# Supporting Dynamically Changing Authorizations in Pervasive Communication Systems

Adam J. Lee, Jodie P. Boyer, Chris Drexelius, Prasad Naldurg, Raquel L. Hill, and Roy H. Campbell

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
{adamlee, jpboyer, drexeliu, naldurg, rlhill,rhc}@cs.uiuc.edu

**Abstract.** In pervasive computing environments, changes in context may trigger changes in an individual's access permissions. We contend that existing access control frameworks do not provide the fine-grained revocation needed to enforce these changing authorizations. In this paper, we present an authorization framework, in the context of the Gaia OS for active spaces, which integrates context with authorization and provides fine-grained control over the enforcement of dynamically changing permissions using cryptographic mechanisms. Our design, implemented in middleware, addresses the limitations of traditional authorization frameworks and the specific access control needs of pervasive computing environments. As part of our proposed framework, we define cryptographic protocols that enforce access to the system's communication channels and provide secure delivery of messages. We also provide a proof of correctness of key agreement and freshness using the standard BAN deduction system.

## 1 Introduction

In pervasive computing environments, a change in context may affect the access permissions of users to different system resources. These context changes may occur often, and the management of access control policies for these environments is inherently more complex. The problem of enforcing access control policies for traditional information systems is well studied. We contend that existing authentication and authorization frameworks for these systems such as Kerberos [1, 2] and public-key certification hierarchies [3] are not adequate for pervasive computing scenarios. To illustrate, in Kerberos, the tickets that represent a user's right to access a service are issued with fixed expiry times, which cannot be determined in advance as a function of changing context. The problem of revocation in hierarchical PKI systems is well known [4]. While the authors in [5] explore the authentication problem for pervasive computing, our focus in this paper is on the adequate enforcement of dynamically changing authorizations, driven by changing context.

We explore these authorization issues in the context of our prototype smart room, consisting of a variety of computing and communication services that can be configured for different activities depending on the context. This infrastructure is orchestrated by Gaia OS [6], which brings the functionality of an operating system to physical spaces. In addition to common operating system functions, such as events, signals, file system, security, processes, process groups, etc., Gaia extends typical operating system concepts to include context, location awareness, mobile computing devices and actuators like doorlocks and light switches.

To illustrate the problem in this setting, consider a typical scenario where users, applications, or sensors may initiate events that trigger a change in context. For example, a smoke detector may generate an emergency alarm event triggered by an actual fire which disables electronic doorlocks and allows occupants of the environment to safely exit the building. This emergency alarm event can be implemented in Gaia as a control message to a software-controlled doorlock service, which in turn sends an unlock control message to the individual doorlocks.

Such event notification messages disseminate context information and have the potential to change the operating characteristics of their environment. Authorization to send these notifications should be tightly controlled. Without such protection, the system is vulnerable to attack. For example, an attacker can trigger an emergency alarm by forging the appropriate event, and open the doors to gain physical access to the environment. An attacker could also repeatedly send lock messages to the doorlock service to keep the doors in a locked state, thereby denying access to authorized users. These two examples illustrate how events that occur in virtual space may affect the physical security of that space.

Data exchanged between users and devices in the course of a pervasive computing scenario is also subject to dynamically changing authorizations depending on the context. Consider a collaborative meeting example where all users should be allowed to send messages using the event channel to a shared electronic bulletin board via their personal handheld devices. When the room is configured as a lecture hall, only a presenter should be authorized to send messages to this bulletin board. Therefore, a user's authorizations to the board should change in response to a change in context. Due to the asynchronous and distributed nature of event generation and notification, as well as data dissemination, this service in Gaia OS is implemented as publish-subscribe channels.

Given the setting, we contend that existing frameworks [1, 3, 7–9] do not provide the fine-grained revocation necessary to enforce changing authorizations. In this paper we present an access control architecture that integrates context with authorization and provides fine-grained control over the enforcement of dynamically changing permissions. Our proposed solution extends the functionality of the Gaia OS and is implemented using distributed objects. The components of our new framework can be replicated easily for load-balancing, are designed to keep minimal state, and work independently without coordination. We believe that this aspect of our design makes our system more efficient and resilient to denial of service (DoS) attacks.

The rest of the paper is organized as follows. Section 2 describes the threat model for attacks on the communication systems of pervasive computing environments. Section 3 presents the limitations of existing authorization frameworks and motivates the need for a new access control architecture, which we present in Section 4. In Section 5, we present the authorization protocols used in our framework. We present an evaluation of our architecture and protocols in Section 6 and conclude in Section 7.

## 2 Threat Model

In this section, we characterize the nature and scope of attacks that can be mounted against the message distribution system of our prototype pervasive computing environment. We assume that attackers can be either *passive* or *active*. Regardless of their physical location, passive attackers can listen to messages anywhere on the network but cannot change the contents of these messages. Active attackers can modify, reorder, replay, or inject messages into the network. Neither active nor passive attackers can perform cryptographic operations with keys that they do not possess. The remainder of this section discusses various attacks and the impact they have on the system.

**Integrity Violations** It is important that an event published by a sender is the same event that arrives at the receiver. If no such guarantees can be made, an active attacker can change the content of messages while they are in transit. Consider an active attacker who changes the sign bit on messages sent out by a temperature sensor. This attack may cause the climate control equipment in the room to activate the heaters, which may adversely impact system hardware.

**Confidentiality Violations** In many cases, the contents of both data and control messages may need to be confidential. Consider an electronic doorlock system that sends events to a central monitoring facility when any door is locked or unlocked. If these messages are sent unencrypted, unauthorized viewers can quickly learn what parts of the building are unlocked. This could lead to intrusions and theft.

**Privacy Violations** While it is difficult to make strong guarantees about privacy such as unobservability, or unlinkability [10], it is important that a message distribution system provides for basic user privacy by preventing disclosure of sensitive personal information. Content publishers should be able to define which entities have access to generated information. For instance, if a user is wearing a location-tracking badge that periodically sends her GPS location, she may specify a list of authorized recipients of this data to preserve her location privacy.

**Authenticity** Message authenticity is necessary in an event distribution system, since the configuration and security of the pervasive computing environment can change as a result of the messages sent. Without such guarantees, the system is open to attacks in which a malicious user injects spurious messages, claiming that they came from legitimate sources. One example of such

an attack is when an intruder forges a message from a smoke detector that will trigger an alarm system. The alarm system may unlock the doors to the building to allow easier exit, or in this case, entrance to an unauthorized user.

**Denial of Service** In addition to traditional denial of service (DoS) attacks, the context-sensitive nature of pervasive computing environments exposes these systems to additional avenues of DoS. For instance, a clever attacker might realize that access permissions in the system change with context. The attacker can repeatedly trigger context changes to force management overheads associated with permission revocation and acquisition on the system. If these overheads are too high, such an attack could impact the distribution of events.

In the rest of this paper, we describe the design and implementation of our secure event distribution system and evaluate the architecture and protocols that we present to show that our system is robust enough to resist the threats outlined in this section. In the next section, we examine the shortcomings of existing security solutions, explore the limitations that they impose when deployed in pervasive computing environments, and motivate the need for our new authorization framework.

## 3 Related Work

In this section, we present the limitations of existing authorization and authentication frameworks with respect to enforcing dynamically changing permissions in pervasive environments. These frameworks include Kerberos [1, 2], SESAME [7], hierarchical PKI [3], SDSI/SPKI [9], and KeyNote [8]. We also distinguish how this work differs from other efforts to incorporate contextual changes into access control frameworks for pervasive environments.

**Kerberos and SESAME** Kerberos is a distributed authentication and authorization framework that enforces access control through the use of time-limited tickets. SESAME provides functionality beyond the scope of Kerberos such as scalability when multiple networks are connected. However, the basic SESAME Security Architecture is essentially an extension of Kerberos. Kerberos service tickets and SESAME certificates are valid for a fixed time interval starting at a predetermined time. Explicit revocation before ticket expiration is not possible. To combat this problem, Kerberos and SESAME administrators may decrease the time interval between ticket activation and expiration. This strategy now acts as revocation mechanism for all tickets, regardless of whether or not permissions have actually changed. This negatively impacts the usefulness of tickets and increases the burden on clients wishing to access resources in both Kerberos and SESAME. This derived revocation mechanism does not solve the revocation problem since there is still no direct link between ticket expiration and context-driven permission changes. Role-based extensions to Kerberos and SESAME including [11] still

suffer from the same fundamental revocation limitation as the basic Kerberos and SESAME architectures.

**Hiearachical PKI** Unlike Kerberos and SESAME, hiearachical PKI systems do have the ability to revoke access rights. However, critical analysis in [4] suggests that distribution of key revocation information may be prohibitively expensive on a large scale. Each time permissions change, a message reflecting the new permission state must be distributed to each host in the system. The dynamic nature of pervasive environments makes this framework insufficient for the revocation needs of these environments.

**SDSI/SPKI** SDSI/SPKI suffers from the same revocation issues outlined above for hierarchical PKI. Though SDSI/SPKI does not support certificate revocation lists (CRLs), a service may exist that provides a list of principals whose keys have been revoked. To maintain system-wide consistency, service providers must continually poll the list of revoked certificates. SDSI/SPKI is not appropriate for ubiquitous environments since certificates may be revoked often.

**KeyNote** Similar to the limitations of the other authorization and authentication frameworks addressed above, KeyNote does not provide sufficient credential revocation mechanisms to handle changing context information. Like Kerberos and SESAME, KeyNote allows credentials to exist for a limited window of time, but explicit revocation is not possible. The KeyNote interpreter, a component used to verify credentials, may be informed of authorization revocation. Similar to SDSI/SPKI above, this places the burden on clients to perform certificate verification. Since we wish to eliminate this client-side burden, the KeyNote framework is insufficient for our needs.

In [12] the authors propose a security middleware for Gaia, that consists of a network of Middleboxes, reconfigurable computing and communications nodes that act as proxies or access points for critical services and mediate access to services by authenticating requesters and negotiating security requirements. Our system implements the generic proxy behavior of Middleboxes and mediates communication between publishers and subscribers by brokering information published by publishers to authorized, registered subscribers. In [13] the authors argue that traditional authentication frameworks that attempt to prove the validity of a claimed identity are not sufficient for pervasive environments, and that contextual attributes like location and manufacturer's certificates should be authenticated to establish higher levels of assurance. This work focuses on establishing trust as opposed to handling dynamically changing permissions. In [14], the authors propose extensions to RBAC that facilitate context-aware access control policies. In their system, users can assume multiple roles, and changes in user permissions are communicated to applications for which the associated roles are active. The authors do not provide a mechanism by which applications can enforce dynamically changing permissions.

In the next section, we introduce our authorization architecture, which is designed to overcome the revocation granularity limitations of existing autho-

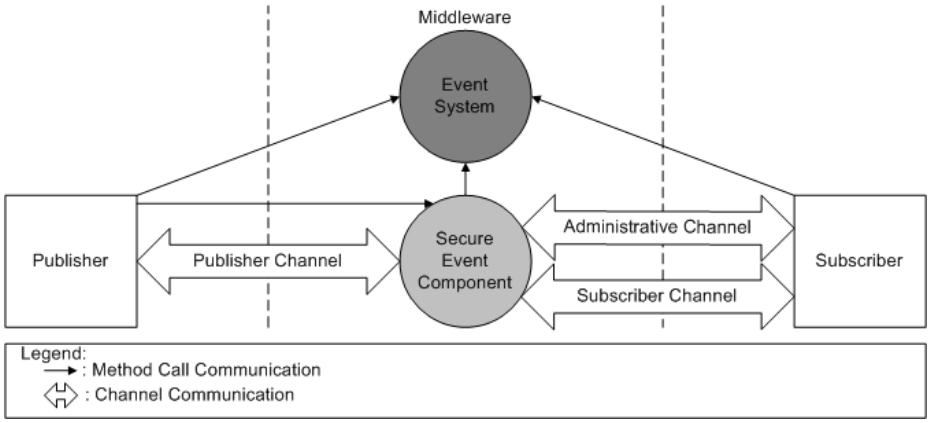


**Fig. 1.** System Architecture

rization and authentication frameworks. We also provide implementation and analysis details to validate our proposed concepts.

## 4 Architecture

Gaia OS [6] is a middleware meta-operating system that exposes APIs which allow users to locate and interact with applications, manage the context of the system, create and utilize communication channels, and otherwise interface with the pervasive computing environment in a meaningful way. Gaia OS utilizes publish-subscribe "event channels" to control the system and disseminate information to interested parties. The specific solution that we present is designed to secure these channels.

### 4.1 System Architecture

We rely on cryptographic mechanims to enforce dynamic authorizations. If an individual has sufficient rights to an information resource, the system can encrypt the information and provide the user with a secret decryption key. This ensures that only authorized individuals have direct access to sensitive data, and anyone snooping on the communication channels is denied access. Dynamic authorization is achieved by controlling the distribution of keys, and key freshness and agreement are essential for proper enforcement of access control policies. Key revocation, an integral part of key management, is used in our framework to deny access when permissions change.

We present a key distribution and management protocol to achieve dynamic authorization in pervasive computing environments. Our proposed solution accounts for denial of service, failures of commodity hardware and the dynamic nature of pervasive systems. Our framework is distributed and provides multiple key management entities. Each management entity maintains minimal state

to facilitate bootstrapping, data migration and replication as well as allows the system to handle attacks on individual components. Figure 1 presents the architecture of the secure publish-subscribe system that we have developed for Gaia OS. We now describe the components of this system.

**Publisher** Publishers are the information providers in our system. They create channels, determine the access control polices for these channels, and publish data. Example publishers include sensors, messaging clients, and other system components.

**Subscriber** Subscribers are the event consumers in our system. Any subscriber can register to consume events that they are authorized to view.

**Event System** The Event System is the underlying messaging system provided by Gaia. This component provides functionality for creating, destroying, and managing event channels.

**Secure Event Component (SEC)** In our system, there are one or more SECs. These components act as distributed reference monitors for the event channels in the system. Each SEC manages access control for one or more event channels. When a channel is created, its creator pushes access control lists (ACLs) controlling read and write access to the new channel. The SEC is responsible for enforcing these ACLs. We describe the SEC in further detail next.

### 4.2 Secure Event Component

The Secure Event Component (SEC) stores encryption keys, checks access rights, and brokers messages securely from publishers to authorized subscribers. A particular configuration of an active space may have multiple SECs that generate and manage symmetric keys used to secure specific event channels. The symmetric keys are stored in a hash table to allow efficient storage and lookup. These symmetric keys must be stored in a secure manner, e.g., using tamper-resistant hardware. Details of the cryptographic protocols between the SEC and the publishers and subscribers are presented in Section 5.

Users in our system can authenticate and obtain their credentials using a public/private key pair issued by the Gaia Certificate Authority. Once the user's identity is established, an attribute certifier, which is not discussed in detail in this paper, can issue them credentials bound to the current context by a counter which attest to various attributes, including their roles in the system. Each device or service maintains an access list, which associates a set of permissions with each role.

When the context of the space changes, the underlying Gaia access control architecture [15] changes the user-role assignment of affected users, according to a meta policy specified by the administrator of the space. The privileges assigned to roles in a space are relatively static, but principals can move in and out of roles in a dynamic fashion, triggered by changes to the current configuration of the space. A Gaia administrator is responsible for setting up these policies. In [15] we show how our access control framework, based on RBAC, can be used

to specify such context-sensitive policies correctly. In this paper, we present a set of mechanisms to guarantee that the permissions that are being enforced are current.

To create a secure event channel, user Alice authenticates with the SEC and receives a symmetric key that she can use to publish to the SEC over a newly created channel. When Bob wants to subscribe to the events published by Alice, the SEC checks Bob's current role permissions to verify whether he possesses the necessary permissions. If authorized, the SEC creates a new channel, negotiates a symmetric key with Bob (one per channel), decrypts the message it receives from Alice and re-encrypts it with Bob's key, and sends it on the appropriate channel.

The binding between a user and their current role can change at any time due to a change in context. The SEC is notified by Gaia's access control service when existing user-to-role mappings are altered by a role change in the system. If any user-role assignments are changed for a particular subscriber channel, the SEC revokes the symmetric key and denies unauthorized access to this information. It is important to note that publishers are unaffected by this change, as the Secure Event Component decouples the publish and subscribe operations by using different keys to communicate with publishers than with subscribers.

In the next section, we present the details of our cryptographic protocols, highlighting their key acquisition and revocation features.

## 5 Protocol Details

In this section, we explain the details of the cryptographic protocols used in our system. These protocols are used to negotiate and transfer keys from the SEC to the publishers and subscribers. We also describe protocols for sending messages and requesting new keys. Finally, we discuss how key revocation works in our system.

It should be noted that throughout this section, we assume that the publisher creates the channels in the system and provides an ACL used to restrict the set of potential subscribers. A straightforward extension of these protocols would allow a third party to create a channel and provide ACLs for both publisher and subscriber groups. Due to space limitations, however, this extended set of protocols is not discussed.

### 5.1 Notation

The following notation is used throughout the remainder of this document:

- $P$ is a publisher, $S$ is a subscriber and $M$ is the SEC.
- $ID(B)$ refers to user $B$'s identity certificate, which is publicly available from a trusted Certificate Authority (CA) in the system.
- $Attr(B)$ refers to object $B$'s attribute certificate (e.g., its system role binding).

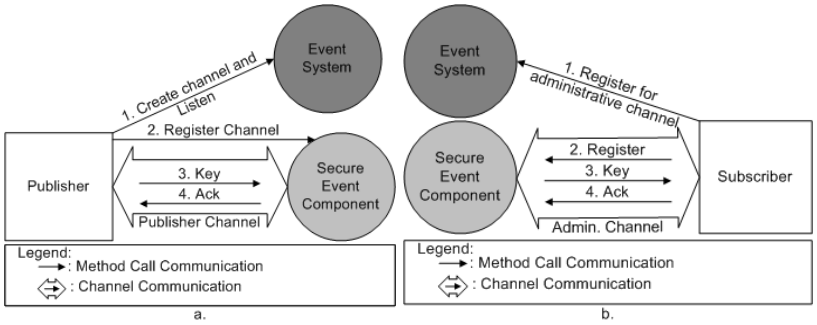


**Fig. 2.** a. The messages required to create a channel. b. The messages required to register as a subscriber for a channel.

- $PK_B$ refers to $B$'s public key and $PK_B^{-1}$ refers to $B$'s private key.
- $K_{PM}$ is a symmetric key shared between the publisher $P$ and the SEC $M$.
- $K_{SM}(P)$ refers to a symmetric key that is shared between the SEC and the subset of authorized subscribers. Note that $K_{SM}(P) \neq K_{PM}$.
- $N_B$ is a nonce generated by user $B$.
- $\{C\}_K$ refers to the message $C$ encrypted with key $K$, whereas $\langle C \rangle_{K'}$ refers to the message $C$ signed by key $K'$.
- $H$ refers to a cryptographic hash function, used to create a MAC.

### 5.2 Authorized Channel Creation

When a publisher, $P$, wishes to publish on a secure channel, he must first create an insecure channel. This is done with the aid of the Event System. This channel, known as the publisher channel, is used by $P$ and the SEC to exchange messages. These messages are shown in Figure 2a. Once the publisher channel is created $P$ and the SEC exchange a key. Key exchange begins when $P$ sends a *secure channel* message to the SEC. This message contains the name of the SEC, a nonce, generated by $P$, an ID certificate for $P$ and a hash of the message signed with $P$'s private key. The entire message is encrypted with the SEC's public key. The SEC responds to $P$ with a message that contains $P$'s name, $P$'s nonce, a new nonce generated by the SEC, a new symmetric key and a hash of the message signed with the SEC's private key. This entire message is then encrypted with $P$'s public key. $P$ acknowledges the receipt of the key by sending a message that contains the SEC's nonce. This message also contains a signed hash and is encrypted with the SEC's public key.

Formally:

$$P \rightarrow M : \{D = (\texttt{SECURECHANNEL},\ \texttt{M},\ N_P,\ ID(P),\ \langle H(D) \rangle_{PK_P^{-1}}\}_{PK_M}$$
$$M \rightarrow P : \{D = (\texttt{OK},\ \texttt{P},\ N_P,\ N_M,\ K_{PM}),\ \langle H(D) \rangle_{PK_M^{-1}}\}_{PK_P}$$
$$P \rightarrow M : \{D = (\texttt{ACK},\ \texttt{M},\ N_M),\ \langle H(D) \rangle_{PK_P^{-1}}\}_{PK_M}$$

This protocol allows for the mutual authentication of the SEC and $P$, along with secure key exchange. It also assures $P$ that the SEC is an authorized system component and binds the actions of $P$ to his identity which ensures nonrepudiation.

We model the protocols using a belief logic. The rules of the Burrows-Abadi-Needham (BAN) formal deduction system [16] are used to draw some conclusions about what guarantees these protocols provide. While the BAN system does not have an independent semantics, it is sufficient here to prove mechanically, given that the assumptions are correct, that key agreement and freshness can be achieved with our protocols. Using only the rules of inference outlined in [16], we prove that after a run of our protocol, a trusted publisher (or subscriber) in our system believes that the SEC believes that he or she shares a symmetric key with the SEC, and that this key is fresh. Likewise, we prove that a trusted SEC believes that $P$ believes that it shares a fresh key with the publisher (or subscriber). We present details of our proof in Appendix A. Note that we cannot prove these results if either the SEC or the publishers and subscribers are malicious and lie about their beliefs.

To prove that we achieve our goals, we assume that all parties communicating can generate fresh nonces. Since each party in the system uses a cryptographic pseudo-random number generator, we feel that this is a valid assumption. Additionally, we assume that all parties can obtain the public-key certificates of the other parties through the Gaia certificate authority. Lastly, we assume that publishers and subscribers trust the SEC to generate appropriate session keys.

Validating that the key agreement and freshness properties can be achieved by our protocols provides us the assurance that our cryptographic mechanisms are being used correctly. It is important to note that these mechanisms only enforce the underlying dynamic policy changes. It is equally important to analyze whether the underlying RBAC framework itself generates the appropriate user-role assignments, consistent with the requirements and goals of our pervasive computing scenarios. We present a proof of this safety property in [15].

### 5.3 Authorized Client Subscription

When a subscriber, $S$, subscribes to a secure channel, she must first acquire an attribute certificate that serves as an attestation of her current system role. The following message exchange is similar to that in Section 5.2:

$$S \rightarrow M : \{D = (\texttt{JOIN},\ \texttt{M},\ N_S,\ Attr(S),\ ID(S)),\ \langle H(D)\rangle_{PK_S^{-1}}\}_{PK_M}$$
$$M \rightarrow S : \{D = (\texttt{OK},\ \texttt{S},\ N_S,\ N_M,\ K_{SM}(P)),\ \langle H(D)\rangle_{PK_M^{-1}}\}_{PK_S}$$
$$S \rightarrow M : \{D = (\texttt{ACK},\ \texttt{M},\ N_M),\ \langle H(D)\rangle_{PK_S^{-1}}\}_{PK_M}$$

This exchange provides the same guarantees as those discussed for channel creation. $S$'s attribute certificate is passed to the SEC in the first message. This allows the SEC to mediate access to the channel based on $S$'s system role.

The client registers as a listener on the administrative channel. All three messages described above are exchanged on the administrative channel. The

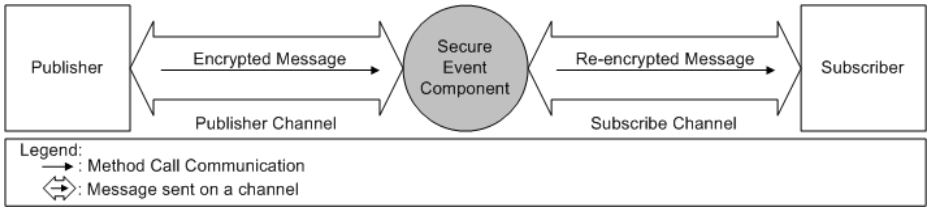


**Fig. 3.** The message publication sequence.

client does not register for the subscriber channel until the final acknowledgment is sent. Figure 2b. illustrates this process.

### 5.4 Publishing a Message

$P$ publishes messages to the SEC encrypted with their shared key. The SEC subsequently decrypts these messages and re-encrypts them with the subscriber key and sends them to the subscribers.

Formally:

$$P \rightarrow M : \{\texttt{MESSAGE}\}_{K_{PM}}$$
$$M \rightarrow S : \{\texttt{MESSAGE}\}_{K_{SM}(P)}$$

This is shown in Figure 3. It is important to note that there may be many subscribers listening to the same channel.

### 5.5 Changing a Publisher Key

$P$ uses a symmetric key to encrypt its messages. Keys that are used for a long period of time are susceptible it is to cryptanalysis, therefore $P$ periodically requests a new symmetric key from the SEC. $P$ requests a key by sending a message that contains a nonce and is encrypted using the key shared between $P$ and the SEC. The SEC subsequently sends a new key to $P$. This message is encrypted with the old key and contains $P$'s nonce and a new nonce generated by the SEC. P acknowledges the receipt of the key by sending the SEC's nonce back, encrypted with the new key.

This exchange is shown as follows:

$$P \rightarrow M : \{\texttt{CHANGE},\ N_P\}_{K_{PM}}$$
$$M \rightarrow P : \{\texttt{KEY},\ N_P,\ N_M,\ K'_{PM}\}_{K_{PM}}$$
$$P \rightarrow M : \{\texttt{KEYACK},\ N_M\}_{K'_{PM}}$$

All of these messages are sent between SEC and $P$ using the publish channel as shown in Figure 4.

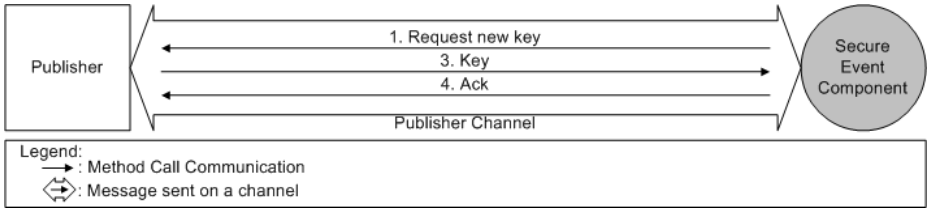


**Fig. 4.** The messages exchanged to get a new key.

### 5.6 Handling a Role Change

In a pervasive computing environment, an individual may have several roles and her active role binding may change for a variety of reasons. The same user may have the role of a lecturer in a smart classroom, and the role of a researcher when a meeting is scheduled in the same room.

Role changes in this system are handled by a unilateral revocation of the keys on the channels affected by the role change. For example, only lecturers may have access to the overhead projectors, so if a lecturer's role changes to a researcher, the projector's publisher key is revoked. In order to accomplish this, the SEC receives a message from a trusted authority, namely the Gaia context system, informing it of the role change. Students subscribed to the projectors display channel need not change their roles or keys.

Unilateral key revocation allows the overhead of revocation to be distributed among the affected entities. These entities are required to reacquire the attributes necessary to continue to use the event channel. This means that the SEC does not need to track the roles and permissions of users subscribed to the channel and also allows key revocation to be fast and efficient.

In the next section, we will evaluate our solution and protocols and show how it meets the needs of pervasive computing environments.

## 6 Evaluation

In this section, we revisit our threat model and argue informally about how our authorization framework addresses our original goals and overcomes the limitations of existing solutions. We revisit each of our threats from Section 2 in turn and examine how we address integrity, confidentiality, privacy, authenticity, and DoS concerns.

Our framework relies on standard public-key certificates for identity authentication. When a new user, Alice, joins our system, she is required to obtain a public-key certificate from our (offline) CA and include this in her communication when she first attempts to contact any server, tries to publish, or subscribe to the event channels. We propose that the user's identity is verified out-of-band before the certificate issued. All trusted entities in the system, including the SEC and the event managers also publish their public keys so that anyone wishing

to communicate with them can use their public key to send them encrypted messages. However, relying on public-key encryption for all confidentiality requirements can be expensive. Therefore the public-key encrypted channels are only used to exchange symmetric keys securely as outlined in our protocols previously.

With the public-key infrastructure, we can also provide integrity protection, non-repudiation, and mutual authentication. For integrity, Alice can create a MAC using a cryptographic hash function and sign this MAC with her private key used for digital signatures. A verifier on the other end can validate the integrity by recomputing the MAC and checking it with the decrypted value embedded in the message. This mechanism can also be used to provide non repudiation of origin. Mutual authentication can be accomplished by a standard challenge-response protocol used in this fashion. Our protocols are optimal in the number of rounds required for mutual authentication.

While our use of the PKI is fairly standard during protocol initiation, all subsequent messages between two mutually authenticated parties are protected by encryption using symmetric session keys for performance reasons. The novelty in our framework is in the use of these keys to enforce authorizations. For example, whenever subscriber authorizations change due to changes in system context, the publishers in our protocol are unaffected. The mediators or the SECs simply change the encryption keys on the channel they share with the subscribers, enforcing this policy change without sacrificing security guarantees. In this case, the subscribers are burdened with the responsibility of obtaining new keys by re-authenticating with the attribute certifiers according to their new roles. This unilateral revocation of keys may inconvenience affected subscribers, requiring them to buffer some event messages from the SECs and delay their processing. On the other hand, when a publisher's authorizations change, subscribers are unaffected. We picked this trade-off deliberately to cause the least amount of impact on the rest of the system, and maximize the precision of enforcement of dynamic access control policies.

Our use of group keys, i.e., one key per role, simplifies key distribution and management. In our initial prototype, we do not optimize group key revocation and sharing. In the future, we plan to explore different hierarchical key distribution frameworks, especially multicast and group-key management [17–20] to optimize this aspect of our protocols.

With respect to the DoS problem, we believe that our distributed mediators (the SECs) make it difficult for an attacker to mount DoS attacks against our infrastructure. In particular, an attacker would need to expend a large amount of resources to target all the different components to cause service outages. Since our protocols themselves are soft-state and our entities can bootstrap easily without needing to coordinate with each other or worry about consistency issues, we claim our solution is resilient to DoS attacks.

However, we realize that a dedicated insider can repeatedly change the context to trigger key revocation and deny service to legitimate users. While our system provides non-repudiation at session start-up, such an attacker can hide

behind the group or role set that he or she belongs to due to our use of symmetric keys at this point. At this point, we may need to look at publish behavior of individual users by analyzing audit logs to catch the culprit. Maintaining these logs becomes more crucial in this context. While our use of symmetric keys at this point makes it harder to catch an attacker quickly in this scenario, we still have accountability at a per-user level.

In the next section, we summarize our work and present our conclusions.

## 7 Conclusion

In this paper we present an access control architecture that integrates context with authorization and provides fine-grained control over the enforcement of dynamically changing permissions. The foundation of this access control architecture is our secure message distribution system which consists of protocols for gaining authorized access to communications channels and securely transmitting messages throughout the system. Our secure message distribution system addresses the threats to the underlying communication system of pervasive computing environments, as well as the limitations of existing access control frameworks.

We have implemented our access control architecture in the Gaia meta-operating system as distributed objects. Our access control and messaging protocols secure Gaia's publish/subscribe communication system. We also present a proof of correctness of our cryptographic protocols using BAN logic.

Our design includes distributed reference monitors, authorization servers, and key managers, which afford maximum flexibility to handle dynamically changing permissions securely. Our SECs maintain ACLs and are directly responsible for enforcing access policy. This design choice enables distributed enforcement of policies without sacrificing consistency. The components of our architecture can be replicated easily for load balancing, are designed to be soft-state, and work independently without coordination. We believe that this aspect of our design makes our system more efficient and resilient to denial of service (DoS) attacks.

## References


1. Kohl, J., Neuman, B.C.: The Kerberos Network Authentication Service (Version 5). Internet Request for Comments RFC-1510 (1993)
2. Neuman, B.C., Ts'o, T.: Kerberos: An Authentication Service for Computer Networks. In: IEEE Communications. Volume 32. (1994) 33–38
3. Housely, R., Ford, W., Polk, W., Solo, D.: Internet X.509 Public Key Infrastructure Certificate and CRL Profile. Internet Request for Comments RFC-2459 (1999)
4. : Public key infrastructure study. National Institute of Standards and Technology (1994)
5. Creese, S., Goldsmith, M., Rosco, B., Zakiuddin, I.: Authentication for pervasive computing. In: Proceedings of the First International Conference on Security in Pervasive Computing, Boppard, Germany, March 12-14th, LNCS, Springer. (2003)

6. Roman, M., Hess, C.K., Cerqueira, R., Ranganathan, A., Campbell, R.H., Nahrstedt, K.: Gaia: A middleware infrastructure to enable active spaces. IEEE Pervasive Computing (2002) 74–83
7. Ashley, P., Vandenwauver, M.: Practical Intranet Security: Overview of the State of the Art and Available Technologies. Kluwer Academic Publishers (1999)
8. Blaze, M., Feigenbaum, J., Keromytis, A.D.: KeyNote: Trust management for public-key infrastructures (position paper). Lecture Notes in Computer Science **1550** (1999) 59–63
9. Rivest, R.L., Lampson, B.: SDSI – A simple distributed security infrastructure. Presented at CRYPTO'96 Rumpsession (1996)
10. Pfitzmann, A., Köhntopp, M.: Anonymity, unobservability, and pseudonymity: A proposal for terminology (2000)
11. Vandenwauver, M., Govaerts, R., Vandewalle, J.: How role based access control is implemented in sesame. In: WETICE. (1997) 293–298
12. Hill, R., Al-Muhtadi, J., Campbell, R., Kapadia, A., Naldurg, P., Ranganathan, A.: A middleware architecture for securing ubiquitous computing cyber infrastructures. In: 5th ACM/IFIP/USENIX International Middleware Conference. (2004)
13. Creese, S., Goldsmith, M., Roscoe, B., Zakiuddin, I.: Authentication for pervasive computing. In: Security in Pervasive Computing. (2003)
14. Wullems, C., Looi, M., Clark, A.: Towards context- aware security: An authorization architecture for intranet environments. In: in the proceedings of the Second IEEE Conference on Pervasive Computing and Communciations Worshops. (2004)
15. Sampemane, G., Naldurg, P., Campbell, R.H.: Access control for active spaces. In: Annual Computer Security Applications Conference. (2002)
16. Burrows, M., Abadi, M., Needham, R.: A logic of authentication. In: Proceedings of the twelfth ACM symposium on Operating systems principles, ACM Press (1989) 1–13
17. McGrew, D.A., Sherman, A.T.: Key establishment in large dynamic groups using one-way function trees. IEEE Transactions on Software Engineering **29** (2003) 444–458
18. Mittra, S.: Iolus: A framework for scalable secure multicasting. In: ACM SIGCOMM '97. (1997)
19. Perrig, A.: Efficient collaborative key management protocols for secure autonomous group communication. In: International Workshop on Cryptographic Techniques and E-Commerce CrypTEC '99. (1999)
20. Steiner, M., Tsudik, G., Waidner, M.: Cliques: A new approach to group key agreement. In: 18th International Conference on Distributed Computing Systems (ICDCS '98). (1998) 380–387

# A BAN Analysis

| Assumptions | Goals | Idealized Protocol |
|---|---|---|
| $P \models \#(N_1)$ | $P \models M \models P \xleftrightarrow{K_{PM}} M$ | $P \to M : \left\{ N_1, \{N_1\}_{K_P^{-1}} \right\}_{K_M}$ |
| $M \models \#(N_2)$ | $M \models P \models P \xleftrightarrow{K_{PM}} M$ | $M \to P : \left\{ N_1, N_2, P \xleftrightarrow{K_{PM}} M, \left\{ N_1, N_2, P \xleftrightarrow{K_{PM}} M \right\}_{K_M^{-1}} \right\}_{K_P}$ |
| $P \models \overset{K_M}{\mapsto} M$ | | $P \to M : \left\{ N_2, P \xleftrightarrow{K_{PM}} M \left\{ N_2, P \xleftrightarrow{K_{PM}} M \right\}_{K_P^{-1}} \right\}_{K_M}$ |
| $P \models \overset{K_P}{\mapsto} P$ | | |
| $M \models \overset{K_P}{\mapsto} P$ | | |

## A.1 Analysis

After step 1 we know: $M \triangleleft \left\{ N_1, \{N_1\}_{K_P^{-1}} \right\}_{K_M}$

$$\dfrac{\dfrac{\dfrac{\bullet}{M \triangleleft \left\{ N_1, \{N_1\}_{K_P^{-1}} \right\}_{K_M}} \quad \dfrac{\bullet}{[M \models \overset{K_M}{\mapsto} M]}\ asmp}{M \triangleleft N_1, \{N_1\}_{K_P^{-1}}}\ sees5 \quad \dfrac{\dfrac{\bullet}{M \triangleleft N_1, \{N_1\}_{K_P^{-1}}}}{M \triangleleft \{N_1\}_{K_P^{-1}}}\ sees1 \quad \dfrac{\bullet}{[M \models \overset{K_P}{\mapsto} P]}\ asmp}{M \models P \mathrel{|\sim} N}\ pubkey$$

After step 2 we know: $P \triangleleft \left\{ N_1, N_2, P \overset{K_{PM}}{\longleftrightarrow} M, \left\{ N_1, N_2, P \overset{K_{PM}}{\longleftrightarrow} M \right\}_{K_M^{-1}} \right\}_{K_P}$

$$\dfrac{\dfrac{\dfrac{\dfrac{\bullet}{P \triangleleft \left\{ N_1, N_2, P \overset{K_{PM}}{\longleftrightarrow} M, \left\{ N_1, N_2, P \overset{K_{PM}}{\longleftrightarrow} M \right\}_{K_M^{-1}} \right\}_{K_P}} \quad \dfrac{\bullet}{[P \models \overset{P}{\mapsto} K_P]}\ asmp}{P \triangleleft N_1, N_2, P \overset{K_{PM}}{\longleftrightarrow} M, \left\{ N_1, N_2, P \overset{K_{PM}}{\longleftrightarrow} M \right\}_{K_M^{-1}}}\ sees5 \quad \dfrac{\dfrac{\bullet}{P \triangleleft N_1, N_2, P \overset{K_{PM}}{\longleftrightarrow} M, \left\{ N_1, N_2, P \overset{K_{PM}}{\longleftrightarrow} M \right\}_{K_M^{-1}}}}{P \triangleleft \left\{ N_1, N_2, P \overset{K_{PM}}{\longleftrightarrow} M \right\}_{K_M^{-1}}}\ sees1 \quad \dfrac{\bullet}{[P \models \overset{K_M}{\mapsto} M]}\ asmp}{P \models M \mathrel{|\sim} N_1, N_2, P \overset{K_{PM}}{\longleftrightarrow} M}\ sees5}{P \models M \mathrel{|\sim} N_1, P \overset{K_{PM}}{\longleftrightarrow} M}\ said \quad \dfrac{\dfrac{\bullet}{[P \models \#(N_1)]}\ asmp}{P \models \#(N_1, P \overset{K_{PM}}{\longleftrightarrow} M)}\ fresh \quad \dfrac{\bullet}{P \models M \mathrel{|\sim} N_1, P \overset{K_{PM}}{\longleftrightarrow} M}}{P \models M \models N_1, P \overset{K_{PM}}{\longleftrightarrow} M}\ nv}{P \models M \models P \overset{K_{PM}}{\longleftrightarrow} M}\ bb$$

After step 3 we know: $M \triangleleft \{\{N_2, P \overset{K_{PM}}{\longleftrightarrow} M, P, M\}_{K_P^{-1}}\}_{K_M}$

$$\dfrac{\dfrac{\dfrac{\dfrac{\bullet}{M \triangleleft \{\{N_2, P \overset{K_{PM}}{\longleftrightarrow} M, P, M\}_{K_P^{-1}}\}_{K_M}} \quad \dfrac{\bullet}{[M \models \overset{K_M}{\mapsto} M]}\ asmp}{M \triangleleft \{N_2, P \overset{K_{PM}}{\longleftrightarrow} M, P, M\}_{K_P^{-1}}}\ sees5 \quad \dfrac{\bullet}{[M \models \overset{K_P}{\mapsto} P]}\ asmp}{M \models P \mathrel{|\sim} N_2, P \overset{K_{PM}}{\longleftrightarrow} M, P, M}\ pubkey}{M \models P \mathrel{|\sim} N_2, P \overset{K_{PM}}{\longleftrightarrow} M, P}\ said}{M \models P \mathrel{|\sim} N_2, P \overset{K_{PM}}{\longleftrightarrow} M}\ said \quad \dfrac{\dfrac{\bullet}{[M \models \sharp(N_2)]}\ asmp}{M \models \sharp(N_2, P \overset{K_{PM}}{\longleftrightarrow} M)}\ fresh \quad \dfrac{\bullet}{M \models P \mathrel{|\sim} N_2, P \overset{K_{PM}}{\longleftrightarrow} M}}{M \models P \models N_2, P \overset{K_{PM}}{\longleftrightarrow} M}\ nv}{M \models P \models P \overset{K_{PM}}{\longleftrightarrow} M}\ bb$$