attributes a recipient must possess in order to decrypt the information object. In CP-ABE the recipient is assumed to possess a set of attributes *RA*. Given an encryption algorithm *E* the encryptor then encrypts *M* with *p* and the

public parameters *PK*, so that the resulting ciphertext *M'* is equal to:

$$E(PK,(p,M))$$

A recipient in possession of a private key *SK* that has been generated based on *RA* is able to decrypt *M'* and produce *M* if and only if *RA* satisfies the policy *p* (notation: *p(RA) = 1*).

In the case of CP-ABE steps 2 and 4 in Figure 7 are modified as follows: in step 2 user *A* encrypts *M* with *PK* and access policy *p*, and in step 4 the private key *SK* is generated with *MK* and based on the set of attributes *RA*.

C. Key-Policy ABE

When encrypting an information object *M* in KP-ABE [7] [11], an access policy *p* is placed in the decryption key *SK* whereas *M* is encrypted based on a set of attributes *MA* that are associated with *M*. Given an encryption algorithm *E* the encryptor then encrypts *M* with *MA* and the public parameters *PK*, so that the resulting ciphertext *M'* is equal to:

$$E(PK,(MA,M))$$

The recipient's private key *SK* is generated based on an access policy *p* that determines to which message attributes *SK* provides access. This access policy *p* is generated per user and expresses for which message attributes *MA* the user is authorized. The recipient is able to decrypt *M'* (into *M*) with *SK* if and only if *MA* satisfies *p* (notation: *p(MA) = 1*).

In the case of KP-ABE steps 2 and 4 in Figure 7 are modified as follows: in step 2 user *A* encrypts *M* with *PK* and the set of attributes *MA*, and in step 4 the private key *SK* is generated with *MK* and access policy *p*.

VI. Application of Attribute-Based Encryption to CPR

The application of ABE to CPR that is described in this paper applies to the case of granular application of CAC, meaning that ABE is applied to each resource of which an information object is composed. In the description of the use of ABE in this section however, the process is explained for an information object consisting of a single resource; the application to multiple resources is a matter of repeating the process.

In the remainder of this paper the use of only CP-ABE for CPR is considered (for reasons given below). In this paper it is assumed that the signing of information objects is based on the use of attribute-based (digital) signatures. Information objects are signed so that their integrity can be verified by the recipient, and so that the CPRESS can verify that an encryptor is authorized to create or modify an information object (in other words: the encryptor has *write* access).

The exact details of the use of attribute-based signatures are not within the scope of this paper (see [12] for more detail), however it is assumed that the ABE authority is able to generate and distribute the required key material to those users who are authorized to create or modify information objects.

A. Suitability of CP-ABE for CPR

In CPR content properties are associated with an information object *M* and by correlating the content properties with the CPR policy, the release conditions and protection requirements for the information object are derived. The release conditions are formulated in terms of user attributes that a user must possess in order to access *M*, and the protection requirements in terms of terminal attributes the terminal must possess in order to store or process *M*. As such, the release conditions can be viewed as an access policy *p_U* and the protection requirements as an access policy *p_T*. Thus, in CPR a user or terminal can access *M* if their attributes satisfy *p_U* and *p_T* respectively. In order for a user to access *M* on a given terminal, both *p_U* and *p_T* must be satisfied.

B. Suitability of KP-ABE for CPR

The goal of CPR is to determine whether or not a combination of user and terminal attributes provides access to an information object that has a given set of content properties associated with it. Translating this to KP-ABE leads to the following two options:

1. An encrypted information object *M'* is associated with attributes in the form of content properties, i.e. the message attributes *MA* are equal to the content properties. The recipient's private key is then associated with a policy *p_CP*, which specifies to which content properties the recipient has access. The recipient is able to decrypt *M'* into *M* if *MA* satisfies *p_CP*.
2. An information object is associated with a simple list of user and terminal attributes *MA* and the private key is associated with an access policy *p* that specifies for which terminal and user attributes the key provides access.

Regarding option 2, in CPR the recipient possesses a specific set of attributes *S* instead of an access policy *p*, hence the recipient can be given a private key only with policy *p* equal to *S*. (Otherwise the recipient will have access to content for which it is not authorized.) This leads to a simple case of CP-ABE in which the information object is encrypted with an access policy consisting of a simple list of attributes. Therefore option 2 of KP-ABE is not considered further.

Regarding option 1 the following disadvantages can be observed:

1. In CPR it is assumed that content properties do not change frequently. This implies that once a private key (associated with *p_CP*) has been distributed, the recipient will from that moment on always have access to information objects that have been encrypted with content properties specified by *p_CP* even if the recipient is no longer authorized.
2. Re-encryption of information objects will not solve the issue described in 1) because encryption will be done based on the same set of content properties *CP*.

In order to avoid the issue described in 1) above, one would have to enforce authentication of user and terminal for each

request for access to an information object, re-compute *p_CP* and distribute new private keys to user and terminal. This approach however does not prevent access to information objects that are accessed outside the control of the CPRESS. Also it clearly defeats the purpose of CAC in support of CPR. For the reasons outlined above KP-ABE is not considered further in this paper.

C. Using ABE with symmetric-key encryption for CPR

Although the explanation of CP-ABE describes the asymmetric encryption of an information object *M* based on an access policy *p*, in the remainder of this paper it is assumed that *M* is encrypted with a symmetric key *K* into ciphertext *M'*, while CP-ABE is used to encrypt *K* into *K'*. Here, *K* is assumed to be an information object itself, however it inherits the content properties from the information object that it encrypts, and is therefore encrypted with the same attributes that *M* would be encrypted with if CP-ABE is applied to *M*. A recipient uses its private key to decrypt *K'* and then uses *K* to decrypt *M'* into *M*.

The reasons for using symmetric-key encryption are threefold:

1. *Computational advantage:* Symmetric-key encryption is substantially faster than asymmetric key encryption (ABE is an example of the latter). This is a particular advantage in case there is a change in the CPR policy that requires all information objects to be re-encrypted.
2. *Less impact when CPR policy changes:* If the CPR policy changes, re-encryption is not necessarily required of the information object; if an encrypted information object *M'* has not previously been shared with recipients that are now no longer authorized to access *M'* under the new CPR policy, it suffices to re-encrypt the symmetric key *K* under the new CPR policy.
3. *Confidentiality of CPR policy:* Use of symmetric-key encryption removes the need for the ABE authority to send *p_U* and *p_T* to the encryptor.

D. Blinding of private keys

A challenge in any ABE system is to prevent recipients from realizing collusion attacks [10]. A collusion attack is realized when two recipients collude to combine their private keys and create a new secret key with the objective of decrypting an information object that neither recipient would be able to decrypt with its private key. A technique to avoid collusion attacks is the so-called *blinding* of keys through randomization values [9]. The essence of this technique is that only private keys that have been issued to the same recipient can be combined. Given that in a CPR architecture it must not be possible that a user colludes with another user, or that a terminal colludes with another terminal, it is assumed that private keys are blinded.

E. Encryption of the symmetric key

When applying ABE to CPR a private key can be issued to individual users and terminals but also to a specific combination of user and terminal. Given that in CPR users and terminals are considered separate entities, this paper assumes that private keys are issued to individual users and terminals only. A consequence of this assumption and the blinding of private keys is that a symmetric key *K*, which is used for the encryption of an information object *M*, must not be encrypted based on one single policy in which release conditions and protection requirements are combined. Instead, *K* is encrypted twice based on separate user and terminal access policies:

1. The first time it will be encrypted based on access policy *p_T*. The result is ciphertext *K'*.
2. The second time it will be encrypted based on access policy *p_U*. The result is ciphertext *K''*.

In order for a user *U* to decrypt *K''* into *K* on a terminal *T*, first *K''* is decrypted into *K'* using the private key of *U*, then user *U* initiates decryption by the terminal and *K'* is decrypted into *K* using the private key of *T*.

The order of encryption must be as described above because if the order were reversed it would allow a user to first apply decryption of *K''* into *K'* by an authorized terminal, and then potentially complete the decryption by the user of *K'* into *K* on a terminal that is not authorized for *M*.

VII. ABE Applied to the Basic Information Exchange Use Cases

In order to apply ABE to the basic information use cases discussed earlier, a number of information management components are added to the architecture. The ABE authority (ABEA) generates private and public key material, as well as the keys that are used for symmetric encryption of information objects. In this paper it is assumed that the ABEA has completed the initial setup phase (see [9] for more details). The CPR Policy Administration Point (PAP) manages the CPR policy and deduces the release policy (*p_U*) and protection policy (*p_T*) for information objects. The NMBS makes content property catalogues available to users and terminals. (The content property catalogue contains the content properties that users and terminals are allowed to access.) In order to do this, the NMBS keeps a copy of the current CPR policy, which it obtains from the CPRESS. The NMBS also verifies digital signatures created by users.

The Attribute Management System (AMS) is responsible for the management of user and terminal attributes and facilitates authentication of users and terminals as well as the verification of user and/or terminal attributes. The ABEA, NMBS and CPRESS will communicate with the AMS.

A user will communicate with the NMBS, CPRESS and ABEA. Further it is assumed that the CPRESS will be the central component, meaning that all communication between the management components will be mediated by the CPRESS. Note that the purpose of this assumption is to simplify the communication model. The NMBS and ABEA are able to authenticate users and terminals. All management components are trusted and able to authenticate one another.

Figure 8 shows the management components and the interfaces between them. The arrows denote the logical

information flows. (The interaction between the AMS and the other management components is not shown.)

A. Information object upload

The steps that comprise the use case *information object upload* are:

1. The user *U* who is logged on to terminal *T* requests a content properties catalogue from the NMBS.
2. The NMBS requests the CPR policy from the CPR PAP and correlates the user and terminal attributes with the CPR policy, resulting in the content properties catalogue for *U* and *T*.
3. The user creates an information object *M* and assigns a set of content properties *CP*.
4. User *U* requests a symmetric key *K* from the ABEA for the set *CP*.
5. The ABEA requests that CPRESS provide *p_U* and *p_T* based on the set *CP*.
6. The CPRESS requests the current CPR policy *p_CPR* from the CPR PAP and correlates *CP* with *p_CPR* resulting in *p_U* and *p_T*. The CPRESS sends *p_U* and *p_T* to the ABEA together with an identifier *p_CPR_ID* that identifies the version of *p_CPR* in effect at the time of generation of *p_U* and *p_T*.
7. The ABEA generates a symmetric key *K* and assigns it a key identifier *K_ID* that is linked to *p_CPR_ID*.
8. The ABEA encrypts *K* with *p_U* and *p_T* into *K''* giving it the same key identifier as *K*, i.e. *K''_ID* = *K_ID*.
9. The ABEA generates a random value *RV* and encrypts it into *RV'* using *K*. Then it sends *K'', RV* and *RV'* to user *U* on terminal *T*.
10. User *U* requests a private key from the ABEA.
11. The ABEA generates a private key based on the attributes of *U* and sends it to *U*. (Note that it is assumed that *T* has already been issued a private key based on the attributes of *T*, and that this private key is securely stored at *T*).
12. User *U* decrypts *K''* into a key *K_U* and retains a copy of *K''*. Note that because of step 2 *K_U* should be equal to *K*. However, the CPR policy may have changed between steps 2 and 6 in such a way that *U* and/or *T* are no longer authorized for *CP*.
13. User *U* then encrypts *RV* with *K_U* into *RV_KU'* and compares *RV_KU'* with *RV'*. If the values match, user *U* concludes that *K_U* = *K*. Note that if *U* is no longer authorized because of a change in the CPR policy occurring between steps 2 and 6, decryption into *K* will fail in which case *U* returns to step 1 and has to modify *M* based on a new content properties catalogue .
14. Assuming that *K_U* = *K*, user *U* assumes the role of encryptor and encrypts *M* into *M'* using K.

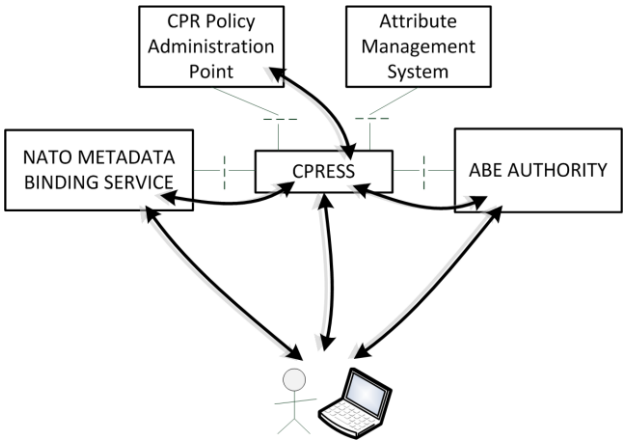

Figure 8. Interfaces between the management components; the arrows indicate logical information flow; the AMS offers authentication and attribute verification services to the CPRESS, NMBS and ABEA; the user communicates with NMBS, CPRESS and ABEA

15. The encryptor binds *CP, K'', RV* and *RV'* to *M'* into a single object *O*, assigns a unique information object identifier to *O*, and signs *O*.
16. The encryptor establishes a session with the CPRESS and initiates upload of *O* to the cloud. The CPRESS proxies the upload of *O*, meaning that the CPRESS will complete the upload of *O* to the cloud.
17. The CPRESS asks the NMBS to verify the digital signature created by *U*.
18. The NMBS verifies the digital signature created by *U* and communicates the result to the CPRESS.
19. If the result from 18) is negative, the CPRESS will delete *O* and log the event. If the result is positive, the CPRESS concludes that *U* is authorized to create and upload information objects and will move to the next step.
20. The CPRESS checks the version of the CPR policy in effect and verifies *K''_ID* against *p_CPR_ID*. Then one of the following steps occurs:

a) If *K''* was issued for an older version of the CPR policy, the CPRESS re-computes *p_U* and *p_T* for the content properties *CP* and compares them with the policies used by the ABEA for the previous version of the CPR policy. If they are different, the CPRESS informs *U* that *K''* has expired and returns *O* to *U*. The user *U* then decrypts *M'* into *M* with *K* and returns to step 4.

b) If *K''* was issued for the current version of the CPR policy, the CPRESS stores *O* in the cloud.

B. Information object request

The steps that comprise the use case *information object request* are:

1. The user *U* who is logged on to terminal *T* requests a private key from the ABEA.

2. The ABEA generates a private key based on the attributes of *U* and sends it to *U*. (Note that it is assumed that *T* has already been issued a private key based on the attributes of *T*, and that this private key is securely stored at *T*.)
3. User *U* requests an information object *O* from the cloud. The request is mediated by the CPRESS.
4. The CPRESS retrieves *O* from the cloud and sends it to *U* on terminal *T*. Note that the CPRESS can first determine whether or not *U* and *T* are authorized to view *O* from the content properties *CP* that are bound to *O*, however because *O* is encrypted this step is not executed .
5. User *U* assumes the role of recipient and asks the NMBS to verify the binding of *O*. If the binding is verified, *U* decrypts *K''* into a key *K_U*.
6. The recipient encrypts *RV* with *K_U* into *RV_KU'* and compares *RV_KU'* with *RV'*. If the values match, the recipient concludes that it is authorized to access *M* and decrypts *M'* into *M*. Note that the recipient may skip this step and decide based on the resulting decryption of *M'* whether or not decryption was successful.
7. If the recipient wishes to make changes to *M*, it starts step 1 of the use case *information object upload*.

Note that the CPRESS compares information object identifiers to determine whether or not the upload of an object *O'* based on a modified version of *M* constitutes a new information object or simply a new version of *O*. In case of the former, the encryptor will have to assign a new information object identifier to *O'*. The management of information object identifiers as well as the management of versions of one and the same information object is assumed to be done by the information management authority of the cloud, and is not further discussed in this paper.

C. *Information object push*

The use case *information object push* can start in one of two ways:

1. A user *U* who is logged on to terminal *T* creates an information object *O* and pushes it to user *V* on terminal *W*.
2. A user *U* who is logged on to terminal *T* requests an information object *O*, possibly modifies it, and pushes it to user *V* on terminal *W*.

In the case of option 1) the following steps are executed:

a) User *U* first executes the use case *information object upload* with the difference that in step 16 *U* does not initiate an upload but instead requests that the CPRESS push *O* to user *V* on terminal *W*.

b) The CPRESS *uploads* new information object *O* to the cloud, but then proceeds to push *O* to user *V* on terminal *W*.

c) Upon receiving *O* user *V* proceeds with steps 5, 6 and 7 from the use case *information object request*.

In the case of option 2) user *U* first executes the use case *information object request*. Then the steps from option 1) above apply with the difference that if *U* has modified *M* in such a way that a new version of *O* will be pushed to user *V*, the CPRESS will interact with the information management system in order to store the information object as a new version of *O*, after which the new version will be pushed to user *V* on terminal *W*.

VIII. FUTURE WORK

The research presented in this paper only touches the surface of the concept of ABE and a number of areas remain to be investigated.

Use of Digital Signatures: The NMBS offers the services of binding verification and binding generation, and relies on a regular Public Key Infrastructure (PKI) in which public key certificates are bound to identities. In an ABE system however, the concept of identity is replaced with that of (possession of) a set of attributes. In an ABE system strong bindings are realized with attribute-based signatures and certificates are associated with a set of attributes. The application of attribute-based signatures must be further investigated and the interaction with the NMBS for the basic use cases be must be revisited.

Determination of the Content Properties Catalogue: We assume that the NMBS determines the content properties catalogue by correlating the user and terminal attributes with the CPR policy. This process however is not yet well defined and further research is required in order to determine its characteristics and feasibility.

Different Types of Access: In this paper the typical types of access rights a user can have on an information object, *no access*, *read* and *write*, are considered. If a user's private key decrypts *K''* into the correct symmetric key *K*, then the user has *read* access. If the user in addition to the above is able to create a verifiable digital signature, the user also has *write* access. If none of the above applies, the user has *no access*. Note that this approach can be further extended by separating decryption keys from encryption keys and by enforcing additional access control on metadata associated with the key material. It is recommended that further research be carried out to determine ways of realizing the aforementioned access rights (see [13] and [14] for more information on potential approaches).

Key Revocation: In the description of the use case *information object upload* the concept of assigning a version identifier to the CPR policy *(p_CPR_ID*) and the symmetric key (*K_ID*) was introduced. Further it was explained how the CPRESS can verify whether or not a change in the CPR policy should prevent a user from creating or modifying an information object. Although this approach helps to prevent the creation or modification of objects based on keys that are no longer valid (i.e. *revoked*) it does not prevent said user from accessing already distributed information objects. The application of ABE to CPR is subject to the same considerations regarding key revocation as a regular

application of ABE, with the difference that in the CPR architecture, a key revocation process must be started with each change of the CPR policy. Despite the use of symmetric keys, the generation of new symmetric keys and the re-encryption of all information objects in the cloud will still be time-consuming. In this paper it is assumed that the CPRESS is able to determine whether or not an information object has been uploaded or requested by a user who after the change in the CPR policy is no longer authorized for said information object. Together with the use of the key identifier, the CPRESS should be able to determine which information objects need to be re-encrypted. The revocation of private keys cannot be enforced by re-encryption. A potential solution that limits the re-use of revoked private keys is to issue 'time-based private keys', i.e. private keys that expire after a certain period of time.

Strength of Encryption: The cryptographic configuration used to encrypt information objects can be based on the expected risk for a particular use case. The correspondence between a risk level and a particular cryptographic configuration can then be captured in the CPR protection policy. It is recommended that further research be carried out into the correspondence between risk levels and cryptographic configurations, as well as how to capture the corresponding requirements in the CPR protection policy.

Federated Distributed ABE: Although we assume a federated identity management capability, the use of a federated distributed ABE needs to be investigated [15].

Scalability: Scalability is related to the use of a federated distributed ABE system. Further research is required in this area in order to determine if a solution based on a single-authority ABE system scales to a federated distributed ABE system. The issue of scalability is already apparent in the case of a single-authority ABE system, for which the number of user and terminal attributes that comprise the attribute space has an impact on the efficiency of an ABE system [9]. Further research is required in order to determine the impact of potential scalability issues when using a federated ABE system in NATO.

Encryption of Content Properties: In architectures in which access control is enforced based on metadata, one can choose to make such metadata accessible only to authorized access control decision functions [14]. One can for example extend the basic use cases with the encryption of the content properties.

Use of Binding Mechanisms: This paper describes the application of ABE at a high level and does not mandate the use of specific technology. This is also the case for the binding mechanism. Experimentation with available binding mechanisms (such as the NL binding mechanism [1]) through prototyping is recommended, in order to assess their suitability to support the architecture described in this paper.

Implications for the Use of CIPE: The Content Inspection and Policy Enforcement (CIPE) service is one of the core services to support the High Assurance ABAC Guard (HAAG) [16]. In order for content (that is captured in an information object) to be inspected, the CIPE service requires the content to be decrypted. In an ABE system it is possible to express in a ciphertext policy that decryption is allowed by services or systems that possess a certain attribute, however the determination of the risk associated with such an approach as well as its computational feasibility requires further research.

Interoperability: Using ABE for the granular application of CAC implies that a recipient may not be able to decrypt all the resources comprising an information object. This may lead to interoperability issues when the recipient views the resources with an application that is not able to process and render encrypted data. The impact of the granular application of CAC on interoperability for the applications that are foreseen to be used by NATO must be investigated.

REFERENCES

[1] S. Oudkerk, I. Bryant, A. Eggen, and R. Haakseth, "A Proposal for an XML Confidentiality Label Syntax and Binding of Metadata to Data Objects," in *NATO Research and Technology Organization Symposium on Information Assurance and Cyber Defence*, 2010.

[2] A. Domingo and H. Wietgrefe, "A NNEC-compliant approach for a Future Mission Network," in *Proc. of the Military Communications Conference (MILCOM)*, 2012.

[3] K. Wrona, S. Oudkerk, and G. Hallingstad, "Designing medium-assurance XML-labelling guards for NATO," in *Proceedings of the Military Communications Conference (MILCOM)*, 2010.

[4] C. Fournet, J. Planul, and T. Rezk, "Information-flow types for homomorphic encryptions," in *Proceedings of the 18th ACM conference on Computer and communications security (CCS)*, 2011, pp. 351–360.

[5] G. Hallingstad and S. Oudkerk, "Protected core networking: an architectural approach to secure and flexible communications," *Communications Magazine, IEEE*, vol. 46, no. 11, pp. 35–41, 2008.

[6] Trusted Computing Group (TCG), "Trusted Platform Module Library Part 1: Architecture," 2012.

[7] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," *Proceedings of the 13th ACM conference on Computer and communications security - CCS '06*, p. 89, 2006.

[8] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proceedings of CRYPTO 84 on Advances in cryptology*, 1985, pp. 47–53.

[9] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Public Key Cryptography–PKC 2011*, 2011, pp. 53–70.

[10] D. Boneh, A. Sahai, and B. Waters, "Functional Encryption : A New Vision for Public-Key Cryptography," *Communications of the ACM*, vol. 55, no. 11, pp. 56–64, 2012.

[11] D. Boneh, A. Sahai, and B. Waters, "Functional encryption: Definitions and challenges," *Theory of Cryptography*, pp. 253–273, 2011.

[12] H. Maji, M. Prabhakaran, and M. Rosulek, "Attribute-Based Signatures," in *Topics in Cryptology – CT-RSA 2011 SE - 24*, vol. 6558, A. Kiayias, Ed. Springer Berlin Heidelberg, 2011, pp. 376–392.

[13] H. Balinsky and S. Simske, "Differential access for publicly-posted composite documents with multiple workflow participants," in *Proc. of the 10th ACM symposium on Document Management*, 2010, pp. 115–124.

[14] M. Kiviharju, "Towards Pervasive Cryptographic Access Control Models," in *SECRYPT*, 2012.

[15] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Advances in Cryptology–EUROCRYPT 2011*, 2011, LNCS, vol. 6632, pp. 568–588.

[16] K. Wrona and G. Hallingstad, "Development of High Assurance Guards for NATO," in *Military Communications and Information Systems Conference*, 2012.