RBAC with ABS
Implementation Practicalities for RBAC Integrity Policies

Mikko Kiviharju
Finnish Defence Research Agency, Riihimaki, Finland

Keywords: MLS, RBAC, CBIS, ABE, Cryptography, Access Control Models.

Abstract: Role-based access control (RBAC) is the *de facto* access control model used in current information systems. Cryptographic access control (CAC), on the other hand, is an implementation paradigm intended to enforce AC-policies cryptographically. CAC-methods are also attractive in cloud environments due to their distributed and offline nature of operation. Combining the capabilities of both RBAC and CAC fully seems elusive, though. This paper studies the feasibility of implementing RBAC with respect to write-permissions using a recent type of cryptographic schemes called attribute-based signatures (ABS), which fall under a concept called functional cryptography. We map the functionalities and elements of RBAC to ABS elements and show a sample XACML-based architecture, how signature generation and verification conforming to RBAC-type processes could be implemented.

1 INTRODUCTION

Role-based access control (RBAC) is very often the most practical model for access control, due its versatility, dynamics and accurate modelling of real world problems. RBAC policies are traditionally enforced by a concept called reference monitor (RM), which is instantiated as a security kernel, typically operating system components.

The usual problem of security kernels is that they need to implement the assumptions of an abstract entity (the RM), which include non-passability, ubiquity, and verifiability (leading to high assurance in its correct and secure functioning). In today's highly distributed and heterogeneous environments achieving these goals become either very expensive or highly localized and suitable for special cases only.

Cryptography, on the other hand, offers high assurance on some aspects and is cost-effective to implement. Solving access control problems with cryptography is called cryptographic access control (CAC), or when applied to RBAC directly, CRBAC. Most of the work on CRBAC focuses on enforcing confidentiality policies or high level models, but there is a "middle ground", between theory and actual cryptographic schemes, that is almost entirely lacking in current literature.

Our aim in this paper is to bridge this gap by presenting a mapping for CRBAC integrity policies from existing standards to existing cryptographic schemes, as well as study their limitations. (By "integrity policies" we mean that the RBAC permission enforced is the `write`-permission, and consider relatively static documents as the RBAC objects.)

A recent cryptographic innovation called functional cryptography, including such concepts as attribute-based encryption (ABE, (Goyal, 2006)) and attribute-based signatures (ABS), resolves many requirements and operations in the RBAC-model. Our contributions in this paper are as follows:

1) We investigate the practicalities of existing ABS schemes w.r.t integrity policy enforcement in reality, including status quo and some overlooked details.
2) We provide an implementation model, which is based on the XACML 3.0 reference architecture (Rissanen, 2013), collaboratively managed documents in a publish-subscribe-model and cloud storage; and which can be used to enforce access control cryptographically for **write**-permissions with ABS. We use a widely adopted standard in order to show the feasibility to transform existing systems or create new systems for the CAC paradigm.

Kiviharju M..
RBAC with ABS - Implementation Practicalities for RBAC Integrity Policies.
DOI: 10.5220/0005122105000509
In *Proceedings of the 11th International Conference on Security and Cryptography* (SECRYPT-2014), pages 500-509
ISBN: 978-989-758-045-1

3) We show how different RBAC standard model elements and commands can be realized in CAC with different ABS-methodologies.

Our main motivation in researching ways to implement RBAC using mainly cryptographic controls is the possibility to move the wealth of existing practice and experience invested in RBAC to an implementation paradigm, which is remarkably more suitable for current cloud environments.

2 PRELIMINARIES

RBAC (Sandhu, 2000) is an access control model, which decouples the user (subject) relations to protected objects via a *role*. We use the RBAC model and terminology described in the RBAC-standard by INCITS (ANSI, 2012) (currently the consolidated version referred to as $RBAC_3$), and restrict ourselves to the administrative commands of the standard in the Core RBAC for brevity. We do, however, consider Hierarchical and Constrained RBAC in the structure and functionality they provide to the whole, but not their dynamics (e.g. adding inheritance relations or modifying dynamic separation-of-duty- sets). Note that we include the concept of an administrative role from ARBAC (Sandhu, 1999).

The RBAC-standard includes a multitude of commands to create, maintain and report the necessary elements and their mappings. The relevant commands we consider are from the Core RBAC, together 13 different commands (see Table 3).

For the purpose of RM-implementations and architecture, we refer to the XACML reference architecture (Rissanen, 2013) and elements, specifically: *Policy Enforcement Point* (PEP), *Policy Decision Point* (PDP), *Policy Administration Point* (PAP) and *Policy Information Point* (PIP)

3 CRYPTOGRAPHIC ACCESS CONTROL

3.1 General

Following the access control meta-model presented in (Barker, 2009) we may assume that there exists a (countable) set of action types, which can be associated with

- resource types, jointly called permissions (to perform some action on a resource)
- resource and principal types, jointly called authorizations (for a principal to have a permission),

For many practical access control policies it is justifiable to present subsets of permissions as combinations of only two types of actions (**`read`** and **`write`**), and a categorization of resource types into data and metadata levels s.t. some combination of these two rights together with appropriate decomposition of the resource translates to the original permission. For example, all the permission types used in MS SQL Server 2000 DBMS, Linux, Windows 7 and Bell-laPadula model were mapped in (Kiviharju, 2012) to a subset of permissions such that the subset of actions used was {**`read, write`**}. The motivation and background for this kind of decomposition was to be able to enforce more policy types than just pure confidentiality and integrity policies with cryptography.

3.2 CAC Models and Related Work

The work on CAC models traditionally has focused on information flow control (IFC) policies and key-assignment schemes (Crampton, 2006; Atallah, 2009). CRBAC is addressed for example in (Crampton, 2006), where it is shown that hierarchical key assignment schemes (HKAS) for IFC can also be used to enforce read-rights in RBAC, and that each ABE-policy also has a HKAS-enforced RBAC-policy. The converse, however, is not shown.

The authors in (Zhu, 2011) develop a cryptographic security model for CRBAC, and they also map the RBAC commands to predicate encryption (PE) scheme functionalities. The CRBAC-paper, too, is somewhat detached from actual crypto scheme practicalities: for example role revocation entails assigning new attributes for roles and complete re-encryption of the part of content, which is both inefficient and against the principles of any IBE-related scheme (i.e. identities should not be changed).

Cryptographic signatures for CRBAC have been considered for example in (Crampton, 2010), but only for inter-domain authorization in the manner of attribute certificates.

The work in (Crampton, 2008) gives a general framework for a role key hierarchy usable for encryption, signing and authentication. Their viewpoint is rather on how to assign keys for different cryptographic policies based on RBAC policies (although for encryption, this is equivalent for actually enforcing RBAC policies).

3.3 Signatures and Integrity

Digital signatures are a class of public-key cryptographic primitives that work towards providing assurance of the origin authenticity of a message (= "*Did the claimed entity produce this message?*"). One of the more modern classes includes functional signatures (FS) and their subclass, attribute-based signatures (ABS). They raise the abstraction level for the authenticity to be dependent on more general attributes (="*Was this message produced in reality with the claimed authorization?*"). We postulate that enforcing the **`write`** -permission cryptographically in the current RBAC-model is most practically performed via ABS schemes.

Protecting data integrity cryptographically with digital signatures has three main goals: to provide the verifier assurance of the data origin authenticity, the absence of unauthorized modifications and also of the content validity (as in: how accurate the information itself is). In the case of conventional signatures, where there is only one "attribute" (the user identity) provided, these facets of integrity are equal. However, in the attribute-based setting the verifier may have a different concept of which combination of attributes are sufficient to prove the validity of the content than the combination used to prove authenticity, such that the verifier would like to specify the policy under which she verifies. The problem is akin to the key-/ciphertext policy dualism used in attribute-based encryption (ABE).

The current ABS-/FS-constructions aim to prove data origin authenticity, thus they tie the policy to the signing key, conforming to the key-policy model (abbreviated here KP-ABS). Thus currently the verifier is not able to select the policy (see table 1).

From the perspective of access control, content validity via a scrutiny of content producers is more akin to creating and viewing log files. The associated permissions then are **`append`**, **`read`**, **`edit`** and **`delete`** to the log file resource class. Thus for the general **`write`** -permission, the history and individual signers can be left out of scope.

4 RBAC AND ABS

4.1 General

Attribute-based signatures (ABS) were introduced as a concept in 2007 (Khader, 2007). With ABS, the signer can prove the authenticity via a predicate over attributes rather than his/her identity. The distinguishing feature of ABS over related concepts is the ability to provide unforgeability, signer (and policy) privacy and collusion prevention all at the same time. Collusion prevention is especially important in filling a loophole present in CAC in general: reference monitors are able to separate different users with (RBAC-/login-)sessions, but CAC schemes need to differentiate between users via other methods, such as personalized credentials.

The first full-fledged ABS was the scheme by Maji et al. (Maji, 2011). On the technical side, it appears that ABS can be realized either via non-interactive proof techniques or based on encryption schemes. (ABE is a derivative of IBE, and enjoys a natural definition for signatures, where one creates a one-time key-pair using the message (or hash of it) as the public key and the private key as the signature, making the IBE/ABE "master key" the signing key). The latter technique offers signatures an order of magnitude shorter than proof techniques.

The ABS scheme employed here is the scheme in (Maji, 2011), referred to as MPR-ABS. Its basic concepts include

- *Credential bundles*, which are personalized per-attribute private keys (personalization is performed e.g. by prepending globally unique user identifiers to attribute information and signing them with Boneh-Boyen signatures (Boneh, 2004) and the actual attribute private keys)
- *Claim predicates*, which are essentially policy formulas applied to the attributes / credential bundles. They are an analogue of access control policies and RBAC active roles in that they can restrict, which attributes user is allowed to use and which not.

Related to an issue discussed above (the content validity) is the concept of individual non-repudiation: if someone has misused her authority to sign content on false grounds, there should be a mechanism to uncover the individual signer behind the attributes. Non-repudiation is, in general, not incorporated in ABS-schemes: although individual signers have personalized key-material (e.g. credential bundles in (Maji, 2011)), this is in the form such that the personalization is not easily extractable (e.g. the signed identifier-attribute combinations in (Maji, 2011). While this feature can be used for testing suspect individuals (if the attributes used are known), it cannot be used to directly extract the user identities behind the signature. There are specific signature schemes to accomplish this, such as (Escala, 2011), but these are separate building blocks not easily integrated to other functions.

Typical of all ABS schemes is that they are inherently group signatures, and that the attributes, the policy or both are tied to "manager"-level private keys. If these master keys are under the authority of one entity, it is usually referred to as *signature trustee*.

ABS schemes have one common security goal, unforgeability. Other goals are different privacy-related goals: verifiers should not be aware, or extract from the signature (without additional information) the individual signer within a group of signers, or the policy under which the signing was performed. Arguably it is case-specific, whether the access control policy itself should be public or not, but we find it more natural and more convincing, if the verifier is given the formula which the signature was generated with. (This type of privacy is possible to incorporate via e.g. DFS (Backes, 2013), but for simplicity, we use schemes that do not allow policy privacy).

4.2 ABS Relation to Policies

In functional cryptography in general it is possible to specify, when and where the access control policy is cryptographically possible to encode. The main categories are key- and ciphertext policies (KP and CP, respectively), although mixtures exist as well. In the KP-model, the policy is encoded in the private key and in the CP-model to the ciphertext. Signature schemes bring more variety to the policy. Ideally policies can be:

- encoded to the private key or the signature
- selected by the signer or the verifier
- can be private of public

The suitability for ABS for enforcing RBAC is explored below, in table 1, with properties relevant to RBAC enforcement given. The conventions used in the table are as follows: S = individual signer, V = verifier, K = private key of S, σ = signature, p = policy, T_s = signature trustee, I_s = identity of S, I_A = identity of attributes.

From Table 1 it can be seen that signature schemes, which are able to enforce authenticity based on policies / attributes can be divided into two categories: original ABS and functional signatures (FS). The fundamental difference (policy-wise) between these is that ABS divide the policy enforcement into two: first creating the attribute private keys and the user randomization (performed by the signature trustee), and then allowing the signer to select the policy (as far as his attributes allow). FS schemes have the signature trustee generate the policy and embed that into the user key. Only the DFS-scheme can be used in such a manner that some policy-defining power is given to the individual signer (although it is not intended for such a purpose).

In the RBAC-model user is decoupled from the resource via the role. Thus user assignment and role activation need to be performed separately. In ABS this decoupling is naturally present via attribute

Table 1: Policy encoding and processing properties of the main ABS/FS schemes.

Scheme	***Novelties***	***Policy Encoding***	***Policy Selection***	***Process privacy*** (1)	***Policy expressiveness***	***Main technique***
MPR-ABS (a)	First ABS	σ	S	I_S, I_A	Monotonic Boolean formulas over attributes	NIWI (h)
DMA-ABS (b)	No signature trustee	σ	S	I_S	Non-monotonic Boolean formulas over attributes	DMA-FE (b)
NM-ABS (c)	Non-monotonicity, small signature size	σ	S	I_S	Non-monotonic Boolean formulas over attributes	CP-FE (i)
R-ABS (d)	"Revocability" (of anonymity of individual signer)	σ	S	$I_S^{(6)}$	Monotonic Boolean formulas over attributes	NIWI (h)
PBS (e)	All policy languages in P	K, σ	T_S	p, I_S, (I_A)	P-language over messages	Groth-Sahai Proofs (h)
FS (f)	Signature size independency of policy size	K, σ	T_S	p, I_S, (I_A)	All policies expr. with a poly-size circuit	NIZKAoK (j)
DFS (g)	Delegation, limited malleability	$K^{(3)}$, σ	S,$T_S^{(2)}$	$p^{(5)}$, $I_S^{(4)}$, (I_A)	Efficiently computable functions	NIZK required (k)

(1): The elements, which are hidden from the verifier (in parenthesis, if not applicable)	a: (Maji, 2011)	g: (Backes, 2013)
(2): Signature trustee is able to assign a family of func. to the signer to delegate further	b: (Okamoto, 2013)	h: (Groth, 2008)
(3): The delegation key with restrictions on the functionalities allowed to be delegated	c: (Okamoto, 2011)	i: (Okamoto, 2010)
(4): Including delegated signers	d: (Escala, 2011)	j: (Bitansky, 2013)
(5): Policy is public for intermediate signers	e: (Bellare, 2014)	k: (Groth, 2006)
(6): Unless revoked	f: (Boyle, 2013)	

private keys generation and user (process) applying allowed policies. In FS, the signature trustee will need to generate new private keys per each new set of active roles. DFS (Backes, 2013) can be employed for a similar purpose, but for simplicity we select ABS as the schemes of choice in our implementation model.

The ABS and FS have differences in the complexity of the policy they are able to express. ABS (including non-monotonic ABS) are limited to the class $\mathbf{NC}^1$. FS-schemes are able to represent general Boolean circuits in complexity class $\mathbf{NC} = \bigcup_i \mathbf{NC}^i$. PBS extends this class of functions even further. The advantage of using larger circuits lies in the fact that some problems are not known be contained within $\mathbf{NC}^1$ (such as graph reachability, which is known to be in $\mathbf{NC}^2$ only). The most common operators, AND, OR, NOT, =, <, >, ADD, MUL, DIV, EXP, LOG, **keyword search** and **regexp** are contained within $\mathbf{NC}^1$. With these, an overwhelming percentage of real-life access control policies can already be expressed, and thus there is no need to go to general circuits from the perspective of this paper.

5 IMPLEMENTATION MODEL

5.1 RBAC Model Mapping

The RBAC model is, as such, functionally more fine-grained than using encryption schemes and signatures without distributing keying material to more than one entity type. For example, the model *Supporting System Functions* require that active roles can be changed within a session, if e.g. environmental conditions change.

If the CAC implementation uses only a single claim-predicate for the user per session, her signatures will either be accepted during the session or not – if a role should be deactivated, the user needs to “log out” or request a new claim-predicate. We introduce a remedy for this in the session management description.

The different RBAC elements and functions are mapped to ABS elements in Tables 2 and 3. Some of the most crucial concepts are elaborated here.

Role: The attributes are natural embodiments for roles (visible also in the XACML-architecture) – however, there is the question whether to bundle attributes into sets to map to one role in CAC. Considering that the reference monitor also makes access choices based on logical formulas over roles, this principle carries easily to CAC as well. An additional note for roles is that in ABS-schemes, each attribute has a private key. This is not directly used by an individual signer, but they are needed in the signing process, as the verification is based on attributes (rather, their public keys) only.

Table 2: RBAC elements mapping.

	KP-ABS ([MPR10] by default)
Object	Document / Message / Content
Operation	`write`
Permission	Private key existence for an attribute
User	User
Role	Attribute
Session (differentiator)	User credential bundle personalization
Session (active roles)	User's possession of her personalized credentials (with a given attribute set) and a defined claim-predicate
PA	Attribute secret key creation
UA	User's possession of personalized credentials (with a given attribute)
Role Hierarchy	Static hierarchy: Attribute delegation
Admin role	Role on metadata
Static SoD	Non-monotonic claim-predicates (NM-KP-ABS, [Oka11])
Dynamic SoD	Non-monotonic claim-predicates (NM-KP-ABS, [Oka11])

Session management: The purpose of the Session-concept in RBAC is to model different profiles for a user. Profiles include different active roles, but they are normally activated in the same batch as the session is created. From the CAC perspective, different sessions are modelled via giving the user multiple identities (which could be a login ID formed by concatenating the user’s personal identity with her current profile identity). When a session is created, it is automatically assigned a default set of active roles. In CAC, and particularly in ABS, this corresponds to defining a claim-predicate over the attributes (essentially a Boolean formula of how and which attributes are used). Any change in the active roles (whether it happens inside a session or via re-establishing a session) requires establishing a new claim-predicate. We note that in the current schemes nothing prevents the signer from selecting a suitable claim-predicate. Ideally, the claim-predicate itself should be time-stamped and authenticated by the signature trustee, or equivalent **OBJ**-subdomain (see the next chapter for the subdomains definition) controller.

Session separation: Different sessions will be separated from each other based on the personalized credential bundles (with differing login IDs, if needed).

User assignment: User can be assigned to the role without yet activating the role. In this mapping UA means delivering the user an item of the

Table 3: RBAC commands mapping.

RBAC command	Applicable function(s)
AddRole	Role mgmt and PAP function
GrantPermission	Create private key for an attribute
AddUser	User mgmt function
AssignUser	Generate user's (new) credential bundle
CreateSession	Create user's current claim-predicate
AddActiveRole	Change user's claim predicate
CheckAccess	**SIG**-subdomain: sign; **VER**-subdomain: Try fetch an instance of the signed content from the **Channel** and verify it
DropActiveRole	Change user's claim predicate
DeleteSession	Invalidate user's claim-predicate
DeassignUser (with loss of auth. [8])	For the deassigned role: Exclude user from next credential bundle update (time-stamped attribute names) and/or Revocation list distribution
DeleteUser	User mgmt function + DeassignUser (for all its roles)
RevokePermission	Exclude attribute from next attribute private key update (time-stamped attribute-names) and/or Revocation list distribution
DeleteRole	Role mgmt and PAP function + RevokePermission for all the permissions of the role

credential bundle corresponding to the attribute, to which user is assigned.

Revocation: Revocation means in this context the deletion of a user assignment or the role-permission association (session-related changes require only changing the claim-predicate). In order to define revocation in CAC for the write-permission, we recall that the policy enforcement is performed in two locations: the **OBJ**-subdomain to allow access to current private key material in the first place, and then in the **USR**-subdomain to actually check the validity of the signature. Since the signing operation as such can be performed by anyone who is familiar with the signing scheme, denying a revoked user access to the signing software/hardware per sé does not solve the problem, if the receiving end does not perform her verification duties. Thus the revocation needs to somehow communicate the revoked / outdated status of the attributes or users to the verifying parties. There are two possibilities for this: time-stamped attributes and revocation list distribution. Handling the revocation in CAC with **`write`** is notably simpler than with **`read`**, since the main part of the enforcement is always performed by non-revoked (and thus benevolent) principals, and it is even possible to extend the domain of revoked content beyond the moment of compromise detection.

A problem with revocation arises, when / if the **`read`**-permission is also enforced cryptographically, and ciphertext revocation is used. Ciphertext revocation assumes the use of functional encryption and provides a way to re-encrypt ciphertexts to be compatible with a new FE-key. We assume hybrid encryption (content encrypted with symmetric algorithm and the symmetric key with FE), leaving the content signatures unchanged. However, if the keying material is also authenticated, there will arise a need to resign it. To address this, there is a concept called sanitizable signatures (SSS, explained e.g. in (Backes, 2013)), which basically allow replacement of structured content elements (not removal or adding as such). In SSS, the modifier creates additional signatures such that they can be verified under the original public key, but the verifier is not able to extract any information of the content previously signed – in this case the encrypted content key before ciphertext revocation. The delegatable functional signatures (DFS (Backes, 2013)) also encompass SSS, and they can be viewed as the attribute-based extension of SSS.

Should a verifier come across content with an invalid or revoked signature, he should merely ignore it and request another copy of the content.

Administrative roles (AR): these are roles that are can create, manage and delete other roles. They can be supported via CAC applied to RBAC metadata, e.g. by attaching AR-signatures to a managed role's attribute and permission descriptors.

5.2 XACML-Conformant Architecture

There are three different subdomains: **SIG**, **Channel** and **VRF**, for the signer, storage medium and verifier functions, respectively. The model depicts a publish-subscribe type of environment, where information is first requested to be published, and the request for subscribing and decryption only follows afterwards. Publishers and subscribers are usually not assumed to establish contact or exchange keying material after the initial handshakes (Ion, 2010). The requesting application for publishing is depicted in the **SIG**-subdomain, and the subscriber in the **VRF**-subdomain, whereas the **Channel** subdomain is responsible for the storage.

Using signatures is essentially a two-party protocol, involving both the signer and the verifier. As such, it is not sufficient to have correct private keys in order to publish authorized material – the verifier action is also needed (in case the signer uses outdated or revoked keys, for example). Thus the enforcement function is necessarily *decoupled* for **`write`** in CAC.

The most frequently used definitions for the XACML Core specification (Rissanen, 2013) data flow architectural elements (the XACML specification itself, RFC 3198, RFC2904, RFC2753 and ISO 10181-3) do not place restrictions on the policy handling point locations as such. Thus we conclude that the decoupling of PEP is not against any previous models, only outside them. The VRF-subdomain is depicted in Figure 1.

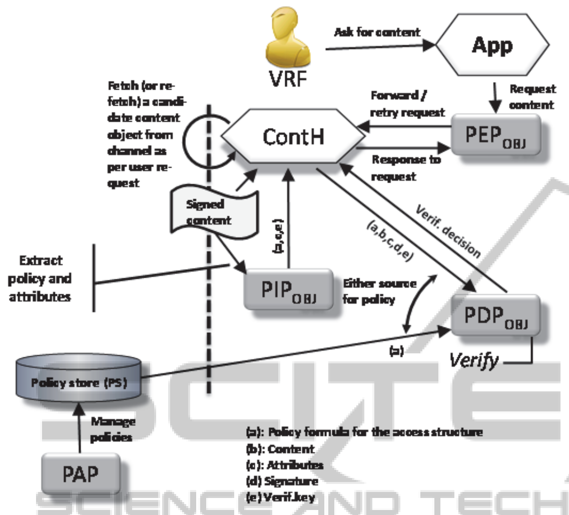

Figure 1: The architecture **VRF**-subdomain.

The verification subdomain (**VRF**) is responsible for the decisions whether the content fetched from the **Channel** is available, endowed with valid signatures and whether to re-fetch another instance from the **Channel**. Mapping verification to XACML-functionality is as follows:

- *PDP*: Verification of the signature requires information of the policy against which the decision is made, and is inherently a decisional function. Although the verification value needs to be computed, the computation is based on the information attached to the content. From the XACML perspective this becomes then the responsibility of PDP. Other components, such and PEP and PIP should not be base their actions on the policy, although they may try to forward it. PDP may get the policy from a common policy store in the **Channel** as well. PDP is assumed to maintain some bookkeeping of the document instances such that in the case of verification failure it can deduce whether it is feasible to find valid instances in the **Channel**. PDP is also assumed to return its decision as belonging to the set: {verified, retry, non-verified}.
- *PIP*: the role of PIP is to gather environmental and object-related attributes. For the purpose of verification of content, all the relevant attributes and policies are (at least with the case of ABS and FS) carried with or encoded in the signature. This leaves little responsibilities for the PIP; it may, signature scheme allowing, try to extract the attributes and policy from the signature.
- *PEP*: since the document verification itself is left for the PDP, PEP is left mainly with forwarding requests and responses. However, if PDP returns retry, PEP will reissue the content retrieval from the **Channel**.
- *ContH*: In XACML, the context handler responsibility is to fetch the actual resource and combine the necessary attributes, policies and resource into one coherent resource context. The only major change in its operation is the way it is fetching the resource: it is not a local disk or database-query, but rather a subscription from a cloud-based environment (the **Channel**).

PAP, PS and Application roles and functions need not be changed.

The signing (**SIG**) subdomain has the main responsibility of enforcing the **`write`**-permission. In order to be able to separate user assignment from role activation (as described in the model mapping) in MPR-ABS scheme functionality, the claim predicate needs to be separately controlled. In the MPR-ABS, the claim predicate can effectively be freely selected by the signer (his attributes allowing), which is more coarse-grained control than the role activation functionality in RBAC requires. Thus in our model we employ trusted computing base (TCB), which controls the claim predicate. In the verification phase the **VRF**-subdomain needs to ascertain that TCB was indeed used in the selection of the claim predicate. This requires further that the claim-predicate can be satisfied only if it includes attributes in possession of only the TCB. This in turn assumes that TCB and user attributes can be combined, which is against the collusion prevention principle usually employed in attribute-based cryptography. There are both scheme-dependent techniques to accomplish this kind of combination/collusion in a controlled setting and more general ways for less expressive formulas using secret sharing schemes, but these techniques are outside the scope of this paper. In the verification phase the **VRF**-subdomain needs to ascertain that TCB was indeed used in the selection of the claim predicate. This requires further that the claim-predicate can be satisfied only if it includes attributes in possession of only the TCB. This in turn assumes that TCB and user attributes can be combined, which is against the collusion prevention principle usually employed in attribute-based

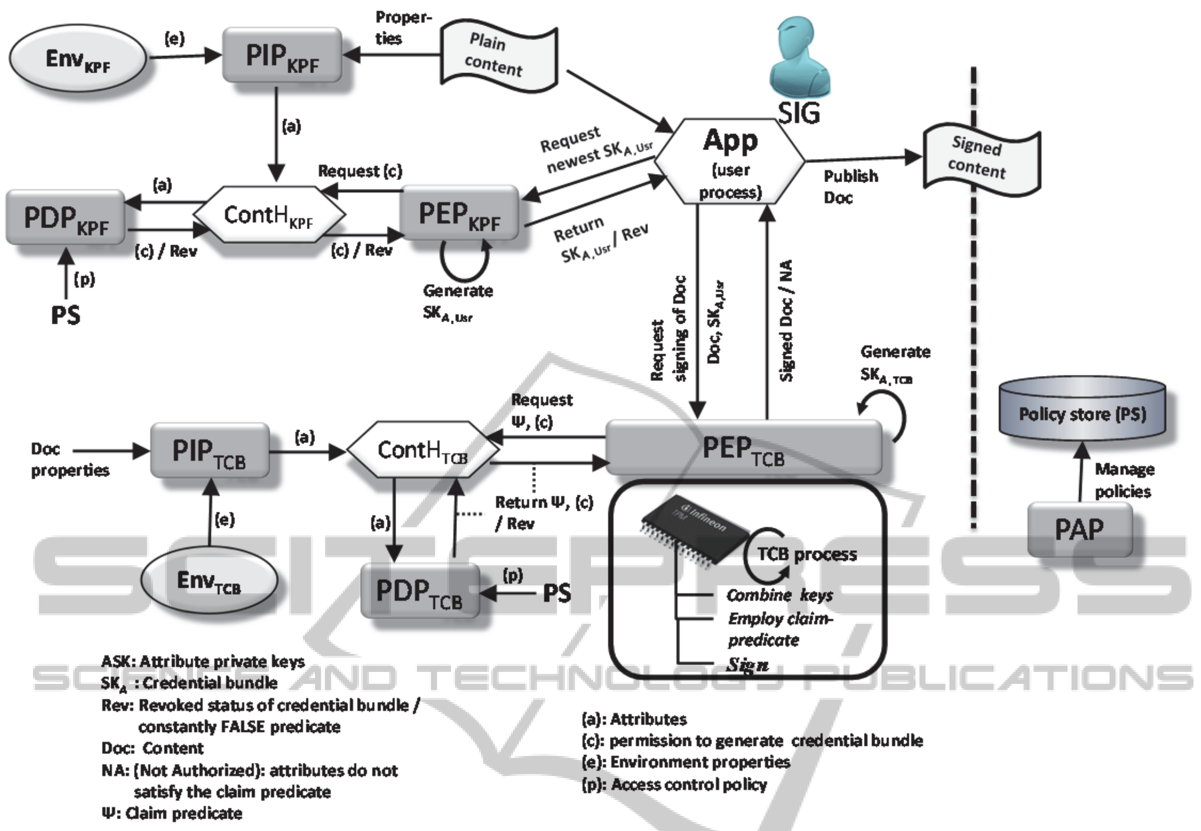

Figure 2: The architecture **SIG**-subdomain.

cryptography. There are both scheme-dependent techniques to accomplish this kind of combination/collusion in a controlled setting and more general ways for less expressive formulas using secret sharing schemes, but these techniques are outside the scope of this paper.

An alternative way of restricting the signer capability in selecting the claim predicate is to integrate the predicate directly in to the private key, as done in FS and DFS. This scenario requires a more available and more broadband connection to the signature trustee, and thus not considered here. The **SIG**-subdomain is depicted in Figure 2.

- *PDP*: PDPs in both portions are responsible for translating the current policy and attributes into a decision, whether individual currently valid signing keys (credential bundles) can be released or not. They also communicate attribute private keys to the PEP to use in the actual creation of the bundles. PDP_{TCB} communicates the active roles information in the form of claim predicate, to PEP_{TCB}.
- *PIP*: PIPs role is not changed from reference monitor realm here.
- *ContH*: Context handler relays and formats the material it handles, it's role in CAC is not very significant.
- *Application*: The application (on the user level) is responsible for coordinating the credential bundle retrieval and for requesting the final signing. It also publishes the signed document to the channel. The user application will act as the XACML application for both the TCB and KPF portions of the **SIG**-subdomain.

Some of the tasks are global (we are assuming only one attribute authority), such as granting permissions for a role, which translates to attribute private key generation. This is implicitly a PAP function and the necessary information is incorporated in the policy received from the policy store.

6 EFFICIENCY CONSIDERATIONS

A cryptographic scheme is a prescription of operations (on varying levels of abstraction) that should be realized in order to have an actual instantiation of the scheme, i.e. it would need to be programmed or burned into a chip.

ABS is mostly based on special elliptic curve groups (ECG) and so-called pairing functions applied to them. There are even existing libraries realizing some ABE schemes already, such as (Bethencourt, 2011). We have investigated previously the bandwidth and computational efficiency of existing schemes on different security levels, and combining them with results given in (Okamoto, 2011), we can give some figures for 128-bit security parameter and access control structure of size 10 (as in number of clauses):

- ABGS (Khader, 2007) (ABE-based, aver.case): 0,9 kB
- MPR-ABS (Maji, 2011) (NIZK-based, worst case): 23,5 kB
- MPR-ABS (Maji, 2011) (NIZK-based, best case): 1,5 kB
- NM-ABS (Okamoto, 2011) (FE-based, best case): 81 B

For implementation purposes the NIZK-based ABS are clearly far less efficient than encryption-based ABS. However, we believe that moving to cryptographically enforced RBAC is no longer just a feasibility study, but something that merely needs integration work to make an actual demonstrable system.

7 CONCLUSIONS

Our research shows that current ABS-schemes can already support the Core RBAC, in a distributed implementation model and considering the **`write`**-permission. There are problems still, especially with dynamic hierarchies and providing support to both role activation separation from user assignment; and strict control of role activation at the same time. The Core RBAC commands can be simulated with ABS, indicating a feasible transformation for RBAC systems from RM to CAC.

The ABS are a sufficient and necessary class of signature schemes for implementing the most common access control needs and policies. The reasons for going beyond ABS to FS would include:

- Complex policies requiring evaluation of arguments beyond $\mathbf{NC}^1$
- Moving the claim-predicate enforcement from trusted hardware to key management (and accepting a more frequent or hierarchical key updates)

Future work will include e.g. designing cryptographic schemes more suitable for content validation, where the verifier is able to select the policy (instead of signer).

REFERENCES

ANSI, 2012. American National Standard for Information Technology – Role Based Access Control, *INCITS 359-2012*, ANSI.

Atallah, M., Blanton, M., Fazio, N., Frikken, K., 2009. Dynamic and Efficient Key Management for Access Hierarchies, In: *ACM Transactions of Information and System Security, Vol. 12, No. 3, Article 18*, ACM.

Backes, M., Meiser, S., Schröder, D., 2013. Delegatable Functional Signatures, In *https://eprint.iacr.org/2013/408*, IACR.

Barker, S., 2009. The Next 700 Access Control Models or a Unifying Meta-Model?, In *SACMAT'09, pp. 187-196*, ACM New York.

Bellare, M., Fuchsbauer, G., 2014. Policy-Based Signatures, In *PKC 2014*, Springer (to appear).

Bethencourt, J., Sahai, A., Waters, B., 2011. Ciphertext-Policy Attribute-Based Encryption-project, in *Advanced Crypto Software Collection, http://hms.isi.jhu.edu/acsc.,*

Bitansky, N., Canetti, R., Chiesa, A., Tromer, E., 2013. Recursive composition and bootstrapping for snarks and proof-carrying data. In *STOC 2013, pp. 111-120*, ACM.

Boneh, D., Boyen, X., 2004. Short signatures without random oracles. In *EUROCRYPT 2004, LNCS 3027, pp. 56-73*. Springer.

Boyle, E., Goldwasser, S., Ivan, I., 2013. Functional Signatures and Pseudorandom Functions. In *https://eprint.iacr.org/2013/401*, IACR.

Crampton, J., Martin, K., Wild, P., 2006. On Key Assignment for Hierarchical Access Control. In *CSF 2006*.

Crampton, J., 2010. Cryptographic Enforcement of Role-Based Access Control, In *FAST 2010*.

Crampton, J., Lim, H., 2008. Role Signatures for Access Control in Open Distributed Systems. In *SEC 2008*.

Escala, A., Herranz, J., Morillo, P., 2011. Revocable Attribute-Based Signatures with Adaptive Security in the Standard Model, In *AFRICACRYPT 2011, pp.224-241, LNCS 6737*. Springer.

Goyal, V., Pandey, O., Sahai, A., Waters, B., 2006. Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data, In *Proc of 13th ACM Conference on Computer and Communications Security, pp. 89-98*, ACM.

Groth, J., 2006. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In *ASIACRYPT 2006, LNCS 4284, pp. 444–459*, Springer, Germany.

Groth, J., Sahai, A., 2008. Efficient non-interactive proof systems for bilinear groups, In *EUROCRYPT 2008, LNCS 4965, pp 415-432*. Springer.

Ion, M., Russello, G., Crispo, B., 2010. Supporting Publication and Subscription Confidentiality in Pub/Sub Networks. *In SECURECOMM 2010, pp. 272-289.*

Khader. D., 2007. Attribute Based Group Signature Scheme. In *http://eprint.iacr.org/2007/159*, IACR.

Kiviharju, M., 2012. Towards Pervasive Cryptographic Access Control Models. In *SECRYPT 2012*.

Maji, H., Prabhakaran, M., Rosulek, M., 2011. Attribute-based signatures". In *CT-RSA 2011, LNCS 6558, pp. 376–392*. Springer.

Okamoto, T., Takashima, K., 2010. Fully secure functional encryption with general relations from the decisional linear assumption. In *CRYPTO 2010, LNCS 6223, pp. 191–208*. Springer.

Okamoto, T., Takashima, K., 2011. Efficient Attribute-Based Signatures for Non-Monotone Predicates in the Standard Model, In *PKC 2011, LNCS 6571, pp.35-52*. Springer.

Okamoto, T., Takashima, K., 2013. "Decentralized Attribute-Based Signatures", In *PKC 2013, LNCS 7778, pp.125-142*. Springer.

Rissanen E. (ed.),2013. Extensible Access Control Markup Language (XACML) Version 3.0, OASIS Standard. In *http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf*, OASIS.

Sandhu, R., Bhamidipati, V., Munawer, Q., 1999. The ARBAC97 model for role-based administration of roles. In *ACM Transactions on Information and Systems Security, 2(1): 105-135*, ACM.

Sandhu, R., Ferraiolo, D., Kuhn, R., 2000. The NIST Model for Role-Based Access Control: Towards A Unified Standard. In *5th ACM Workshop on RBAC, pp. 47-63*.

Zhu, Y., Ahn, G-J., Hu, H., Ma, D., Wang, S., 2013. Role-Based Cryptosystem: A New Cryptographic RBAC-system Based on Role-Key Hierarchy. In: *IEEE Transactions on Information Forensics and Security*. IEEE.