Trusted Service Discovery through Identity Management

Anders Fongen and Trude Hafsøe Bloebaum
Norwegian Defence Research Establishment (FFI)
Emails: {anders.fongen,trude-hafsoe.bloebaum}@ffi.no

***Abstract*—Service oriented environments face threats from unauthorized clients and fake or compromised services. The threats exist both during service discovery and service invocation, and should be mitigated through the same security framework. Through the use of a modern identity management system which offers a combination of key attestation and attributes for access control, more threats can be appropriately addressed. The combination of discovery and identity management results in a more comprehensive threat mitigation, scalable maintenance of security related information and easier federations of security domains. The architecture and protocols of this system combination are presented and discussed.**

Keywords-Identity management, Authentication, Integrity, Service Discovery

I. Introduction

The term *Service Discovery* (SD) refers to the broking process through which service clients may learn about relevant service providers. In service-oriented systems, having a reliable SD capability is essential in order to achieve loose coupling between providers and clients, particularly in dynamic environments. This makes SD an integral part of the Service-Oriented Architecture (SOA) concept. The SD mechanism is responsible for providing potential clients with information about the *identity* and the *location* of a service. In short, this means that the SD must be able to provide a service description and a service endpoint when requested.

SD can be implemented either as a directory-based registry, or as a distributed mechanism. No matter how the mechanism is implemented, it must support the communication illustrated in Figure 1. Service providers must be able to publish their services by providing the SD mechanism with the endpoint and the description of their services (step 1), clients must be able to lookup services (step 2), and be able to bind to a service using the information provided by the SD (step 3). Because the SD acts as a directory in which clients can look up services, we will throughout this paper refer to this mechanism as the *Directory Service* (DS).

SD involves the use of *selection criteria* which are used to filter the set of services presented to the client. If SD is performed using a pull based mechanism, where the client actively queries for services, the selection criteria is used to limit the set of services included in the query response. For push based interaction the services announced to the client are either selected on the service side, or on the client side.

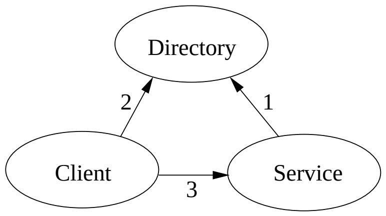

Fig. 1. The SOA triangle of client, service and directory

In both cases the selection is based on a filtering expression decided by the client.

For a man-in-the-middle attack in a SOA system, the DS is an attractive attack vector. An attacker can misguide a client into the use of a fraudulent service if the DS is compromised. This would be analogous to what is known as *phishing attacks* in a web environment.

Security requirements of an SD system are straightforward and intuitive: *The services should be genuine and the clients should be authorized*. The security concerns of the interacting parties can furthermore be formulated as follows:

Client:

- The discovery service should be genuine, e.g. belong to the trust domain and conduct operations as expected
- The indicated services must meet the selection criteria
- Queries are private and should only be known to the discovery service

Service:

- The discovery service should be genuine
- Only authorized clients should attempt to invoke the service

Directory Service:

- Only authorized clients should make queries
- Only authorized services should make announcements

The words *authorized* and *genuine* are used numerous times in the list of requirements and their use in this paper needs an explanation: Subject to *authentication*, where the party (client or service) proves its identity, an authority will determine the *attributes* that are associated with that identity. During invocation of services, a boolean function called the *access requirement* will use the attribute set as parameters and return a true/false value which decides if the invocation can be completed, i.e. the party is authorized for the operation. This

Dette er en postprint-versjon / This is a postprint version.
DOI til publisert versjon / DOI to published version: 10.1117/12.920657

evaluation principle is elsewhere known as “Attribute Based Access Control “ (ABAC).

The term *genuine* indicates that the conduct of an operation by a party happens in accordance with expectations and agreements. A party infected by malware may well be authorized but not genuine.

The traditional approach to access control is to let the service be its own authority and maintain its own registry of users and access rights. This is a simple and well understood approach but requires multiple registration of subjects, which is known to scale poorly. In *identity management systems* (IdM), the authority is a centralized instance called *identity provider* (IdP) with the responsibility for maintenance of identity information on behalf of the entire community.

The IdP will issue *identity statements* which are attestations of the attributes and public keys which belong to identities. The information is bound to the identity for a time period through an *expiration date* and a *signature*. The IdP is the trust anchor of the community and its signature is trusted by everyone.

For the purpose of building a proof-of-concept prototype, an existing experimental IdM has been used. The *Gismo IdM* has been built for the purpose of testing IdM systems in military tactical domains using mobile nodes and wireless communication. This IdM system is now being extended to improve the security and trust in discovery service operations.

The contributions of this position paper is: (1) A proposed security model for discovery services (2) An integrated approach to authorization during discovery and invocation (3) The concept in *attested genuineness* applied to discovery services. (4) The reference implementation of the proposed solution

The identity management mechanism presented in this paper is a generic concept that can be applied to a number of different SD mechanisms, as it does not rely on solution specific features such as provider registration. Note that the interaction between provider, client, and DS is often implemented as services. This means that the information exchange can, as soon as the identity of the involved partners have been established, be protected using the same standardized security mechanisms as those used to protect other services.

The remainder of the paper is organized as follows: Section II proposes a set of security requirements for SD systems. Section III introduces the Gismo IdM in more details, while Section IV provides details on how to apply its access control support in a SD system. Section V presents some related research while Section VI concludes the paper and identifies remaining issues on this matter.

II. A Security Model for Discovery Services

The stakeholders should set the access requirements. In other words, the clients and services should formulate the requirements necessary to protect their assets and interests. In the model presented in this paper the discovery service (DS) is merely a broker which ensures that the formulated access requirements are met during the discovery operation. The access requirements formulated by the client during a discovery operation must be met by the service in the sense that the attributes contained in the identity statement of the service evaluates the boolean function to `true`.

In the opposite direction, the service formulates the access requirements which must be met by the attributes of the client.

The mutual access requirements must be met during the invocation phase as well as the discovery phase, since control during discovery alone leaves some unsolved risks:

1) The discovery service may be compromised and fail to enforce the access requirements
2) The client may reuse old discovery information to invoke services it no longer has access to
3) The client may pass the discovery information on to non-authorized clients.
4) The service has changed its access requirements, but the new requirements has not yet propagated to the discovery service

In Chapter IV, where the implementation is presented, it will be shown how access requirements are represented as serialized objects which are transferred to the directory and stored there. It will also be shown how the *liveness property* of the services is maintained through periodic updates of the identity statements.

Finally, additional selection criteria may apply to the discovery operation, like cost, service quality and semantic properties. This will be discussed in Section IV-D.

III. Gismo IdM

The presence of an identity management system is essential to the management of subject keys and attributes. The identity provider (IdP) will serve as a trusted third party (TTP) and issue attestation of both keys and attributes.

For the efforts on establishing a proof-of-concept prototype for trusted service discovery, an existing IdM called *Gismo IdM* has been employed. Gismo IdM was developed in order to study the necessary properties for an IdM used in a multi-domain wireless mobile network used by a coalition tactical force.[3]

In Gismo IdM, existing PKIs are kept for reasons of investment protection, but encapsulated by a number of Identity Providers (IdP), each serving a “community of interest” (COI). The members of a COI share the IdP’s public key as their trust anchor. The IdP issues *Identity Statements* (IS) to attest the public key and attributes of a subject. The IS is given a short lifetime and sealed with the signature of the IdP. Due to the short lifetime, no revocation arrangement is necessary.

The architectural overview of Gismo IdM is shown in Figure 2. Observe that the COI members are never exposed to PKIX protocols or data objects (X.509 certificates or revocation lists). The key properties are explained in the following paragraphs:

A. Authentication support

The identity provider (IdP) issues *Identity Statements* (IS) which bind the public key of a subject to its identity, analogous

Dette er en postprint-versjon / This is a postprint version.
DOI til publisert versjon / DOI to published version: 10.1117/12.920657

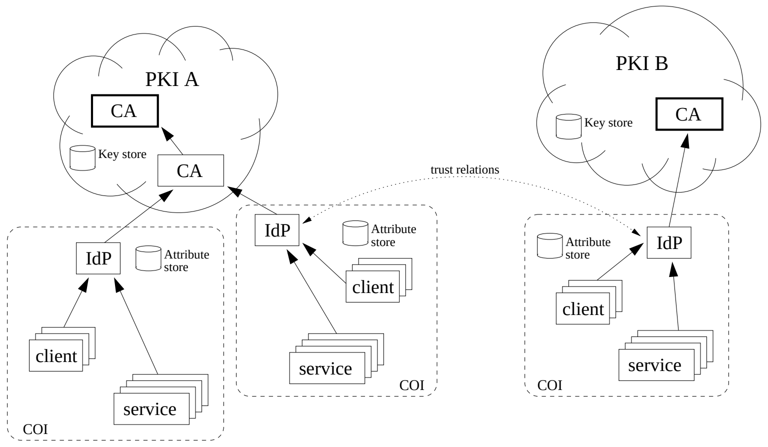

Fig. 2. The functional components of Gismo IdM. Observe that the IdP serves one single COI. Key management is handled by the PKI whereas the attribute management is done by the IdPs on the COI level

.

Subject Distinguished Name
Subject Public Key
Subject Attributes
Valid from−to
Issuer Distinguished Name
Issuer's Signature

Fig. 3. The structure of the Identity Statement

to X.509 Certificates. Identity statements are issued to local subjects registered in the IdP, as well as to subjects who can display an IS issued by a different IdP to which this IdP has a *trust relationship*. The structure of an IS is shown in Figure 3.

The subjects (either client or server) authenticate themselves during discovery or service invocation by the use of their identity statements and their private keys. Different authentication protocols have been designed with the purpose of generating as little network traffic and as few protocol round trips as possible. [4]

B. Integrated access control

Included in the identity statement is a set of attributes which describes properties of the subject in the form of name-value pairs. The attributes can describe *roles* of the subject and enter into access control decisions based on the Role Based Access Control (RBAC) or Attribute Based Access Control (ABAC) model. They can also describe other properties of the subject, e.g., preferred language, proficiency level etc.

The attributes are sealed inside the identity statement with the signature of the IdP, so they cannot be changed once issued.

C. Cross-COI operations

Clients can invoke servers in a different COI as indicated in Figure 2, provided that there exists a trust relationships between the two COIs. A client obtains an identity statement from its IdP, then passes on that IS to the IdP of a foreign COI. The foreign IdP can issue a *guest IS* containing the same information, but signed by the foreign IdP. Since the guest IS's signature will be trusted by servers in the foreign COI, it can be used to authenticate to these servers. Server authentication requires a *cross domain IS* issued from one IdP to the other, so a signature chain back to the client's trust anchor can be constructed. The middle part of Figure 4 shows the protocol that takes care of this. The IdP of COI A, termed IdP_a, issues a "native" identity statement to the client, which is given to IdP_b, which in turn issues a guest identity statement.

D. Attested genuineness

The integrity of the software running in a computer can be inspected by a hardware unit called *Trusted Platform Module* (TPM) which will issue certain cryptographic proof only if the integrity is approved. [10] It has been shown in [6] that the cryptographic proof can be verified by the trusted third party (the IdP) which will issue identity statements with certain reserved attribute values. These values will attest to anyone in the community that the subject operates from a computer with an inspected and verified software stack.

The concept of attested genuineness extends the trust in an operation from the identity of the controlling subject to the conduct of the subject, and that the operation takes place in a bona-fide manner unaffected by potential malware attacks.

E. Statefullness of services

During invocation of services, two different authentication mechanisms were developed: One simple protocol for *stateless services*, which are services whose system state is not affected by invocations, e.g. a lookup service. Replay attacks do not threaten the integrity of a stateless service, and costly replay protection may therefore be replaced with an encrypted response in order to render the replay useless to an attacker.

For a stateful service, characterized by a state change during service invocation, replay attacks must be detected before actual invocation takes place. This is a more costly mechanism that requires clock synchronization and a state memory in the

Dette er en postprint-versjon / This is a postprint version.
DOI til publisert versjon / DOI to published version: 10.1117/12.920657

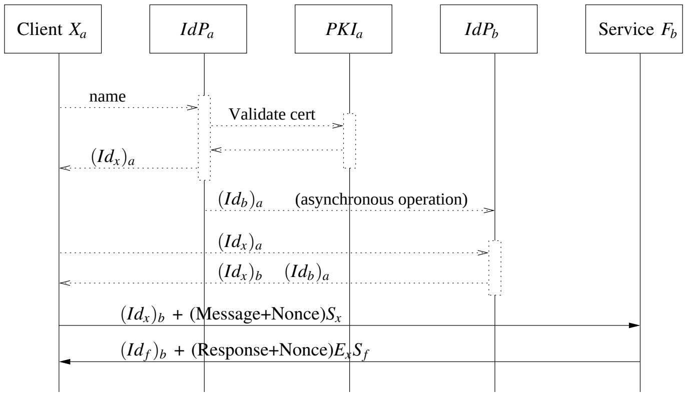
Client X_a
IdP_a
PKI_a
IdP_b
Service F_b
name
Validate cert
$(Id_x)_a$
$(Id_b)_a$ (asynchronous operation)
$(Id_x)_a$
$(Id_x)_b$ $(Id_b)_a$
$(Id_x)_b$ + (Message+Nonce)S_x
$(Id_f)_b$ + (Response+Nonce)$E_x S_f$

Fig. 4. The authentication protocol for a stateless service. The symbol $(Id_x)_a$ indicates the identity statement for Subject x issued by the IdP for COI a. $(Id_b)_a$ indicates the cross-COI for the IdP in COI b, issued by the IdP in COI a. S_x indicates signed by subject x, E_x encrypted to subject x.

server. The invocation protocol designed for this purpose is described in [4] and is functionally equivalent to the stateless variant in Figure 4.

IV. Implementation of access control during discovery

Section III describes the authentication and access control support during service invocation. The security mechanisms related to service discovery are based on the same mechanisms and will now be described in more detail.

The security requirements listed in Section I mandates the mutual authentication between clients, services and directory. Prior to discovery operations (queries and announcements), all parties must obtain identity statements from the IdP and authenticate themselves using the protocol shown in Figure 4. During the discovery operations, the necessary authentication elements (IS, signature) are piggybacked on the discovery service message (query or announcement).

A. The service table

The directory will keep a table of live services containing information pertaining to access control, metadata matching, service invocation and liveness control. Each row of the service table has these elements, some of which are to be explained later in the paper:

- The service endpoint URL
- The *service description tuple*
- The access requirement function
- The service invocation *parameter object*
- The Identity Statement of the service

B. Service announcement messages

Services should announce themselves once they are in operation. They must locate the IdP, get their IS, locate the directory, then authenticate with it and send an announcement message. The announcement message contains the information stored in the respective row of the service table. The structure is as follows:

- The service endpoint URL
- The service description tuple
- The access requirement function
- The service invocation *parameter object*

The IS is already included in the authentication part of the message as shown in Figure 4.

As mentioned in Section III, the IS expires after a while, after which it is disregarded from discovery operations. The service must therefore resubmit the the announcement message when necessary in order to be regarded as live. This is how the liveness property of the service is maintained, as defunct services will be removed from the service table.

The response from the directory is a simple status message which also authenticates the DS to the service as shown in Figure 5.

C. The service query/response messages

In order to query for live services, the client need to authenticate with the directory and send a service query. The query response from the directory will contain the authentication proof of the directory as shown in Figure 4. The service query consists of these elements:

- A *service description template*
- The access requirement function

In order to match the query to a row in the service table, all three tests must pass:

1) The service description template matches the service description object, cf. Section IV-D
2) The client's subject attributes (contained in the IS) meet the access requirements of the service and vice versa, cf. IV-E
3) The IS of the service is not expired (liveness property)

The matching process creates a result set of service representations which is sent back to the client as a response message. Each representation contains information necessary for the client to *rank* the services before invocation takes place, as well as endpoint and syntax information necessary for the actual invocation. The structure is as follows:

- The service endpoint URL
- The service description tuple
- The service invocation parameter object
- The distinguished name of the service IS
- The expiration time of the service IS

D. Service matching

The process through which a service query is matched with a set of service descriptions is not a part of the security model and consequently not one of the contributions of this paper. Any matching mechanism may be fitted to this system. For the experimental prototype, a *tuple matching* mechanism was chosen somewhat similar to what is found in some OODB systems. The specific mechanism is taken from a Tuplespace implementation called *SmallSpaces*.[2]

In the tuplespace retrieval mechanism the *template* consists of an ordered set of values or wildcards, which matches tuples

Dette er en postprint-versjon / This is a postprint version.
DOI til publisert versjon / DOI to published version: 10.1117/12.920657

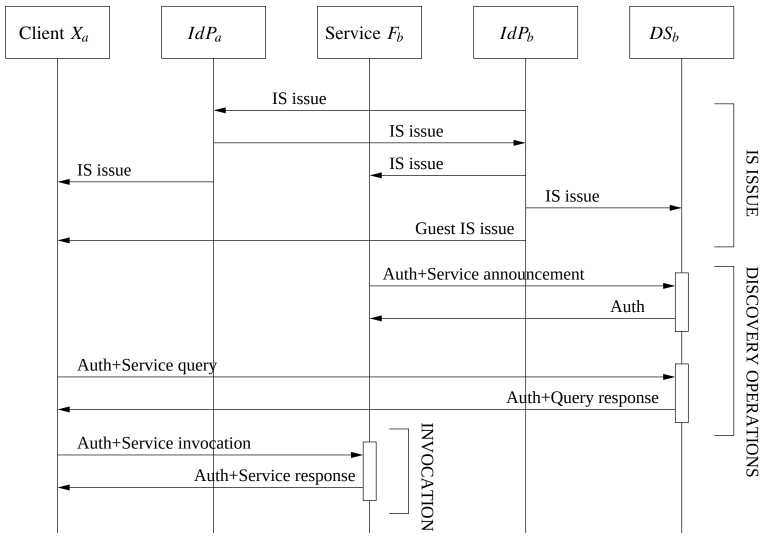

Fig. 5. The total sequence of IS issuing operations, discovery operations and service invocation. Details regarding IS issue and authentication are shown in Figure 4. Several different ordering of messages may be valid.

with the same arity (number of values) as well as same values in the non-wildcard position of the template.[7] Matching can be typed as well, and in an object oriented environment the class of a template element can match tuple elements of any subclass. Value matching is done through the Java `equals(..)` method (or similar).

Object oriented tuple matching allows for a flexible range of matching arrangements where hierarchical and orthogonal type checking can be combined with different value matching strategies.

E. Access requirement evaluation

Access requirement are expressed in the form of boolean expressions. During evaluation the variables are given values from the identity statement attributes. An example of an access requirement is

```
nationality=="uk" OR sec_clearance=="NATO secret"
```

which during evaluation is using the values of the attributes named `nationality` and `sec_clearance` from the identity statement of the subject under inspection.

Access requirements are represented as expression trees in Java object form and serialized during network transport. It is included in service announcements and service queries and evaluated by the directory during the service matching process.

F. Invocation parameter object

A service implementation in Gismo IdM is a Java servlet which inherits from `ServiceContainer` and follows certain coding conventions, some of which relates to discovery, other to invocation.

The parameters used for service invocation are provided as parts of a *parameter object* which is given the correct values prior to invocation. During the invocation, the serialized object is passed to the servlet endpoint URL through a POST operation. The traditional servlet parameter passing mechanism is not used.

The structure of the parameter class becomes the syntax description of the service, analogous to a WSDL (Web Services Definition Language) document in a Web Services environment.

An instance of the parameter object is returned as a part of the service query result set. The client may execute static code for instantiating this object, or it may employ the reflection API to manipulate the object during run-time. The role of the object is to transport values, not to provide any logic operations. It will therefore mostly contain getter and setter methods.

The replacement of a WSDL-based syntax description with a parameter object simplifies the client code considerably and allows for easier experimentation with ad-hoc selection and invocation.

G. Cross-COI operation

Section III-C introduced the cross-COI mechanisms in Gismo IdM. Cross COI operation allows the client and service to belong to different Community of Interest and rely on different IdPs. This is still possible now when Service Discovery has been introduced, since the SD is no more than another service. When the service and the client belongs to different COIs, the directory can belong to one of them, or even to a third. As long as the required trust relations are established between the IdPs of the different COIs, the service discovery mechanisms may well be invoked across COI borders.

H. The liveness property

Since identity statements are issued with a limited lifetime, the subject attributes cannot be regarded as attested after IS

Dette er en postprint-versjon / This is a postprint version.
DOI til publisert versjon / DOI to published version: 10.1117/12.920657

expiration. It is the duty of the service to repeat the announcement process so often that the directory service always has a non-expired IS.

This mechanism will also maintain the *liveness property* of the service description: If the server computer stops its operation, it will also stop refreshing the service announcement, and as the IS expires the directory will regard this service as defunct.

Since many existing SD mechanisms, most notably the standardized Web service registries, lack liveness support, a side effect of using the identity statement lifetime in this manner is improved functionality of the SD mechanism itself.

V. Related research

In most part of the research on Discovery Services the focus has been on scalability and efficiency of the architecture, expressive power of the selection process, and sometimes on security issues. In 1999, Czerwinski et al. [1] suggested the use of asymmetric keys and encryption to ensure that only authorized clients were allowed to make service queries. In their model, the access requirements are set by the directory service, not by the parties themselves. They rely on a traditional PKI architecture, which makes cross-domain operation much harder. But most important to this discussion is the lack of binding to the authentication of the invocation operation. Without such binding it is not possible to ensure that the invoked service really is the one referred to by the discovery process.

In 2007, Trabelsi et al. [9] performed an early study into the security requirements of SD, and suggested a security mechanism based on attribute based encryption. Here, encryption is used to limit which services a given client can receive information about in a setting where service information is distributed using multicast messages. This mechanism can be used to protect sensitive information in the SD messages, but it relies on the fact that information about user identity and attributes is available through other mechanisms.

Among more recent works are the experiments by Mochetta et al. on ubiquitous computing [8] where a distributed approach to security and privacy is investigated. Their security solutions are not founded on any threat analysis, and there is little discussion on matters of integrity and availability. Decentralized authorities as found in some peer-to-peer security systems offer only probabilistic trust relations, not deterministic as elsewhere. The authors believe that probabilistic trust will not be accepted in commercial applications.

With respect to SOA-specific security, there exists a number of standards that address different aspects of security. The Security Assertion Markup Language, in conjunction with WS-Trust, enables the usage of security tokens to establish trust between parties. In addition, WS-Security is commonly used to apply security functions such as encryption and integrity checking to SOAP messages. These standards focus on providing authentication, confidentiality, and integrity of each individual SOAP message exchange, but they do not address the full security requirements of the Discovery Service. These standards could however be used as components in a Web Service-based implementation of the concept suggested in this paper.

To the authors' knowledge, there is no published results which incorporates a federated identity management system into both the discovery and invocation phase, and leaves the security requirements to be formulated by the stakeholders, not the system administrator of the middleware components (like the discovery service). The work presented in this paper is likely to be novel and to leverage the trust between the user and the provider of a service that they both operate correctly and in a bona-fide manner.

VI. Conclusion and further research

The results presented in this paper demonstrate that a Service Discovery system in combination with an IdM system may improve the security and trust between the parties of a SOA transaction. The parties formulate their own access requirements which must be met both during discovery and invocation, and the liveness property is maintained through the identity statement expiration mechanism.

The issuing and authentication protocols are designed for the use in low bandwidth, wireless networks and consumes little network resources. The program code has been ported to Android so that smartphones can enjoy the same security services as ordinary computers.

Future research on trusted discovery systems involves the use of the TPM (Trusted Platform Module [10]) for the sake of integrity protection and integrity attestation. The effort is denoted *Attested Genuineness* [5], and aims to heighten the trust in a computer which has its software integrity inspected by a hardware unit and attested by a trusted third party.

References

[1] Steven E. Czerwinski, Ben Y. Zhao, Todd D. Hodes, Anthony D. Joseph, and Randy H. Katz. An architecture for a secure service discovery service. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, MobiCom '99, pages 24–35, New York, NY, USA, 1999. ACM.

[2] Anders Fongen. SmallSpaces tuplespace server. http://www.fongen.no/?docname=SmallSpaces, 2007. Accessed: 11/03/2013.

[3] Anders Fongen. Architecture patterns for a ubiquitous identity management system. In *ICONS 2011*, Saint Maartens, Jan. 2011. IARIA.

[4] Anders Fongen. Federated identity management in a tactical multi-domain network. *Int. Journal on Advances in Systems and Measurements*, Vol.4, no 3&4, 2011.

[5] Anders Fongen and Federico Mancini. Attested genuineness in service oriented environments. In *ICDIPC 2013*, Dubai, UAE, 2013.

[6] Anders Fongen and Federico Mancini. The integration of trusted platform modules into a tactical identity management system. In *MILCOM*, San Diego, USA, 2013.

[7] David Gelernter. Generative communication in linda. *ACM Trans. Program. Lang. Syst.*, 7(1):80–112, 1985.

[8] Eduardo Moschetta, Rodolfo S. Antunes, and Marinho P. Barcellos. Flexible and secure service discovery in ubiquitous computing. *J. Netw. Comput. Appl.*, 33(2):128–140, March 2010.

[9] Slim Trabelsi, Laurent Gomez, and Yves Roudier. Context-aware security policy for the service discovery. In *AINA Workshops (1)*, pages 477–482, 2007.

[10] Trusted Computing Group. Trusted Computing Group. http://www.trustedcomputinggroup.org/. Online, Accessed Mars 2012.

Dette er en postprint-versjon / This is a postprint version.
DOI til publisert versjon / DOI to published version: 10.1117/12.920657