An Approach for Secure Role Assignment

Wolfgang Apolinarski, Marcus Handte, Pedro José Marrón
Networked Embedded Systems Group
University of Duisburg-Essen
Duisburg, Germany
{wolfgang.apolinarski|marcus.handte|pjmarrron}@uni-due.de

***Abstract*—One of the key characteristics of pervasive computing applications is their ability to adapt to their environment. One possible approach to simplify the development of such adaptive applications is to use middleware to automate adaptation decisions. However, in many application scenarios, adaptation decisions can have critical security implications. Thus, when designing middleware to support adaptation it is necessary to take security constraints into account. In previous work, we presented generic role assignment [1] as a middleware abstraction to support adaptation. In contrast to other approaches, role assignment enables adaptation within a single and across multiple smart environments in a flexible, application-overarching manner. In this paper, we extend the concept to support security constraints. Furthermore, we describe the necessary modifications to our existing role assignment system. Finally, we evaluate our approach showing resulting benefits and limitations. In comparison with existing approaches, the proposed approach is more flexible and does not introduce a single point of trust.**

***Keywords*-Role Assignment, Security, Access Control**

I. Introduction

The vision of pervasive computing is to provide the user with seamless and distraction-free task support. This task support is realized by pervasive applications that are running on devices integrated into everyday objects. Using sensors, they are able to perceive information about their physical context and using actuators, they may manipulate parts of it. Due to their integration, the devices are heterogeneous and may exhibit mobility. These factors lead to a dynamic environment. Because of this, applications need to adapt to the environment continuously. If the environment changes frequently, a manual adaptation by the user is not feasible as this might cause significant distraction. Therefore, it is necessary to automate the adaptation. Implementing this automation at the application layer is a complicated and error-prone process and leads to a considerable development overhead.

To avoid this, it is possible to implement the automation at the middleware layer. An application can then specify its adaptation requirements to the middleware. The middleware automates the adaptation decisions and may even execute the resulting changes. In many application scenarios, adaptation decisions can have critical security implications. Consider an application for document editing that automatically migrates its output from the small screen of the user's mobile phone to a larger screen available in the environment. If the screen has been set-up by a malicious person, it is possible to copy the displayed documents. This problem is amplified, if a service in a remote environment is mediated over the Internet. As a result, it is necessary to consider security constraints when designing a middleware that automates adaptation.

In previous work [1], we presented generic role assignment as a lightweight but flexible abstraction to automate adaptation at the middleware layer. Role assignment assumes that devices have some knowledge about their context and are able to perceive parts of it at runtime. Examples for the former may be the device type or owner. Examples for the latter might be the device's GPS coordinate or the surrounding temperature. In generic role assignment the middleware automatically assigns so-called roles based on this context. A role can be seen as a tag that may be assigned to one or more devices. By definition, a role is assigned to any device as long as there are no further constraints. To enable the use of roles in different applications, role assignment introduces so-called rules that define contextual constraints on devices. Rules can be categorized into filter and reference rules. A filter rule simply constraints the set of devices to those that exhibit a particular context. A reference rule references other assignments which enables hierarchical composition. To express a set of requirements, a developer can define combinations of roles and rules within a so-called role specification. Since the roles are just tags that are assigned to devices, they can be used at both, the application as well as the middleware layer. At the middleware layer, roles may be used, for example, to adapt the set of devices that interact with each other. An example use case at the application layer is to use roles to adapt the configuration of a distributed application.

In this paper, we extend generic role assignment to support security constraints. To do this, we first define a simple yet flexible model to capture the trust relationships between different devices. On the basis of this, we extend role assignment with an additional class of rules to model security requirements and we discuss the necessary changes to enable their automatic enforcement at runtime. As a validation, we describe a prototypical implementation of a secure role assignment system and we describe several system services that we built on top of it. Finally, in order to evaluate the approach, we determine its performance impact experimentally and we discuss experiences from implementing several security critical applications. The evaluation indicates that secure role assignment is flexible enough to support a broad range of services and applications without introducing high performance overheads.

The structure of the remainder is as follows. In the next section, we briefly outline our design goals for secure role assignment. Thereafter, in Section III, we describe our approach. In Section IV, we describe our prototypical implementation which we evaluate in Section V. Finally, we describe related work in Section VI and we conclude the paper in Section VII.

II. Design Goals

Our primary goal for secure role assignment is the ability to use its resulting role assignments as basis to develop security critical pervasive computing applications. Based on this, we can derive the following three design goals.

- **Configurable:** To support a broad range of applications, the role assignment must be configurable. The same holds true for the concepts required to realize secure role assignment. Specifically, it is necessary to enable the developer to specify security requirements at different levels of abstraction. For example, a developer might want to directly limit role assignment to a set of devices that is known at development time. For other applications, a developer might want to perform role assignment only with the set of devices that are trustworthy from a user's perspective. Furthermore, since not all adaptation decisions must be secured, the security should be optional.
- **Light-weight:** The devices in the pervasive computing domain are often heterogeneous and may include resource-constrained devices like sensors. To be able to use such devices, secure role assignment should be light-weight. In cases where security is not needed, its security mechanisms should not introduce additional costs. Furthermore, if security is required the resulting overheads should not introduce high performance penalties.
- **Secure:** Since the decisions made during role assignment may have critical security implications, the security provided by role assignment must be high. Despite the fact that devices can be resource-constrained, malicious persons may have a significant amount of resources at their disposal. This is especially true, if role assignment is performed across an insecure network such as the Internet. Consequently, the concepts used to secure role assignment should allow the use of cryptographic methods that exhibit Internet-level strength.

III. Approach

In the following, we describe our approach to realize the design goals described previously. To do this, we first provide an overview of the basic idea. Thereafter, we discuss the individual concepts in detail.

A. Overview

The basic idea of our approach is to secure the input data that is used during role assignment. Since generic role assignment introduces two classes of rules, this affects two types of data. For filter rules, it is necessary to ensure that the context used during their evaluation corresponds to the actual situation. From a systems perspective[1], this can be achieved by ensuring that the situation has been observed by a trustworthy sensor[2] and that the observation has not been altered. Furthermore, since the context may change over time, it may also be necessary to ensure that the observation has been made recently. For reference rules, it is necessary to ensure that the roles used during reference resolution have been assigned properly. To do this, it is necessary to ensure that the role has been assigned by the device responsible for a particular role specification and that the role has not been altered. As role assignment is a dynamic process, it may also be necessary to ensure that the role is not outdated. Thus, we need to secure the authenticity and integrity of context and roles and optionally, we also need to ensure their freshness.

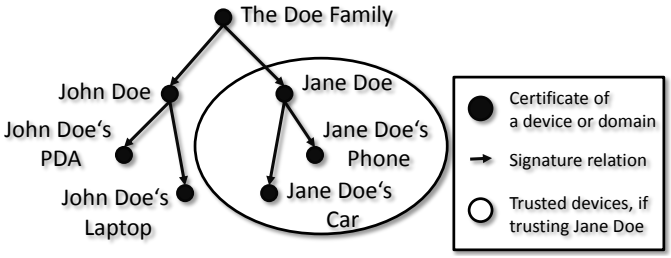

Fig. 1. Certificate Hierarchy and Trust Model Example

As a first step, it is necessary to define the set of trustworthy devices. To do this, we introduce a simple trust model that is based on certificate hierarchies. In the model, a certificate is attached to every device. The certificates are used to identify devices and capture the devices' memberships in different administrative domains. As a second step, we then extend the role specification with an additional class of so-called security rules which define the set of devices or domains regarded as trustworthy. In order to enable user-specific definitions of trust, we introduce trust-levels that can be defined for each device. The levels can be referenced in an abstract manner within the security rules. At runtime, the levels are then automatically resolved into the proper set of device-specific certificates. Due to the fact that the security rules cause additional effort during the role assignment process, the rules can be attached to individual filter or reference rules. Besides from reducing the overhead, this approach also increases the configurability. As a third and last step, we then ensure the enforcement of the security rules during the evaluation of the corresponding filter and reference rules which realizes the design goal of security.

B. Trust Model

As a first step, it is necessary to define the set of trustworthy devices. Trust can be modeled in many different ways [2]–[4]. For our approach, we chose to base our trust model on the widely used concept of certificate hierarchies [5] depicted

[1] We are aware of the fact that there are other possibilities. For example, one might argue that a correspondency is given, if n arbitrary sensors make similar observations. However, such definitions are orthogonal to our approach.

[2] If certain facts are not observed directly but modeled by a human (e.g. device ownership, etc.), the modeling person must be trustworthy.

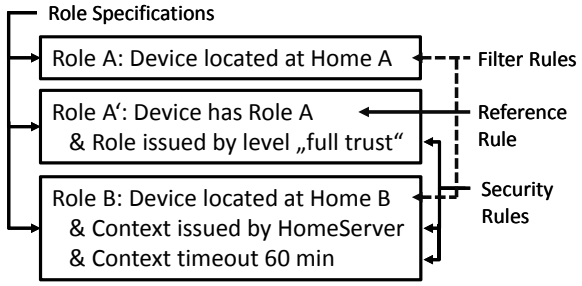

Fig. 2. Role Specification with Security Rules Example

in Figure 1. The reason for this is twofold. First, certificate hierarchies exhibit both, scalability at deployment as well as at runtime. By means of hierarchical relationships, devices can be grouped into an arbitrary number of administration domains. Furthermore, it is possible to add new devices to established domains at any point in time (i.e. by signing a device's certificate with the corresponding domain certificate). At runtime, it is then possible to automatically detect and verify a device's membership in a decentralized manner (i.e. by validating the signatures of a known sub-tree). Second, as a side-effect of using certificates, it is possible to generate a unique, cryptographically strong device identifier (e.g. by using the certificate's fingerprint as device id).

Using the certificate hierarchies, we enable the user to assign different levels of trust to one or more certificates. An example for this is depicted in Figure 1. If the certificate represents an administrative domain, intuitively, the trust recursively expands to all devices that it contains. For example, in Figure 1, by putting a particular level of trust in the certificate that represents Jane Doe, the trust expands to all of Jane Doe's devices. In order to reduce the modelling effort, we order the trust levels and assume transitivity such that a lower level includes the higher levels. Consequently, the two extremes are full trust and no trust. Between those two levels, we allow an arbitrary number of additional levels with an application-defined meaning. Additionally, applications can modify the trust level of certificates based on their needs.

C. Security Rules

To secure the input data that is used during role assignment, we extend the role specification by introducing a new class of rules, security rules. Security rules enable the specification of two types of security requirements. They allow the specification of a minimum amount of trust that is required to consider a data source during role assignment. This can be done either directly by referencing devices or administration domains by the means of their certificates. Alternatively, it is possible to indirectly reference devices by requiring a particular trust level. Second and in addition to trust, the security rules enable the specification of a maximum age of data.

The security rules can then be assigned to filter and reference rules in a fine-granular manner. An example for the resulting role specifications is depicted in Figure 2. The figure shows three role specifications. The first one assigns a role *Role A* to a device whose location is at *Home A*. The second specification references *Role A* in the previous specification, but in addition it requires that the role ought to be assigned by a device that is classified as *full trust*. Consequently, this role will only be applied to devices that are equipped with a certificate that exhibits the trust level *full trust*. Note that due to the trust model, this may apply to both, devices that have been identified as such directly as well as devices that exhibit a certificate whose signatures are rooted in a *full trust* certificate. The third example specification defines *Role B* which is only assigned to devices that are located at *Home B*. Using security rules, the distribution of this role is restricted to devices whose location has been observed by a device called *HomeServer* and this observation has been made at most 60 minutes ago.

D. Enforcement

After having defined the trust model and the security rules, the third step and final step is to ensure the security enforcement during role assignment. Due to the fact that the security rules can be added to both, filter rules and reference rules, it is necessary to enable the enforcement for context as well as roles. In the following, we briefly describe how the enforcement can be realized for each type of data.

1) Context: To enable the enforcement with respect to context we must be able to ensure the authenticity of the source. Furthermore, we must be able to ensure the integrity of the context and we must be able to determine its age. To do this, we apply the *generator*, *storage* and *validator* principle that we developed previously. For the sake of brevity, we briefly discuss the problem mapping in the following and outline the basic idea. The detailed protocol descriptions can be found in [6]. For our application scenario, *generators* are mapped to devices that gather context by means of their built-in sensors. *Storages* are mapped to devices that store this context and make it accessible for role assignment[3]. *Validators* are mapped to devices that evaluate role specifications and perform the role assignment. Using the update and validation protocols described in [6], validators can verify the authenticity, integrity and freshness of the context gathered by generators, even if the storage device is malicious. To do this, the protocols utilize cryptographic signatures as well as some additional meta information that is attached to the context. Note that in order to support resource-constrained devices, the protocols do not require asymmetric cryptography on all devices. Instead, they can also apply hash-based message authentication codes as a light-weight replacement on resource-poor devices.

By applying this mapping, validators – that is devices evaluating a role specification – can directly validate the authenticity, integrity and freshness of the context supplied by the (storage) devices which they test. They must compare the requirements defined in the role specification with the context's meta information and they must validate the cryptographic signatures. Due to the use of certificate hierarchies, this might require the additional exchange of the certificate

[3]Note that generators and storages can be represented by the same device.

chain of the generator. Towards this end, the generator has to be modified in such a way that it attaches the chain as part of the update protocol – which transmits context from the generator to the storage.

2) Roles: The role assignment process is described in detail in [1], here we focus on the security requirements. To enforce the security requirements with respect to roles, there are two possible options. First, it is possible to apply the principles applied to context analogously. To do this, assigned roles must be extended with meta information by the source device that assigns a role to a target device. Just like with context, this meta information must include a time stamp, the source and the target device identifiers as well as the certificate chain of the source device. The result can then be protected against manipulation using a cryptographic signature before it is transferred to the target device. When a reference rule is extended with a security rule, any device can then validate the role assignment by the following process. If the security role requires a certain freshness, the age of the assignment is validated by checking the time stamp. Now, it is necessary to validate whether the source device exhibits (at least) the required trust level. So, the source identifier and the identifiers contained in its certificate chain must be matched against the certificates that exhibit (at least) the required trust level. If this is the case, the signatures along the certificate chain as well as the signature of the role need to be validated. As an alternative, it is also possible to enforce the security requirements by retrieving the role and the meta data directly from the source device through a secure connection. This alleviates the need to cryptographically sign the role. To establish such a connection, it is possible to use the devices' certificates to guarantee authenticity and to generate a shared symmetric key which can then be used to guarantee integrity as well as secrecy.

Both approaches have advantages and disadvantages. The main advantage of the first possibility is that it does not require a direct connection between the source device that assigns a role and the device that aims at validating the role. However, in order to support validation, it is either necessary to use comparatively heavyweight asymmetric cryptography or to have a shared symmetric key which must be distributed somehow. For the former, the resulting overhead can be high, especially if many roles must be validated or if roles need to be validated periodically. In such cases, it is more efficient to opt for a secure connection. The additional effort for connection establishment can then be easily amortized over time. Since both approaches can be beneficial, depending on their usage, we propose to apply both of them on a case-by-case basis.

IV. Implementation

To validate our approach, we have extended the implementation of our basic role assignment system presented in [1] with the security related concepts and mechanisms described previously. Furthermore, we have used it to develop a number of additional security critical middleware services. The resulting implementation is available as open source under BSD license[4]. In the following, we briefly describe the necessary architectural changes. Thereafter, we outline the interaction of the components. Finally, we describe a number of middleware services that we have built on top of secure role assignment.

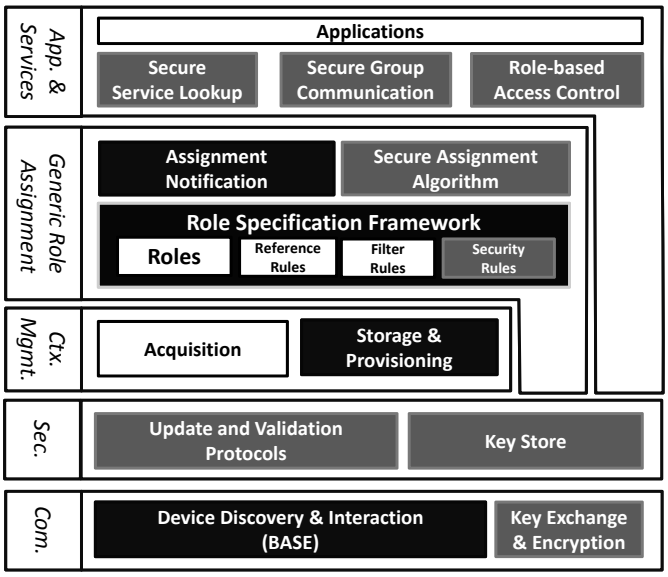

Fig. 3. Secure Role Assignment System Architecture

A. Architecture

As described in [1], for our original implementation of generic role assignment, we used the BASE [7] middleware to enable spontaneous interaction between devices. On top of that, we implemented a context storage that stores arbitrary context in the form of RDF triples that can be structured using OWL ontologies. The storage makes the local context available to other devices. A role assignment framework enables developers to define role specifications, which consist of Boolean expressions of SPARQL queries (for filter rules) and references to roles of other specifications (for reference rules). A generic assignment service implements the algorithm to automatically assign roles. Roles are assigned by executing queries and resolving the references to other assignment services. When the assignment changes, the devices that receive or loose a role are notified. On top of that, middleware services and applications may use the assignment to adapt their behavior.

In order to implement the concepts described in Section III, we extend the basic architecture with a number of additional components. The resulting architecture is depicted in Figure 3. The previous components are marked in black whereas the new components are shaded gray. At the communication layer, we extend the architecture with two additional BASE plugins that provide secure connections using symmetric cryptography (via AES and HMAC) and enable different variants of the Diffie-Hellman key exchange (i.e. DH and ECDH). To implement the trust model and to manage pre-shared symmetric as well as asymmetric keys and certificates (i.e. RSA and ECC), we

[4]Documentation and code are accessible via http://peces.googlecode.com.

add a key store component. Besides from managing pre-deployed keys and certificates, the key store is also able to cache symmetric keys (and verified certificate chains) that are established (and verified) using the key exchange plug-ins. Furthermore, we add a component that is capable of signing and validating context information and roles using the update and validation protocols described in [6]. To define security requirements, we extend the role specification framework with the security rules described in Section III-C.

B. Interaction

To clarify the architecture described previously, we provide an overview of their interaction in the following. In order to support secure role assignment, each device must be configured with an asymmetric key pair as well as the chain of certificates that define its membership in different domains. Optionally, it may also be equipped with a number of pre-shared keys. In addition, if a device wants to start secure role assignments, it also needs to know the certificates of domains that are trustworthy for the application. These can either be specified directly as part of a role specification, or they can be stored in the key store with their associated trust level - which is then resolved at runtime. Intuitively, this trust configuration does not have to be performed by the developer but could also be done by an administrator or an advanced user.

Once a role specification is injected into the assignment service, the algorithm first interprets the security requirements of the specification. If a filter rule is associated with a security rule, the query evaluation for the filter rule cannot be executed on the context storage of the remote device – as the remote device could easily send an arbitrary response. Instead, it must be done locally on the device hosting the assignment service and it should only be done using context from a trustworthy source. In order to do that in an efficient manner, the assignment service sends the query to the remote device. Using the query, the context storage of the remote device determines and transmits the subset of RDF triples that is required to produce the query result. Once the triples are received, the assignment service validates them as described in Section III-D1. Thereafter, it locally executes the query on the subset of triples that meet the security requirements. Although, this approach requires a duplicate query execution – on the remote device as well as on the device hosting the assignment service – the filtering of RDF triples can often significantly reduce the amount of communication, especially in cases where devices store a lot of context.

If a reference rule is associated with a security rule, the assignment service first determines the device responsible for the assignment as described in [1]. Using the key store, it first checks whether the two devices already share a symmetric key. If this is not the case, the key exchange plug-in is used to establish a symmetric key. Using either the cached or the newly established key, the device then creates a secure connection to the device. Using the secure connection, the assignment service can then retrieve roles and validate meta information directly.

Once the role assignment has been computed, the role assignment service distributes the roles by informing the necessary devices through their notification component. Depending on the usage of the roles, this may either entail the creation of a signature – in cases where the role must be validated without establishing a secure connection – or it may entail the establishment of a secure connection to the receiving device. As discussed next, from an efficiency perspective, either one might be favorable depending on the use case.

C. Integration

Since roles are just generic tags, they can be used as a basis to implement further adaptation support. In order to test and validate our implementation, we have extended some of the services described in [1] to enable their secure use. Furthermore, we have developed additional security critical middleware services on top of secure role assignment. In the following, we briefly discuss their integration with the secure role assignment system described above.

At first, we extended the role-based service registry as well as the role-based group communication service described in [1]. The group communication service implements group communication using roles (i.e. similar to multicast) whereby groups are identified by a particular role. To use this service in a secure manner, we added optional encryption using a group key. A developer can define security requirements on the group by defining a role with security rules. The role assignment system then automatically enforces them and ensures that only appropriate devices will receive the role. Before the roles are distributed, we first create a symmetric group key which we then distribute together with the role. To avoid overhearing, the role is transmitted to each device using a pair-wise encrypted channel. If devices leave the group, the role is updated. Once a role is received or updated, the new key is used for encryption. This ensures that only group members are actually able to participate in the communication.

The role-based service registry, uses a slightly different approach. Similar to the secure group communication service, the role assignment can be secured by the application developer using security rules. Instead of using a secure transfer, they are signed by the assigning device. Thus, when a service search is executed, the devices that should retrieve the search query can be evaluated in a scalable, decentralized manner, without requiring interaction with the device that assigned the roles.

Besides from these extensions, we also developed new services such as the role-based access control service. Here, we apply a similar approach as for the role-based service registry. Instead of using asymmetric signatures, we use HMAC with a symmetric key. The resulting interaction is depicted in Figure 4. To secure a service with role-based access control, the developer first specifies roles that represent a client's possible access rights. To ensure that the roles are assigned properly, the developer secures them with security rules.

When the service is started, the system injects the role specification into an assignment algorithm – either on the local device or a trustworthy device in the vicinity. To ensure that

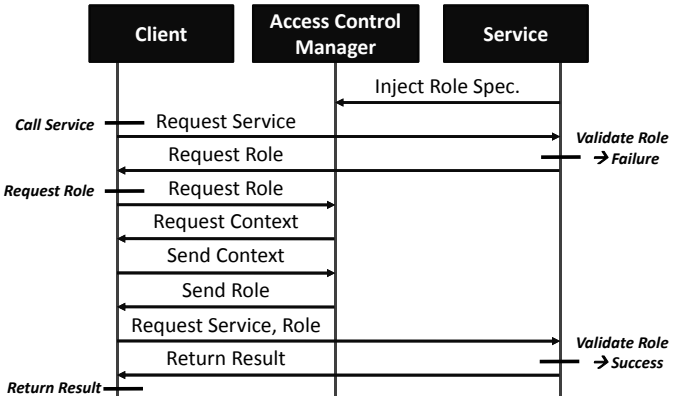

Fig. 4. Role-based Access Control Example

the role specification is not altered, this is done via a secure connection which will establish a key between the device hosting the service and the device performing access control. When a client attempts to access the service, the service first determines whether the client request contains the necessary role. If the request does not contain this role, the system generates an exception which is returned to the client. The exception notifies the client that the service requires a particular role that is assigned by a particular device responsible for access control. In order to get this role, the client contacts the access control manager and requests it. Then the access control manager performs the normal sequence of secure role assignment with this device. This involves sending the query for filter rules, resolving references, validating signatures, etc. Once this process has completed, the access control manager signs the roles using HMAC and the symmetric key of the device that hosts the service. Then it forwards the role to the client. Thereafter, the client attaches the roles and repeats the request. The service can then validate the role by validating the HMAC using the key shared with the access control manager. If the validation succeeds, the service is executed and the result is returned. For subsequent calls, the client simply reattaches the role to the request as long as it is fresh enough. Once the role is expired, the procedure repeats from the beginning. Although, this might look like a fairly complicated procedure, our implementation of role-based access control hides most of the details in automatically generated stubs and skeletons. The only interactions that are visible to the client developer are the initial call and the failure handling in cases where a service call fails due to a lack of privileges. Similarly, using the trust model, the skeleton can hide the selection of a device that is suitable to perform access control. With the exception of the definition of a role specification and the API calls to validate the roles, secure role assignment is completely transparent.

V. Evaluation

To evaluate our secure role assignment system and the associated services, we have used it in the PECES European research project to develop several prototype applications for various scenarios including traffic management and e-health. In the following, we briefly describe our experiences from implementing these applications and relate them to the design goals of creating a *configurable* and *secure* abstraction. Thereafter, we present a number of experiments that show the resulting overhead and we discuss the results in relation to the design goal of creating a *light-weight* abstraction.

A. Experiences

In most practical applications, ensuring security is a mandatory requirement. As an example consider the following excerpt from our traffic management application: The traffic management application automatically prepares trips on behalf of the user. When a user enters his car, his smart phone automatically extracts the destination from the user's calendar and uses it to configure the navigation system. The navigation system and the user's smart phone apply role assignment to limit the access to the user's personal data and service. If applicable, the phone automatically pays road tolls and it is also able to reserve parking lots. To do this, the phone determines a suitable parking lot using the secure service registry. This ensures that only trustworthy parking lots are used. When reserving a toll road or a parking lot, the phone transmits the car's license plate which is then used for automated billing. While the car is on the road, the user may subscribe to the latest commercial traffic announcements which are then distributed to all subscribed cars using group communication. Preventing illegitimate access, the data provider uses secure role-based group communication which ensures that only paying customers are able to receive the traffic announcements.

As an example on how to involve security rules here, consider the access limitation to the user's personal data. The limitation consists of a role that allows access and checks the following rules: A filter rule *Device is allowed access* and a security rule *Context issued by a fully trusted device*. This role specification is then injected into the user's smart phone and the access control is managed as pictured in Fig. 4.

Although, the traffic management scenario is fairly complicated and exhibits a number of different entities with varying security goals, secure role assignment provides the basis for realizing all of them. The same holds true for the e-health application as it automatically manages the flow of medical data between patients, nurses and doctors. Clearly, as role assignment is just a basic abstraction for automatic adaptation, the application of role assignment requires the provisioning of additional services on top of it. The fact that we were able to implement a range of security critical middleware services on top of it can be seen as an indicator for fulfilling the design goals of having a flexibly *configurable* and *secure* abstraction.

B. Experiments

During the development of the applications in PECES, we have used the secure role assignment system on a broad variety of devices. This includes both, resource-rich devices such as traditional servers, laptops and tablets as well as resource-constrained systems such as phones, wifi routers (running

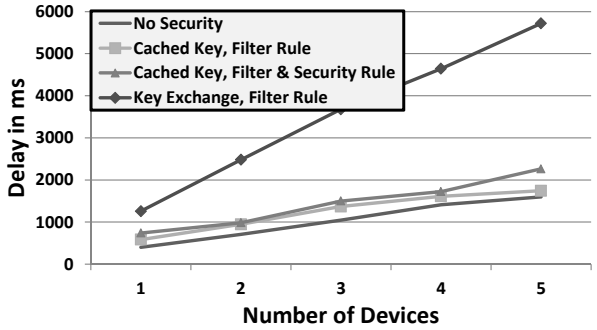

Fig. 5. Secure Role Assignment Delay

OpenWRT) and sensor nodes (i.e. SunSPOT). During the development and testing, we found that the resulting application performance was not impacted in a dramatical way. However, in order to experimentally determine the overheads of adding security to generic role assignment, we performed a number of laboratory measurements in which we systematically enable or disable various security features.

For the experiments, we used a Lenovo T61 laptop (Intel T5670 1.8 GHz Dual-Core CPU, 3 GB RAM) running Windows 7 and the Oracle Java JRE 1.7 to perform all secure role assignments. The laptop is connected to a varying number of HTC Tattoo devices (Qualcomm MSM7225 528 MHz CPU, 256MB RAM) that are running Android 1.6. On each device, an individual 160-bit ECC certificate is pre-deployed. The security provided by 160-bit ECC is at least as high as 1024-bit RSA, a standard that is widely used in the Internet. For the wireless connection, we used a dedicated IEEE 802.11g network which was created by a Netgear WNR3500L. All measurements show the average delay of 100 repetitions.

As a baseline (*No Security*), we measured the total delay of assigning a single role to a varying number of HTC Tattoos. The resulting delay is depicted in Figure 5. The time shown is the total time that passes between starting a role specification on the laptop until all devices have received the role. For 5 HTC Tattoo devices, the total time stays well below 1.6 seconds. The second (*Cached Key, Filter Rule*) and third cases (*Cached Key, Filter Security Rule*), show the overhead for role assignment if the resulting roles shall be transmitted securely to their receivers but the key to establish a secure connection has been cached already. As can be seen from the measurements, the additional encryption (128-Bit AES) adds a small overhead to the overall process. This overhead is increased by the addition of a filter rule, as this requires the additional validation of an (asymmetric) signature on the laptop. With at most 2.3 seconds for role assignment with 5 devices the total overhead remains low. The last measurement (*Key Exchange, Filter Rule*) is identical to the second case, but in addition, it requires the exchange of a key using ECDH. As depicted in Figure 5, the key exchange drastically increases the overhead to 5.8 seconds for 5 devices due to the high computational effort for ECDH. However, due to the fact that the key store can cache the resulting symmetric key between different interaction, this overhead is only experienced once. With an approximate overhead of less than 1 second per device for the key exchange, we are argue that role assignment is light-weight and can be applied in many scenarios.

VI. Related Work

Many pervasive middleware systems make extensive use of context or role assignment, but often security mechanisms are not considered, implemented or marked as future work. Sometimes, the security mechanisms are described as important for the future work [8] or security is only mentioned without providing any further details about the implementation [9]. Focusing more on privacy-awareness, Ni et al. [10] describe an extensive framework to model role-based access control. It allows to assign roles based on hierarchical and conditional constraints, similar to our approach. However, they are not considering how roles or context are being secured if they are exchanged over an insecure communication channel.

In our previous work [6], we described an approach for a peer-based secure context distribution framework. The framework secures context gathered from heterogeneous devices with built-in sensors. The devices communicate on a peer-to-peer basis, they are not connected to or rely on any kind of infrastructure. We used this work as a basis for the secure role assignment system presented here. In contrast to the secure context distribution framework, which distributes context actively, our system provides support for active and re-active role assignment. Additionally, the role assignment system enables automatic key exchanges and therefore secure communication based on symmetric cryptography for a resource-efficient communication, which is not covered by [6].

Many approaches for security systems in the pervasive computing domain introduce a central server responsible for security. This contradicts our view of an environment which consists of decentralized applications and services. Therefore, a centralized approach for security is not suitable. Although we support re-active secure role assignment with the possibility to externalize secure role assignment to another device in the environment, there is no central security service. This stands in contrast to GAIA [11] or Vigil [12] which use central services or agents. Similarly to our approach, Vigil uses certificates for devices in the environment. Their environments (SmartSpaces) each consist of a Service Manager that announces public services and is used by devices to register with the environment. All Service Managers are organized in a tree-structure and have trust relationships established with each other. Similarly, each SmartSpace has a Security Agent that is responsible for the secure service access in the environment. Our approach does not need central authorities or devices, instead, every device can validate certificates, roles and context on behalf of another device, if they have previously agreed on working together. GAIA uses central servers for access control and for data storage. All data is stored (encrypted) on a central server that verifies certain requirements (e.g. context), before the data can be accessed by a device. Using our security system, the data is stored on each device individually, also the

access to services can be controlled by the device executing the service itself. This results in a decentralized environment where central authorities are not needed.

Our description of trust in devices is, in contrast to Lagesse et al. [2], not reputation-based. Also, trust values are stored on each device individually and, beside the certificate tree-structure, there is no trust group formed. Reputation-based systems [3], [4] can be misused by attackers. If attackers use a device that behaves well until it reaches a high reputation level, it is possible to modify the trust relationships to their needs. A certificate hierarchies avoids these kind of attacks, because a valid certificate is needed before trusted interactions are possible. Nevertheless, a modification of our trust model could add reputation-based trust to the currently implemented certificate-based trust model. Another approach to model trust relationships is shown by Takabi et al. [13]. Here, fuzzy relation equations are used to describe the trust in users. Roles can require a particular trust level, before they are assigned. In our system, the secure role assignment may have additional constraints about the context's source and freshness, but not about the trust in the role's target devices. This enhances the seamless transition between environments. Consider a device which is currently located in a remote environment. It may still access services there, if it possesses the necessary context. Instead of adding a different parameter (i.e. trust) to roles, we allow trust constraints to be modeled with regard to context.

The key exchange component that we provide in our secure role assignment system, could be extended with different approaches. In SPATE [14], small groups of users can exchange a common secret (like a key) by comparing hash codes (T-Flags). This enables SPATE to be independent of a certificate hierarchy at additional costs, i.e. all users have to recognize and compare images and all devices need a display showing the image. Additionally, SPATE has a more complicated setup which requires the devices to scan a bar-code before the key exchange process starts (requiring a built-in camera on each device). Although currently not supported, this approach could be modified and used to extend our key exchange mechanism. Mathur et al. [15] describes a key exchange protocol that retrieves a shared key from RF signals. To obtain a key, it is necessary to put the pairing devices into physical proximity. According to the authors, the protocol is resistant to attackers that are more than 6.2 cm away (at 2.4 GHz). The protocol could be used to establish a pre-shared key between devices replacing our current key exchange. Nevertheless, this is only successful if devices are in direct vicinity. Our approach therefore uses a different key exchange to enable the interaction between devices that are located at different places.

VII. Conclusion

In many application scenarios, adaptation decisions can have critical security implications. If the adaptation decisions are automated using generic role assignment, the role assignment must be secured. Here, we presented secure role assignment as a secure extension to generic role assignment and we described a prototypical implementation of a secure role assignment. Using this implementation, we showed that the system can provide configurable and secure support to enable the use of generic role assignment in security critical applications. As indicated by our measurements, given a suitable combination of cryptographic methods, the resulting system is light-weight enough to be applicable to a broad range of devices. In the future, we plan on extending our secure role assignment system to support supplemental models of trust. Specifically, we focus on extending our trust model with a reputation-based system to add more flexibility to trust-based decisions.

Acknowledgments

This work has been partially supported by CONET (Cooperating Objects Network of Excellence), PECES (PErvasive Computing in Embedded Systems) and GAMBAS (Generic Adaptive Middleware for Behavior-driven Autonomous Services), all funded by the European Commission under FP7 with contract numbers FP7-2007-2-224053, FP7-224342-ICT-2007-2 and FP7-2011-7-287661.

References

[1] W. Apolinarski, M. Handte, and P. Marron, "Supporting environment configuration with generic role assignment," in *Intelligent Environments (IE), 2011 7th International Conference on*, july 2011, pp. 1 –8.

[2] B. Lagesse, M. Kumar, J. M. Paluska, and M. Wright, "Dtt: A distributed trust toolkit for pervasive systems," in *Proc. of the 2009 IEEE Int. Conf. on Perv. Comp. and Communications*. IEEE Computer Society, 2009.

[3] K. Krukow, M. Nielsen, and V. Sassone, "A framework for concrete reputation-systems with applications to history-based access control," in *Proc. of the 12th ACM conf. on Comp. and com. security*, ser. CCS '05. New York: ACM, 2005.

[4] G. D. M. Serugendo, "Trust as an interaction mechanism for self-organising systems," in *Int. Conf. on Complex Systems (ICCS)*, 2004.

[5] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," RFC 5280, IETF, May 2008.

[6] W. Apolinarski, M. Handte, and P. J. Marrón, "A secure context distribution framework for peer-based pervasive systems," in *PerWare Workshop at the 8th Annual IEEE PerCom*, March 2010.

[7] M. Handte, C. Becker, and G. Schiele, "Experiences - extensibility and minimalism in BASE," in *Workshop on Sys. Support for Ubi. Comp. (UbiSys) at Ubicomp*, 2003.

[8] H. Schmidt, F. Flerlage, and F. Hauck, "A generic context service for ubiquitous environments," in *IEEE Int. Conf. on Pervasive Computing and Communications*, 2009.

[9] S. Kang, J. Lee, H. Jang, H. Lee, Y. Lee, S. Park, T. Park, and J. Song, "Seemon: scalable and energy-efficient context monitoring framework for sensor-rich mobile environments," in *Proc. of the 6th int. conf. on Mobile sys, app., and services*, ser. MobiSys '08. ACM, 2008.

[10] Q. Ni, E. Bertino, J. Lobo, C. Brodie, C.-M. Karat, J. Karat, and A. Trombeta, "Privacy-aware role-based access control," *ACM Trans. Inf. Syst. Secur.*, 2010.

[11] J. Al-Muhtadi, R. Hill, R. Campbell, and M. Mickunas, "Context and location-aware encryption for pervasive computing environments," in *4th Annual IEEE Int. Conf. on Perv. Comp. and Com. Workshops*, 2006.

[12] L. Kagal, J. Undercoffer, F. Perich, A. Joshi, and T. Finin, "A security architecture based on trust management for pervasive computing systems." in *G. Hopper Celebration of Women in Com.*, October 2002.

[13] H. Takabi, M. Amini, and R. Jalili, "Trust-based user-role assignment in role-based access control," in *IEEE/ACS Int. Conf. on Com. Sys. and Applications (AICCSA '07)*, 2007.

[14] Y.-H. Lin, A. Studer, H.-C. Hsiao, J. M. McCune *et al.*, "Spate: small-group pki-less authenticated trust establishment," in *Proc. of the 7th int. conf. on Mobile sys., app., and services*, ser. MobiSys '09. ACM, 2009.

[15] S. Mathur, R. Miller, A. Varshavsky, W. Trappe, and N. Mandayam, "Proximate: proximity-based secure pairing using ambient wireless signals," in *Proc. of the 9th int. conf. on Mobile sys., app., and services*, ser. MobiSys '11. ACM, 2011.