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

Department of Computer Science
University of Rome - La Sapienza

ABSTRACT

Access control models need to be interoperable when administrative domains with heterogeneous access control models need to collaborate. Even, collaboration among homogeneous access control models is not straight-forward due to the different security orderings they might employ. In this paper, we briefly put forward an overlay formation mechanism based on chameleon hash functions. The mechanism allows collaborators to map their collaborating entities into a new collaboration specific security ordering that is agreeable to the peer collaborator. Collaborators use overlays as interoperation interfaces. By digitally signing each others' overlays, organizations enter into collaboration. Since overlays are virtual mappings, defining an overlay does not interfere with the access control model of the host organization. The use of overlays hides the internal security ordering of an organization from its collaborators. The trapdoor collision property of chameleon hash function ensures the privacy of collaboration agreements.

Categories and Subject Descriptors

D.4.6 [**Operation System**]: security and protection—*Authentication, Cryptographic controls*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms

Security, Design, Management.

Keywords

access control, interoperability, name spaces, chameleon hash.

1. INTRODUCTION

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, work-flow 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, *et al.* Depending upon the size and the functional requirements, organizations deploy different types of access control models; standard or proprietary. Such a heterogeneity in access control models demands their interoperability when organizations need to enter into collaborations. Our goal is to provide a simple and manageable mechanism that allows collaborators to easily integrate each others' users and resources, irrespective of the heterogeneity in their underlying access control (AC) models.

The primitive goal of an access control framework is to efficiently manage the entities (users and resources) under its control. A typical deployment of an access control *framework* 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 among them. The policy languages are employed to provide properties that are difficult to achieve under AC model alone, e.g., context-sensitive access requests. And, the enforcement mechanisms are employed to enforce outcomes of access requests to a resource. They are also used in situations where certain requirements are contrary to the inherent properties of the underlying AC model. 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.

An 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 mechanisms* component of access control frameworks, i.e., by integrating cryptographic primitives. Therefore, the choice of PKI plays a pivotal role in deciding the autonomous and dynamic nature of the resulting collaborative environment. The use of X.509 type of PKI – a top-down, centralized architecture – does not allow the AC frameworks to remain truly decentralized. SPKI [1] is an alternative PKI suitable for distributed environments. Our approach is partially motivated by the treatment of names in SPKI. However, our approach provides following distinct advantages over SPKI-based approach for interoperability: i) usability – XML based security assertions can be readily used without making them part of certificates, ii) privacy – collaborators cannot reveal the collaboration specific details to others without peers' consent, and iii) identifying asymmetric key pairs as a common denominator among collaborators, nullifies the heterogeneity of their deployed PKIs.

*This work was partially funded by the WEB-MINDS project supported by the Italian MIUR under the FIRB program.

†The author is supported by "Fellowship for young Indian researcher, 2005" from MIUR.

ASIACCS'07, March 20-22, 2007, Singapore.

2. DESIGNING OVERLAYS

To achieve interoperability among heterogeneous AC models, we have empowered the *enforcement mechanisms* component of access control frameworks by providing an utility to form overlays. The utility derives its strength from the trapdoor collision property of chameleon cryptography and the naming treatment of SPKI. In the following we shall briefly explain the properties of chameleon hash function and our approach to use these properties to devise a naming scheme. Then we shall explain the use of this naming mechanism to form overlays and use of such overlays as collaboration interfaces to forge collaborations.

2.1 Chameleon hash function

DEFINITION 1. [3] *A* chameleon hash function *is a one-way hash function like any other universal hash function, e.g., SHA-1, except that the function is public-key dependent and the corresponding private-key gives an ability to efficiently find a pre-image 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 [3].

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 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 [3].

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.

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.

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.

2.2 Naming mechanism

Our approach of name treatment differs from SPKI in the technique used to define and bind names. SPKI uses certificates to define and bind names while we use chameleon hash functions. A principal uses *commitment hash* to define names and *commitments* to bind entities to names.

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 (administrator of collaborative domain "*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 $CH_{K_A}(M_A,R_A) = X_A$.

$M_A :=$

Name	:=	K_A *CID*411*Users*
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

By the 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". The above name definition in four-tuple form: $< K_A, M_A, R_A, X_A >$, that is $CH_{K_A}(M_A,R_A) = X_A$.

Binding a subject to name: 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 *CID*411*Users*" by constructing m_{A_1} as shown below and finding r_{A_1} such that $CH_{K_A}(M_A,R_A) = CH_{K_A}(m_{A_1},r_{A_1}) = X_A$, holds. Therefore, K_{U_1} can use the *commitment* $< K_A, m_{A_1}, r_{A_1}, M_A, R_A, X_A >$ as its membership proof to $< K_A, M_A, R_A, X_A >$. In this fashion K_A binds together all the users, irrespective of their current location in the security ordering, that shall participate in collaboration "*CID*411". In the name binding example shown in Figure 2, subject's identity is directly specified. K_A may opt to bind a set of subjects defined under other local name. This kind of indirect binding produces *extended names*. Due to space restrictions, we could not elaborate construction of extended names and membership proofs constructed using them. For complete details, readers are encouraged to refer [4].

m_{A_1} := | Subject := K_{U_1} |

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

Conferring authorizations: Authorizations are conferred over names or subjects by issuing an authorization commitment hash which 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 such definitions are called *authorization commitments*. Figure 3 shows a typical construction of name commitment-hash-message M_B by principal K_B (administrator of collaborative domain "*B*") to define name "K_B *CID*244" with authorizations specified under the construct "*PERMS*".

M_B :=

Name	:=	K_B *CID*244
PERMS	:=	PRMS (ROLE.Manager) - PRMS.Update
Validity	:=	TRUE(Task.*T*)

Figure 3: Typical usage of *name commitment-hash-message* to confer authorizations

Note the composition of construct "*PERMS*". Principal K_B, the owner of name "K_B *CID*244" has used a set of permissions at its disposal by the underlying access control model; RBAC [2] in this case. Also note the construct "*Validity*", the authorization is valid until the task "*T*" is true. The above authorization commitment hash is intended to regulate access requests (by placing it in the ACL of shared resource of domain "*B*".)

2.3 Forging collaboration

We have seen the constructions for name and authorization definitions and their binding with subjects. Domain administrators make use of name and authorization definitions to form overlays. An overlay in its simplest form consists of at least one name or authorization definition. The set of *name definitions* in an overlay are called *out definitions* since they are meant for mapping outgoing users from local domain into a new collaboration specific ordering as per the requirements of a peer collaborator. The set of *authorization definitions* in an overlay are called *in definitions* since they empower the incoming users from a collaborating domain.

Let us briefly explain a simple collaboration involving two administrative domains (domain "*A*" and "*B*") keeping in perspective the name and authorization definitions given in the previous subsection. To forge a collaboration between domain "*A*" and domain "*B*", the respective domain administrators K_A and K_B negotiate designs of their respective overlays. K_A binds its participating users under the *out definition* "K_A *CID*411*Users*". On the other hand to accommodate the incoming users from domain "*A*", K_B makes authorization provisions under its *in definition* "K_B *CID*244". As a collaboration forging step, K_B binds its *in definition* with the *out definition* from K_A by issuing an authorization commitment that makes "K_A *CID*411*Users*" member of "K_B *CID*244" and the collaborators digitally sign each others' overlays.

In the above collaboration only the users from domain "*A*" and resources from domain "*B*" are taking part. Intuitively, a user from domain "*A*" composes its authorization proof with the help of commitments that vouch its membership to domain "*A*'s" *out definitions* and the authorization commitment issued by domain "*B*" to couple together domain "*A*'s" *out definitions* with domain "*B*'s" *in definitions*.

2.4 Distinct advantages

1. Our mechanism to define names, authorizations and binding entities to them allows collaborators to arrange their 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 (*in definitions*) for visiting users from collaborating domain need to be incorporated in the host domain's resource ACLs. Upon completion of collaboration, the rules in ACLs shall lapse and pre-collaboration functionality is 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 assertions using such enriched pre-image messages are useful in realizing complex policies.
3. Apart from standard signatures, undeniable signatures with full convertibility are 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 [4].

3. CONCLUSION

Overlays allow collaborating domains to map their users and resources into an ordering suitable to peer collaborators' requirements. The approach to constitute overlays with two set of definitions: *in definitions* – a new ordering of shared resources and permissions for incoming users, and *out definitions* – a new ordering of outgoing users compatible with the ordering defined by *in definitions* of peer, gives a clear understanding of authorization flows to domain administrators. Domain administrators enforce collaboration by issuing authorization commitments, i.e., by coupling the *in definitions* with the peer's *out definitions*. Such authorization commitments become part of the authorization proofs presented with each inter domain access requests and revoking these commitments brings the collaborating domains to their pre-collaboration functionality immediately. This is an important requirement in ephemeral, dynamic collaborations.

4. REFERENCES

[1] 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.

[2] 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.

[3] H. Krawczyk and T. Rabin. Chameleon hashing and signatures. In *Proceedings of the ISOC Symposium on Network and Distributed System Security*, pages 143–154, 2000.

[4] V. Patil, A. Mei, and L. Mancini. Addressing interoperability issues in access control models. Technical report, Dept. of Comp. Sc., University of Rome - La Sapienza, 2006. http://www.dsi.uniroma1.it/˜patil/pub/TR/webminds-62.pdf.