Towards Pervasive Cryptographic Access Control Models

Mikko Kiviharju
Finnish Defence Forces Technical Research Centre, Lakiala, Finland

Keywords: Cryptographic Access Control, CAC, Permissions, Access Control, CBIS.

Abstract: Access control lies at the heart of any technical information security and information assurance system. Access control is traditionally enforced by reference monitors, which are assumed to be able to reliably monitor and mediate all traffic from users to objects. An alternative view to enforcement is cryptography, referred to as cryptographic access control (CAC). CAC has gained popularity with the emergence of distributed computing, especially cloud computing and "everything as a service". CAC is not a formal model, but an enforcement paradigm. In this paper we propose an extension to the current CAC framework and discuss the limits, where it is in general feasible to extend CAC as a paradigm over reference monitors.

1 INTRODUCTION

Cryptographic access control (CAC) paradigm refers to replacing reference monitors (RM) by cryptography in enforcing access control. A reference monitor is a common name for an active process checking all access requests aimed at protected resources. First explicit examples of the CAC-paradigm date back three decades, and some significant advances have been made technologically since then, notably the introduction of attribute-based cryptography (Bethencourt, 2007); (Khader, 2007). However, the profound difference between a reference monitor as an active component and cryptography as a passive technique suggests rather far-reaching conclusions.

CAC has gained popularity due to the rise of distributed computing systems, described by the lack of ability to place reference monitors close to the data (or at all reachable from the data). Examples include digital rights management (DRM), cloud computing and high-assurance environments:

- In a typical cloud computing environment, the data owner has to place her trust concerning the security of her information on the cloud service provider, whose systems are usually not possible to be audited clearly. However, distributed information can be nearly impossible to erase, as is the case for example with different personal information in the internet.
- Distributors of copyrighted material cannot control the access for their content with trusted reference monitors, once the content is purchased, as the processing environment is owned by the user of the material.
- Military environments with classified information have high assurance needs, yet high-assurance reference monitors are expensive, rigid, highly application-specific and prone to become bottlenecks.

Current CAC-schemes, on the other hand, are lacking mainly in two senses:

- The "depth" of cryptographic enforcement – data might be cryptographically protected, but first-layer of key management still requires reference monitors;
- Access rights formalization, resulting in human responsibility in defining, designing and implementing more complex access right types;

In this paper we propose an extended framework for CAC aimed at the possibilities for extending the paradigm and lessening the dependence on reference monitors. The framework is based on the decomposition of access rights into different levels of metadata and basic types of access rights.

This paper is organized as follows: chapter two reviews some of the related work in the area; chapter three motivates our setting and explains the scenario; in the fourth chapter we present our model and finally chapter five concludes the paper.

Kiviharju M..
Towards Pervasive Cryptographic Access Control Models.
DOI: 10.5220/0004042502390244
In *Proceedings of the International Conference on Security and Cryptography* (SECRYPT-2012), pages 239-244
ISBN: 978-989-8565-24-2
Copyright © 2012 SCITEPRESS (Science and Technology Publications, Lda.)

2 RELATED WORK

The basic idea of CAC is simple – almost every piece of information is encrypted and signed. The simplicity of the idea is why models, schemes, implementations and even standards abound. The previous work on CAC can be roughly divided in two: symmetric and asymmetric key methods. Asymmetric key methods, such as XML encryption, do not generally formalize their access control models.

Symmetric-key CAC-schemes stem from the world of DRM and broadcast encryption, distributed file servers, outsourced data and cloud computing. Symmetric-key CAC-schemes use formalization to show the optimality of their metrics (Naor, 2001); (De Capitani di Vimercati, 2007) or that the described method actually does implement the desired policy (De Capitani di Vimercati, 2007). Formalizations are not on the access control concept level, the way MAC and RBAC have been formalized (Bell, 1973); (Ferraiolo, 1992).

A major advance within the CAC paradigm in asymmetric-key methods was achieved with the introduction of attribute-based cryptography. Attribute-based cryptogrpahy can enforce read-rights very flexibly via ABE and (in principle) write-rights through ABGS.

ABE encodes subject properties, called attributes (such as rank, clearance, etc.) to a set of keys, and the object access control list is used as a logical formula to encrypt the actual plaintext.

The CBIS-concept (Content-based Information Security) experimented by US DoD between 2000 and 2005 as an Advanced Concept and Technology Demonstrator (McGovern, 2001); (Savoie, 2004) and researched later e.g. in (Kiviharju, 2010) was based on similar threat model we use here.

In conjunction with CBIS it was presented the idea of using metadata in helping to extend the CAC paradigm to additional security services, but the idea was not elaborated further, nor applied to specific types of rights.

The recent interest in FHE - (fully) homomorphic encryption (Gentry, 2009) for cloud computing solves many confidentiality problems for performing computations remotely without the data storage being able to decipher the meaning of the computations. FHE, however, does not cover a wide area in the access control domain: as a program using FHE is able to run normally without decrypting any of its input or output, it would seem that the **`execute`**-permission is covered by FHE as well. However, this is not the case: even with sufficiently efficient FHE the encryption scheme does not stop anyone from actually running the code, or make the execution flow itself encrypted.

3 THE CASE FOR EXTENDED CAC MODELS

Our main setting considers a cloud-based storage service with typical dynamic subjects and collaboratively modifiable objects. The main focus here is on the properties of the cloud: it is assumed to be reliable in the sense of availability (as documents are distributed, backed up, shared and synchronized, it becomes less and less likely to actually lose information accidentally or by malevolent interaction).

Specifically, we lay out the following assumptions:

- A server (in the cloud) acts as a storage or an execution platform only, focusing on availability and speed. It may have the capability to remove (all its copies of the) files on an authorized request. The server does not have the capability to perform key-management or cryptographic duties related to the stored content.
- A storage-server is almost always assumed to be able to provide at least one "clean" copy of the requested data, although it may not have the ability to identify the correct instance.
- There are no unpassable reference monitors "close" to the data. For authorized users, some RM-functionality is expected, but it is also possible to read and write (including deletion) the data by bypassing these RMs.

We propose to extend the current CAC-paradigm to cover the full spectrum of access control rights by:

- Including most conventional reference-monitor-enforced permissions.
- Extending the cryptographic enforcement further from the content / data, in terms of metadata.

If more types of access rights than just **`read`** on data and selected metadata are to be protected, there is need to encode the whole access control matrix somehow to be enforced cryptographically. This is, however, more complex than just encrypting and signing the whole access control matrix as metadata to the protected data. On the other hand, metadata encryption itself seems a viable option.

One of the main promises of some of the cryptographically enforced access control implementations seems to be that since data is encrypted, there is no need for reference monitors, as there is no way to read the contents without the cryptographic keys. This is, however, an oversimplification of the whole access control concept:

- Types of access rights form a hugely varied category beyond data confidentiality;
- Cryptographic keys need to be guarded as well as data;
- Access control entails many dynamic supporting functions (bookkeeping, adding and deleting users, objects and rights) that need to be accomplished somehow in addition to the actual access control, requiring access to different levels of metadata.

However old-fashioned or unscalable reference monitors may seem to be, they can still emulate the intuitive functions of access control easily, for example enforcing access right revocation: they simply close and further disallow the connection between subject and object – which is possible since by definition every access should pass through the RM.

In the traditional view of information security, cryptography is able to answer properly to confidentiality and integrity only. When this is mirrored to access control:

- Disable read = encrypt (“*make text incomprehensible*”, not “*disable viewing*”)
- Disable write = sign (“*make it possible to detect modification*”, not “*disable bit-flips, deletions and insertions*”)

Cryptographic techniques themselves are not able to enforce much anything, since they are passive. This passiveness leads to a shift in responsibility from data or service provider to its consumer. The responsibility shift means, for example, that outsourced data server may not be responsible for the data integrity; instead the data consumer should check the validity of the data if she so chooses.

4 THE EXTENDED CAC MODEL

4.1 Dimensions in Rights and Data

Our main contribution in this paper is an extended view on cryptographic access control, based on data-metadata hierarchy and decomposition of RM-based types of access rights into combinations of **`read`** and **`write`**, applied to different types of data enforceable with cryptographic methods.

The access control matrix (ACM) uses a flat model to lay out objects, and does not fix any types of access rights. In ACM, subjects and object metadata can all be equally objects along with more traditional files and directories.

However, in real systems, types of access rights or permissions tend to be more varied and complex. Permissions are specified on top of each other, e.g. who is allowed to change or audit permissions. Permissions and object form data-metadata hierarchies, which can be used to our advantage in reducing permission types to those usable in CAC.

Our model performs permission type decomposition according to three parameters, shown as axes: r/w, metalevel and data-type axis. The r/w-axis shows, which CAC-enforceable type (encryption for **`read`** or signing for **`write`**) should be used; data-type specifies on which data the permission applies to; and metalevel-axis shows, whether the type of the data type is data, metadata on data or even metadata of metadata.

The data-type axis has several types, but the CAC model only cares about whether the protected data is content (“payload”) or access-control related. Other types are optional, and shown only to show the connection to table 1. The dimensions are shown in Figure 1.

The benefits of the decomposition are straightforward: **`read`**- and **`write`**-permissions can readily be enforced cryptographically, provided that there is something to encrypt or sign. Dividing the targets into metadata-levels conceptually places more abstract functions into the data-plane and enables their representation with known methods, such as structured data (e.g. XML-documents). This in turn enables different classes of data administrators to perform their duties independent of their rights to the payload content.

The underlying idea in using hierarchy is to represent most of the access rights as existing (structured) data, with each type of metadata placed parallel to the actual content node. Access control metadata would typically include encrypted symmetric keys (along with their metadata) and signatures.

Each conventional access right is assumed to be able to be represented by a “small” number of points in the decomposition space. Enumerating and canonizing the rights this way avoids the translation issues between permission types expressed in natural language between different systems, and clearly states, what is expected of the cryptographic scheme proposed to protect that particular permission type.

As an example, consider the **`own`**(**`Take Ownership`**)-permission from Bell-LaPadula (Windows): it can be seen as both **`read`**- and

write- permission pertaining to a datatype *access control*, and as it is additional information on data, it lies on the *metadata* level. Thus that particular permission would be set of two points on the line defined by *access control* and *metadata*.

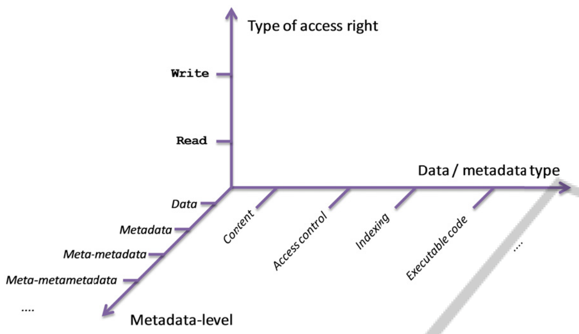

Figure 1: Decomposition model for types of access rights.

It should be noted that it is not sufficient to merely show that a permission type can be broken into a set of **read**s and **write**s: if the target of the read/write-operation is not persistent data, cryptography can help only in principle (to encrypt server process internal structures is probably pointless).

We do not expect that any concievable permission can be reduced to a set of mere **read**- and **write**-rights. However, we note that the general models for computing, such as the Turing machine, use only read- and write operations, complemented by decisional logic and state information (i.e. metadata).

We also concur that some rights are more efficient and / or practical to enforce with reference monitors, even considering that establishing trusted RMs in suitable locations may be expensive.

4.2 Modeling Access Rights

The model needs to show that different types of access rights can be enforced cryptographically. This goal is twofold:

1) Showing that a type of access right can be reduced to a combination of **read**- and **write**-rights in general;

2) Showing that enforcing each decomposition of access rights type at the data consumer's domain is sufficient to realize the same kind of protection as with a reference monitor.

Not all permission types are practical to turn into their **r**/**w**-decomposition. We selected a set of rights we deemed somehow general and relevant to the consumer's computing environment. In order to tie the model closer to real systems, in addition to general types, we included types of access rights from MS SQL Server 2000 database management, Windows XP / Windows 7 - called "special" permissions (Microsoft, 2007) - and Bell-laPadula model (Bell, 1973). This totals 11+11+10+4 types, respectively and including the general types.

Table 1: Types of access rights and their decomposition to cryptographically enforcable types and targets.

Type of access right	Source	r/w	the target of the r/w
create	gen.	w	data storage metadata
delete	gen.	w	data storage metadata
log actions	gen.	r	system actions metadata
		w	system log data
audit logs	gen.	r	system log data
delegate	gen.	w	access control table metadata (with subject-based restrictions)
execute	gen.	r	executable code data AND
		w	program execution data & metadata (in a meaningful way)
search	gen.	r	indexing metadata OR
		r	data (in a meaningful way)
revoke, grant, control	BLP	w	access control table metadata
own	BLP	r	data and all related metadata AND
		w	data and all related metadata
Traverse Folder	Win7	r	data storage metadata
Read Attributes	Win7	r	file metadata

The permissions and their decomposition into **r**-/**w**-types for selected permission types joined with a suitable target are shown in table 1.

In table 1, different types of access rights are grouped according to their source: general, Bell-LaPadula, Windows XP / 7 and Microsoft SQL Server 2000.

We excluded most of the investigated rights from the table due to their triviality (e.g. **read**, **write**), host domain (strongly at the storage, such as database statement permissions), or because they essentially duplicate the decomposition type from another type or permission.

It should be noted that each permission is itself metadata of the target it addresses. Thus cryptographic protection of a permission is always one ladder higher on the metadata axis than its target.

`Delegate` (**`grant`** in BLP) is a permission on a permission, or a meta-permission. The **`delegate`**-right here means that a subject has a permission to grant selected rights further, subject to a set of additional restrictions.

The enforcement of the **`search`**-right depends on the actual implementation of the search: if an indexing structure is built during the search, that structure can be encrypted; if there is no indexing structure, one may use searchable encrypted databases, using searchable encryption techniques or homomorphic encryption.

The BLP-model addresses permissions that affect the access control matrix itself. We call these rights access control metadata, also referred to as security-related metadata (Kiviharju, 2010). Permissions such as **`grant`** and **`revoke`** address the questions of who is allowed to change the access control enforcement function, i.e. the security administrator role. This implies that access control metadata needs to be addressed differently than other metadata. For metadata other than access control, there is no need to elevate the metadata-level for more than one, and CAC adds just another layer on top of traditional structured documents' protection.

4.3 Responsibility Shift

The shifting of responsibility becomes essential in those types of access rights, for which the misuse can directly result in breach of a security attribute outside the data consumer's domain (such as service disruptions).

We assumed that the data storage facility is able to provide at least one instance of data "clean" that has otherwise been somehow deleted or modified via unauthorized channels. This implies that some rights can be left for the responsibility of the data storage in the cloud:

- **`delete`** and **`update`** (in such a way that it erases original data altogether)
- extensive **`append`** and **`create`** (to create a DoS attack)
- **`control`** and **`revoke`**

By our assumptions, consumer processes data in a platform distinct from the storage server. However, some rights affect metadata, which is needed by the server in order to perform the storage function.

Rights that may affect operation inside the storage domain include the permissions listed above and most notably **`execute.`** Execution refers to data that is interpreted as program code, and run on some platform. In order to be able to execute the code, the subject has to have the general right to access the code altogether (**`read`** rights) and then have an access to the execution system on a specific processor (**`write`** rights). **`Read`** can be accomplished by encrypting the executable file, but write rights to the processor are more complex to accomplish cryptographically. It is not meaningful to sign the code, since the platform owner is not necessarily responsible for verifying signatures. Alternative approach could be e.g. instruction set encryption per platform, in which case the executable would be twice encrypted: first the instruction set permutation and then block encryption. The instruction set encryption's key management should be the responsibility of the execution platform owner

4.4 Restrictions on Permissions

Reference monitor-based access control systems are able to restrict access based on different criteria, such as time, history (based on subject's previous actions), location and role. The basis of the restriction is often trusted, as the reference monitor (assumed to be non-corruptible) can also be assumed to be able to synchronize to a reliable time source reasonably accurately and possess a complete set of information concerning the exercise of the rights (as all permission usage should travel via the RM) for auditing purposes.

In CAC, as the encryptor/verifier generally has little or no control over the decryptor's/signer's environment, the existence and use of permission restrictions based on environmental factors cannot be trusted.

In CAC-schemes the restrictions are almost always based on key-management: the subject keys are assumed to be available only if the intended restriction does not apply. However, the consumer is only bound by the possession of these attributes (keys), not by the particular circumstances related to the use of the attributes.

In a more controlled environment it could be possible to enforce some portions of the computing platform to have higher-than-usual assurance in their integrity, by using TPM or other tamper-proof hardware. In these cases ABE could be used to place environment-based restrictions by dividing the attribute-keys between the user and the platform:

- Attribute-keys pertaining to the user are given to the user (as files or tokens);
- Attribute-keys pertaining to the environment are embedded into the tamperproof hardware modules;
- The encryption / signing is performed with a combination of these two types of keys' public parts.

Some restrictions are needed to express certain rights at all, for example **`Traverse Folder`** requires that **`read`** rights are checked also for other objects in the folder hierarchy above the target.

5 CONCLUSIONS AND FUTURE WORK

In this paper we have discussed the current status of cryptographic access control paradigm from the point of view of access control models, and suggested extensions to the current CAC-paradigm.

The proposal concerned extending the cryptographic enforcement to multiple permission types other than just just **`read`**- and **`write`**. We accomplished this to a practical set of rights by decomposing conventional permissions to a set of read- and write-operations on certain data, and showing that for most of them it is feasible and practical to enforce them cryptographically as well.

There still remain many open questions and problems, such as efficient revocation. However, less elusive problems, such as cryptographical role binding are natural cryptographic efforts to tackle; from the modeling point of view it would be instructive to create actual XML-schemas or formal cryptographic RBAC-models.

REFERENCES

Bell, D., LaPadula, L., 1973. *Secure Computer Systems: Mathematical Foundations*. MITRE Technical Report 2547, vol. 1. MITRE.

Bethencourt, J., Sahai, A., Waters, B., 2007. Ciphertext-Policy Attribute-Based Encryption. In *IEEE Symposium on Security and Privacy 2007,* IEEE Computer Society. pp. 321 – 334.

De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P., 2007. Over-encryption: Management of Access Control Evolution on Outsourced Data. In *Very Large Databases (VLDB) 2007 – Conference Proceedings*, ACM. pp. 123-134.

Ferraiolo, D., Kuhn, D., 1992. Role Based Access Control. In *15th National Computer Security Conference – Conference Proceedings*. pp. 554-563.

Gentry, C., 2009. A Fully Homomorphic Encryption Scheme. PhD Dissertation in Stanford University. [Online], Available http://crypto.stanford.edu/craig/ [11.3.2012].

Khader, D., 2007. Attribute Based Group Signature Scheme. [Online], Available: http://eprint.iacr.org/2007/159 [8.3.2012].

Kiviharju, M., 2010. *Content-Based Information Security (CBIS): Definitions, Requirements and Cryptographic Architecture*, FDF Technical Research Centre.

McGovern, S., 2001. *Information Security Requirements for a Coalition Wide Area Network Thesis in Naval Postgraduate School, Monterey, California,* NPS/CISR.

Microsoft, 2007. How to set, view, change, or remove special permissions for files and folders in Windows XP. [Online], Available: http://support.microsoft.com/kb/308419 [11.3.2012].

Naor, D., Naor, M., Lotspiech, J., 2001. Revocation and Tracing Schemes for Stateless Receivers. In *CRYPTO 2001 – Conference Proceedings*, Springer-Verlag. pp. 41-62.

Savoie, J., 2004. *A Strong three-factor authentication device: TrustedDAVE and the new Generic Content-Based Information Security (CBIS) architecture*, Technical Memorandum TM 2004-198, DRDC Ottawa.