Access Control in Publish/Subscribe Systems

Jean Bacon David M. Eyers Jatinder Singh
Computer Laboratory
University of Cambridge
Cambridge CB3 0FD, UK
{firstname.lastname}@cl.cam.ac.uk

Peter R. Pietzuch
Department of Computing
Imperial College
London SW7 2AZ, UK
prp@doc.ic.ac.uk

ABSTRACT

Two convincing paradigms have emerged for achieving scalability in widely distributed systems: ***publish/subscribe communication*** and ***role-based***, policy-driven control of access to the system by applications. A strength of publish/subscribe is its many-to-many communication paradigm and loose coupling of components, so that publishers need not know the recipients of their data and subscribers need not know the number and location of publishers. But some data is sensitive, and its visibility must be controlled carefully for personal and legal reasons. We describe the requirements of several application domains where the event-based paradigm is appropriate yet where security is an issue. Typical are the large-scale systems required by government and public bodies for domains such as healthcare, police, transport and environmental monitoring.

We discuss how a publish/subscribe service can be secured; firstly by specifying and enforcing access control policy at the service API, and secondly by enforcing the security and privacy aspects of these policies within the service network itself. Finally, we describe an alternative to whole-message encryption, appropriate for highly sensitive and long-lived data destined for specific domains with varied requirements. We outline our investigations and findings from several research projects in these areas.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*

General Terms

publish/subscribe, role based access control, security, encryption

1. INTRODUCTION

Our work is concerned with how to support and manage information security within and between large-scale, independent, widely distributed application domains. Information security has well-known facets: availability, integrity and confidentiality. Access control most directly targets confidentiality, which is the main focus of this paper.

Achieving security in large, heterogeneous distributed systems is a challenging task because tightly-coupled access control mechanisms often have requirements that are at odds with the loosely-coupled nature of scalable communication paradigms. For example, many traditional access control mechanisms depend on knowing the precise identities of information receivers. However, group communication and dynamic, fine-grained information dissemination often make it hard to provide such guarantees without sacrificing scalability.

Two recently emerging paradigms for achieving scalability in secure distributed systems are asynchronous, *publish/subscribe* communication and *role-based access control* (RBAC). Publish/subscribe is a highly-decoupled distribution model, where (generally) publishers produce information irrespective of consumers. This information may propagate across domains of control (e.g. organisations) and is delivered to subscribers according to their interests. RBAC uses the concept of a role to introduce a level of indirection between principals and protected objects when making access control decisions.

In this paper, we show how the paradigms of publish/subscribe communication and RBAC can be integrated, allowing us to control information dissemination, while retaining the benefits of publish/subscribe communication, such as decoupling of clients and overall system scalability. Following this approach, we describe a secure middleware capable of supporting fine-grained control of communication within and between the domains of a multi-domain architecture.

We also touch on our research into interaction control. Whereas access control tends to focus on binary accept/deny permissions, interaction control facilitates more subtle modification of data in transit within a distributed publish/subscribe system. For example, event delivery in healthcare systems might apply transformations that collect data values into buckets: useful information is transmitted without the data being quite so sensitive. This approach may be more appropriate for highly sensitive, long-lived data.

The paper is structured as follows. Section 2 provides the required background by introducing our models of a widely-distributed system, in terms of domains, publish/subscribe communication and access control. Section 3 describes prototype implementation of publish/subscribe middleware and role-based access control systems that follow the models presented. We also survey existing efforts in integrating access control and publish/subscribe. Section 4 introduces how RBAC and publish/subscribe can be integrated. Section 5 presents our multi-domain architecture in more detail, covering how system-wide unique names are ensured, how event types are managed, and how policies are expressed and administered. We also discuss two aspects of transmission control in the broker network overlay: the encryption required to ensure that untrusted brokers do not see sensitive data; and the control at domain boundaries of the flow of sensitive data out of a domain. Section 6 describes our ongoing work into interaction control. This includes integration of database systems into publish/subscribe systems to facilitate expressive, dynamic filtering. Section 7 describes application scenarios from the policing and healthcare domains. Section 8 summarises and concludes.

2. BACKGROUND

In this section, we introduce the concepts of domains, publish/subscribe and role-based access control. These provide the basis for our work in securing a publish/subscribe service in a multi-domain environment.

2.1 Domain Architecture

In our model of a large-scale distributed application, we define a *domain* to be an independently administered unit in which a domain manager has, or may delegate, responsibility for naming and policy specification. The following motivating scenarios have in common a dedicated communication infrastructure shared by independently administered domains, some of which are strongly related and have similarly named roles for principals. The bulk of the communication is likely to be within a domain but there is also a clear need for inter-domain communication.

1. **Police infrastructure**: A number of county-level police domains need support for intra- and inter-domain messages. Incident reports may be sent within and between domains for real-time response and may also be stored as part of an audit or record-keeping process. Databases for court records and the licensing of drivers of vehicles are accessible from all domains.
2. **Healthcare systems**: The communication infrastructure of a national health service is shared by many independent hospitals, clinics, primary-care practices, etc. Caring for a patient in their home involves carers from many domains. This includes sharing information with various care providers [25], making aspects of patient information persistent in centralised health record services [13] and auditing data flows, to monitor compliance with procedures [24], and to investigate anomalies.
3. **Environmental monitoring**: Traffic, noise, pollution, and weather conditions are monitored in a city to provide real-time information for citizens [1]. All data is recorded for historical analysis to aid prediction and for use by Local Government for planning purposes.

2.2 Publish/Subscribe

Publish/subscribe [9] is emerging as an appropriate communication paradigm for large-scale systems. It allows loose coupling between mutually anonymous components and supports many-to-many communication. For consistency with other publish/subscribe systems, we use *event* as synonymous with the term *message* for an item of communicated data. An event is a data-rich occurrence, encapsulating a particular semantic. Typically, an event instance consists of a set of *(attribute, value)* pairs, conforming to a named event type definition.

In the publish/subscribe paradigm, a principal takes the role of a publisher and/or a subscriber. Subscribers register their interest in receiving an event through a subscription. Publishers produce events without any dependence on subscribers. Principals connect to the publish/subscribe middleware to communicate. This occurs through an *event broker*, which routes, typically in cooperation with other brokers, events from publishers to subscribers. An event is delivered to a subscriber if it matches a subscription. This process is termed *notification.*

Publish/subscribe systems are classified as *type/topic* or *content/attribute-based* [9]. Topic-based publish/subscribe involves the use of an event channel dedicated to a particular named topic/type. Producers publish events to the appropriate channel, while subscribers express their interest in receiving messages of a certain type. Content-based publish/subscribe considers message content: a subscriber defines their interest in receiving particular events based on the type and attribute values of the event instance.

Large-scale, publish/subscribe messaging systems often comprise a network of brokers that provide a communication service and lightweight clients, that use the service to advertise, subscribe to and publish messages [6, 5]. A broker network can have a static topology (e.g. Siena [6] and Gryphon [19]) or a dynamic topology (e.g. Scribe [**?**] and Hermes [18]). Our approach for securing publish/subscribe applies to both cases. A static topology enables the system administrator to build trusted domains and in that way improve the efficiency of routing by avoiding unnecessary encryptions (see Section 5.7), which is more difficult with a dynamic topology. On the other hand, a dynamic topology allows the broker network to dynamically re-balance itself when brokers join or leave the network either in a controlled fashion or as a result of a network or node failure. Broker networks are subject to failures of nodes and links, and brokers may join and leave dynamically. Thus, a communication service must be robust under these conditions, fault-tolerant and dynamically reconfigurable. For this reason the message brokers may exploit an overlay network [19], since peer-to-peer naming and protocols provide the necessary robustness.

2.3 Role-based Access Control

Role-Based Access Control (RBAC) [23] is an established technique for simplifying scalable security administration by introducing *roles* as an indirection between *principals* (i.e. users and their agents) and *privileges.* Privileges, such as the right to use a service or to access an object managed by a service, are assigned to roles. Separately, principals are associated with roles. This separates the administration of people, and their association with roles, from the control of privileges for the use of services (including service-managed data). The motivation is that users join, leave and change roles in an organisation frequently, and the policy of services is independent of such changes. Service developers need only be concerned with specifying access policy in terms of roles, and not with individual users.

Here we focus on securing access to the communication service using RBAC. Authentication into roles must be securely enforced to control the use of all protected services and access to the data they manage [2]. Domain managers, or their delegates, specify communication policy in terms of message types and roles; that is, which roles may create, advertise, send and receive which types of message. Inter-domain communication is achieved through negotiated agreements, expressed as access control policy, on which roles of one domain may receive (which attributes of) which types of message of another.

The notion of role is ideally suited to a multicast communication style. For example, the Cambridge police domain may define a role officer-on-duty and message topics such as burglary, traffic-accident with associated attributes. Officers on duty can subscribe to receive notifications of incidents for which they should take responsibility. RBAC causes principals to be anonymous whereas parametrised RBAC gives the option of anonymity or identification, for example officer-on-duty (station-ID, police-ID). The use of parametrised roles can also help to avoid an explosion in the number of roles required when RBAC is used in large systems.

For the communication service, RBAC policy indicates the visibility (to roles, intra- and inter-domain) of specified attributes of message types. The fact that advertisement is required before messages can be published, and that both are RBAC-controlled, prevents the spam that pervades email communication between humans. Without such control denial-of-service through publication or subscription flooding could degrade large-scale inter-software communication in the same way that it consumes resources in email management. With our approach, a spammer could only be an authorised, authenticated member of a role and therefore could be held accountable.

2.4 Secure Communication

If the network and message brokers could be guaranteed 100% secure and trustworthy, then RBAC would achieve precisely the visibility specified by policy. It is already standard practice to protect confidential data on the wire by means of encryption, since the network is vulnerable to listeners. In addition, we observe that the brokers are not likely to be trusted universally with all data. Our experiments have investigated fine-grained security, where message attributes are encrypted selectively, with key management transparent to the client level. Encryption overhead per se does not need to be justified, and our evaluation indicates that our approach can often incur less overhead than using whole-message encryption.

3. PROTOTYPE SYSTEMS

Although our approach is generally applicable, our design and implementation are based on specific prototypes, *Hermes* publish/subscribe and *OASIS* RBAC. This section provides a brief overview of these prototype implementations, describing the features specific to them.

3.1 Hermes Publish/Subscribe

Hermes [19, 18] is an architecture for distributed, content-based publish/subscribe with an integrated programming model. It consists of two kinds of component: *event brokers* and *event clients*, the latter being *publishers* and *subscribers*. Event clients publish, or subscribe to, events in the system. An event client has to maintain a connection to a *local event broker*, which then becomes *publisher-hosting*, *subscriber-hosting*, or both. We assume that clients fully trust this broker. A local broker is usually either part of the same domain as the client, or it is owned by a service provider trusted by the client. An event broker without connected clients is called an *intermediate broker*.

Event brokers form the application-level overlay network that performs event propagation by means of a content-based routing algorithm. Most publish/subscribe systems, including Hermes, optimise content-based routing of events with a *subscription coverage relation*, which states which subscriptions are subsumed by others [6]. This allows brokers to reduce the number of events sent through the system by enabling them to filter non-matching events as close as possible to the publisher; these filters become increasingly specific as events approach subscribers.

Hermes supports strong *event typing*: every published event (or *publication*) in Hermes is an instance of an *event type*. An event type has an *event type owner*, an *event type name* and a list of typed *event attributes* so that, at runtime, publications and subscriptions can be type-checked by the system. Subscribers express their interest in the form of *subscriptions* that specify the desired event type and a conjunction of (content-based) filter expressions over the attributes of this event type. Hermes event types are organised into inheritance hierarchies, but our work does not depend on this. We show later how inheritance can be used within domains when it is available.

Each event type defined within a domain is registered by its owner via a local event broker. This causes encryption status and keys to be set up within the domain and a rendezvous node to be selected for peer-to-peer routing. Before a publisher can publish an event instance, it must submit an *advertisement* to its local event broker, indicating the event type that it wishes to publish. A Hermes publication consists of an *event type identifier* and a set of attribute value pairs. The type identifier is the SHA-1 hash of the name of the event type. It is used to route the publication through the event broker network. It conveniently hides the type of the publication, i.e. brokers are prevented from seeing which events are flowing through them unless they are aware of the specific event type name and identifier.

The rendezvous node for an event type is selected by hashing the type name to a broker identifier—an operation that is supported by the peer-to-peer routing substrate [22]. Advertisements and subscriptions are routed towards the rendezvous node, and brokers along the path set up filtering state for them.

For reliability reasons, rendezvous nodes are replicated for each event type. In Hermes, a rendezvous node keeps an authoritative copy of the event type definition, which is cached at other brokers throughout the system for type-checking advertisements, subscriptions, and publications. In our current work, authoritative, domain-specific type information is stored within the originating domain and rendezvous nodes hold a copy.

3.2 OASIS Role-Based Access Control

The *Open Architecture for Secure Interworking Services* (OASIS) [3, 4], provides a comprehensive rule-based means to check that users can only acquire the privileges that authorise them to use services by activating appropriate roles.

A role activation policy comprises a set of rules, where a role activation rule for a role r takes the form

$$r_1, .., r_n, a_1, .., a_m, e_1, .., e_l \vdash r$$

where r_i are prerequisite roles, a_i are appointment certificates (most often persistent credentials) and e_i are environmental constraints. The latter allow restrictions to be imposed on when and where roles can be activated (and privileges exercised), for example at restricted times or from restricted computers. Any predicate that must remain true for the principal to remain active in the role is tagged as a *role membership condition*. Such predicates are monitored, and their violation triggers revocation of the role and related privileges from the principal.

An authorisation rule for some privilege p takes the form

$$r, e_1, .., e_l \vdash p$$

An authorisation policy comprises a set of such rules. OASIS has no negative rules, and satisfying any one rule indicates success.

OASIS roles and rules are parametrised. This allows fine-grained policy requirements to be expressed and enforced, such as exclusion of individuals and relationships between principals, for example treating-doctor(doctor-ID, patient-ID). Without parametrisation it becomes necessary to define an unmanageably large number of roles for an organisation of any significant size.

3.3 Related Work

Securing publish/subscribe systems using access control is an active research area. Wang et al. [29] propose a number of considerations for publish/subscribe access control. Miklós [12] provides semantics defining a security ordering based on event attribute values. Scoping [10] uses grouping structures to control the visibility of occurrences within the infrastructure.

Some work empowers principals with control over the information they produce. *Symmetric publish/subscribe* [27] allows publishers to couple constraints with their publications. Events are matched by computing the intersection of the publication and subscription constraints. Oprychal et al. [14] define *event owners*, who conditionally license other principals with event privileges. Their approach gives control to publishers, who can be defined to own the event instances they produce.

Wun and Jacobsen [30] define a distributed *post-matching* policy model, enabling a broker to perform actions after a particular content-based matching operation. Although generic, they provide examples as to how such a model could be used for security aspects. A similar approach underlies our work on interaction control [26].

4. ACCESS CONTROL POLICY FOR PUBLISH/SUBSCRIBE CLIENTS

In OASIS RBAC, the authorisation policy for any service specifies how it can be used in terms of roles and environmental constraints. Here, we use OASIS to protect the publish/subscribe service in this way at a local broker. This implements security at the publish/subscribe network edge. The service's methods include the following:

```
define(message-type)
advertise(message-type)
publish(message-type, attribute-values)
subscribe(message-type, filter-expression-on-attributes)
```

OASIS policy indicates, for each method, the role credentials, each with associated environmental constraints, that authorise invocation. OASIS role parameters can be used to limit privileges to particular message_types. The define method is used to register a message type with the service and specify its security requirements at the granularity of attributes. On advertise, publish and subscribe, these requirements are enforced. We can therefore support secure publish/subscribe within a domain in which roles are named, activated and administered.

A domain-structured OASIS system is engineered with a per-domain, secure OASIS server, as described in [3], and a per-domain policy store containing all the role activation and service-specific authorisation policies. This avoids the need for small services to perform authentication and secure role activation. The domain's OASIS server carries out all per-domain role activation and monitors the role membership rule conditions while the roles are active. This optimisation concentrates role dependency maintenance within a single server and provides a single, per-domain, secure service for managing inter-domain authorisation policy specification and enforcement.

5. LARGE-SCALE BROKER NETWORKS

In this section we consider broker networks of sufficient scale that the brokers cannot all be trusted unconditionally. We present a scalable, multi-domain mechanism for naming and defining event types. In order to support efficient routing, but without releasing sensitive information, we introduce attribute encryption to allow some event data to remain opaque to some brokers. Decoupling the encryption from the whole event requires secure associations between type names and encryption keys. The secure event types presented in the previous section are used to make these secure associations. This work was originally presented by Pesonen et al. in [15, 16, 17].

There are several levels of access control. At the first level, participants are vetted as they try to join the publish/subscribe network. Unauthorised event brokers will not be permitted to join. Note that in terms of link-level security, all the connections between brokers use *Transport Layer Security* (TLS) [8] in order to prevent unauthorised access to data within lower layers of the network stack.

Clients must connect to local event brokers in order to access the publish/subscribe system's services. Thus the clients cannot directly interfere with publish/subscribe behaviour. The brokers are able to touch all events that pass through them. Brokers can examine the level of traffic and names of attributes that exist in an event (when attribute-level encryption is used). It may be possible to obfuscate event streams (e.g. by inserting dummy events) in order to make traffic analysis more difficult

5.1 Management of Event Names and Policies

When constructing policy-secured, multi-domain publish/subscribe systems, a mechanism is needed through which to agree on the naming of event types. We assume that domains are allocated unique names within the system as

Name tuple:	1	Type issuer's public key
	2	User-friendly name
	3	Version number
Body:	4	Attributes
Digital signature:	5	Delegation certificates
	6	Digital signature

Figure 1: Contents of a secure event type definition

a whole and that roles are named and managed within a domain. Each domain provides a management interface through which role activation policies and service authorisation policies can be specified and maintained.

A group of domains may have a parent domain from which an initial set of role names and policies is obtained. For example, county police domains may agree to use a nationally defined set of police roles; health service domains may start from an initial national role-set. The domain management interface allows local additions and updates, for example when government changes national policy. Parametrised roles allow domain-specific parameters, for example sergeant(domain-ID, police-ID). This allows relationships to be captured as well as avoiding excessive numbers of roles in large-scale systems.

In an evolution from the single-domain Hermes publish/subscribe system, we introduce a format of event type definitions that binds the type name and definition together in a secure manner. Public key cryptography is used to guarantee the authenticity and integrity of this type information. Thus we protect the system against forged or tampered event type definitions. We reduce the chance of accidental name collisions, and provide a unique handle through which policy can refer to the names of types and attributes.

We require that all participating brokers in the publish/subscribe system have a key-pair. We can thus require that event-type issuers incorporate this public key into the type name. This facilitates an event naming scope for each particular type issuer. Since event type names include a public key, it is intuitive that the event type definitions should be signed by the corresponding private key. This binds the type definition to the type name, and facilitates verification of event type integrity and issuer authenticity.

The six items that make up a secure event type definition are shown in Figure 1. Items 1–3 identify the name of the type. The Attributes item 4 indicates the core event type definition, and items 5 and 6 contain a digital signature of the event type.

Adding a public key to the type name eliminates event name conflicts. While this is desirable, a user-friendly name is also maintained. The user-friendly name is able to encode useful aspects, such as hierarchical naming. This enables administrative grouping of type definitions across multiple event type owners. Orthogonal to both of those concerns is type evolution, hence the provision of a version number. Releasing a new version of an event type definition will not conflict with previous instances still in use within the publish/subscribe system. Indeed, to avoid race conditions for version numbers when multiple type managers are releasing updated event definitions, a UUID scheme is used for version numbers. UUIDs are 128-bit values that are coupled with a practically collision free generation algorithm.

Item 4 in the event type definition describes the event type structure. Each attribute definition itself consists of a user-friendly name, a unique identifier (UUID), and an attribute type identifier. The set of types supported depends on the subscription filter language used. Friendly names for attributes are intended to be used by clients of the publish/subscribe system, whereas the UUID is used by intermediate brokers during distributed event routing. The friendly names only need to be unique within the context of one particular version of an event type definition. When a publisher-hosting broker receives an event to route, it looks up the friendly names used by its client using the publisher's event type definition. This allows the UUID fields to be correctly populated. The reverse of this process occurs at subscriber-hosting brokers. The UUIDs allow multiple versions of an event type to exist within the publish/subscribe system at a point in time.

The digital signature of an event type provides a guarantee of the authenticity and integrity of the type definition. The signature is calculated over all the type definition items except the signature itself: items 1–4 in Figure 1. It thus binds the type definition and the name tuple.

The delegation certificates (item 5) facilitate Internet-scale management of event types. Since key-pairs are involved in signing event types, without delegation certificates the type owner would need to re-sign all updates to the type. Delegation certificates facilitate a digitally-signed path of trust from the original event type owner to type managers: parties that are allowed to update event types on their behalf. The delivery of delegation certificates to type managers can be performed out-of-band. The delegation certificates also provide the means to specify fine-grained access rights. Our prototype implementations have typically supported rights such as addAttribute, removeAttribute, editAttributeName and editAttributeType.

5.2 Capabilities for Publish/Subscribe Access Control

This section presents the set of capabilities required to support multi-domain, distributed, publish/subscribe access control. Each domain will contain a number of publish/subscribe clients, and an *access control manager*. The access control manager grants privileges to brokers and clients within that domain according to the domain's access control policy. In order to bootstrap the overall network infrastructure, one of the domains is designated as the *coordinating domain*. The coordinating domain invites other domains into the shared publish/subscribe system.

There are two main reasons that domains may wish to join the shared infrastructure: either to facilitate applications that communicate across domains, or to increase the reliability and coverage of the broker network. Mutually distrusting domains will only cooperate to achieve these ends if they trust that access control policy will be enforced.

There are four main aspects that need to be controlled:

1. Nodes (i.e. brokers and clients) joining and leaving.
2. Event type and topic definition.
3. Event type and topic modifications.
4. Event clients accessing the publish/subscribe API.

In all four cases the capabilities are rooted at an appropriate resource owner. In the case of the first two, the resource owner is the coordinating domain. In the latter two, the resource owner is the event type or topic owner.

5.3 Delegating Authority

In our model, resource owners grant delegation certificates to the access control manager of the target domain. The delegation certificate authorises the access control manager to perform further intra-domain delegations as appropriate. Clearly the intra-domain policy can be as complex as the access control manager wishes to support.

The separation of inter- and intra-domain concerns means that resource owners have to trust access control managers within a domain to behave appropriately. The resource owner has no control over the certificates issued to domain members by a domain's access control manager. However, since the access control manager can have all its authority revoked by a resource owner, it is unlikely that the inter- and intra-domain management separation will prove problematic.

A client requesting an action will present the capability it received from its domain's access control manager, and the chain of delegation certificates linking the access control manager to the resource owner. Thus the access control manager's key-pair is the link between the capability and delegation certificate chains. Any verifier of a client request will satisfy itself that all the certificates in the chain are satisfactory.

Next we describe the access control actions that can be requested. The authorities that are described below can be aggregated into single certificates if the issuer has sufficient authorisation. So a coordinating domain could issue a certificate granting connect and install rights to a particular access control manager (action: connect | install). The authorities can also contain wild-cards. For example, action: * enables all possible actions. Of course the resource owner can only grant authority to their own resources.

Broker network access

The most fundamental access control decisions in the system relate to nodes joining the broker network. Authority is rooted at the coordinating domain: the owner of the shared publish/subscribe system. The access right is binary. The authority also indicates the name of the shared publish/subscribe network:

```
network: TestNetwork1
action: connect
```

Mutual checking should be done between both parties at connection time.

Introducing new types/topics

As mentioned previously, the authority to install new types or topics into the shared publish/subscribe system is also granted by the coordinating domain. As for the case of broker network access, the access right is binary, and the name of a particular shared publish/subscribe network is given:

```
network: TestNetwork1
action: install
```

Extending types/topics

Extending types and topics is a concern for event type owners. Any given type extension authority must come from the owner of the type being inherited from. The granted authorities are coarse, providing a binary answer as to whether inheritance of a particular type is permitted. Using wildcards in the type name can significantly reduce the number of certificates required:

```
type: Test*
action: extend
```

The extended type's definition must include the delegation certificate that grants authority for the original event type to be extended.

Accessing the publish/subscribe API

Finally, clients accessing the publish/subscribe API need to be controlled. Any publication will be related to a particular event type (topic), and thus the topic owner is the resource owner of the API. The topic owner will issue delegation certificates to appropriate access control managers.

The access rights for the publish/subscribe API can be controlled in a fine-grained manner. Authorities can choose to grant clients access to a wide range of types with corresponding attributes for publication and subscription. However, the authorities can also choose to issue rights over a particular type, and can limit the specific actions that can be taken:

```
type: TestType1
action: subscribe | publish
attributes: Attr1 & Attr2 = 500
```

In the above example, the authority has placed a restriction on event content: at publication time the publisher-hosting broker will implicitly force the attribute value to 500 (but accept the client request). Similarly the subscriber-hosting broker will silently add the above filter on Attr2 to the subscription.

5.4 Credential Propagation

A client will present its credentials to its local broker when using the publish/subscribe system API. The local broker can then assess whether the client is authorised to take the requested action. Since routing events in a wide-area publish/subscribe system may involve multiple inter-broker message hops, the client's credentials need to be passed through the publish/subscribe network alongside the event data to enable intermediate brokers to verify access rights. For this, the client's credentials, and a timestamp (to preclude replay attacks), are signed by the client.

We assume that brokers trust each other when propagating the client's credentials. Brokers are permitted to strike a balance between verification and speed. Comparative paranoia will dictate the need to verify a larger proportion of the events that pass through a broker.

As in many scalable peer-to-peer infrastructures, soft-state is used within the network. Rather than maintaining connectivity in order to signal revocation of particular privileges, soft-state will simply time-out privileges if they are not refreshed periodically.

We extend the above capability-based access control mechanism into one that controls access to encryption keys using the same certificate structures. A publisher who is authorised to publish events of a given event type will also be deemed authorised to access the encryption keys used to protect events of that type.

5.5 Encrypting Event Content

To enforce access control within a large broker network, we control access to particular encryption keys. Because clients have to trust their local broker, the event content encryption can be managed without involving the clients: encryption tasks are delegated to brokers. This lowers the number of nodes that need to have access to particular encryption keys. We also change the specific keys used over time in response to changes in the broker topology, and periodically in order to reduce the use of any one key. There are three main benefits of this approach:

1. Fewer nodes need to handle confidential encryption keys, reducing the probability of keys being disclosed.
2. The key refresh operations involve fewer nodes, and thus incur lower overheads.
3. Local brokers only need to perform decryption once when delivering a given event to local subscribers.

A fundamental choice is whether to encrypt event data as a whole, or to encrypt attribute data separately. Whole-event encryption is simpler to implement, requires fewer keys, causes fewer cryptographic operations, and thus is usually faster. On the other hand, attribute encryption provides finer-grained control over event data, and may be beneficial in cases where brokers can selectively route events based on unencrypted attributes.

5.5.1 Whole-event encryption

The simplest approach to protecting event content is to encrypt entire events. For routing purposes, the event type identifier is left intact and unencrypted. In transit, events consist of a tuple containing the type identifier, a publication timestamp, ciphertext and a message authentication tag. Each event type in the system will have its own individual associated encryption key.

Event brokers authorised to access the event data will be able to acquire the current encryption key, decrypt the event, and potentially perform content-based routing. Event brokers without this authorisation will have to route the event based on its event type alone. Potentially this will reduce the quality of routing decisions made.

Whole-event encryption requires one encryption per publication, performed on the publisher's local broker, one decryption at each filtering intermediate broker, and finally a decryption at each broker local to any subscribers.

5.5.2 Attribute encryption

Finer-grained control can be achieved by associating an independent encryption key with each attribute. The encryption key is indexed by the attribute's UUID. Event type identifiers are left intact to allow all brokers to perform some degree of routing. Authorised brokers will be able to obtain the keys for each attribute, and thus carry out content-based routing over those attribute values.

When in transit, events using attribute encryption contain an event type identifier, a publication timestamp, and a set of attribute tuples. Each attribute tuple contains an attribute identifier, some ciphertext and an authentication tag. In our prototype implementation, the attribute identifier is the SHA-1 hash of the attribute name used within the event type definition. The hash prevents unauthorised parties from learning the attributes included in a given event.

Attribute encryption usually results in higher overhead than whole-event encryption. The initialisation of encryption algorithms tends to be computationally expensive, and usually dominates the total encryption time for attributes without much data. The initialisation has to be repeated for each attribute. The evaluation presented in [15, 16] demonstrates these performance issues.

The advantage of attribute encryption is that type owners can provide clients with different levels of access to the same event type. It also enhances the ability of intermediate brokers to perform content-based routing. This has the desirable effect of reducing the number of messages sent between brokers when content-based filtering is significant.

We can emulate attribute encryption by introducing a new event type for each level of subscriber authorisation. Publishers can then publish multiple events that are effectively views on a single real event. A drawback of this approach is that it becomes difficult to manage when there are many event attributes, or many levels of subscriber authorisation.

5.6 Encrypting Subscriptions

Since intermediate brokers are not considered universally trustworthy in our model, subscriptions must also be encrypted. This enforces that only authorised brokers can submit subscriptions to the publish/subscribe network, and that unauthorised brokers do not gain information when establishing subscription paths.

When doing whole-event encryption, the subscription filters are also completely encrypted. Even so, the event type identifier within the subscription is left intact so that brokers are still able to route events based on topic, even when they are not authorised to access the subscription filter. Unauthorised brokers have to assume that events with hidden subscription filters have no filters.

In the case of attribute-based encryption, each attribute filter is encrypted individually. Again, the attribute identifiers are left intact.

5.7 Avoiding Unnecessary Encryption

If two communicating brokers have compatible levels of authorisation, an optimisation is to avoid cryptographic operations. In particular, this will be the case for communication between brokers within the same domain. To establish the degree of compatibility, brokers examine each others' credentials at connection time and add them to their routing tables for future reference. A publisher-hosting broker always encrypts content, since it is computationally cheaper to do so once for the entire event dissemination tree.

To avoid unnecessary decryption operations, a plaintext content cache is attached to otherwise encrypted events. The cache is filled on-demand, and discarded whenever the event routing traverses a link between brokers that do not have mutual trust. In general, the increased communication cost of this cache is dominated by the computational cost of encryption operations.

5.8 Key Management

Irrespective of the chosen approach to event encryption, the encrypted data has a UUID. This identifier is used to determine the key for encryption or decryption. By controlling access to the encryption key, we enforce access control over the encrypted event content.

Access to encryption keys occurs within *key groups* of brokers. Key groups are the basis of issuing and re-issuing encryption keys to brokers. A *key group manager* is responsible for verifying that a broker is authorised to join a given key group. Therefore the key group manager requires the trust of the event type owner that access control will be enforced appropriately. The key group manager may be a member of the type owner's domain, or some other trusted third party. The same type of capability structure used for managing publish/subscribe requests (see Section 5.2) is also used for authorising membership of key groups. Although the mechanisms for enforcing access control are different for clients and brokers, we maintain consistency in the capability representation.

For any client request, both the client and its local broker must have sufficient authorisation to carry out the request. The local broker has to check the client's credentials against access control policy. However, since the local broker performs encryption on behalf of the publishing client, the broker will also need to be authorised to access the necessary encryption keys.

5.8.1 Secure group communication

Decentralised, multi-domain, encrypted publish/subscribe communication can be seen as a type of secure group communication. In any secure group communication mechanism, the key management system must scale well in the number of clients. It is desirable that the communication mechanism is effective over widely-dispersed participants, that high rates of node churn (joining and leaving) do not affect safety or liveness, and that all members are in close time synchronisation.

A number of scalable key management protocols are surveyed in [?]. Our prototype implementation uses the *one-way function tree* (OFT) [11] protocol. The OFT protocol uses a binary tree that places participants at the leaves of that tree. For n participants, the algorithm scales in $log_2 n$ in processing and communication costs, and in the amount of state stored at each participant. These results were experimentally verified in [15, 16, 17]. OFT is comparatively simple to implement and yet performs well. The same structured overlay network used for inter-broker event communication can also be used for the OFT protocol.

5.8.2 Key refreshing

Group key management schemes tend to regenerate encryption keys for two main reasons: *key freshness* and *membership changes*. Regenerating keys periodically avoids large amounts of data being encrypted with the same key. This reduces data exposure should a key become compromised. Keys are also regenerated when group membership changes to provide forward and backward secrecy: members who have left the group cannot access new data, and members who join cannot access old data. In our context, a broker that is a member of a given key group must hold the corresponding capability.

Even with state-of-the-art key management protocols, rekeying is an expensive operation in terms of network traffic and distributed coordination, and is best avoided as much as possible. The simplest approach to ensure freshness of capabilities is to limit their validity periods. To avoid unnecessary key refreshes, the key manager only checks the validity condition when members join or leave. For joining members, whose capabilities were valid at the time of the most recent key refresh, no further refresh is required. Similarly, when a member leaves, key refresh can be held off until the capability expires. These cases are illustrated in Figure 2. In general, the validity of brokers' capabilities will be far less dynamic than those of clients of the publish/subscribe system.

Since data may still be travelling through the publish/subscribe network for a small period of time after key refreshes have occurred, old keys will need to be maintained for that time. To address this issue, new encryption keys are tagged with a timestamp. The timestamp allows the correct encryption key to be used based on an event's publication timestamp. The key destruction delay employed depends on the nature of the application and the size of the network.

6. INTERACTION CONTROL

The publish/subscribe access control mechanisms that are discussed above involve binary (permit/deny) protection to events, through restrictions and encryption. However, rather than denying access, information can be made appropriate for disclosure, through functions that perturb, fuzzify, summarise or translate between formats. Our recent work on *interaction control* provides the means for data to be modified as part of the dissemination process. This enables a domain to exercise fine-grained control over the content of information released. Customising data to context can protect sensitive information, by releasing to a recipient only that data *required* in the circumstance.

Interaction control enables a broker to transform an event instance either upon publication (receipt by the broker—to render it suitable for local processing), or before notification to a particular subscriber (to customise the information to the recipient). Transformations may alter the values of an event instance, or convert an event into another type to encapsulate a different semantic. Thus a broker can degrade, enrich or produce some loosely related event instances. Transformations are conditional, in that they only apply in defined situations.

Interaction control is suited to application environments, e.g. healthcare, where domains are responsible for the information they handle. In situations where an event is relevant to multiple subscribers, domain policy can tailor an event, providing to each only that information relevant for their task: for example, events recording a patient's physiological state might fuzzify information (e.g. location details) except in emergency situations; a drug-auditor requires information regarding prescriptions, but with patient information removed. As transformation occurs in the middleware, publishers are not burdened with the data policy requirements of other principals.

Interaction control is compatible with encryption schemes, though the aim is to customise information to specific requirements. While this may include attribute-level control, it can also involve perturbation, or even the enrichment of events. In some situations, e.g. on a battlefield, data is only sensitive for a particular time period. However, in scenarios where data remains sensitive, it may not be appropriate to liberally distribute encrypted information, due to key-

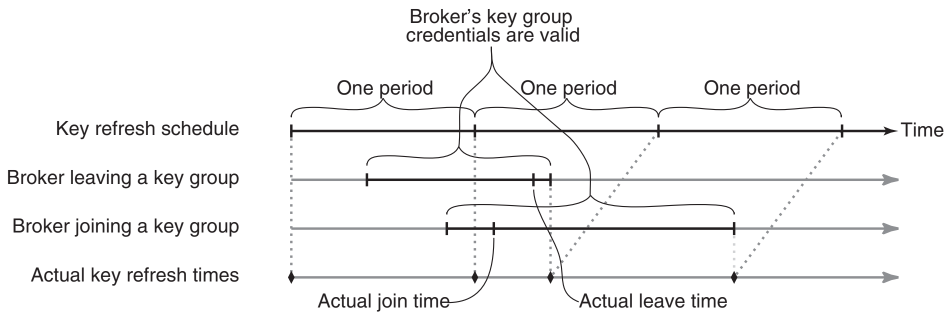

Figure 2: A key refresh schedule adjusted based on the validity periods of brokers' capabilities

management concerns.

6.1 Implementation

We have integrated publish/subscribe messaging capabilities into the PostgreSQL database engine to allow a database instance to function as an event broker [28]. As events encapsulate data, often this information must be stored, if only for auditing purposes. Databases are a common component of enterprise infrastructure, and given their well-established data handling capabilities, provide an obvious point for messaging-system integration. The coupling of publish/subscribe mechanisms with database environments allows us to build on the relational model, query languages, transactions and backup/auditing capabilities, while facilitating data persistence, communication and replication — all under a common type interface. Databases may be subscribers to facilitate replication and maintain an audit or log of selected events. Databases may advertise the events they are prepared to publish so that their subscribers may be notified of conditions that are triggered within the database.

We have extended this implementation to include policy-based mechanisms for interaction control [25, 26]. This brings application-level data control policy into messaging and storage middleware. Event restriction and transformation operations can use information about current environment, event content, publisher or subscriber details, stored data, external functions, etc. Further, we exploit transactional and auditing facilities, which is important for monitoring data disclosure, in addition to detecting policy errors and anomalies.

7. MULTI-DOMAIN APPLICATIONS

In this section we describe the application of the models presented above to real-world multi-domain environments. These environments are similar to those we have explored in recent research projects.

7.1 Police

Figure 3 shows a multi-domain publish/subscribe network based on the United Kingdom Police Forces, with three particular sub-domains:

Metropolitan Police Domain. This domain contains a set of CCTV cameras that publish information about the movements of vehicles around the London area. We have included Detective Smith as a subscriber in this domain.

Congestion Charge Service Domain. The charges that are levied on the vehicles that have passed through the London Congestion Charge zone each day are issued by systems within this domain. The source numberplate recognition data comes from the cameras in the Metropolitan Police Domain. The fact that the CCS are only authorised to read a subset of the vehicle event data will exercise some of the key features of the enforceable publish/subscribe system access control presented in this paper.

PITO Domain. The Police Information Technology Organisation (PITO) is the centre from which Police data standards are managed. It is the event type owner in this particular scenario.

In our example scenario, the Congestion Charge Service would only be authorised to read the `numberplate` field of vehicle sightings — the location attribute would not be decrypted. We thus preserve the privacy of motorists while still allowing the CCS to do its job using the shared publish/subscribe infrastructure.

Let us assume that a Metropolitan Police Service detective is investigating a crime and she is interested in sightings of a specific vehicle. The detective gets a court order that authorises her to subscribe to `numberplate` events of the specific number plate related to her case.

7.2 Healthcare Environments

Health information is sensitive: its confidentiality protected by oath, codes of conduct and law. Health data concerns patients; however, the users of patient data are typically care providers. Thus, they have a responsibility to respect the privacy of information obtained as part of the care process.

Collaboration is central to healthcare. In modern health environments many forms of collaboration exist — from specialist referrals to remote diagnosis to end-of-shift handovers in a hospital ward. Therefore, while health information must be protected for reasons of privacy, it must also be shared to afford proper care.

Health systems must allow for definition of the circumstances in which it is *appropriate* that data be shared. This involves the use of policy that accounts for current context, including the identity and credentials (roles and privileges) of the principals involved, and environmental state. Health infrastructure tends to be controlled, providing information on users. This information is useful for access control decisions, auditing procedures and the detection of policy errors.

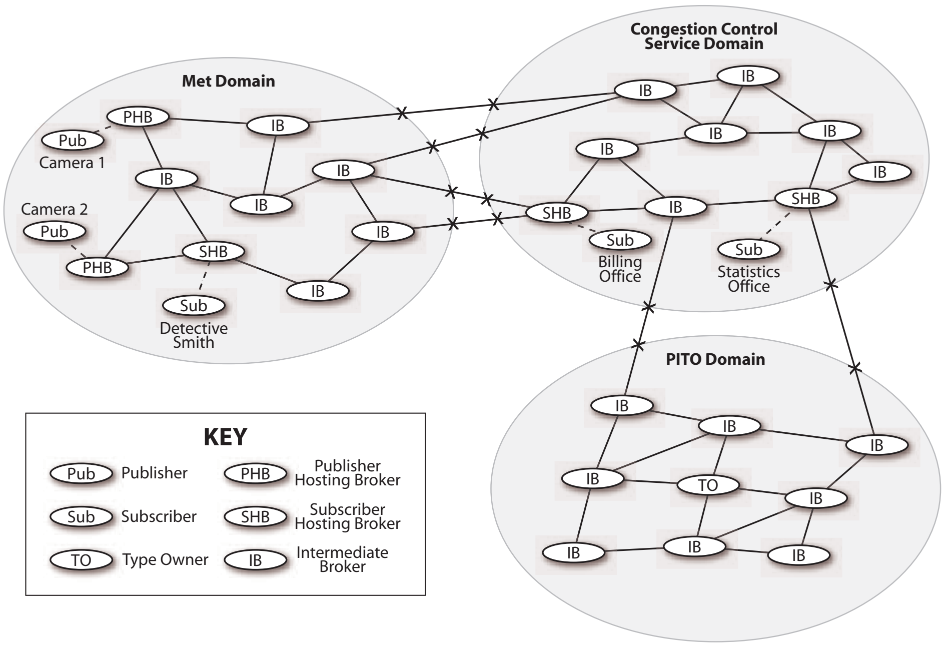

Figure 3: Overview of infrastructure

7.2.1 *Future healthcare — Homecare*

A majority of care concerns chronic (ongoing) conditions. With the ageing population, there is a push to manage such conditions better, to improve quality of life, while reducing the burden on health services. As such, there is a global movement towards intermediate care services: providing care outside traditional care institutions, closer to the home. The envisaged benefits for homecare include increased freedom, mobility and quality of care for patients, while allowing for better health resource allocation.

Homecare environments are small domains, dynamically created for the treatment of a patient [24]. As these lie outside an existing domain structure, homecare environments lend themselves to cross-organisational collaboration. For example, there may be interactions between doctors, care nurses, specialists, pharmacies, technical support, billing services, etc., in addition to the range of technologies (sensors) supporting monitoring services. Homecare environments are highly data driven, where particular incidents (e.g. a sensor reading, a particular treatment action) are of relevance to various principals. As such, these environments are amenable to event-based models, where dissemination must be controlled to ensure that (only) the correct parties receive appropriate information.

The current focus of the National Health Service (England) is on integrating existing health services under a common framework, where confidentiality issues arise when accessing data records. However, to support future healthcare, infrastructure is required to support event-driven intermediate care services.

7.2.2 *Policy-based publish/subscribe data control*

As principals and domains in healthcare are known, and the environment is controlled, it is possible to reason about data-distribution. We have developed a model for controlling information dissemination in event-driven healthcare environments [25, 26]. Our approach is based on interaction control (see Section 6), allowing broker-specific policy to define the conditions in which data is released, i.e. *what* data, to *whom*, and under *what* circumstances. This functionality is realised through the following two operations:

Restrictions defining the circumstances for event delivery to subscribers. Restrictions can control access to events by permitting/denying a subscription request through evaluation of context sensitive conditions. Restrictions can also act as filters to define the circumstances within which notification should occur. Such filters are imposed silently so as not to reveal to the subscriber any (potentially) sensitive data encoded in the restriction.

Transformations customising an event to circumstance.

The above are defined using context-aware conditional clauses, which reference the credentials of principals, system and environmental state. It is through these conditionals that information control policies define access privileges, such as the rights associated with a role (e.g. doctors can access Prescribe events for patients they treat) and the content of data received (e.g. patient monitoring data has the location perturbed, except in situations of emergency).

Figure 4 shows the event flow for a Prescribe event as it moves from the home environment to interested parties. This involves transformation into a Prescription for the Pharmacy and a Controlled_Drug_Auth event for the Auditor, who requires notification but with patient specifics removed. In this scenario, a doctor must treat the patient to which the event instance relates, and a restriction ensures that the auditor only receives notifications for drugs of a particular class. For more detail of this scenario see [25] and [26].

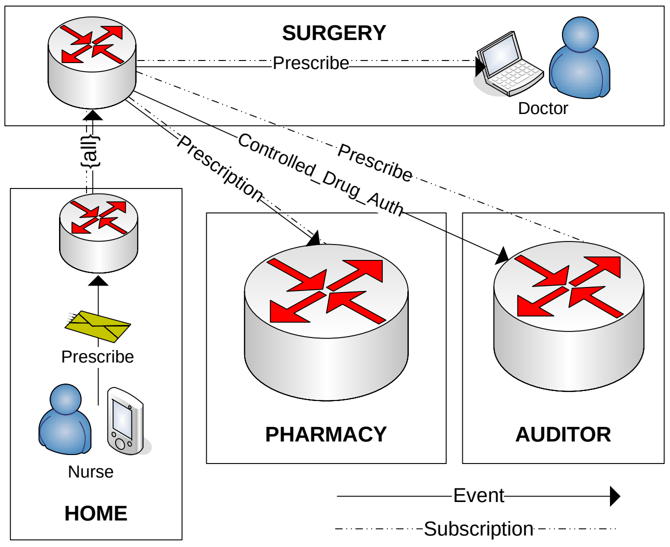

Figure 4: Transformations and restrictions for a Prescribe event as it moves across domains

8. CONCLUSIONS AND OPEN ISSUES

Our system architecture comprises multiple administration domains sharing a dedicated event-broker network. We have found this to be appropriate for many applications. We also assume a secure server per domain that manages credentials and activates roles according to policy. With access control functionality located in the client-hosting brokers, we are able to enforce RBAC on the publish/subscribe clients. In general, separating event-management functionality into a dedicated event service makes access control easier to enforce than in a peer-to-peer approach where the client and event service are colocated. The latter seems inappropriate for applications transmitting sensitive data.

We have assumed content-based routing, for efficiency of communication, rather than broadcast or gossip-based routing. When some brokers are not trusted to see certain sensitive data this style of routing can still be used, with the modifications we describe.

Maintaining the required confidentiality of data depends on those brokers given access to keys continuing to be trustworthy and reliable. For sensitive data that persists long term this may not be a sufficient guarantee. The domain that creates and owns the data may have legal obligations relating to its transmission. Also, it may be that entire domains have reduced requirements on the data, known to the data source. In this case, data transformation can be used to augment the security mechanisms that are used when domains are fully trusted and have an established need for full access to data.

Acknowledgements

We acknowledge the support of the UK Engineering and Physical Sciences Research Council (EPSRC) via grants EDSAC21 (GR/T28164), CareGrid (EP/C53718X) and TIME-EACM (EP/C547632).

9. REFERENCES

[1] Jean Bacon, Alastair Beresford, David Evans, David Ingram, Niki Trigoni, Alexandre Guitton, and Antonios Skordylis. Time: An Open Platform for Capturing, Processing and Delivering Transport-Related Data. In *Proceedings of the Fifth IEEE Consumer Communications and Networking Conference, CCNC,* pages 687–691, Las Vegas, January 2008. IEEE Press. Session on Sensor Networks in Intelligent Transportation Systems.

[2] Jean Bacon, David M. Eyers, Ken Moody, and Lauri I. W. Pesonen. Securing publish/subscribe for multi-domain systems. In *Gustavo Alonso, editor, Middleware, volume 3790 of Lecture Notes in Computer Science,* pages 1–20, Grenoble, France, November 2005.

[3] Jean Bacon, Ken Moody, and Walt Yao. Access control and trust in the use of widely distributed services. In *Middleware '01, IFIP/ACM International Conference on Distributed Systems Platforms*, volume 2218 of LNCS, pages 295–310. Springer, November 2001.

[4] Jean Bacon, Ken Moody, and Walt Yao. A model of OASIS role-based access control and its support for active security. *ACM Transactions on Information and System Security (TISSEC)*, 5(4):492–540, November 2002.

[5] Guruduth Banavar, Marc Kaplan, Kelly Shaw, Robert E. Strom, Daniel C. Sturman, and Wei Tao. Information ow based event distribution middleware. In *Middleware Workshop at the International Conference on Distributed Computing Systems 1999*, 1999.

[6] Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems*, 19(3):332–383, August 2001.

[7] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron. Scribe: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in communications (JSAC)*,

20(8):1489–1499, October 2002.

[8] T. Dierks and C. Allen. The tls protocol version 1.0. *RFC 2246*, January 1999.

[9] Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys*, 35(2):114–131, 2003.

[10] L. Fiege, M. Mezini, G. Muhl, and A. P. Buchmann. Engineering event-based systems with scopes. In *European Conference on Object-Oriented Programming (ECOOP)*, pages 309–333, 2002.

[11] D. A. McGrew and A. T. Sherman. Key establishment in large dynamic groups using one-way function trees. May 1998.

[12] Zoltan Miklos. Towards an access control mechanism for wide-area publish/subscribe systems. In *1st International Workshop on Distributed Event-Based Systems (DEBS'02)*, ICDCS, pages 516–524. IEEE, July 2002.

[13] Ken Moody. Coordinating policy for federated applications. In *4th IFIP WG3 annual working conference on Data and Application Security*, pages 127–134. Kluwer, August 2000.

[14] Lukasz Opyrchal and Atul Prakash. Secure distribution of events in content-based publish subscribe systems. In *10th USENIX Security Symposium*, August 2001.

[15] Lauri I. W. Pesonen and Jean Bacon. Secure Event Types in Content-Based, Multi-domain Publish/Subscribe Systems. In *SEM '05: Proceedings of the 5th international workshop on Software Engineering and Middleware*, pages 98–105, Lisbon, Portugal, September 2005. ACM Press.

[16] Lauri I. W. Pesonen, David M. Eyers, and Jean Bacon. A capabilities-based access control architecture for multi-domain publish/subscribe systems. In *Proceedings of the Symposium on Applications and the Internet (SAINT 2006)*, pages 222–228, Phoenix, AZ, January 2006. IEEE.

[17] Lauri I. W. Pesonen, David M. Eyers, and Jean Bacon. Encryption-Enforced Access Control in Dynamic Multi-Domain Publish/Subscribe Networks. In *Proceedings of the International Conference on Distributed Event-Based Systems (DEBS'07)*, pages 104–115. ACM Press, June 2007.

[18] Peter R. Pietzuch and Jean M. Bacon. Hermes: A distributed event-based middleware architecture. In *1st International Workshop on Distributed Event-Based Systems (DEBS'02)*, ICDCS, pages 611–618. IEEE Press, July 2002.

[19] Peter R. Pietzuch and Jean M. Bacon. Peer-to-peer overlay broker networks in an event-based middleware. In *2nd International Workshop on Distributed Event-Based Systems (DEBS'03)*, ICDCS. ACM SIGMOD, June 2003.

[20] Peter R. Pietzuch and Sumeer Bhola. Congestion Control in a Reliable Scalable Message-Oriented Middleware. In *M. Endler and D. Schmidt, editors, Proc. of the 4th Int. Conf. on Middleware (Middleware '03)*, pages 202–221, Rio de Janeiro, Brazil, June 2003. Springer.

[21] S. Rafaeli and D. Hutchison. A survey of key management for group communication. *ACM Computing Surveys*, 35(3):309–329, 2003.

[22] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *Middleware '01, IFIP/ACM International Conference on Distributed Systems Platforms*, pages 329–350, November 2001.

[23] Ravi Sandhu, Edward Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.

[24] Jatinder Singh, Jean Bacon, and Ken Moody. Dynamic trust domains for secure, private, technology-assisted living. In *Proceedings of the the Second International Conference on Availability, Reliability and Security (ARES'07)*, pages 27–34, Vienna, April 2007. IEEE Computer Society.

[25] Jatinder Singh, Luis Vargas, and Jean Bacon. A Model for Controlling Data Flow in Distributed Healthcare Environments. In *Pervasive Health 2008: Second International Conference on Pervasive Computing Technologies for Healthcare*, Tampere, Finland, January 2008. IEEE Press.

[26] Jatinder Singh, Luis Vargas, Jean Bacon, and Ken Moody. Policy-based information sharing in publish/subscribe middleware. In *IEEE Workshop on Policies for Distributed Systems and Networks (POLICY 2008)*, IBM Palisades, New York, June 2008. IEEE Press.

[27] Anthony Tomasic, Charles Garrod, and Kris Popendorf. Symmetric publish/subscribe via constraint publication. *Technical Report CMU-CS-06-129R*, Carnegie Mellon University, 2006.

[28] Luis Vargas, Jean Bacon, and Ken Moody. Integrating Databases with Publish/Subscribe. In *Proceedings of the 4th International Workshop in Distributed Event-Based Systems (DEBS'05)*, pages 392–397. IEEE Press, June 2005.

[29] C. Wang, A. Carzaniga, D. Evans, and A. Wolf. Security issues and requirements in internet-scale publish-subscribe systems. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02)*, page 303. IEEE, 2002.

[30] Alex Wun and Hans-Arno Jacobsen. A policy management framework for content-based publish/subscribe. In *Middleware '07*, Lecture Notes in Computer Science 4834, pages 368–388. Springer, 2007.