Addressing Interoperability Issues in Access Control Models

Vishwas Patil
patil@di.uniroma1.it

Alessandro Mei
mei@di.uniroma1.it

Luigi V. Mancini
mancini@di.uniroma1.it

Dipartimento di Informatica
Università degli Studi di Roma - La Sapienza
Via Salaria 113, 00198 Roma, Italy

ABSTRACT

An access control framework is responsible for efficiently and effectively managing an organization's resources on which its users can perform actions. Depending upon their requirements, different organizations deploy different kinds of access control frameworks. For a common goal, organizations often collaborate by contributing their resources and users. To integrate each other's resources and users, their access control frameworks should be interoperable. To help a collaboration realize, several models [14, 41, 42] exist. These models facilitate the collaboration among homogeneous access control frameworks. In practice, collaborators may have heterogeneous frameworks that may not share any similarity in their security orderings [10] which may prove to be a serious hitch for integrating each others' resources and users at an appropriate order. Here, we present a utility that allows one to form an overlay of definitions specific to the collaboration. Such definitions map new names for the existing definitions available within the framework. Thus, the new security order formed through overlay definitions can be presented as an interoperation interface to the collaborators. The use of overlays hides the internal security ordering of an organization from its collaborators and we shall see how collaboration specific context information can be captured and used in our approach. The post-collaboration setup should provide an efficient mechanism for authentication-cum-authorization of participants consistent with the local policies and ensure non-repudiation of any inter-organization communication. We have come across a cryptographic primitive, called chameleon hash, that has allowed us to efficiently realize the above mentioned requirements and properties. A preliminary analysis of our approach shows an advantage over existing certificate based practices [11, 15, 20, 24, 43] in terms of manageability, privacy and communication overheads. Our scheme should be the best implementation choice for dynamic and ephemeral collaborations where preserving pre-collaboration functional setup during the span of collaboration and also after the collaboration is important . Actually, this is a pressing need for organizations coping with globalization.

In this paper, our goal is to devise an enforcement mechanism to facilitate concurrent collaborations in a distributed environment with a focus on the manageability, interoperability and privacy of collaborators. Privacy to the collaborators is a new unique feature provided under our approach.

Keywords: access control, interoperability, name spaces, chameleon cryptography.

1. INTRODUCTION

Owing to the advent of digital revolution it has become possible to integrate almost any kind of electronic device that has computing and communication capability into the day to day life of individuals and organizations. This extends the reach of such devices beyond the physical boundaries and demands protection against misuse. In other words, availability of such resources to their owners becomes a security issue. Computers, printers, card-readers, sensors, digital photocopiers, etc., are typical examples of digital resources. Furthermore, even the services provided by such resources (e.g., databases) and applications developed on top of them (e.g., web-services) can be collectively referred to resources. Therefore, to ensure the availability of resources to the intended users of an organization, the organization needs a comprehensive mechanism called access control framework. Depending upon the size and functional requirements, different organizations deploy different types of access control frameworks. For instance, a small organization might be content with an access control framework of type access matrix [19] or mandatory access control (MAC) [6] or discretionary access control (DAC) [26]. Military organizations are/have been traditional practitioners of MAC and DAC. In commercial environments with large number of users and resources (e.g., banks), role-based access control (RBAC) [18] has emerged as a de facto standard. An organization may also have a tailored flavor of these frameworks suitable to its requirements or may have a proprietary framework in place. Such a heterogeneity in models for access control brings forward the challenge in their interoperability when used to form a collaborative environment.

The primitive goal of an access control model is to efficiently manage users and resources (entities) under its control. Different AC models achieve this goal differently. A typical deployment of an access control system is a combination of the following three logical components: an *access control model*, *policies*, and *enforcement mechanisms*. The access control model provides means to arrange, efficiently manage entities and define relations amongst them. For example, in MAC, entities are arranged in a matrix where cells of the matrix define the relation between users listed in rows with resources placed in columns. In RBAC, entities are assigned to abstract names inherent in the model (e.g., ROLE, OBS, etc.) and the model provides means to express relationships amongst such pre-defined abstract names. Policy languages are employed to provide properties that are difficult to achieve under AC model alone, e.g., context-sensitive access requests. Enforcement mechanisms are employed to enforce the outcome of an access request to a resource and also in situations where certain requirements are contrary to the inherent properties of the underlying AC model; for example, private or external role hierarchies [31, 34] in RBAC. The interfluve of functional scope of these three logical components is not strict and may vary in actual deployment, according to the requirements and nature of the setup.

Collaboration amongst organizations is a pressing need in the current trend of globalization and outsourcing. Organizations collaborate for a common goal by contributing their resources and users. For example, workflow systems spanning across several autonomous organizations (administrative domains), computational grids where computational resources and service users belong to different administrative domains, military and intelligence coalitions, collaboration through outsourcing, etc., face interoperability issues amongst their autonomous administrative domains that constitute the collaboration environment. Some of the prominent models that help in realizing collaboration are [14, 41, 42]. However, these models facilitate collaboration among homogeneous access control frameworks. In practice, collaborators may have heterogeneous frameworks and may not share any similarity in their security orderings [10], the orderings help collaborators in judging appropriate levels for accommodating each others' resources and users. Several other models that address certain issues while heterogeneous autonomous domains collaborate have been proposed. These approaches have varying reliance on the three components of access control system described above. In [14, 41, 42], RBAC model has been extended to facilitate collaboration among homogeneous domains. The additional requirement for collaboration in distributed environment is that the communication across participating domains should have authenticity and non-repudiability properties. These properties can be provided by enhancing the enforcement mechanism component (i.e., by integrating cryptographic functions or a PKI) of access control systems. In [11, 15, 20, 39, 43, 29], PKI (X.509) assisted AC models have been proposed to facilitate a secure collaboration. The choice of PKI, X.509 in the above mentioned proposals, plays a pivotal role in deciding the autonomous and dynamic nature of the resulting collaborative environment. The use of X.509 PKI (which is a top-down architecture and centralized in nature) in the above proposals does not allow them to remain truly decentralized. In [1, 2, 8, 9, 10, 13, 21, 22, 24, 33, 28], formal models based on the policy languages are proposed for collaboration in a distributed environment. These models and their policy engines rely on security assertions provided by the underlying enforcement mechanisms to evaluate resource access requests. Briefly speaking, several combinations of the above mentioned components are possible to facilitate collaboration in distributed environment with a varying dependence on the three components to achieve security and manageability in the resulting domain.

In a collaborative domain, the *enforcement mechanisms* component has a greater role to play than its role in the stand-alone access control system. Authentication, non-repudiation, security assertions that can be verified off-line, etc., are the additional properties expected from the *enforcement mechanisms* component. To achieve these additional properties in actual implementation of any of the above listed proposals facilitating collaboration in a distributed environment, they shall rely on cryptographic primitives, plausibly asymmetric keys. Thus, having a pair of asymmetric keys with the entities involved in collaboration, we provide a name and authorization binding utility that greatly simplifies forging dynamic collaborations and post-collaboration management of not only the resulting setup but also of individual participating domains. The utility derives its strength from the collision property of chameleon cryptography [25] and the naming philosophy of SPKI/SDSI [13]. Given a public key, corresponding unique chameleon hash function can be efficiently derived. This function possess all the properties of universal one-way hash functions except that the owner of the private key (trap-door) can produce collisions for any hash value with a different pre-image. We use the chameleon hash function to introduce local names and exploit the collision property to bind entities or externally defined local names to trap-door owner's local names. The utility of introducing names and their binding allows potential collaborators to form an overlay over their shared resources, thus maintaining their post-collaboration autonomy and shielding their actual access control framework from collaboration specific modifications. This is very essential while collaborations are ephemeral. Though our utility is based on keys, the key management issues are out of the scope of this paper and we assume that the same practice should be followed as in other proposals that make use of cryptographic primitives to achieve the properties required in a distributed environment.

Taming heterogeneity: as the collaborators can be dissimilar in terms of their underlying access control models, governing policies/policy engines and enforcement mechanisms, we identify a common denominator across them – asymmetric keys. Our overlay formation utility based on asymmetric keys allows collaborators to generate a common agreeable interface between themselves, without making modifications in their respective autonomous functional setups. In [28], this requirement is acknowledged and addressed through means of "common vocabularies." Identifying asymmetric key pairs as a common denominator also nullifies the heterogeneity collaborators may have in their use of a PKI (PGP or X.509, for example).

Organization of the paper: In next section we introduce our mechanism to define names and binding entities to them. The naming mechanism is central to formation of overlays as an interoperation interface to the collaborators. In Section 3, we explain usage of overlays for a typical collaboration scenario. Section 4, briefly provides our experimental results and lists advantages of our mechanism. We provide the related work in Section 5 and conclude the paper with Section 6.

2. FORMING OVERLAYS – BRIDGES FOR COLLABORATORS

We introduce overlays as an interoperation interface to the collaborators. Central to this utility is a flexible inter-linkable naming mechanism based on chameleon hash functions. We begin this section with a sub-section on highlighting the importance of names as a mnemonic handle in access control, its usage in current practice and make case for local name spaces. Then we present the cryptographic primitive – chameleon hash function – on which our naming mechanism is based. A brief scenario showing usage of overlays is also presented at the end of this section.

2.1 Importance of names as a mnemonic handle in access control

The most important function of a name is to serve as a mnemonic handle for some human user, it is important that users be able to create names rather freely using well-chosen identifiers [13]. Especially, names have been proved very useful when they refer to a group of entities (plausibly of same type) since one can use such a handle to specify and enforce a policy over members of the group just by referring the name [18]. Time-line of access control systems shows that, for potentially large setups, we had to move from flat subject-object capability list (AC-matrix) to RBAC family [18, 14, 41, 42] for the sake of efficient management of the setup. Manageability is an important aspect of an access control framework. Efficient management of resources ensures their availability which is, in fact, an important requirement for secure systems. RBAC [18], the usual contender for access control in large setups, introduces special names (introduced as RBAC abstract elements) like USERS, ROLES, OBS, PRMS, etc., to group together its users, their roles, objects, and permissions over objects. Then it specifies relations among such abstract elements to achieve efficient management of the setup. For example, the many-to-many set relation between abstract elements ROLES and PRMS; where members of the group ROLES are mapped to the members of set PRMS and vice versa. Thus assigning an user a role from ROLES group essentially empowers the user with permissions assigned to that role. This indirect binding of users (through ROLES) to the possible permissions over a resource (OBS) in RBAC framework provides the following advantages. These advantages, due to the *access control model* component of access control system, come at the cost of granularity. The granularity requirements of the system should be achieved through the *enforcement mechanisms* component, so that the former retains its simplicity and manageability.

- It helps in writing manageable policies using the abstract elements like; USERS, ROLES, PRMS, etc., and enforcing actual authorizations by resolving entity's membership to appropriate sets.
- The resource need not maintain the actual list of users and their respective set of permissions. Thus reducing the size of ACL (Access Control List), and the ACL look-up time.
- Therefore, addition or deletion of users from the setup need not be reflected in the ACL.
- Similarly, temporary suspension of set of permissions or introduction of new permissions need not be reflected in the ACL.

However, the utility of these special handles (names or abstract elements) under RBAC is limited to the administrative domain in which they are defined. In other words, the abstract name ROLES in one RBAC domain is different from the abstract name ROLES in another RBAC domain, i.e., the name definitions are local to the domains. It would be very useful in collaborative environments to have an ability to refer to the names defined in other administrative domains [27]. The challenge lies in devising a mechanism with minimum inter-domain communication costs. Since, in a stand alone RBAC implementation the access control decisions essentially boil down to set-membership queries. In other words, a user requesting some permission over certain object must have its membership in a role that has been mapped to the requested permission. In a dynamic stand alone RBAC setup, the members of abstract elements (e.g., ROLES, PRMS, etc.) are continuously updated by the domain administrator and this state change is readily available within the domain while making access control decisions.

Frameworks like CBAC [14], exploit above listed advantages for collaborative environment by modeling abstraction over the abstract elements from collaborating RBAC domains (introducing a set of abstract elements, for example, COALITION, PARTNERORGANIZATION, ORGANIZATIONASSETS, etc.) The resulting entity after integrating the domains is a virtual organization and RBAC specifications will be used to manage the new virtual abstract entities. The model loses its manageability if one goes on integrating the virtual organizations in further collaborations. The state changes in any of the participating domain create cascading effect. Also, in such frameworks (e.g., [14, 41, 42]), it is not possible to accommodate a domain with non-RBAC framework since it does not have the abstract elements defined. In the following we introduce the chameleon hash function and its properties.

2.2 Chameleon hash function and its properties

DEFINITION 1. *A* chameleon hash function [25, 12] *is a one-way hash function like any other universal hash function like SHA-1* [32], *except that the function is public-key dependent and the corresponding private-key gives an ability to efficiently find a pre-image* [36] *colliding to a pre-computed hash generated with another pre-image.*

A *chameleon hash function* is associated with a pair of public and private keys (the latter called a *trapdoor* or *collision key*) and has the following properties [25].

1. Anyone who knows the public key can compute the associated hash function.
2. For those who don't know the trapdoor the function is collision resistant [36] in the usual sense, namely, it is infeasible to find two inputs which are mapped to the same output.
3. However, the holder of the trapdoor information can easily find collisions for every given input.

Let, K and SK denote an asymmetric key pair, where K is a public key (or *hash key*) while SK represents the corresponding private key. $CH_K(.,.)$ denotes the associated chameleon hash function, which can be computed efficiently given the value of K. On input (pre-image) a message m and a random string r, this function generates a hash value $CH_K(m,r)$ which satisfies the following properties [25].

Collision resistance There is no efficient algorithm that on input the public-key K can find pairs m_1, r_1 and m_2, r_2 where $m_1 \neq m_2$, such that $CH_K(m_1, r_1) = CH_K(m_2, r_2)$, except with negligible probability.

Trapdoor Collision There is an efficient algorithm that on input the trap-door SK, any pair m_1, r_1, and any additional message m_2, finds a value r_2 such that $CH_K(m_1, r_1) = CH_K(m_2, r_2)$.

Uniformity All messages m induce the same probability distribution on $CH_K(m,r)$ for a given r chosen uniformly at random.

Henceforth, we shall use a principal's name as subscript to his public key, i.e., K_A denotes public-key of principal A and the use of corresponding private key SK_A is implied when reference is made to find chameleon hash collisions by the principal. Therefore, CH_{K_A} denotes the chameleon hash function associated with principal K_A. We alternatively refer a principal by his public key.

DEFINITION 2. Commitment hash: *A principal "A", denoted by its public-key K_A, constructs a message M_A and randomly chooses a number R_A to obtain chameleon hash value X_A by applying $CH_{K_A}(.,.)$ over M_A and R_A, i.e., $CH_{K_A}(M_A, R_A) = X_A$. The message M_A used to produce the commitment hash is called* commitment-hash-message.

A commitment hash X_A produced over pre-image (M_A, R_A) by principal K_A is denoted by a four-tuple: $< K_A, M_A, R_A, X_A >$.

DEFINITION 3. Commitment: *Principal "A" issues a commitment for a message m_i over commitment hash $< K_A, M_A, R_A, X_A >$ by finding r_i, such that $CH_{K_A}(m_i, r_i) = CH_{K_A}(M_A, R_A) = X_A$. The message m_i used to produce the commitment is called* commitment-message.

A commitment r_i, issued by principal K_A (using its trapdoor), over a commitment hash $< K_A, M_A, R_A, X_A >$ for a given commitment-message m_i is denoted by a six-tuple: $< K_A, m_i, r_i, M_A, R_A, X_A >$. Commitments for a given commitment hash can only be found with the knowledge of trapdoor.

Having provided the definitions and properties of chameleon hash functions, we would like to proceed to our name definition facility in next sub-section. We would like to note that, the input messages (*commitment-hash-message* and *commitment-message*) to chameleon hash functions are text strings. We are free to decide the contents of these strings. One can use this fact to convey desired semantics. A principal can utilize the *commitment-hash-message* M to convey certain semantics, by choosing an R and generating the hash X. Thus, the commitment hash $< K, M, R, X >$ is an assertion made by the principal K about M. To enforce the semantics in M, the principal K issues commitments to intended principals by embedding their identities (e.g., public-key) into the *commitment-messages*. For example, $< K, m_1, r_1, M, R, X >$ is a commitment issued by principal K to principal K_V, if $m_1 = K_V$. In the following sub-section, we exploit this setting to define names and binding entities to them.

2.3 Defining names and binding entities to names

The concept of empowering a domain administrator to aggregate collaboration specific entities under local and extended names is motivated by SPKI/SDSI [13] philosophy. However, our approach differs from SPKI/SDSI in the technique used to define and bind names. SPKI/SDSI uses certificates to define and bind names while we use chameleon hash functions. The discussion of relative merits of our mechanism are deferred until Section 5. We borrow the following definitions from SPKI/SDSI and give our constructions to define and bind names using chameleon hash functions.

All principals are represented by their public keys. A principal is an individual, process, or active entity whose messages are distinctively recognizable because of the cryptographic operations (commitment hash and commitments) they perform on them using the public key that represents them. It is convenient to say that the principal *is* its public key.

DEFINITION 4. [13] *An* identifier *is a word over some given standard alphabet.*

EXAMPLE 1. *"Collaborators", "Employees", "TeamDBA" are examples of valid identifiers.*

A local name is a pair consisting of a public key and an arbitrary identifier. Each public key has its own associated local name space.

DEFINITION 5. [13] *A* local name *is a sequence of length two consisting of a key followed by a single identifier.*

EXAMPLE 2. *"K_A Collaborators", "K_B Collaborators", "K_A Users", "K_B TeamDBA" are valid local names.*

Local names in different name spaces are unrelated to each other, even if they use the same identifier. There are many reasons to use local names:

- To provide a convenient user-friendly handle for referring to another principal.
- To provide a level of abstraction that separates name one uses to refer to the principal from the keys the principal uses, since the later may change.
- To allow another party to provide the desired definition, by having one name defined in terms of a name defined by another party.
- To have a name that refers to a collection (or *group*) of principals.
- To have a name that can be used as an binary attribute–by defining the group of principals that possess that attributes.

DEFINITION 6. [13] *An* extended name *is a sequence consisting of a key followed by* two or more *identifiers.*

EXAMPLE 3. *"K_A CID411Users TeamDBA" is a valid* extended name *which is bound to local name "K_A CID499Users" in the following way.*

In the following we provide our constructions for defining local names, extended names, authorizations and binding subjects to them.

Defining a Local Name: A principal chooses an arbitrary *identifier* and constructs the *commitment-hash-message* in a manner shown in Figure 1, to generate a *commitment hash* by applying its chameleon hash function. For example, principal K_A defines a local name "K_A *CID*411*Users*" by constructing M_A as shown below and producing commitment hash X_A such that $CH_{K_A}(M_A, R_A) = X_A$. By identifier string "*CID*411*Users*" we try to convey principal K_A's intention to club together users taking part in a collaboration identified by number "411". To distinguish other potential name definitions by principal K_A, we put an additional (small-letter) subscript to the commitment-hash-message, the corresponding random seed, and the commitment hash. And, the same subscript will follow for respective commitment messages used for name bindings. Thus, we would like to denote the name definition mentioned above, as: $CH_{K_A}(M_{Aa}, R_{Aa}) = X_{Aa}$ and the corresponding four-tuple notation by: $< K_A, M_{Aa}, R_{Aa}, X_{Aa} >$.

Binding Subjects to Local Names: To bind a subject to a local name, owner of the local name constructs a *commitment-message* in a manner shown in Figure 2, to generate a *commitment* for a given commitment hash (i.e., local name). For example, principal K_A binds a subject K_{U_1} to its local name "K_A $CID411Users$" by constructing m_{Aa_1} as shown below and finding r_{Aa_1} such that $CH_{K_A}(M_{Aa}, R_{Aa}) = CH_{K_A}(m_{Aa_1}, r_{Aa_1}) = X_{Aa}$, holds. Unlike SPKI/SDSI, where both name definition and binding are done just by issuing a name certificate, one must issue a commitment hash pertaining to a local name definition in order to bind subjects to it. For the sake of brevity, we use the following notation to show name binding:

$$K_A\ CID411Users \longrightarrow K_{U_1} \tag{1}$$

Similarly, to bind principals K_{U_2}, K_{U_3} to local name "K_A $CID411Users$", principal K_A constructs commitment-messages m_{Aa_2} and m_{Aa_3} in similar fashion shown above and finds r_{Aa_2} and r_{Aa_3}, respectively. Therefore,

$$K_A\ CID411Users \longrightarrow K_{U_2} \tag{2}$$

and,

$$K_A\ CID411Users \longrightarrow K_{U_3} \tag{3}$$

For the sake of convenience and simplicity, we collectively denote Equations 1, 2, and 3 by the following:

$$K_A\ CID411Users \longrightarrow \{K_{U_1}, K_{U_2}, K_{U_3}\} \tag{4}$$

A subject can be a local name. Therefore, following assignment is valid.

$$K_A\ CID411Users \longrightarrow K_{U_3}\ TeamDBA \tag{5}$$

where K_{U_3}'s name definition for "*TeamDBA*" is given in Figure 3, and its members are bound by Equation 6 below.

$$K_{U_3}\ TeamDBA \longrightarrow \{K_{U_3}, K_{U_4}\} \tag{6}$$

Therefore,

$$K_A\ CID411Users \longrightarrow \{K_{U_1}, K_{U_2}, K_{U_3}, K_{U_3}\ TeamDBA\} \tag{7}$$

We have seen that a subject can be a key or a local name. Following is an example where subject is an extended name which is bound to local name "K_A $CID499Users$".

$$K_A\ CID499Users \longrightarrow K_A\ CID411Users\ TeamDBA \tag{8}$$

The meaning of extended names is defined in terms of the meaning of related local names. Informally, in above binding, members of local name "K_A $CID499Users$" are members of name "*TeamDBA*" defined in name space of principals belonging to local name "K_A $CID411Users$". Therefore, intuitively;

$$K_A\ CID499Users \longrightarrow \{K_{U_3}, K_{U_4}\} \tag{9}$$

A name is thus either a local name or an extended name.

Name Membership Proofs: Local names and extended names can be used as rules in ACLs of protected resources. Consider a scenario in which principal K_B puts "K_A $CID499Users$" and "K_A $CID411Users$" into positive ACLs of resources under its control. That is, any requester that can prove its membership to one of the listed names in ACLs is allowed to access the resource. The same algorithm for name rewriting and certificate chain discovery [13] can be used for our cryptographic constructions. On input a set of commitments (name bindings), the algorithm efficiently finds name membership proofs, if any. Due to space limitations we exclude elaborating the algorithm for proof construction and show it only intuitively. Principals $K_{U_1}, K_{U_2}, K_{U_3}, K_{U_4}$ can successfully access resources under the control of principal K_B. Proofs for principals K_{U_1}, K_{U_2}, and K_{U_3} are straight forward since their respective *commitments* (cf. Equations (1), (2), and (3)) prove their membership to name "K_A $CID411Users$". Whereas, proofs of principals K_{U_3}, K_{U_4} consists of chaining of two *commitments* – one from principal K_{U_3} and other from K_{U_1}. Principal K_{U_3} can access resources under K_B's control in two different capacities (roles), since it possesses two proofs satisfying the ACLs. K_{U_4}'s proof is sketched below.

Since, $K_A\ CID411Users \longrightarrow K_{U_3}\ TeamDBA$ (cf. Equation (7))
and, $K_{U_3}\ TeamDBA \longrightarrow K_{U_4}$ (cf. Equation (6))
$\therefore\ K_A\ CID411Users \longrightarrow K_{U_4}$

Extending **Definition 2**, we say that;

DEFINITION 7. *A commitment hash produced in order to define a local name is called* name commitment hash*. The respective commitments issued to bind subjects to the name are called* name commitments.

EXAMPLE 4. *All constructions shown above are examples of name commitment hash and name commitments.*

DEFINITION 8. *An* authorization commitment hash *is similar to name commitment hash by construction except that the commitment-hash-message contains an additional construct* "PERMS" *to indicate what all permissions members of the* "Name" *construct (i.e., name) shall inherit. The respective commitments issued to bind subjects to the name are called* authorization commitments

EXAMPLE 5. *Figure 4 shows a typical construction of name commitment-hash-message M_{Ba} by principal K_B to define name "K_B $CID244$" with authorizations specified under* "PERMS" *construct.*

Note the composition of "PERMS" construct. Principal K_B, owner of the name "K_B $CID244$" has used the set of permissions at its disposal by the underlying access control model; RBAC in this case. The above authorization commitment hash is intended to regulate access requests (by placing this *authorization commitment hash* in ACL of protected resource, say a database, under K_B's administration) from members of K_B's "$CID244$" group, which is binded to its collaborator K_A by the following binding (*authorization commitment*).

$$K_B\ CID244 \longrightarrow K_A\ CID411Users \tag{10}$$

Thus, principals K_{U_1}, K_{U_2}, K_{U_3}, K_{U_4} – all members of "K_A $CID411Users$" group, can exercise all permissions over K_B's protected resource that are allowed to the “Manager” ROLE in K_B’s administrative domain; except the “Update” operation.

Name commitments are distinguished from authorization commitments by the presence or absence of the “PERMS” construct. Similarly, commitment-hash-messages.

Enriching the commitment-hash-message: As mentioned before, the *commitment-hash-message* part of the pre-image to a chameleon hash function is a text string and its composition can be done as per the requirements. Here we provide one more useful construct that is typically required in collaborations – accommodating collaborator until the life time of a task. In our previous examples of commitment-hash-message compositions we have seen the construct “Validity” used to specify literal time intervals. We show another example of using this construct to hold a temporal variable “TASK”. Figure 5 shows one such composition validating the authorizations for group “K_B $CID244$” only for the life time of TASK “T”. In Appendix A we show how the facility of delegation can be incorporated by introducing a construct “Also-honor”.

2.4 Overlays as Bridges for Collaborators

In the previous sub-section, we have seen the ability of principals to define and bind names and authorizations. Here we shall see how principals can utilize these abilities to form overlays for the purpose of collaboration. An *overlay* is an interface provided to a peer collaborator in order to accommodate each others’ resources and users. An overlay in its simplest form consists of a pair of name and authorization definitions. Name definitions for aggregating users of host domain, and authorization definitions for specifying what authorizations on host domain’s shared resources are permissible for users from visiting domain. Therefore, for a collaboration, the collaborators design their respective overlays taking into consideration each others’ requirements. In other words, a collaborator (say, K_B) can demand a certain group-wise structure (cf. Equation 4) or hierarchy (cf. Equation 5, and 6) over the collaborating users from visiting domain (i.e., K_A’s administrative domain). Vice versa, K_A’s requirements shall be incorporated in name definitions constituting K_B’s overlay.

For example, consider two administrative domains $\mathbb{A}$ and $\mathbb{B}$ willing to collaborate. Let principals K_A and K_B be the administrators controlling users and resources in domains $\mathbb{A}$ and $\mathbb{B}$, respectively. For the sake of simplicity, let us assume that resources from domain $\mathbb{A}$ and users from domain $\mathbb{B}$ are not participating in the collaboration (In next Section we provide a comprehensive example where both counterparts are contributing their users and resources). Therefore, overlay of domain $\mathbb{A}$ will have only name definitions and overlay of domain $\mathbb{B}$ will have only authorization definitions. Upon mutual agreement, following are their overlays.

Overlay from domain $\mathbb{A}$:

$$< K_A, M_{Aa}, R_{Aa}, X_{Aa} > \tag{11}$$

Overlay from domain $\mathbb{B}$:

$$< K_B, M_{Bb}, R_{Bb}, X_{Bb} > \tag{12}$$

Enforcing Collaboration: To enforce the collaboration, principal K_B i) empowers the users of K_A by issuing an authorization commitment (for the name definition provided by K_A in its overlay), and ii) signs the commitment hash of name definitions from K_A’s

$M_A :=$

Name	:=	K_A $CID411Users$
Validity	:=	not-before “2006-09-01_00:00:00”
		not-after “2007-08-31_23:59:59”

Figure 1: Typical usage of *commitment-hash-message* part of the pre-image to define local names

$m_{Aa_1} :=$

Subject	:=	K_{U_1}

Figure 2: Typical usage of *commitment-message* part of the pre-image to bind subjects to local names

$M_{U_3a} :=$

Name	:=	K_{U_3} $TeamDBA$
Validity	:=	not-before “2006-09-01_00:00:00”
		not-after “2007-08-31_23:59:59”

Figure 3: K_{U_3}’s name definition for “*TeamDBA*”

$M_{Ba} :=$

Name	:=	K_B $CID244$
PERMS	:=	PRMS (ROLE.Manager) - PRMS.Update
Validity	:=	not-before “2006-09-01_00:00:00”
		not-after “2007-08-31_23:59:59”

Figure 4: Typical usage of *name commitment-hash-message* to define local names with authorizations

$M_{Bb} :=$

Name	:=	K_B $CID244$
PERMS	:=	PRMS (ROLE.Manager) - PRMS.Update
Validity	:=	TRUE (TASK.T)

Figure 5: Containing validity of authorizations through TASK variable

overlay.

$$K_B\, CID244 \longrightarrow K_A\, CID411Users \tag{13}$$

$$\{X_{Aa}\}_{SK_B} \tag{14}$$

Principal K_A need not issue any such authorization commitments for K_B, since its resources are not taking part into collaboration. In this fashion, K_B formed a bridge with the help of K_A to facilitate users from domain $\mathbb{A}$ to access resources in domain $\mathbb{B}$. In next section, we will see a comprehensive scenario where both the collaborators are actively participating with possible concurrent collaborations with third parties. We shall also see the privacy implications in such multi-layer collaborations.

3. A TYPICAL COLLABORATION SCENARIO

In this section we shall see full potential of our mechanism in terms of the ease it brings in forging concurrent collaborations, interlinking collaborators, and privacy.

We explain these with the help of a scenario, graphically depicted in Figure 6. Figure 6(a), shows two autonomous administrative domains (collaborators) $\mathbb{A}$ and $\mathbb{B}$ negotiating for a collaboration. Let us assume, $\mathbb{A}$ is a software firm that provides IT related services. $\mathbb{B}$ is a big industrial organization that is willing to engage $\mathbb{A}$ to cater its IT related needs. *A* and *B* are the actual sub-domains of these respective organizations that are actively collaborating. Let K_A and K_B be the public keys of administrators responsible to manage these sub-domains. For this probable collaboration to go forward, *B* needs to open up its resources so that experts from domain *A* can perform their jobs for *B*. And auditors from *B* need to access an auditing tool licenced to *A*. In order to accommodate these mutual requirements, *A* and *B* propose their overlays to each other as shown in Figure 7(a). Note that the specifics of their exact collaboration related tasks are abstracted under variable "T_1". However, name definitions have explicit time interval specified. In other words, name definitions and corresponding bindings are valid for the specified time interval but the authorization definitions and corresponding bindings are valid only during the life time of "TASK".

Figure 6(b) and 7(b) show the steps involved in enforcing the collaboration among *A* and *B*, where *A* empowers the users from *B* by issuing an authorization commitment and similarly *B* does it for *A*'s users. *A* and *B* sign the chameleon hash values of each others' name definitions as an agreement for collaboration. We denote collaboration by • operator and place a subscript to it that holds context of the collaboration. Collaboration among *A* and *B* is thus denoted as: $A \bullet_{T_1} B$. Concurrently, *A* starts negotiating another collaboration with *C*. The motivation is, *B* comes forward with some IT job (say, compiling a huge data set of its customers and their spending habits) which *A* should do but has no expertise in data-mining technology. Therefore, *A* wants to take help from data-mining experts from *C*. *A* and *C* negotiate their respective overlays for this collaboration. For this collaboration, *C* is participating only with its users, where *A* is offering its resources (which are actually *B*'s resources – the data set). This is possible because of the "Pre-enforcement" setting done in *A*. The pre-enforcement settings and enforcement steps by *A* for *C* are listed in Figure 7(c). But, there is a caveat. The context for $A \bullet_{T_2} C$ is T_2. The data-mining experts from *C* can work on the data set of *B* iff $T_2 \subseteq T_1$. This provides a mean to address the typical requirement of decomposing a task into sub-tasks and satisfying sub-tasks from different concurrent collaborations.

Privacy – In collaboration enforcement phases, collaborators are signing chameleon hash's computed by peers as an agreement for collaboration. These are *chameleon signatures – that provide with an undeniable commitment of the signer to the contents of a signed document (as regular signatures do) but, at the same time, do not allow the recipient of the signature to disclose the contents of the signed information to any third party without the signer's consent* [25]. Thus, collaborations formed using our mechanism enjoy privacy.

The strong arrowed lines in Figure 6(b) and 6(c) indicate the bridges for users, from autonomous administrative domains, to access resources.

4. EXPERIMENTAL ANALYSIS

We have implemented three different flavors of chameleon hash functions based on i) simple factorization, ii) discrete logarithm (both from [25]), iii) advanced factorization (from [38]); and the results are tabulated in Table 1. Implementation of these schemes can be categorized into two phases: Hash Computation/Generation and Finding Collision. These schemes produce hash of length 160-bits. The values are taken over the average of 100 runs.

The implementation is carried out on a GNU/Linux (i486) platform with gcc-3.3.5, OpenSSL 0.9.7e library for cryptographic primitives (without any external cryptographic acceleration) and numerical analysis. To get a fair computational estimation, we did not use any code optimization of gcc while building our executables.

4.1 Approach to Compute Execution Time

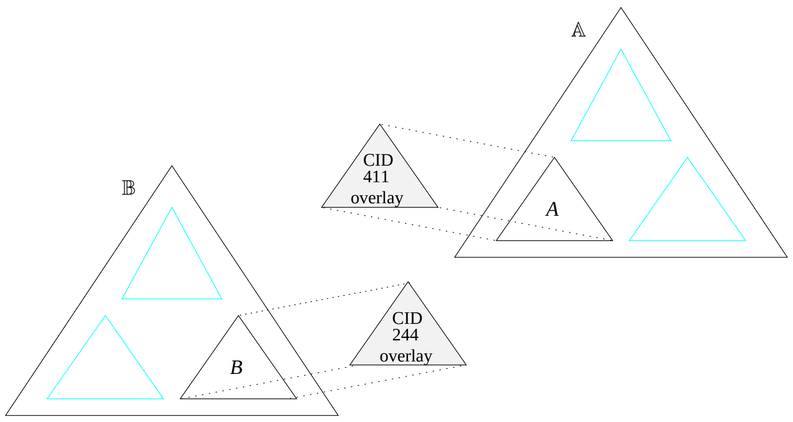
Various approaches are possible to audit the process execution time. We employed the method of tracking CPU cycles consumed during execution of a function of our interest. The experiments are carried out on an AMD 750MHz machine, that complies to the IA32 architecture (which provides cycle counter; a 64-bit, unsigned number). The IA32 counter is accessed with the `rdtsc` (read time stamp counter) instruction. This instruction takes no arguments. It sets register `%edx` to the high-order 32 bits of the counter and register `%eax` to the low-order 32 bits. Based on this methodology, a pair of functions are integrated with our code that allows us to measure the total number of cycles that elapse between any two time points:

```
#include "clock.h"
void start_counter(); /* Starts the counter */
double get_counter(); /* Returns: Number of cycles
                         since last call to
                         start_counter */
```

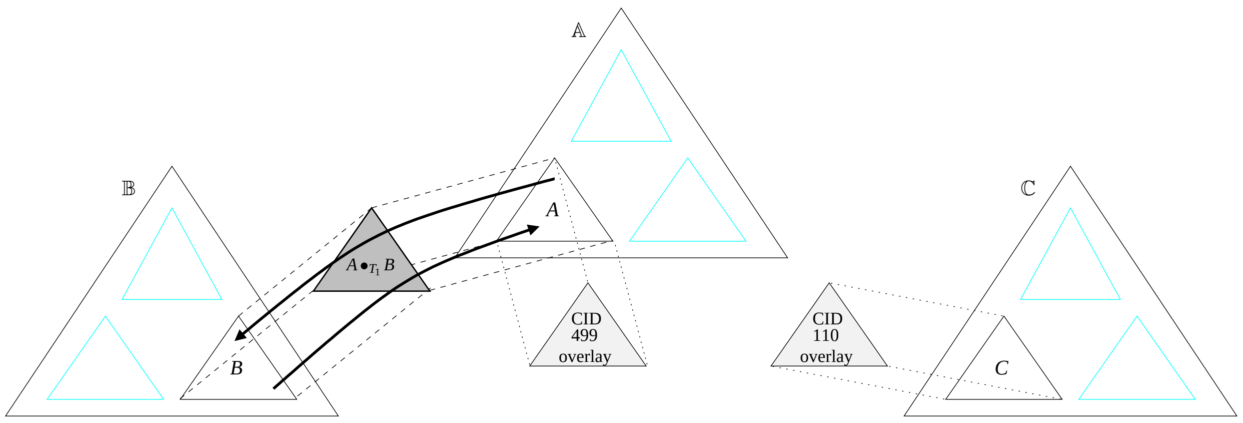
To verify the precision of this approach we marked the counter before and after `sleep(sleeptime);` function call (where sleeptime equals to one). We obtained 756,154,624.0 as return value (i.e., 756.2 MHz). We run each function of our interest for 101 times and discarded the first value of execution time in favor of cache warming process. Furthermore, results are gathered in run-level 1; to minimize interference from other processes.

4.2 Comparative Analysis

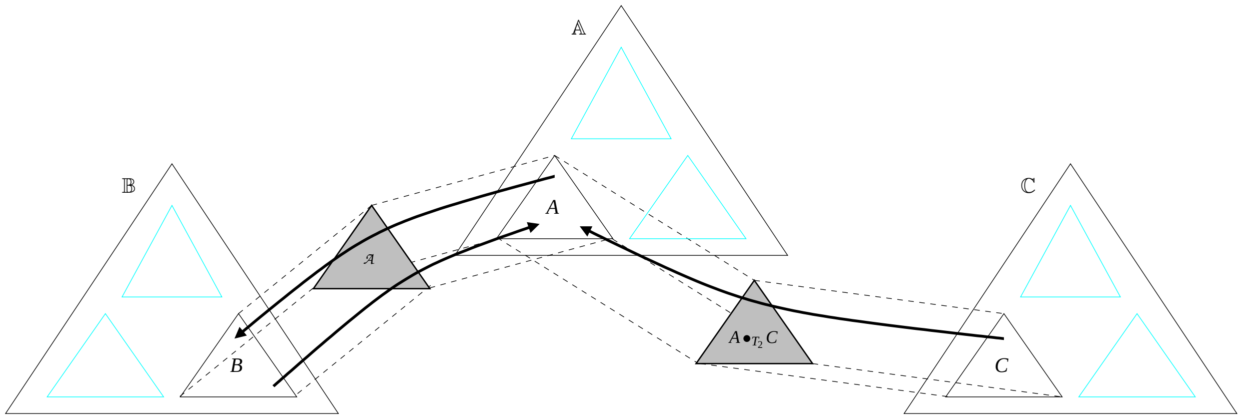
Chameleon scheme based on simple factorization gives the least time required for hash computation while the scheme based on advanced factorization gives the least time required for computing a hash collision. Generally, in a collaboration setup, there will



9(a) A and B proposing their collaboration specific overlays



9(b) A and B form collaboration for task T_1, concurrently A negotiates another collaboration with C



9(c) A and C form collaboration for task T_2, building a uni-directional bridge for C's users to reach A's resources

Figure 6: Inter-domain collaboration scenario

$CH_K(.,.)$ **Schemes**	**Simple Factorization**		**Discrete Logarithm**		**Advanced Factorization**	
	Hash Computation	Finding Collision	Hash Computation	Finding Collision	Hash Computation	Finding Collision
Time (in *ms*)	14.375	46.503	140.881	0.887	56.139	0.720

Table 1: Flavors of chameleon hash functions and their computational costs

Overlay from A:

$< K_A, M_{Aa}, R_{Aa}, X_{Aa} >$, where

M_{Aa} :=

Name :=	K_A $CID411Users$
Validity :=	not-before "2006-09-01_00:00:00"
	not-after "2007-08-31_23:59:59"

and,
$< K_A, M_{Ab}, R_{Ab}, X_{Ab} >$, where

M_{Ab} :=

Name :=	K_A $CID411$
PERMS :=	PRMS (ROLE.Auditor)
Validity :=	TRUE (TASK.T_1)

Overlay from B:

$< K_B, M_{Bp}, R_{Bp}, X_{Bp} >$, where

M_{Bp} :=

Name :=	K_B $CID244Users$
Validity :=	not-before "2006-09-01_00:00:00"
	not-after "2007-08-31_23:59:59"

and,
$< K_B, M_{Bq}, R_{Bq}, X_{Bq} >$, where

M_{Bq} :=

Name :=	K_B $CID244$
PERMS :=	PRMS (ROLE.Manager) - PRMS.Update
Validity :=	TRUE (TASK.T_1)

9(a) Internals of overlays from A and B

Enforcement by A for B:

K_A $CID411 \longrightarrow K_B$ $CID244Users$
and,
$\{X_{Bp}\}_{SK_A}$

Enforcement by B for A:

K_B $CID244 \longrightarrow K_A$ $CID411Users$
and,
$\{X_{Aa}\}_{SK_B}$

Overlay from A:

No name definitions,
Users are not participating.

$< K_A, M_{Ac}, R_{Ac}, X_{Ac} >$, where

M_{Ac} :=

Name :=	K_A $CID499$
PERMS :=	PRMS (ROLE.DBA) - PRMS.Update
Validity :=	TRUE (TASK.T_2)

Overlay from C:

$< K_C, M_{Ca}, R_{Ca}, X_{Ca} >$, where

M_{Ca} :=

Name :=	K_C $CID110Users$
Validity :=	not-before "2006-09-01_00:00:00"
	not-after "2007-08-31_23:59:59"

No authorization definitions,
Resources are not participating.

9(b) Steps involved in enforcing collaboration between A and B, Overlays from A and C

Pre-enforcement setting by A:

K_{U_3} $TeamDBA \longrightarrow K_A$ $CID499$

Enforcement by A for C:

K_A $CID499 \longrightarrow K_C$ $CID110Users$
and,
$\{X_{Ca}\}_{SK_A}$

9(c) Pre-enforcement setting done by A and steps involved in enforcing collaboration between A and C

Figure 7: Inter-domain collaboration scenario

be relatively small number of name and authorization definitions (hash computations) than the number of corresponding commitments (hash collisions) for these definitions. But the proof verification process involves hash computations, in order to verify the authenticity of commitments used in proofs, and takes the overall number of hash computations performed during the span of collaboration above the number of hash collisions performed by collaborators together. An exception to this generalization happens while the inter-domain interaction among collaborators is little and their intra-domain user assignments are frequent. Therefore, loosely speaking, collaborations can be categorized in three types – the two mentioned above and the third in which only one peer is actively participating. Again under this third category, there can be sub-categories similar to the two former types mentioned above. This categorization of collaborations makes sense while choosing the chameleon hash scheme. Collaborations in which the collective hash computations by collaborators are much higher than their relative commitments, the scheme based on simple factorization is suitable. On the other hand where the collective number of commitments is very high than the collective number of chameleon hashes, the scheme based on advanced factorization is suitable. It is interesting to know that the choice of chameleon scheme for collaboration itself is a negotiation aspect among collaborators as it decides the overall computational cost in their individual domains. This is very useful if one of the administrative domain is computationally constrained, for example an environment consisting sensors (or imagine futuristic personal area networks of electronic gadgets), where computationally powerful collaborator agrees for a scheme in which its overall computational costs are higher than its peer domain.

We hope that the importance, and capability of chameleon schemes will bring forward more efficient implementations to existence. Our implementations are available at [5].

4.3 Advantages of our mechanism

1. Our mechanism to define names, authorizations and binding entities to them allows collaborating partners to arrange their respective collaboration specific entities in a manageable and understandable form. This abstraction of collaborating entities from rest of the underlying access control setup keeps the modifications in pre-collaboration setup to the least possible – only new rules for visiting users from collaborating domain are need to be integrated in host domain's resource ACLs. Upon completion of collaboration, the rules in ACLs shall lapse and pre-collaboration functionality will be automatically restored.

2. The fact that the commitment-hash-message and commitment-message are text strings, allows us a free hand at their internal composition as per requirements. One can also utilize this fact to incorporate the XACML/SAML structure to compose these messages. The resulting definitions/assertions using such enriched pre-image messages are very useful in realizing complex policies.

3. Apart from standard signatures, sanitizable signatures [3] and undeniable signatures with full convertibility are also readily available, courtesy chameleon hash functions. The use of chameleon signatures, which is an efficient type of non-interactive undeniable signatures, as an agreement for collaboration gives a unique privacy property to collaborations formed using our mechanism.

4. Having the requirement of just an asymmetric key pair, our mechanism addresses heterogeneity of collaborators in terms of their underlying access control models, type of PKIs they use, and also their computational capabilities.

5. Keeping aside the usefulness of our naming mechanism for collaboration purpose, the mechanism is even useful in stand alone access control setups. For example, i) to design new security ordering on top of the existing one, ii) to handle requirements that are contradictory/exceptional in underlying access control model – to define private roles, over-riding hierarchy in RBAC [31, 44].

5. RELATED WORK

RBAC [35] (Role-based access control) was introduced in 1996 as a solution to the shortcomings of MAC [6] (mandatory access control) and DAC (discretionary access control) [26] frameworks. RBAC [18] became a de facto standard in large organizations with a large number of users and resources to be managed. RBAC provided a systematic way to organize users and resources. It does so by mapping users to organizational roles and permissions over resources to the set of organizational roles [18]. Subsequently, the trend of globalization and interdependence of large organizations necessitated introduction of a family of frameworks [41, 42, 14] on top of the RBAC framework.

The first of these three models – TMAC [41] (team based access control) introduced the notion of "team" to refer to a group of collaborating users acting in various roles and provided a way to assign permissions to the "team". TBAC [42] (task based access control) was introduced to synchronize access permissions with ongoing tasks and workflow instances spanning across organizations. CBAC [14] (coalition based access control) was introduced to capture the notion of "coalition" of organizations working for common task. There were also similarly motivated works [4, 23] on these lines, independent of RBAC framework.

The access control decisions in the RBAC family frameworks (i.e., TMAC, TBAC, CBAC) are based on set membership queries, as discussed in Section 1. That is, when these models try to address collaborations in a distributed environment they need to rely on a mechanism that communicates internal state of collaborating domains to all collaborators or a mediator if the collaborations are mediator facilitated. This essentially turns the whole environment into an on-line environment. These models facilitate collaboration across domains that have RBAC as their underlying access control model.

Interoperability of access control frameworks becomes a hindrance when domains with heterogeneous access control frameworks need to collaborate. Bonatti, Sapino and Subrahmanian [10] points out that even with frameworks of same type the collaborators may not use the same security orderings. Furthermore, collaborative environments also need a mechanism to authenticate and authorize requests originating from collaborating domain. SAML/XACML [37, 17] (Security Assertion Markup Language/eXtensible Access Control Markup Language) is a methodology to perform and convey inter-domain authentication and authorization. The naming scheme is canonical and the setup is on-line in nature. Also, to ascertain the properties like authenticity of assertions and their non-repudiation needs integration of a PKI. In [30] other shortcomings of XACML are discussed. X-RBAC [22] gives a XML-based specification language for multi-domain environments' policy-specification needs. X-GTRBAC [7] is a XML-based administration model for multi-domain environment that aims at enabling administration of RBAC policies in the presence of constraints with support for conflict resolution.

The reliance on a PKI is compelling in a collaborative environment formed of independent autonomous administrative domains, since the inter-domain communications also require non-repudiation. Acknowledging this real need for realizing multi-domain collaborations, several innovative approaches [29, 11, 43, 39, 21, 15, 9, 28] have been proposed. These proposals are based on X.509 type of PKI. X.509 is a centralized PKI and intended for identification [16]. Therefore schemes based on X.509 type of PKI use digital certificates and its extensions to securely authenticate users in distributed environment and then take authorization decisions. SPKI/SDSI [13] was proposed to address the shortcomings of traditional X.509 type of PKI. SPKI/SDSI uses two different certificates – name and authorization certificates. Authorization certificates are introduced to communicate authorizations in distributed environment securely. However, X.509 is the most widely deployed and used PKI in real world.

SPKI/SDSI is a very flexible and expressive framework for achieving authentication and authorization in a distributed environment. An overlay mechanism similar to ours proposed in this paper can also be achieved using SPKI/SDSI except the feature of privacy to collaborators. SPKI/SDSI also has a `tag()` field in its authorization certificates (equivalent to the freedom of composing chameleon-hash-message and commitment-message under our mechanism) where developers can introduce constructs as per their requirements.

6. CONCLUSION

We have introduced a name and an authorization definition scheme based on chameleon hash functions. We have shown how to interlink names and utilize this facility to form overlays for collaboration. With a comprehensive scenario we have explained how collaborators build bridges to accommodate each others' users and resources. We have also seen how the context of a collaboration is captured and its relation to other concurrent collaborations of collaborators. The use of chameleon signatures as a collaboration agreement provides privacy to the collaborators. Overlays provide an ease of understanding and manageability to administrators in charge of setups. Overlays also reorganize heterogeneous collaborating setups into new security orderings that are acceptable to collaborators. We also have made a case through our implementation results for computational heterogeneity among collaborators, whereby choosing an appropriate scheme the computational load can be shifted to a peer collaborator.

The facility of name interlinking and reliance of collaborators on each others name bindings, like SPKI/SDSI, leads towards a flexible and expressive trust management system with additional benefits of privacy. Usefulness of our mechanism for realizing *incomplete contracts* is worth investigating. Incomplete contracts is a practical way of signing contracts (most of real world contracts fall under this category) where all the minor details of obligations are not enlisted or cannot be explicitly specified. The property of having a fixed hash value (say, of a contract document) and then finding collisions for this hash value while keeping the signature over the fixed hash value intact, is complementary for realizing incomplete contracts.

7. REFERENCES

[1] M. Abadi, M. Burrows, B. Lampson, and G. Plotkin. A calculus for access control in distributed systems. *ACM Trans. Program. Lang. Syst.*, 15(4):706–734, 1993.

[2] X. Ao and N. H. Minsky. Flexible regulation of distributed coalitions. In *ESORICS '03, 8th European Symposium on Research in Computer Security*, volume 2808 of *Lecture Notes in Computer Science*, pages 39–60. Springer, 2003.

[3] G. Ateniese, D. H. Chou, B. de Medeiros, and G. Tsudik. Sanitizable signatures. In *ESORICS '05: Proceedings of the 10th European Symposium on Research in Computer Security*, volume LNCS (3679), pages 159–177. Springer-Verlag, 2005.

[4] V. Atluri and W.-K. Huang. An authorization model for workflows. In *ESORICS '96: Proceedings of the 4th European Symposium on Research in Computer Security*, volume LNCS (1146), pages 44–64. Springer-Verlag, 1996.

[5] M. authors. Implementation code (Blinded, as per submission instructions), 2005. http://.

[6] D. Bell and L. Lapadula. Secure computer systems: Mathematical foundations and model. Technical Report M74-244, Mitre Corp, Bedford, Mass, 1975.

[7] R. Bhatti, B. Shafiq, E. Bertino, A. Ghafoor, and J. B. D. Joshi. X-GTRBAC admin: A decentralized administration model for enterprise-wide access control. *ACM Transactions on Information and System Security (TISSEC)*, 8(4):388–423, 2005.

[8] C. Bidan and V. Issarny. Dealing with multi-policy security in large open distributed systems. In *ESORICS '98: Proceedings of the 5th European Symposium on Research in Computer Security*, pages 51–66. Springer-Verlag, 1998. Lecture Notes In Computer Science; Vol. 1485.

[9] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis. The KeyNote Trust-Management System Version 2. RFC 2704, Internet Engineering Task Force, 1999.

[10] P. A. Bonatti, M. L. Sapino, and V. S. Subrahmanian. Merging heterogenous security orderings. *Journal of Computer Security*, 5(1):3–29, 1997.

[11] D. W. Chadwick and A. Otenko. The permis x.509 role based privilege management infrastructure. In *SACMAT '02: Proceedings of the seventh ACM symposium on Access control models and technologies*, pages 135–140. ACM Press, 2002.

[12] X. Chen, F. Zhang, and K. Kim. Chameleon hashing without key exposure. In *ISC '04: Information Security, 7th International Conference*, volume LNCS (3225), pages 87–98. Springer-Verlag, 2004.

[13] D. Clarke, J.-E. Elien, C. Ellison, M. Fredette, A. Morcos, and R. Rivest. Certificate chain discovery in SPKI/SDSI. *Journal of Computer Security*, 9(4):285–322, 2001.

[14] E. Cohen, R. K. Thomas, W. Winsborough, and D. Shands. Models for coalition-based access control (CBAC). In *SACMAT '02: Proceedings of the seventh ACM symposium on Access control models and technologies*, pages 97–106. ACM Press, 2002.

[15] G. Denker, J. Millen, and Y. Miyake. Cross-domain access control via PKI. In *POLICY '02: Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks*, page 202. IEEE Computer Society, 2002.

[16] C. Ellison. Improvements on conventional PKI wisdom. In *First annual PKI Research Workshop*, 2002. http://www.cs.dartmouth.edu/ pki02/Ellison/paper.pdf.

[17] eXtensible Access Control Markup Language (XACML 2.0). OASIS Standard, http://www.oasis-open.org/committees/xacml/, 2005.

[18] D. F. Ferraiolo, R. S. Sandhu, S. I. Gavrila, D. R. Kuhn, and R. Chandramouli. Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security*, 4(3):224–274, 2001.

[19] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman. Protection in operating systems. *Commun. ACM*, 19(8):461–471, 1976.

[20] A. Herzberg, Y. Mass, J. Michaeli, Y. Ravid, and D. Naor. Access control meets public key infrastructure, or: Assigning roles to strangers. In *S&P '00: Proceedings of the IEEE Symposium on Security and Privacy*, pages 2–14. IEEE Computer Society, 2000.

[21] D. Jonscher and K. R. Dittrich. Argos – configurable access control system for interoperable environments. In *Proceedings of the ninth annual IFIP TC11 WG11.3 working conference on Database security IX : status and prospects*, pages 43–60, London, UK, UK, 1996. Chapman & Hall, Ltd.

[22] J. B. D. Joshi, R. Bhatti, E. Bertino, and A. Ghafoor. Access-control language for multidomain environments. *IEEE Internet Computing*, 8(6):40–50, 2004.

[23] M. H. Kang, J. S. Park, and J. N. Froscher. Access control mechanisms for inter-organizational workflow. In *SACMAT '01: Proceedings of the sixth ACM symposium on Access control models and technologies*, pages 66–74. ACM Press, 2001.

[24] H. Khurana, V. Gligor, and J. Linn. Reasoning about joint administration of access policies for coalition resources. In *ICDCS '02: Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)*, page 429. IEEE Computer Society, 2002.

[25] H. Krawczyk and T. Rabin. Chameleon hashing and signatures. In *NDSS '00: Proceedings of the ISOC Symposium on Network and Distributed System Security*, pages 143–154, 2000.

[26] B. W. Lampson. Protection. *ACM SIGOPS Operating Systems Review*, 8(1):18–24, 1974.

[27] N. Li. Local names in SPKI/SDSI. In *CSFW '00: Proceedings of The 13th Computer Security Foundations Workshop*, pages 2–15. IEEE Computer Society Press, 2000.

[28] N. Li, J. C. Mitchell, and W. H. Winsborough. Design of a role-based trust-management framework. In *SP '02: Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 114–130. IEEE Computer Society, 2002.

[29] J. Linn and M. Nyström. Attribute certification: an enabling technology for delegation and role-based controls in distributed environments. In *RBAC '99: Proceedings of the fourth ACM workshop on Role-based access control*, pages 121–130. ACM Press, 1999.

[30] P. Mazzoleni, E. Bertino, B. Crispo, and S. Sivasubramanian. XACML policy integration algorithms: not to be confused with XACML policy combination algorithms! In *SACMAT '06: Proceedings of the eleventh ACM symposium on Access control models and technologies*, pages 219–227. ACM Press, 2006.

[31] J. D. Moffett. Control principles and role hierarchies. In *RBAC '98: Proceedings of the third ACM workshop on Role-based access control*, pages 63–69. ACM Press, 1998.

[32] NIST. Secure Hash Standard (SHS). *Federal Information Processing Standards Publication 180-1*, April 1995.

[33] V. Patil and R. K. Shyamasundar. Towards a flexible access control mechanism for e-transactions. In *EGCDMAS '04: International Workshop on Electronic Government, and Commerce: Design, Modeling, Analysis and Security*, pages 66–81. INSTICC, 2004.

[34] R. Sandhu. Role activation hierarchies. In *RBAC '98: Proceedings of the third ACM workshop on Role-based access control*, pages 33–40. ACM Press, 1998.

[35] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.

[36] B. Schneier. *Applied Cryptography, Second Ed.* John Wiley and Sons, 1996.

[37] Security Assertion Markup Language (SAML V2.0). OASIS Standard, http://www.oasis-open.org/committees/security/, 2005.

[38] A. Shamir and Y. Tauman. Improved online/offline signature schemes. *CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, Springer-Verlag, LNCS(2139):355–367, 2001.

[39] D. Shands, R. Yee, J. Jacobs, and E. J. Sebes. Secure virtual enclaves: Supporting coalition use of distributed application technologies. *ACM Trans. Inf. Syst. Secur.*, 4(2):103–133, 2001.

[40] M. Shehab, E. Bertino, and A. Ghafoor. Secure collaboration in mediator-free environments. In *CCS '05: Proceedings of the 12th ACM conference on Computer and communications security*, pages 58–67. ACM Press, 2005.

[41] R. K. Thomas. Team-based access control (TMAC): a primitive for applying role-based access controls in collaborative environments. In *RBAC '97: Proceedings of the second ACM workshop on Role-based access control*, pages 13–19. ACM Press, 1997.

[42] R. K. Thomas and R. S. Sandhu. Task-based authorization controls (TBAC): A family of models for active and enterprise-oriented autorization management. In *Proceedings of the IFIP TC11 WG11.3 Eleventh International Conference on Database Securty XI*, pages 166–181, 1998.

[43] M. Thompson, W. Johnston, S. Mudumbai, G. Hoo, K. Jackson, and A. Essiari. Certificate-based access control for widely distributed resources. In *"SEC '99: Proceedings of the 8th USENIX Security Symposium"*, pages 215–228. USENIX, 1999.

[44] X. Zhang, S. Oh, and R. Sandhu. PBDM: a flexible delegation model in RBAC. In *SACMAT '03: Proceedings of the eighth ACM symposium on Access control models and technologies*, pages 149–157. ACM Press, 2003.

APPENDIX

A. DELEGATION

Consider Figure 6(c), where collaboration between domain A and domain B, i.e., $A \bullet_{T_1} B$ is denoted by $\mathcal{A}$. $\mathcal{A}$ is a new virtual domain which can forge new collaborations further. To do so, either A or B has to control this virtual domain in order to construct an overlay for further collaborations arising out of $\mathcal{A}$. Let us assume A is controlling this new virtual domain and made responsible to handle further collaborations of $\mathcal{A}$. To make this happen, B need to delegate rights over its shared resources for $A \bullet_{T_1} B$ to A. We introduce a delegation construct "Also-honor" for this purpose, see Figure 8. This a way to inform resources of B to accept proofs that contains authorization commitments issued by A.

M_{Br} :=

Name	:=	K_B CID244
PERMS	:=	PRMS (ROLE.Manager) - PRMS.Update
Validity	:=	TRUE (TASK.T_1)
Also-honor	:=	K_A

Figure 8: Delegating authorization

One can also think of an additional construct to specify the status of further delegation by subjects.

Security violations arising out of this delegation facility are not addressed within the mechanism. However, the mechanism proposed in [40] can be used to check the violations in respective collaborating domains.