Enforcing Spatial Constraints for Mobile RBAC Systems

Michael S. Kirkpatrick
Department of Computer Science
Purdue University
West Lafayette, IN, USA
mkirkpat@cs.purdue.edu

Elisa Bertino
Department of Computer Science
Purdue University
West Lafayette, IN, USA
bertino@cs.purdue.edu

ABSTRACT

Proposed models for spatially-aware extensions of role-based access control (RBAC) combine the administrative and security advantages of RBAC with the dynamic nature of mobile and pervasive computing systems. However, implementing systems that enforce these models poses a number of challenges. As a solution, we propose an architecture for designing such a system. The architecture is based on an enhanced RBAC model that supports location-based access control policies by incorporating spatial constraints.

Enforcing spatially-aware RBAC policies in a mobile environment requires addressing several challenges. First, one must guarantee the integrity of a user's location during an access request. We adopt a proximity-based solution using Near-Field Communication (NFC) technology. The next challenge is to verify the user's position continuously satisfies the location constraints. To capture these policy restrictions, we incorporate elements of the UCON$_{ABC}$ usage control model in our architecture.In this work, we also propose a number of protocols, describe our prototype implementation, report the performance of our prototype, and evaluate the security guarantees.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*authentication*

General Terms

Security

Keywords

Location-based access control, RBAC, continuity of usage

1. INTRODUCTION

Role-based access control (RBAC) is widely used in modern enterprise systems, as it eases the burden of administration by crafting policies based on roles, rather than identities. The increasing usage of mobile devices in enterprise settings, though, presents a new challenge as users need access to protected resources from a variety of settings. Existing solutions, such as using a virtual private network (VPN), are inadequate, as the access control decision does not consider the context of the request. For example, no distinction is made between a VPN established through a public (and potentially malicious) wireless hotspot and one initiated from a secured home network. As such, there is growing interest in incorporating contextual factors, primarily the user's location, into RBAC systems.

Location-constrained RBAC systems can offer robust fine-grained access control in a number of application scenarios. One example would be to improve the privacy of patient records in a health care system [13]. A limitation of current systems is the "bored but curious" employee; such a person may access the record of a celebrity undergoing treatment in the same hospital, despite having no valid reason to do so. Incorporating spatial constraints could restrict access to the patient's record only to workers in the ward in which he is being treated.

In a government or military setting, secure processing of confidential material might require restricting such accesses to a single room or set of rooms. Simplistic, but undesirable, solutions could be to require a different set of credentials for use in the room or to restrict access to machines permanently stored in that location. A more flexible approach would be to permit users to bring in their (employer-assigned) mobile devices and present the same credentials they use otherwise. That is, enforcement of the spatial constraints would be transparent to the user.

Previous work addressing the topic of spatially-constrained RBAC have focused on developing the policy models for such systems. However, enforcement of these models offers a number of interesting challenges that have not been considered. In this work, we focus on addressing two such challenges. First, the system must provide a secure means to authenticate the user's claim to a particular location. Second, as users are assumed to be mobile, the system must be able to enforce access control as the user's location changes.

We address the first challenge by developing a novel *proof-of-location* protocol, based on the assumption that a number of *location devices* are pre-deployed in known physical positions. A user retrieves the proof, including a timestamp, which he presents to a *resource manager*, along with other relevant credentials. The resource manager consults with a *role manager*, and grants a ticket for the resource if the

SACMAT'10, June 9–11, 2010, Pittsburgh, Pennsylvania, USA.

request is approved.[1] We have also developed a prototype implementation of this protocol, where the user retrieves the proof-of-location using a cell phone equipped with NFC technology.

Our design for this protocol is based on the GEO-RBAC [9] model. The main feature of GEO-RBAC is to associate *spatial extents* with traditional roles. Furthermore, GEO-RBAC supports hierarchical definitions of both roles and locations. For example, an organization could grant some basic permissions to <*Employee*,*Third Floor*>, and these permissions would automatically apply to <*Manager*,*Room 305*> without requiring any redundant definitions.

To address the challenge of location changes, we combine GEO-RBAC with elements of UCON$_{ABC}$. The GEO-RBAC model makes a distinction between role *enabling* (entering the spatial region) and *activation* (exercising the role's permissions). UCON$_{ABC}$ requires that certain obligations and conditions be upheld, even after the access decision is made. By combining these models, we can express the constraint that access must be revoked once the user moves outside the role's spatial extent. That is, our design enforces "continuous access control" as the user moves.

We highlight three contributions of this paper. First, we define an architectural model for enforcing spatially-aware RBAC. In presenting this model, we identify our design goals, assumptions of device capabilities, principal responsibilities, and storage requirements. Second, we present a base protocol for incorporating the user's location into an access request. We also describe extensions of this base protocol for addressing mutually exclusive roles and continuity of access. Third, we describe challenges and issues addressed during the development of a prototype spatially-aware RBAC system using NFC technology. We also provide details about our implementation, document our performance measurements, and offer an informal analysis of the security guarantees of our design.

2. RELATED WORK

Role-based access control (RBAC) [28] is commonly used to model information system protections, including hierarchical designs [29, 30]. Several extensions to the basic RBAC model have been proposed, including some that incorporate temporal logic [18] and spatial constraints [12, 3, 9, 27, 7, 4]. Additional work [8, 10] has focused on securing mobile and context-aware systems. These approaches have focused on abstract models to represent the spatial and temporal constraints, whereas our work focuses on creating an enforcement architecture and an implementation for such a system. In particular, our work expands on the GEO-RBAC model by examining the design necessary to enforce such constraints.

Our work also examines the role of usage control models within such a system. The UCON$_{ABC}$ model [33, 26] describes the various methods for checking access requests. This model describes conventional access control as *preA*, indicating the access check is performed before access is granted and is not performed again. In contrast, *onA* systems continue to enforce the access constraints while the resource is accessed. These continuous checks are important for mobile systems, where a user can move outside the permitted region after being granted access to a resource. We incorporate *onA* checks into the design of our architecture.

To bridge the gap between abstract policies and real implementations, Sandhu *et al.* have proposed the notion of PEI (policy, enforcement, implementation) models [31]. That is, they have introduced a distinction between policy goals, which are traditionally high-level and abstract, and enforcement mechanisms, which define a usable structure for creating an implementation. While GEO-RBAC describes the high-level policy, our current work is to define the enforcement mechanism.

XACML is an open standard for defining the structure of an access control enforcement mechanism [25]. The portions of XACML that are relevant to our current work includes the definition of key points or principals, such as the policy decision point (PDP), the policy enforcement point (PEP), and the policy information point (PIP). The PIP is responsible for providing relevant information to the PDP in regard to a user's access request. Once the PDP has determined whether or not the access is granted, the decision is passed to the PEP, which is responsible for carrying out the decision. For example, the PEP may be a server that generates tickets that can be used to access the data. Our work examines the principals necessary to server as the PIP, PDP, and PEP for a location-based RBAC system.

NFC is an RFID-based proximity-constrained technology that provides contactless communication between a device and a reader/writer. However, NFC has a number of advantages over traditional RFID mechanisms, such as a very restricted broadcast range that is typically 10 cm in radius. This limited range is clearly sufficient to provide evidence of the user's presence in a room or building. Additionally, NFC defines a peer-to-peer mode that can be used to read and write data in a single contactless session. While recent work has uncovered attack vectors on NFC phones [19, 21], these attacks have focused on reading data stored for passive retrieval from an NDEF tag. Our design does not store such data, so these attacks are not related to our work.

There are two works that are similar to ours. The first [15] enforced access control according to the user's context. This work, as does our own, emphasized the importance of authenticating the user's claim to a particular context. In their approach, the users were aboard a train and made a claim to a location and velocity. This claim was compared to that of a trusted party aboard the train. In contrast, our trusted location devices are immobile. Additionally, this prior work does not consider the needs of an RBAC environment.

The other similar work is the approach developed for the Grey [5] project. Grey is a smartphone-based system that is used to control access to secure rooms. In contrast to Grey, our aim is to incorporate NFC technology into an access control mechanism for information systems, not just to physical spaces. Furthermore, the Grey project does not consider or support RBAC policies.

3. ARCHITECTURE

In this section we describe our architecture. We start by discussing the goals that shaped our design, then describe the assumed capabilities of the principals involved.

3.1 Design Goals

Our design approach was to keep our architecture as general as possible to provide for a diverse selection of imple-

[1]Note that the *resource manager* and *role manager* are both servers, so the request decision is fully automated.

mentations. In order to accomplish this approach, we defined the following goals for our design.

Maximize efficiency. The creators of the Grey smartphone-based access control system state, as a principle for designing security systems, "Perceived speed and convenience are critical to user satisfaction and acceptance." Consequently, any such access control protocol should be as efficient as possible. Our design aims to achieve to this goal by minimizing the number of communication steps and cryptographic operations for successful completion.

Separation of server duties. In a spatially-aware RBAC system, there are two necessary steps to any access request. First, the requesting user[2] must be mapped to a role. Second, the role and request must be checked against the protected object's set of permissions. We model these distinct steps by designating separate principals for each.

Pseudonymize requests. When a user requests access to a resource, the server responsible for protecting the resource has no need for the user's identity information or the location. This server only needs to know what roles the user has activated. Only the server that maps the user to a role needs knowledge of the user's identity. We protect this data by encrypting it with a key known only to the user and the principal managing the role mappings.

Continuity of access. In mobile systems, a user could move outside the extents of the region for which a role has been defined. At that point, any request that was granted according to the user's original location should be revoked. We enforce this constraint by using a continuity of access model that requires users to re-confirm their locations after a certain period of time. If the user has moved outside the allowed region, he will be unable to confirm his location, and his existing permissions will be revoked.

Generalized client design. In our design, we strive to make our system model as general and applicable as possible. That is, we desire to minimize any assumptions regarding the client's performance or security capacities. For example, we do not assume the user's mobile device is capable of multiple complex cryptographic operations, nor do we assume specialized hardware security mechanisms. Such assumptions would be barriers to adoption. Consequently, we cannot place any reliance on the trustworthiness of the client for determining the correct location. If the system were based on GPS, for instance, the server could not distinguish between a device that reported the true coordinates and a corrupted device that provided false locations.

3.2 Principals

From a high-level perspective, our design is based on a ticket-granting architecture in which a user submits an access request to the resource manager that owns the desired resource. If the request is granted, the manager issues a ticket the user can submit to the resource for the duration of the session. It is important to note that the resource itself is responsible for checking the validity of the ticket. However, as ticket validation is not directly related to the enforcement of location constraints, we consider such an issue to be outside the scope of this paper. Consequently, our architecture does not explicitly model the resource as a separate principal.

Although our discussion assumes a ticket-granting architecture, our design can also be applied when tickets are not involved. That is, if the system is set up so that the user issues an access request directly to the desired resource, then the resource itself is acting as its own manager. Once the access decision is made, the resource then grants access immediately without the additional step of issuing a ticket. However, for the simplicity of discussion, we will continue to refer to a ticket-granting design.

The following four principals form the core of our architecture.

- *User* – the principal making the request. This principal generally refers to the device used for the request, although one can also interpret it as the person making the request in some instances. When a distinction is necessary, we use the uncapitalized "user" to refer specifically to the person, whereas *User* would indicate the device.

- *Location Device (LD)* – the physical device storing location information. We assume that *LD* is installed in a pre-defined location and cannot be moved. For example, *LD* may be installed inside a wall or another immovable structure. *LD* serves as one part of the PIP, as it provides contextual information relevant to a request.[3]

- *Resource Manager (RsM)* – the resource manager responsible for the requested resource. *RsM* acts as the PDP and, in combination with the resource itself, as a part of the PEP. If the policies regarding access to the resource grant permission based on *User's* currently active roles, then *RsM* approves the request and generates a ticket. As previously described, the resource itself (which is not modeled as a separate principal) also acts as a part of the PEP by confirming the validity of the ticket before granting access.

- *Role Manager (RoM)* – the role manager that maps a user to a set of roles. *RoM* is responsible for evaluating the location claim and the credentials presented. It then returns a list of active roles to *RsM*, which evaluates the request in relation to the defined policy. As such, *RoM* acts as the PIP. Although we assume *RoM* consists of a single, centralized server, we believe our architecture could be applied to a distributed server, as well. We leave such a consideration for future work.

[2]Note that there is some inherent ambiguity in relation to the term "user." We generally use the capitalized term *User* to refer to the physical device or the device software making the request, while the uncapitalized "user" typically refers to the actual person behind the request. In some cases, *User* may consist of multiple physical devices. For example, an NFC-enabled cell phone may be used for communication with the location device, while the actual access request is submitted to a resource after connecting the phone to a laptop. In such a design, *User* consists of the combination of the phone and the laptop.

[3]Our design differs slightly from the basic XACML structure in relation to the use of the PIP. Normally, the PIP is consulted by the PDP when it receives a request. However, this approach would require additional communication overhead, as *RoM* would have to contact *LD*, which could delay the access decision. Instead, *RoM* just needs to authenticate the data *User* gathered from *LD*. Consequently, our approach reduces the number of communication steps required.

In practice, there would be multiple location devices and resource managers within the system. However, our protocol is designed to focus on a single access request at a time. In that view, *User* contacts a single *LD* for proof of location, then contacts a single *RsM* to request the access. Consequently, we only mention a single *RsM* and *LD* in our protocol definitions.

3.3 Communication

Figure 1 models the communication channels that exist in our architecture. With the exception of the channel between *LD* and *User*, we make no assumptions about the underlying network medium. That is, the other connections can be either wireless or wired, and we place no restrictions on this choice. However, as our design is based on the presumption *User* is mobile, we require the communication between *LD* and *User* to guarantee proximity.

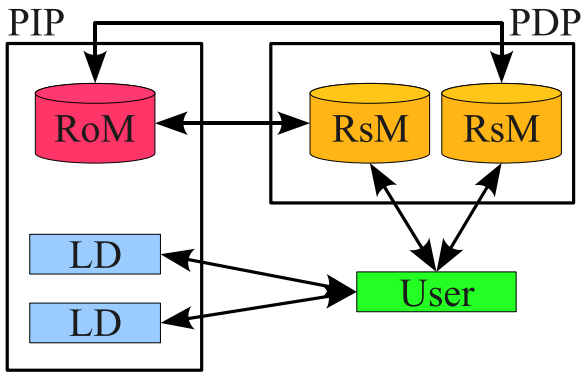

Figure 1: Communication channels within a spatially-aware RBAC architecture

It is important to note that proximity is a relative concept, and the required precision depends on the application. Our primary application requires only loose constraints on location, such as *User's* presence in a particular room or suite. Consequently, absolute precision of the location is not required. Our design choice is to use the NFC technology that is available in certain Nokia cell phones. As previously stated, NFC's limited broadcast range of 10 cm is certainly adequate for ensuring that *User* is in the desired room.[4] In fact, if each cubicle has a separate *LD*, the granularity of the request could be further increased. Another advantage of this technology is that NFC supports a peer-to-peer mode that allows *User* and *LD* to exchange information simultaneously. This capability is beneficial for the first two steps of our protocol.

3.4 Capability and Storage Requirements

As we previously described, we assume only a limited amount of computational power on the client-side principals, *User* and *LD*. Specifically, we assume that *LD* is able to perform a cryptographic hash algorithm, such as SHA-256. *User* must be able to perform symmetric key encryption, which entails the ability to store cryptographic keys securely.

On the server side, *RsM* must be able to perform symmetric key cryptography. *RoM* must be able to do the same, as well as perform the same cryptographic hash algorithm as *LD*. Additionally, *RoM* must be able to sign and verify certificates. As no other principal actually inspects the certificates used in our protocol, the signatures can be implemented using symmetric key cryptography. Doing so would improve the efficiency of the protocol. To simplify our implementation, we chose the symmetric key approach for the certificates.

In addition to these capabilities, each principal must store a limited amount of data. We summarize these storage requirements as follows.

Location device. Each location device contains a certificate $Cert_{LD}$ signed by *RoM*. The certificate contains a unique device identifier ID_{LD} and its physical coordinates, $Coords_{LD}$. The device also contains a password Pwd_{LD}.

User. Like *LD*, *User* stores a certificate $Cert_U$ that was signed by *RoM*. The certificate contains a unique identifier ID_U. *User* also has a password Pwd_U and shares a symmetric key K_U with *RoM* to encrypt its requests. In our design, K_U is stored in the secure element of the NFC cell phone, which means it can only be accessed by a trusted application. Additionally, *User* has a unique hardware identifier ID_U. The main purpose of ID_U is to bind the current request to *User*. As we will describe in Section 6, performing this binding is not a trivial feat. In the protocol, ID_U is revealed to *LD*, who binds the identifier to the current request as part of a cryptographic hash.

It is important to note that ID_U and K_U are associated with the physical device *User*. However, the certificate $Cert_U$ is associated with the person operating the device. If multiple people share the same device (for example, when nurses in a health care setting share a laptop), then the device would need a mechanism for switching certificates. Similarly, if the battery in one user's device is almost out of power, the user can use a Bluetooth connection or flash storage to transfer his $Cert_U$ to another device.

Resource manager. Each resource manager is responsible for controlling access to a set of resources by granting tickets to users. The manager stores its access control policy, *RolePerms*, that maps permissions for resources to roles. *RsM* also maintains a list, denoted by *CrntTix*, of valid tickets it has issued. Tickets are removed from this list when they become invalid, which can occur when the ticket expires, the user deactivates the role, the user activates a conflicting role, or the user moves out of the spatial extents of the region associated with the role. Recall that *RsM* does not have knowledge of *User*'s identity. Instead, *RsM* generates a unique session identifier ID_S for each access request, and associates the ID_S with the corresponding ticket granted.

Role manager. The role manager maintains the authoritative *RoleMap*, which maps users to roles and roles to geographic locations. For each *User*, *RoM* stores a list of *ActivatedRoles*. *RoM* also keeps a map, *UserResMap*, that associates *Users* with *RsMs* based on requests made.

[4]It is important to understand the difference between NFC's peer-to-peer functionality and passive RFID communication. In passive RFID, the proximity of the reader and the tag is dependent on the amount of power supplied to the tag; hence, a more powerful reader can be used to read tags from longer distances. In NFC peer-to-peer connections, both *LD* and *User* are active, and the broadcast range is set by both devices.

That is, $UserResMap$ stores pairs of the form $<RsM,ID_S>$ for each *User*. The utility of this map is evident when considering mutually exclusive roles. If *User* activates a role that conflicts with a previously activated role, *Role* must inform the appropriate *RsM* that the previous role has become de-activated. When a *RsM* determines that a session has ended, it sends a request to *RoM* to remove the relevant $<RsM,ID_S>$ entry from $UserResMap$.

RoM also maintains a number of cryptographic keys and tokens. *RoM* shares a unique symmetric key K_U with each *User*. The key is identified by the token ID_U. *RoM* also stores a key (or key pair) for signing and verifying certificates. As previously described, in our implementation, we used symmetric key encryption for the certificates, as no other principal needs to verify the certificate. Finally, *RoM* stores the set of passwords for each user and *LD*.

3.5 Setup Phase

Setting up an implementation of our protocol requires a number of steps. First, *RoM* must generate and sign the certificates for each *User* and *LD*. For *LD*, generation of the certificate and creation of the password Pwd_{LD} must occur *before* the device is installed in its physical location. Consequently, proper controls must be in place to prevent a malicious administrator from installing *LD* in a false location.

For *User*, the certificate generation occurs when the user registers with the system. The user can create an initial password when registering, and can change the password at any time. Next, ID_U and K_U must be generated and installed in the phone. As K_U must not be leaked, we install it in the phone's secure element, which restricts access to trusted, signed applications.

3.6 GEO-RBAC

Our architecture is based on the GEO-RBAC model for spatially-aware RBAC. GEO-RBAC has a number of unique features that we use in our design. In traditional RBAC, users are assigned to roles, such as *Doctor*. A permission grants the ability to perform an action on an object to any user that can activate the associated role. GEO-RBAC expands the notion of role to include a designated *spatial extents*. For example, a *spatial role* can be defined as the ordered pair <*Doctor*, *Emergency Room*>.

Next, GEO-RBAC makes the distinction between an *enabled role* and an *activated role*. A spatial role is automatically enabled once a qualified user enters the spatial extents. Role activation, however, is performed only in response to a specific request by the user. The role must first be enabled before the user can request its activation. He must also provide the requisite authorization credentials.

When *User* submits an access request, he specifies the role to activate in order to satisfy the access control policy. To process the request, *RoM* must compute the appropriate set of activated roles. The algorithm in Figure 2 describes the calculation. First, the set of currently active roles is intersected with the currently enabled roles. That is, if *User* has left the spatial extents of a previously active role, it is no longer enabled and cannot be considered active. If the requested role is in this intersection, that means it is enabled and has previously been activated. If the role is not in the intersection, then it is added to the set of activated roles. However, as GEO-RBAC supports mutually exclusive roles, the algorithm must remove any conflicting roles that have previously been activated. The algorithm then returns the set of activated roles for the user.

Input: Activated role r
Location l
Output: Set of active roles R
1. $R \leftarrow$ *current active roles*
2. $E \leftarrow \emptyset$
3. $S \leftarrow$ *spatial roles*
4. **Foreach** s **In** S
5. **If** l inside $SpatialExtents(s)$
6. $E \leftarrow E \cup \{s\}$
7. **End If**
8. **End Foreach**
9. $R \leftarrow R \cap E$
10. **If** $r \notin R$
11. $R \leftarrow R \cup \{r\} - ConflictingRoles(r)$
12. **End If**
13. **Return** R

Figure 2: Algorithm for computing *ActivatedRoles*

4. PROTOCOLS

Our architecture requires multiple protocols for granting and maintaining access. In this section, we start by describing the protocol for making an initial request, and then present the protocol for maintaining continuity of access according to the $UCON_{ABC}$ model.

4.1 Initial Request

The initial access protocol, in which the user requests access to a resource, consists of the following steps. Graphical representation of this protocol is shown in Figure 3.

1. [$User \rightarrow LD : ID_U$] The user's device sends its hardware identifier ID_U to the location device, which binds the proof of location to the requesting device.

2. [$LD \rightarrow User : Cert_{LD}, T, H^*$] Using the hardware identifier received in step 1, *LD* generates a timestamp T. *LD* then computes a cryptographic hash $H^* = H(ID_U, Pwd_{LD}, T)$. This binds *User* to the current location at the time given by the timestamp.

3. [$User \rightarrow RsM : ID_U, T, E^*$] *User* creates an encrypted package, denoted E^*, containing the requested role to activate (*Role*), the hash H^* that provides proof-of-location, the user's password Pwd_U, and the two certificates signed previously by *RoM*. That is, $E^* = E_{K_U}(Role, H^*, Pwd_U, Cert_U, Cert_{LD})$. The encryption is performed with a symmetric key that is shared between *User* and *RoM*. The hardware identifier ID_U is sent in an unencrypted form to permit *RoM* to look up the corresponding key E_{K_U} for decryption.

4. [$RsM \rightarrow RoM : ID_U, T, E^*, ID_S$] *RsM* forwards the ID_U and E^* packages received from *User*, along with the timestamp T. In addition, *RsM* generates a session identifier ID_S. This identifier allows *RoM* to create a mapping between *User* and the current request, but without revealing the actual request to *RoM*.

5. [$RoM \rightarrow RsM : ActivatedRoles$] After receiving the data from *RsM* in the previous step, *RoM* uses ID_U

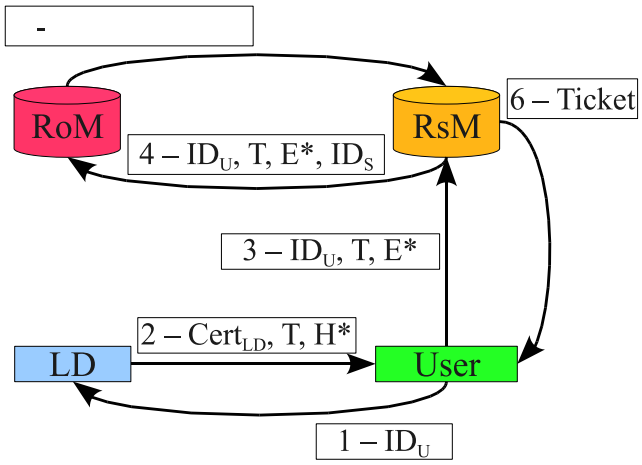

Figure 3: Data transferred at each step of our protocol

to look up the key associated with *User* and decrypts E^*. *RoM* validates the certificates, checks the user's password Pwd_U, looks up the password for *LD*, and reconstructs H^* using ID_U and the given timestamp. If the hashes and the passwords match, *RoM* computes *ActivatedRoles* by executing Algorithm 2. The resulting list is stored and set to *RsM*.

6. [$RsM \rightarrow User : Ticket$] *RsM* examines the received *ActivatedRoles* list and applies the access control policies. If the policy is satisfied, *RsM* issues a ticket that can be used to access the requested resource. While the specifics of the ticket-granting service are beyond the scope of this paper, we assume that the *Ticket* is bound to the requesting device, such as with public key cryptography.

There are a couple of key features to our initial protocol that may not be intuitive and require justification. First, consider the session identifier ID_S. While the use of this identifier may not be obvious, it is beneficial for mutually exclusive roles and continuity of usage. As such, we will describe its use in the following sections.

Second, our design assumes a certain amount of synchronization, the most important of which is time. When *LD* generates the timestamp T, this time should be approximately the same as the time on *RsM* at the same moment. As *RoM* is a central system, both *LD* and *RsM* could initiate a secure time synchronization protocol with *RsM* on occasion. Furthermore, since *RsM* specifies the access control policies, it is at liberty to provide varying degrees of timeliness for requests. For more sensitive resources, *RsM* could require a smaller time frame for the receipt of the request.

Similarly, the system needs to ensure that passwords and certificates are updated on occasion, especially if a device has been compromised. As with the issue of time, *RoM* again acts as a central authority, as it both signs and verifies the certificates when used. We consider this routine maintenance to be orthogonal to our protocol design.

Next, the need for both steps 3 and 4 requires consideration. One could argue that it would be more efficient for *User* to send the request directly to *RoM*, rather than *RsM*. One problem with such an approach is that step 3 is sent *along with the request*. That is, step 3 involves additional information that we do not explicitly model, as it is dependent on the application scenario. The two steps help to preserve pseudonymity, as *RsM* has knowledge of the request, but not the requester's identity or location; *RoM*, on the other hand, is aware of the identity, but has no knowledge of the request being made.

An additional advantage of keeping steps 3 and 4 distinct is that it maintains the locality of policies in *RsM*. One resource manager may require strict adherence to a time frame and may reject any request where the timestamp is more than a couple of seconds old. Another may allow requests if the timestamp shows the user was at the location an hour ago. Clearly the latter creates a much looser interpretation of the location constraint (i.e., the user only needs to prove that he *was* in that location, not that he still is), but our approach allows the administrators of the resource managers the flexibility to implement such a policy.

More subtly, the preservation of steps 3 and 4 is a defensive mechanism against a denial-of-service attack. Recall that we assume a single, centralized *RoM*, but several *RsMs* distributed throughout the network. If a compromised *User* sends a large number of requests to *RoM*, that server could become overloaded and the entire system could crash. If a number of the *RsMs* are enabled with mechanisms to detect the attack, they could prevent the attack on *RoM*.

A second possible criticism of our protocol is the lack of cryptography protecting the messages between principals. This choice is deliberate, as our protocol is intended as an overlay on top of an existing network infrastructure. That is, we assume that system implementers apply cryptographic techniques as necessary. Cryptography could be used in step 5 to prevent an attacker from modifying the *ActivatedRoles* in transit. However, in some applications, *RoM* and *RsM* may exist on the same physical machine, so encryption may be unnecessary. Thus, we only explicitly model the cryptographic mechanisms required to achieve our stated goals. In our implementation, *RoM* and *RsM* exist on the same machine, so encrypting steps 4 and 5 was unnecessary.

We also observe that steps 1 and 2 depend on the choice of proximity-enforcing technology. That is, some implementation choices would result in step 1 being unnecessary, while others would modify the hash value H^*. Our implementation choice is to use the peer-to-peer mode available in the Nokia NFC technology. This mode allows *User* to send ID_U to *LD* and receive the resulting data all within a single contactless connection. For other technology choices, designers may have to modify these steps as appropriate.

4.2 Mutually Exclusive Roles

Our base protocol, as described above, does not fully support mutually exclusive roles. While this may be appropriate for some applications, there are others that require such support. Assume that $ActivatedRoles_n$ denotes *User's* current set of activated roles. For a new request, *RoM* computes the new set $ActivatedRoles_{n+1}$. If $ActivatedRoles_n \subset ActivatedRoles_{n+1}$, *User* has activated a previously unused role. In some settings, broadcasting $ActivatedRoles_{n+1}$ may be required to ensure this new role does not violate the policy of some *RsM*, but this message is unnecessary in general. However, if $ActivatedRoles_n \not\subset ActivatedRoles_{n+1}$, the newly activated role has forced the disabling of a previ-

ously used role. As a result, whatever permissions *User* is exercising as a result of the previous role should be revoked. This revocation is enabled by ID_S.

Recall that *RoM* maintains the data structure $UserResMap$ for each *User*. This structure consists of pairs of the form $<RsM,ID_S>$. *RoM* can use these entries to notify the relevant *RsM* of the updated $ActivatedRoles_{n+1}$. Specifically, we introduce two optional steps to our protocol.

4a. [$RoM \rightarrow RsM^* : ActivatedRoles_{n+1}$] *RoM* sends a broadcast to the list RsM^* consisting of the resource managers with which *User* currently has an active session (according to $UserResMap$). This message lets these resource managers revoke any outstanding tickets, if necessary, according to the policies maintained locally by the *RsMs*.

4b. [$RsM^* \rightarrow RoM : Ack$] This optional step can be used to enforce strict mutual exclusion by requiring that the new request can only be approved after the *RsMs* have had a chance to revoke the necessary tickets. If strict requirements are not needed, this step can be omitted.

4.3 Continuity of Access

In terms of the UCON_{ABC} model, our basic protocol defines a *preA* approach to access control. This means that the permissions and credentials are checked only before the access is granted. In systems where users are mobile, one could move out of the extents of the spatial role after being granted access to a resource. As a result, we would like to enforce *onA* constraints, as well. These constraints require *User* to confirm his position while accessing the resource.

User can be required to confirm his location in either a proactive or reactive manner. As *RsM* serves as the PDP, it would control which method is applied. In the proactive approach, *User* would initiate the confirmation protocol; to reduce the reliance on a person's memory, the client software would display a reminder. In the reactive approach, *RsM* would connect with *User* and request confirmation. In either approach, the burden on the user should be minimized.

Once confirmation is required, the previous protocol is modified as follows.

1. [$User \rightarrow LD : ID_U$] This step works as before.
2. [$LD \rightarrow User : Cert_{LD}$, T, H^*] This step works as before.
3. [$User \rightarrow RsM : T$, E^*] The hardware identifier ID_U is unnecessary, as *RsM* has stored this value for the session. In most cases, the intent is simply to re-establish *User's* location, the role and *User's* credentials are not needed. That is, $E^* = E_{K_U}(H^*, Cert_{LD})$. However, in a more secure setting, the user may be required to re-enter his password, and E^* would be constructed as above.
4. [$RsM \rightarrow RoM : ID_U$, T, E^*] As *RsM* has stored ID_U, it adds the identifier to the message to *RoM*. Doing so prevents a collusion attack in which a different *User* device is used for a false confirmation. The ID_S is not needed at this point, so it is omitted.
5. [$RoM \rightarrow RsM : ActivatedRoles$] *RoM* computes the new set of *ActivatedRoles* and sends the updated list to *RsM*. As there is no new request to activate a role, the computation is simply checking whether the existing roles are still enabled according to the new location.

If the updated *ActivatedRoles* list continues to satisfy *RsM's* policy, the access is allowed to continue. Depending on the ticket-granting implementation, *RsM* may issue a new ticket, it may contact the resource directly and inform the resource of an updated expiration date for the ticket, or no action may be necessary. Similarly, if *User* is unable to confirm his location as required by the *RsM's* policy, *ActivatedRoles* would be null, and *RsM* could begin the process of ticket revocation. In either case, the action mechanics are external to the issue of location constraints. As such, we consider this topic to be beyond the scope of this paper and leave the question to implementation designers.

In considering continuity of access, step 5 of the protocol becomes much more complicated. In Section 4.2, we introduced the notion of broadcasting the updated *ActivatedRoles* list to other *RsMs* before responding to the current request. If *User* has successfully confirmed his location and all activated roles are still enabled, then no action is necessary. However, if the confirmation fails, or the *ActivatedRoles* has changed, a number of possibilities arise.

If the confirmation fails, a conservative approach would be to inform all *RsMs* so that they can provide an appropriate response. In general, though, we believe such an approach is undesirable. Consider the case where the user makes a mistake re-typing his password. Revoking all of this accesses would not be appropriate. Instead, a better response would be simply to let the current *RsM* handle the failure. Other *RsMs* would continue to enforce the continuity of usage according to their own policies, and there would be no overhead penalty of false-alarm messages.

Another possibility is that the confirmation succeeds, but a subset of *ActivatedRoles* have become disabled. That is, the user has confirmed both his location and his identity, so there is no chance of a false-alarm as above. In this case, *RoM* should broadcast the new list to the *RsMs* in the entries in $UserResMap$. The *RsMs* would then have the ability to adapt to the change in environment. Unlike the case of mutually exclusive roles, though, we do not see an advantage in delaying the confirmation step. Thus, after broadcasting the updated *ActivatedRoles*, *RoM* immediately informs the current *RsM* of the confirmed roles.

Note that in this continuity of access model, there is a delay between when *User* leaves the spatial role's extents and when the role is deactivated. In many cases, this would be acceptable. In high-security settings, this delay could be eliminated by having explicit entrances and exits for the spatial role. Then, the door can be treated as a resource and the role required for access is mutually exclusive with any role enabled by the location device within the spatial extents.

5. IMPLEMENTATION AND EVALUATION

We have developed a prototype implementation of our architecture using Java. To implement *RoM* and *RsM*, we have adapted the source code of the GISToolkit [1]. This library implements the OpenGIS Geography Markup Language (GML) encoding standard [24]. We have extended the library to define spatial regions appropriate for modeling a building. Specifically, our extension models floors,

rooms, and suites. We have also incorporated XML files to represent the role definitions, spatial role extents, and permissions.

On the client side, we have split the behavior of *User* into two components. The first component consists of a Nokia 6131 NFC-enabled cell phone [23, 22]. We use this device to connect with an Advanced Card Systems (ACS) NFC reader, model ACR 122 [2], which serves as *LD*. The second component of *User* is a Java client application written for a more traditional computing device, such as a laptop. The data generated by *LD* in step 2 of our request protocol is transferred from the phone to the laptop manually.

Implementation of *User* in these two components is problematic. First, it is inconvenient. More importantly, it breaks the guarantee that the user of the laptop is in the claimed location; colluding users could simply communicate the data from *LD* via some side communication channel. Both of these criticisms can be addressed by our vision of integrating NFC technology into a custom computing device.[5] Access to ID_U could then be controlled by trusted hardware, such as a Trusted Platform Module (TPM), ensuring that the request is bound to a known and trusted device.

Communication between *User*, *RsM*, and *RoM* is accomplished using traditional sockets. Implementing the communication between *User* and *LD* was more challenging. On the Nokia 6131 NFC phone, we deployed a MIDlet that transmitted data using the Java JSR 257 Contactless Communication API [16]. The ACR122 reader was connected via USB to a Windows XP workstation.

Implementing the behavior of *LD* required the development of a custom Java application to communicate with the reader. The ACR122 uses the standard Windows CCID interface. While the SDK provides documentation and sample code for traditional smart cards (e.g., only reading or only writing NDEF-formatted tags), this approach is undesirable for our design. That is, this basic approach would require connecting the phone to the reader twice; the first touch would send data from the phone to reader, and the second touch would be the response. Furthermore, the user would have to intervene manually to change the modes between the touches.

A better approach is to use the peer-to-peer extension of JSR 257, which is supported by both the Nokia 6131 NFC and the ACR122. Although the SDK provided by ACS does not contain examples of this functionality, we were able to adapt code from the nfcip-java library [17]. Combining the APDUs (instructions for communicating with the ACR122) with the basic structure of the ACS SDK, we were able to implement the peer-to-peer communication.[6] The phone is designated as the initiator of the request (as it sends data first), while the reader is then designated as the target.

Two machines were involved in our performance evaluation. The first test machine was running Windows XP on an Intel Pentium M CPU running at 1.60 GHz with 1.3 GB of memory clocked at 333 MHz. We attached the ACR122 reader to this machine via a USB connection to measure the amount of delay experienced by the user as a result of the NFC communication. Initiating the peer-to-peer connection 50 times, we observed an average of 131.4 ms delay from time the request is sent from the phone until a response is received.

Our second machine was used to test the overhead of the back-end server portion of our design. This machine was running Ubuntu Linux, version 9.04 (Jaunty Jackalope) on an Intel Core 2 Duo CPU running at 2.26 GHz with 3 GB of memory clocked at 667 MHz. To eliminate network delay from our measurement, our implementations of *User*, *RsM*, and *RoM* were all executing on this machine, using sockets to communicate. We measured the delay from the time that *User* submitted the request to *RsM* until it received a response. On average, our protocol yielded an overhead of 24.4 ms. Clearly, the overhead imposed by the local computations of our architecture is minimal, and most of the delay users observe would result from normal network communication. As such, we argue that implementing our design for a real spatially-aware RBAC is certainly feasible.

6. SECURITY ANALYSIS

In this section, we present an informal security analysis that primarily focuses on two threat models. As our primary concern is to secure access to the protected resource, we mainly examine the threat from a malicious *User*. However, we also consider the threat posed by *RsM*, as one of our design goals is to ensure the pseudonymity of *User*. The aim of our security analysis was to address common attacks on authentication protocols [20, 14, 32, 11, 6]. These attacks include *replay*, *collusion*,*reflection*, *denial-of-service*, and *typing*. We do not consider *eavesdropping* and *modification*, as our system is built on the assumption that appropriate cryptographic mechanisms are used to protect messages at the lower layers of the network stack.

Assumptions. Given these threat vectors, we state a number of assumptions. First, we assume the integrity of *RoM*. As no other principal has knowledge of the mapping between users and spatial roles, we do not see a defense against a corrupted *RoM*. That is, if a corrupted *RoM* reports false *ActivatedRoles*, the attack may be detected, but there is no mechanism for correction. Consequently, we assume that the *ActivatedRoles* reported in step 5 is correct.

Additionally, we assume the coordinates stored in *LD's* certificate are correct. Note that our assumption does not preclude the chance that a malicious *User* can attempt to use a false certificate. Next, we assume that *RsM* is acting in good faith to protect the resource. If not, *RsM* could simply issue tickets without regard for the protocol.

Malicious *User*. In this attack, the primary goal of the attacker is to gain illicit access to the protected resource. A secondary goal would be a denial of service for others. Our threat model assumes the attacker is computationally bound, *i.e.*, he cannot break the cryptographic hash or encryption primities employed. Under this model, our protocol is secure against the following attacks as explained below.

1. **Replay** The goal of a replay attack is for *User* or an eavesdropper to reuse a piece of data as part of a false request. Clearly, the $Cert_{LD}$ and timestamp T are

[5]While one may object to this solution, we argue that an organization requiring a spatially aware RBAC system is likely to have the resources available to design such a device, or contract to a company that will do so.

[6]One may question why we did not use the nfcip-java library as written. We were unable to get this library working with our model of the reader. After contacting the library's author, we learned that the firmware of our model may be incompatible with the library. We are grateful to the author for his willingness to help with this problem.

tied to the given request (and the *User* through the hash $H(ID_U, Pwd_{LD}, T)$. If the timestamp is modified or a different $Cert_{LD}$ is sent, then the hash would not match. Similarly, if the hardware identifier ID_U is changed in transit, again, the hash would not match. Furthermore, ID_U is tied to the given *User*, as the key is only shared between that *User* and *RoM*.

Another advantage of the timestamp T is that it ensures the timely use of the location information. That is, *User* cannot hoard the data received from *LD* for use after moving out of the spatial extents. First, the data may be marked as invalid if T is beyond an acceptable time frame. Additionally, T can be used to enforce an ordering of the requests. That is, if $T_1 < T_2$, but T_2 arrives at *RoM* first, the request with T_1 would be denied as an expired request.

2. **Collusion** There two possibilities for collusion between two users. The first attack is for $User_1$ to obtain the proof-of-location from *LD*, and send the proof to $User_2$ via a side channel. This attack is essentially identical to *ghost-and-leech* attacks on RFID readers. As described in Section 5, our vision for deployment with a custom device would obviate this attack vector, as only known and trusted devices can submit requests to *RsM*. A second possibility for collusion would be for $User_1$ to send a valid ticket to $User_2$. Although we generally consider the implementation of the ticket-granting service to be beyond the scope of this paper, as noted previously, we assume that remote attestation with a TPM or public key cryptography could be used to bind the $Ticket$ to $User_1$.

3. **Reflection** In a reflection attack, the attacker would engage in a protocol with a target to get data that could be reused as part of a request. In our design, the target would have to be another *User*, as no other principal reveals credentials that the attacker could attempt to reuse. However, in our protocol, *User* only initiates the protocol. That is, a malicious $User'$ cannot initiate the protocol with a targeted *User*. Thus, a reflection attack is not applicable in our architecture.

4. **Typing** For a successful typing attack to occur, there must be more than one piece of data of the same type. That is, the attacker tries to get the victim to accept one piece of data as another based on the two pieces having the same format. In our protocol, there are no instances of two pieces of data having the same type. As a result, a typing attack is not possible.

5. **Denial-of-service** Our protocol is designed so that all *Users* must submit their requests through a distributed number of *RsMs*. If *User* attempts a denial-of-service attack against a particular *RsM*, he may succeed depending on how robust the *RsM* is against such an attack. However, this attack vector is not the result of our design, but is an inherent danger of a networked system. *RsM* could mitigate the damage from these attacks by employing appropriate measures, such as blacklisting suspected nodes.

   RoM, as a centralized server, can also be a target for a denial-of-service. However, our assumption is that the *RsMs* are behaving properly. As a result, *RoM* could send a request to the *RsMs* to throttle their requests as appropriate. Furthermore, if *RoM* is equipped with software to analyze recent logged requests, it could identify a potentially misbehaving *User* by identifying any ID_U identifiers that are associated with an abnormal number of requests. Thus, while both of these types of denial-of-service are possible, it is our observation that these attacks are not contingent on the design of our architecture, and a number of mitigation techniques are possible.

Malicious *RsM*. The goal of a malicious (or corrupted) *RsM* would be to bind the user's identity to the requests. However, the encryption of the $Cert_U$ ensures that *RsM* is restricted to only the pseudonymized identifier ID_U. That is, if the attacker discovers which user had possession of a particular *User* device at a given time, he could discover the identity by pairing the user with the associated ID_U. Although this is a breach of strict confidentiality, our design goal was to provide pseudonymity, rather than pure anonymity. That is, without the ability to pair a user with the *User* device, the attacker cannot discover the identity through the request alone.

7. CONCLUSIONS

In this work, we have proposed a novel architecture for enforcing spatial constraints in an RBAC environment. We identified a number of goals that such an architecture should meet, and constructed protocols that accomplish these goals. We have demonstreated that our architecture is flexible and can be applied to a number of settings with varied security requirements by localizing the policies in the individual *RsMs*. Our design incorporates concepts from the $UCON_{ABC}$ usage control model to enforce continuous access checks while the user accesses the protected resource.

We have implemented a prototype of our architecture that provides a proof of concept. Our prototype uses a Nokia 6131 NFC cell phone to communicate with a ACR122 reader connected to a workstation. We have addressed the challenges we encountered in adopting this technology, and described the performance we observed in our experimental evaluation. We have also provided an informal analysis of the security guarantees of our design.

Based on the results in this paper, we find a number of promising directions for future research in spatially-aware RBAC systems. First, one must be able to determine when spatial boundaries are crossed. Without relying on the trustworthiness of the reporting device, determining when this happens presents a challenge. Additionally, one could consider how to handle a user that leaves the area and returns. One approach would be to require the user to re-activate all of the previous roles. Alternatively, activation could occur automatically when the user crosses the boundary. Creating such an event-based model for continuity of usage presents a possible area of future research. Addressing these and other research challenges will offer organizations the ability to provide robust fine-grained access to protect resources on the basis of access locations.

8. ACKNOWLEDGEMENTS

The work reported in this paper has been partially supported by the NSF grant 0712846 "IPS: Security Services for

Healthcare Applications," and the MURI award FA9550-08-1-0265 from the Air Force Office of Scientific Research.

9. REFERENCES

[1] GISToolkit. http://gistoolkit.sourceforge.net/.

[2] Advanced Card Systems Limited. ACR122 NFC contactless smart card reader software development kit. http://www.acs.com.hk/acr122-sdk.php.

[3] S. Aich, S. Sural, and A. K. Majumdar. STARBAC: Spatiotemporal role based access control. In *OTM Conferences*, 2007.

[4] V. Atluri and S. A. Chun. A geotemporal role-based authorisation system. In *International Journal of Information and Computer Security*, volume 1, pages 143–168, 2007.

[5] L. Bauer, L. F. Cranor, M. K. Reiter, and K. Vaniea. Lessons learned from the deployment of a smartphone-based access-control system. In *Proceedings of the 3rd Symposium on Usable Privacy and Security (SOUPS)*, 2007.

[6] C. Boyd and A. Mathuria. *Protocols for Authentication and Key Establishment*. Springer, 2003.

[7] S. Chandran and J. Joshi. LoT RBAC: A location and time-based RBAC model. In *Proceedings of the 6th Internation Conference on Web Information Systems Engineering (WISE '05)*, pages 361–375. Springer-Verlag, 2005.

[8] M. J. Covington, W. Long, S. Srinivasan, A. K. Dev, M. Ahamad, and G. D. Abowd. Securing context-aware applications using environment roles. In *Proceedings of the 6th ACM Symposium on Access Control Models and Technologies (SACMAT '01)*, pages 10–20, 2001.

[9] M. L. Daimani, E. Bertino, B. Catania, and P. Perlasca. GEO-RBAC: A spatially aware RBAC. In *ACM Transactions on Information and System Security (TISSEC)*, 2007.

[10] M. L. Damiani and E. Bertino. Access control and privacy in location-aware services for mobile organizations. In *7th Internation Conference on Mobile Data Management*, 2006.

[11] S. Gritzalis and D. Spinellis. Cryptographic protocols over open distributed systems: A taxonomy of flaws and related protocol analysis tools. In *16th International Conference on Computer Safety, Reliability and Security (SAFECOMP '97)*, pages 123–137, September 1997.

[12] F. Hansen and V. Oleschuk. SRBAC: A spatial role-based access control model for mobile systems. In *Proceedings of the 8th Nordic Workshop on Secure IT Systems (NORDSEC '03)*, pages 129–141, October 2003.

[13] F. Hansen and V. Oleshchuk. Application of role-based access control in wireless healthcare information systems. In *Scandinavian Conference in Health Informatics*, pages 30–33, 2003.

[14] J. Heather, G. Lowe, and S. Schneider. How to prevent type flaw attacks on security protocols. In *Proceedings of the 13th IEEE Workshop on Computer Security Foundations (CSFW '00)*, 2000.

[15] R. J. Hulsebosch, A. H. Salden, M. S. Bargh, P. W. G. Ebben, and J. Reitsma. Context sensitive access control. In *Proceedings of the Tenth ACM Symposium on Access Control Models and Technologies (SACMAT '05)*, 2005.

[16] JSR 257 Expert Group. JSR 257: Contactless communication API.

[17] F. Kooman. The nfcip-java project. http://code.google.com/p/nfcip-java/.

[18] L. Lamport. The temporal logic of actions. In *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 1994.

[19] G. Madlmayr, J. Langer, C. Kantner, and J. Scharinger. NFC devices: Security and privacy. In *The 3rd International Conference on Availability, Reliability and Security (ARES)*, pages 642–647. IEEE Computer Society, 2008.

[20] S. Malladi, J. Alves-foss, and R. B. Heckendorn. On preventing replay attacks on security protocols. In *In Proceedings of the International Conference on Security and Management*, pages 77–83. CSREA Press, 2002.

[21] C. Mulliner. Attacking NFC mobile phones. In *EUSecWest 2008*, May 2008.

[22] NFC Forum. NFC forum tag type technical specifications. http://www.nfc-forum.org/.

[23] Nokia. Nokia 6131 NFC SDK programmer's guide.

[24] Open GIS Consortium. Open GIS geography markup language (GML) implementation specification. http://www.opengeospatial.org/standards/gml, 2003.

[25] Organization for the Advancement of Structured Information Standards (OASIS). eXtensible Access Control Markup Language (XACML). http://www.oasis-open.org/committees/tc_-home.php?wg_abbrev=xacml/.

[26] J. Park and R. Sandhu. The UCON$_{ABC}$ usage control model. In *ACM Transactions on Information and System Security*, volume 7, pages 128–174, 2004.

[27] I. Ray, M. Kumar, and L. Yu. LRBAC: A location-aware role-based access control model. In *Proceedings of Internation Conference on Information Systems Security (ICISS)*, pages 147–161, 2006.

[28] R. Sandhu. Role-based access control models. In *IEEE Computer*, Feb. 1996.

[29] R. Sandhu. Role hierarchies and constraints for lattice-based access controls. In *Fourth European Symposium on Research in Computer Security (ESORICS)*, 1996.

[30] R. Sandhu. Role activation hierarchies. In *3rd ACM Workshop on Role-Based Access*, 1998.

[31] R. Sandhu, K. Ranganathan, and X. Zhang. Secure information sharing enabled by trusted computing and pei models. In *ASIACCS '06: Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security*, pages 2–12, New York, NY, USA, 2006. ACM.

[32] P. Syverson. A taxonomy of replay attacks. In *Proceedings of the 7th IEEE Computer Security Foundations Workshop*, 1994.

[33] X. Zhang, J. Park, F. Parisi-Presicce, and R. Sandhu. A logical specification for usage control. In *Proceedings of the Ninth ACM Symposium on Access Control Models and Technologies (SACMAT '04)*, 2004.