

Secure communication for ad-hoc, federated groups

Andreas Sjöholm
Swedish Institute of Computer
Science, Box 1264, 164 29
Kista, Sweden
Axiomatics AB, Electrum 223,
164 40 Kista, Sweden
andreas@axiomatics.com

Ludwig Seitz
Swedish Institute of Computer
Science
Box 1263, SE-16429 Kista,
Sweden
ludwig@sics.se

Babak Sadighi
Swedish Institute of Computer
Science, Box 1264, 164 29
Kista, Sweden
Axiomatics AB, Electrum 223,
164 40 Kista, Sweden
babak@axiomatics.com

ABSTRACT

Ad-hoc federated groups are getting increasingly popular as means of addressing collaborative tasks that require information sharing. However, in some application scenarios, the security of the shared information is vital. Managing the communication security of such groups in an efficient way is a difficult task.

This paper presents an architecture that enables secure communication for ad-hoc, cross-organisational groups. Our architecture covers group admission control, group key management and secure group communication. The groups in question are expected to be ad-hoc groups where the potential participants have no prior knowledge of each other and thus federation mechanisms need to be used to establish group admission rights. In order to handle group admission we use the SAML and XACML standards, for group key management we use the TGDH protocol. Our approach thus supports decentralised management of the most important tasks in secure group communication using an integrated approach based on established security standards. We have also produced a demo implementation to show the feasibility of our architecture.

This research was pursued as part of the TrustDis project funded by the Swedish Governmental Agency for Innovation Systems (Vinnova).

Categories and Subject Descriptors

K.6.5 [Management of Computing and Information Systems]: Security and protection

Keywords

Access Control, Tree-based Group Diffie-Hellman, Secure Group Communication

1. INTRODUCTION

The core concept of the Internet has always been to share information distributed among different locations. As Vir-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

IDtrust '08, March 4-6, 2008 Gaithersburg, MD
Copyright 2008 ACM 1978-1-60558-066-1 ...\$5.00.

tual Organisations became increasingly popular, a new aspect of information sharing got into the focus of attention: cross-organisational, ad-hoc collaborations. Today clients for such applications include the scientific community, business and the military. Even other public organisations (e.g., health-care, firefighters, police) are increasingly interested in tools facilitating ad-hoc collaboration for achieving common goals.

As an application scenario, imagine the occurrence of a major traffic incident on a major public road. Several vehicles are involved in a pile-up, including trucks with dangerous chemicals. The harsh weather is complicating the rescuers work. A police car is quickly on site followed by additional rescue workers such as ambulances and firefighters.

It soon becomes apparent that many other public and governmental authorities and agencies such as the road administration authority, the national meteorological institute, a foreign embassy and the hazardous materials unit must be involved as their competence are necessary or they need to be informed. These actors need to share information, coordinating their efforts and avoiding fragmentation.

Such ad-hoc collaborations are expected to work without extensive infrastructure and formation should require minimal set-up time by being parallel to informing a group of actors. This type of collaborative group can be expected to accommodate for several hundreds of users spread amongst different organisations with little or no knowledge of each other except for a goal or interest in common.

Frequently, information sharing in such heterogeneous groups is subject to security related questions such as confidentiality and integrity. While these questions have clear solutions in client-server and even decentralised architectures, implementing the same solutions for ad-hoc groups without losing the advantages of ad-hoc group collaboration is far from obvious.

The lack of a controlled infrastructure and a central authority in an ad-hoc collaborative environment makes it difficult to maintain both security and manageability. Who can join and be part of a collaborative network? How can sensitive information be exchanged in a group of mostly unknown members? How can a user, away from his home network, be authenticated? Who enforces the rules of the network? How can administration be kept manageable under such circumstances?

A solution which addresses the questions above, applicable to large ad-hoc groups, is needed. This paper presents a novel architecture that deals with the problems of authori-

sation, authentication, confidentiality as well as availability in such groups.

The rest of this paper is structured as follows: In section 2 we define the problem at hand. Section 3 presents related work. In section 4 we present our architecture designed to solve the problems. Our demo implementation of the architecture is presented in section 5. We then discuss the advantages and drawbacks of the architecture in section 6. Finally we summarise, give a conclusion and point to future work in section 7.

2. PROBLEM STATEMENT

The two main problems we face with respect to secure group communication are[1]:

- Who may join the group?
- How can the group members share cryptographic keys in order to ensure confidentiality and message integrity?

A sensible solution to the first problem is a combination of federated identity management and a policy based access control system. This requires a Policy Decision Point (PDP) which issues access control decisions, Policy Enforcement Points (PEP) that enforce the PDP's decisions, and Policy Information Points (PIP) that provide and the policies used by the PDP [14]. The federated identity management system is needed in order to provide certified information about the user (e.g., roles, affiliations, qualifications) to the PDP.

A common solution to the second problem are group key agreement schemes that allow all participants of the group to compute a common group key. Such a scheme needs to provide new group keys to the members in reaction to the following events [9]:

- Single member join: A new member joins the group.
- Single member leave: A current member leaves the group.
- Group merge: Two groups merge into one.
- Group partition: A group is split into new groups.

In order to be cryptographically secure, the key agreement scheme needs to fulfil the following cryptographic requirement [9]:

Definition 1. Key Independence:

Assuming that the key agreement scheme has produced the sequence of keys $K = \{k_1, k_2, k_3, \dots, k_m\}$ in reaction to a sequence of group events.

A passive adversary who knows a proper subset of group keys $K' \subset K$ cannot discover any other group key $k_i \in (K \setminus K')$.

This requirement also ensures that a new group member can not decrypt any of the messages sent before the join event, and an ex-member can not decrypt any messages sent after the leave event (*Forward and Backward Secrecy*).

Secondary problems that need to be addressed are:

- Scalability: The group communication architecture must scale well with the number of users.

- Full representation of the group: All members shall have a complete, consistent representation of the group and its members.
- Distributed functionality: It must be possible to distribute functionality (such as PDPs), thus preventing single points of failures.
- Implementation of leader role with possibility of delegation: A group leader shall exist with the role of creating the policies that regulate group admission control. It shall be possible to issue constrained delegations of this role.
- Fault tolerance and self healing: Link disruption must not jeopardise the security and operability of the group more than that the functionality can be restored when the link is restored. During disruption when members are lost or cut off from each other, the group must be functional as partitioned and autonomous groups.

Summarising one can say, that the problem is to find an architecture that is able to solve all of these problems in an integrated, efficient, and decentralised way.

3. RELATED WORK

We present related work in the areas of group key agreement schemes and of group admission control. Further we present related work that combines both to create secure group communication solutions. A large number of group key management systems use trusted third parties (TTPs) to distribute the keys to the group members (e.g., [10, 15]). The use of a TTP greatly simplifies key management and update issues, due to the centralisation. However, the centralisation also disqualifies such schemes for our application scenario (we further discuss this in subsection 4.1).

In the area of contributory group key agreement schemes, the Tree-based Group Diffie-Hellman (TGDH) protocol by Kim, Perrig, and Tsudik [9] is an approach that fulfils our requirements. Therefore we choose it as component of our architecture.

Furthermore the Dynamic SubTree (DST) group key agreement scheme by Mao *et al.* [11] could be considered, since its average time cost is less than TGDH's. However, DST uses the unrealistic assumption that group member departure times is known in advance.

Another promising approach is the PFMH tree based contributory group key agreement protocol suite (PACK) by Yu, Sun, and Liu [16]. PACK seems to have better theoretical performance than TGDH for single user join, while the author's simulation results suggest worse performance for single user leave.

Kim, Mazzocchi, and Tsudik propose a number of approaches for admission control in peer groups [8]. The paper's main focus is on admission by voting whereas the use of access control mechanisms is limited to Access Control Lists.

The commercial product *MindAlign* from *Parlano* advertises secure communication for federated groups. No detailed information on the technical details could be obtained¹.

¹Actually since Parlano has been acquired by Microsoft, there is no information at all available from the Parlano website, our main source of information was <http://en.wikipedia.org/wiki/MindAlign>.

Judge and Ammar propose a Group Access Control Architecture for Secure Multicast and Anycast [7]. This architecture provides group policy management, member authorisation and group key management. However, their key agreement scheme is not *key independent*. Moreover, the access control engine used in this architecture is not further specified.

Agarwal *et al.* suggest an integrated solution for secure group communication [2]. This approach uses GDH, the predecessor of TGDH for group key agreement. Furthermore the non-standard access control system Akenti is used to determine group admission. Since work both on this approach and on Akenti appears to have ceased (last update of Akenti was 2005), this does not seem like a promising candidate for solving our problem.

We can therefore summarise, that a number of promising approaches for contributory group key agreement schemes exist. Their main difference lies in the performance optimisation of specific group events. Thus alternative schemes can be investigated, should future experiments indicate performance problems with our TGDH based approach.

As for comprehensive architectures including group admission control, related work seems to either rudimentary or based on outdated, non-standard access control technology.

4. SECURITY ARCHITECTURE

In this section we present our architecture for secure group communication *TgdhXacml* and explain some of the choices we made.

4.1 Contributory key schemes versus TTPs

As we pointed out in section 3 our main problem with TTPs is their centralisation. Setting up a TTP and reconfiguring it when the group structure changes drastically, is bound to introduce delays in availability.

TTPs availability can also become problematic when the number of users increases dramatically, furthermore the TTP can be a single point of failure and therefore a prime target for attacks. Finally the use of a TTP raises the question of who should provide and maintain this service, which can increase the level of distrust between group members and thus limit the information they are willing to provide. All these points speak against using a TTP in our ad-hoc, federated group scenario. Therefore we have not considered TTP based key management systems for our approach.

4.2 XACML and SAML

When choosing systems for authorisation and for supporting federated identity, our main objective was to use widely accepted standards. This choice stems from the facts that a range of different actors have to interoperate and to agree on a common authorisation model as well as both a direct (corporate's policy) and an indirect (through SOX²) requirement to exclusively use open standards.

XACML [6] is a widely accepted access control standard with good support both from the industry and the scientific community [3]. Furthermore a range of useful support tools are available both specifically for XACML³ and for process-

²Sarbanes-Oxley Act

³see: <http://www.oasis-open.org/committees/xacml> under the heading *Additional Information*

ing XML.

Another determining factor in our choice of XACML is the fact that the upcoming version 3.0 of XACML [13] has support for delegation, a feature that is very important for our decentralised group management approach. The delegation mechanism allows authority holders to delegate a precisely constrained part of their authority to others, thus facilitating decentralised administration, without the risk of giving away too much authority.

Using these delegation features, the broad access control policies for group communication can be specified by a co-ordination authority, while power to create fine-grain authorisations (e.g., for specific subjects) can be delegated to the organisations participating in the group.

Using SAML [4] as assertion language for supporting federated identity is an obvious choice when combined with XACML. In order to evaluate a specific request, an XACML PDP needs to establish an evaluation context. This includes collecting attributes for the subject of the request. Providing such attributes in a secure way is a typical application for SAML.

Having chosen SAML as syntax and protocol suite for attribute assertions, an Attribute Authority (AA) is needed to generate the SAML assertions and to manage the federation of attributes. The Assertion Server⁴ is a system closely integrated with XACML 3.0 and SAML. It's main function is to provide attribute assertions and to manage attributes. Besides this the Assertion Server handles attribute hierarchies, delegation of attribute authority and parallel administration of multiple sources of attribute authority. It can be queried according to the protocol defined in the SAML V2.0 [4] standard.

Internally it uses XACML policies to represent attribute value assignments (including attribute hierarchies) and attribute authority (including authority delegation), however a user never needs to see these internal policies.

Using the Assertion Server, a PDP can gather all necessary attributes in order to evaluate a request. Multiple Assertion Servers can serve one PDP and one Assertion Server can serve multiple PDPs.

Figure 1 shows an example set up with a policies and some attributes provided externally by an Assertion Server. All XACML policies have been greatly simplified to make them less verbose. In our application scenario from the introduction, a crisis coordinator has the authority over a group resources allocated to the incident, which has been named *Incident23*. He issues a root policy delegating authority over these resources to the *Police board*, however the authority constrains the possible subjects to members of the police force (*Delegated Subject* must have *organisation = Police*).

Alice who is a member of the police board, coordinates the police operation. In the second policy, she allows Bob who is the policeman on site to access said resources. In order to be valid, this policy needs to be supported by the root policy. This is the case, since the Assertion Server confirms both that Alice has the *role = Police board* attribute and Bob has the *organisation = Police* attribute.

4.3 Architecture

We now proceed to tie together the different components of our approach into an integrated architecture. The in-

⁴Available from:

http://www.sics.se/spot/assertion_server.html

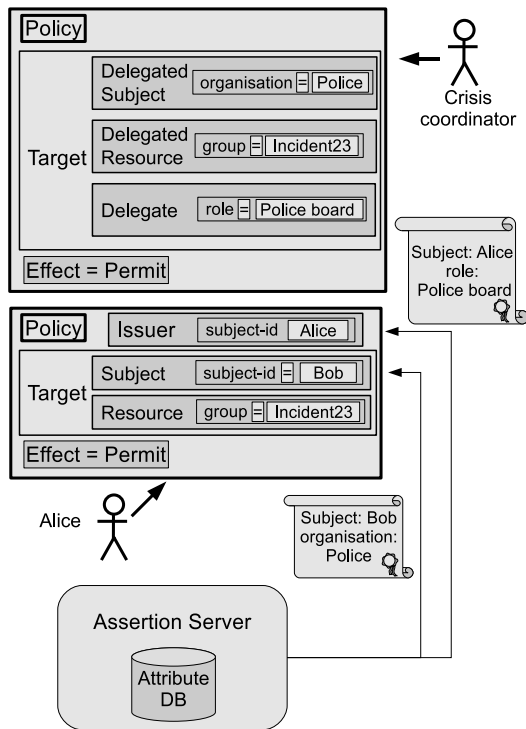


Figure 1: Example for a delegation and related attribute assertions provided by an Assertion Server.

tegrating component deployed with all group members is called the *context handler*. The purpose of the context handler is to coordinate information and integrate the services needed for TGDH and XACML in a ad-hoc environment. This means that a group member's context handler must implement the following functionality:

- Locate information for XACML contexts
The most important functionality of the context handler. TgdhXacml stores a list of addresses where certain information and services can be found in the distributed group (such as PDPs and Assertion Servers). The TgdhXacml context handler uses these addresses to provide the access control component with additional information (e.g., policy databases, assertion server locations).
- Respond to queries about the TgdhXacml environment
All members in the group should have the same information about services and newly joined members need to be updated on this knowledge. Therefore the context handler must be ready to provide the TGDH sponsor with information about the environment, such as lists of PDPs and Assertion Servers.
- Start up (additional) PDPs and Assertion Servers
Upon creation of the group the group leader's context handler starts up one PDP, one Assertion Server and one PIP and publishes this information in the list of available services. Other members can later start additional modules if the workload is too high for a single member or just to distribute services in the group.

This could be made voluntarily by asking a member to take responsibility for lowering the workload on a stressed member or to start up dummy members that do not participate in the group communication and just provide additional services.

- Synchronisation of policy and assertion databases
Since it should not make a difference which PDP or Assertion Server a member is using, databases containing attribute and policy information must be synchronised. An approach would be to use a replicating database server. If the underlying distributed network layer supports data replication this feature can be used.
- Communication security
The context handler gets provided with a group master key from the TGDH layer which allow it to encrypt and decrypt messages. The group master key is used as seed to generate an AES-256 content encryption key (CEK). Thus even if a CEK is compromised, the current group master key remains secure.
Our TGDH layer also provides DSA keys for all group members, which are used for digital signatures. DSA has been chosen for message authenticity instead of a MAC algorithm because DSA is asymmetric and thus the DSA public key can be sent in clear text together with the join group request.
The raw data communication protocol for the group is managed by the network layer.
- Implement API
As the TgdhXacml provides no user functionality, applications are to be plugged in on top of the TgdhXacml. This can be any kind of service that provides or retrieves information.

Figure 2 shows an example layout of the TgdhXacml architecture for a group with four members. Every member M_i has a context handler, coordinating the other modules. All members are also running the TGDH protocol and a PEP. Additionally some members are running applications, PIPs and Assertion Servers.

In the example the members $M1$ and $M3$ provide an Assertion Server. The Assertion Server's attribute database needs to be synchronised between these two members. Members $M2$ and $M4$, both provide a PIP, therefore the policy database must also be synchronised. The top level of the architecture is the application layer.

Observe that member $M3$ is not running any application, which implies that $M3$ is a virtual member that was only set up to lower workload of the other members and increase availability of the attribute database.

4.4 Authentication

Since all Diffie-Hellman offspring lack authentication and are susceptible to active attacks (such as man-in-the-middle attacks), TgdhXacml needs a mechanism to authenticate group members. This authentication is to provide a unique identifier that can be used to bind attribute assertions to specific group members.

In order to prevent man-in-the-middle attacks, this authentication infrastructure needs to be set up over secure channels prior to joining the group. We use a X.509 v3

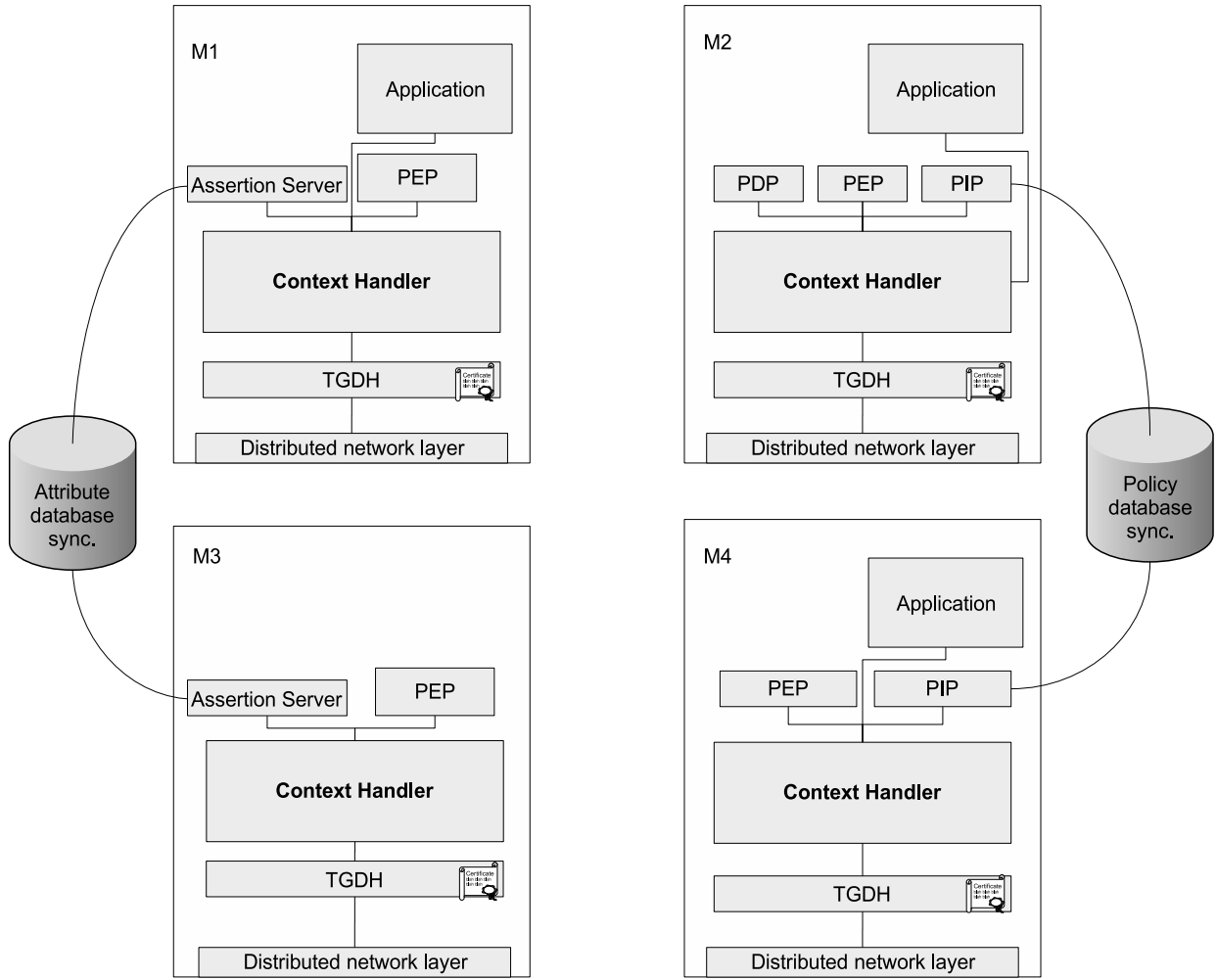


Figure 2: The architecture of our group communication system.

based public key infrastructure for our approach, because it is the most widespread standard and is supported in several open source libraries.

The group creator either generates a self-signed certificate or requests a certificate from a Certificate Authority (CA). Other members' certificates must be signed by a CA according to a certificate infrastructure policy. We then use the Distinguished Names (DN) from these certificates as subject identifiers in the XACML policies and SAML assertions.

In our application scenario it is reasonable to assume pre-existing authentication structures within the organisations participating in the group. Nevertheless chances are that incompatibilities between different authentication systems would require some further federation mechanisms in order to become interoperable. This however is out of the scope of this article.

4.5 The modified join protocol

This section describes the protocol used when a new member wants to join the group. It is slightly modified from the classical TGDH protocol in order to accommodate for the access control and the transfer of DSA public keys we added.

A join request must contain the TgdhXacml group name one wishes to join, the blinded key for the TGDH join-protocol, the potential member's DSA public key and authentication certificate.

The join request is broadcasted to all members of the group. One existing TgdhXacml context handler declares itself as the sponsor and further on only that context handler will react to the join message. The sponsor now checks the validity of the authentication certificate and requests an admission decision from a group PDP, in order to determine if the new member should be allowed to join. Thus the sponsor acts as PEP, creating a valid XACML request and then enforcing the PDP's decision.

If the join request is denied, the sponsor notifies the potential member and takes no further action. If the join request is permitted the sponsor sends a message containing vital tree information such as tree structure, blinded keys, sequence numbers, PDP and Assertion Server lists to the new member. The sponsor also blocks the whole TGDH tree and broadcasts an update message containing the new blinded keys in the tree.

Figure 3 shows a sequence diagram of a join.

4.6 Management

One important goal of this architecture is to support security management of ad-hoc collaborative groups. This mainly boils down to supporting the following tasks:

1. Controlling who may join a group based on attributes
2. Delegating administrative power
3. Controlling access to information at the application level

The first task is accomplished by creating XACML access control policies that define the conditions of group admission. After these have been defined, the job of deciding on group admission is automatically performed by the PDP. Updates of the group admission policies may be necessary, but they will certainly occur less frequently than actual group admission decisions.

Tools that support the creation of XACML policies exist⁵, however analysing these and integrating them into our group management architecture is out of the scope of this paper.

The second task is supported by the XACML v3.0 Administrative Policy standard [13]. An example for the basic principle of delegation using administrative policies was shown in figure 1. A full XACML delegation policy is shown in appendix A.

The third task is supported by restricting access to the group key, used to encrypt all communication between group members. Thus only group members will have access to the applications available within the group. Furthermore these applications can be programmed to enforce additional access control policies towards members of the group trying to access them. Such application access control could also make use of the access control infrastructure already available within the group (PDPs, Assertion Servers, PIPs).

5. DEMO IMPLEMENTATION

In order to demonstrate the feasibility of our architecture, we have produced a demo implementation for the main elements of our architecture. This includes the context handler, the TGDH layer, XACML PDP and PEPs, SAML assertions and Assertion Servers.

The code was written in the Java programming language and includes a basic graphical user interface shown in figures 4 and 5.

The interface allows users to

- join and leave the group,
- to display a list of PDPs available in the group,
- send and receive encrypted messages,
- display the TGDH tree (Show Tree),
- show the current master group key, the number of members and the position of this member in the tree (Show info),
- and to start a new PDP for the group.

The distributed network layer has been implemented by using IP Multicast.

The TGDH layer is based on the TGDH library from Lijun Liao at Ruhr-University Bochum, Germany⁶.

The Assertion Server and the XACML PDP are open source projects previously published by SICS⁷. We currently use non-synchronised flat files as attribute and policy databases and a simple PIP has been localised with each PDP, performing non-optimised policy retrieval.

The core components that were build from scratch according to our architecture are the Context handler and the PEP. The Context handler is configured by a simple file, specifying the group name the multicast address and port and the DSA key parameters for this member.

As a simple application layer we implemented a group chat sending and receiving strings.

⁵For example: <http://xacml.dif.um.es/>

⁶<http://www.nds.ruhr-uni-bochum.de/research/top/gc/tgdh/> used with kind permission

⁷See: <http://www.sics.se/spot> for more information

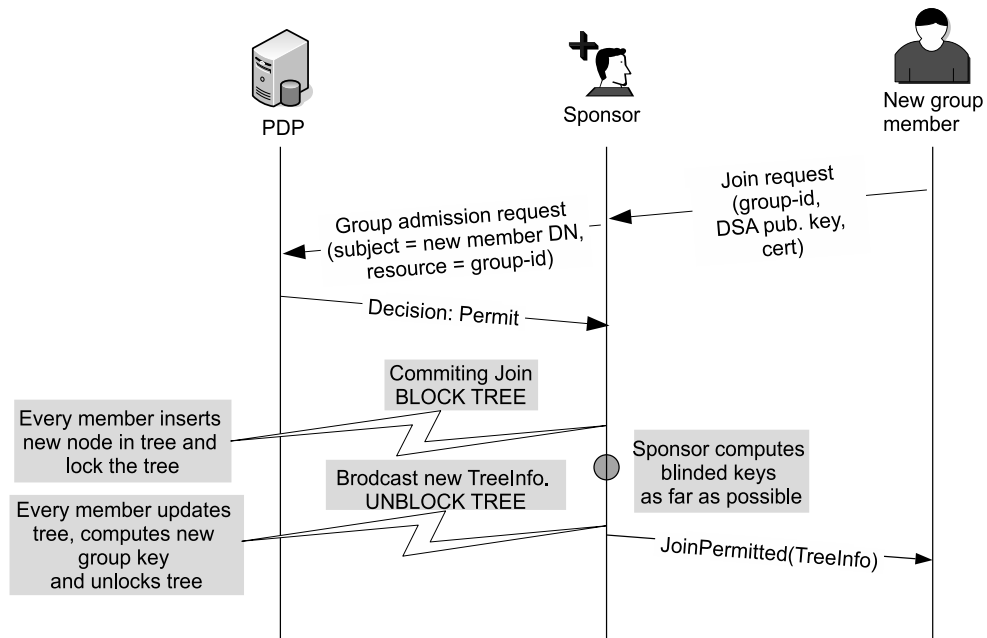


Figure 3: The sequence of diagram of the join protocol.

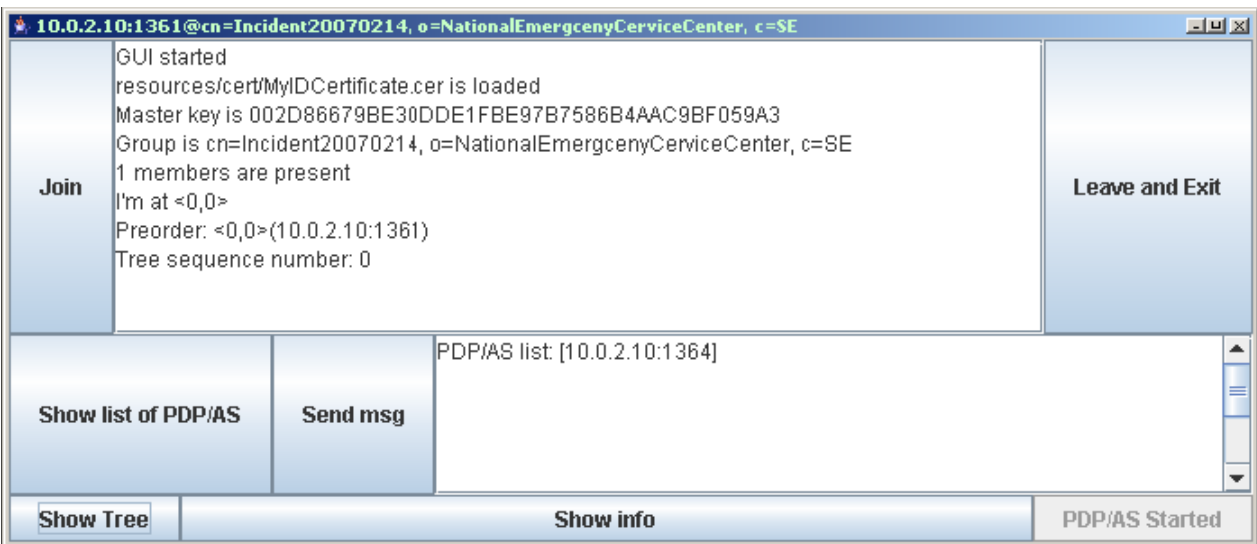


Figure 4: GUI at start.

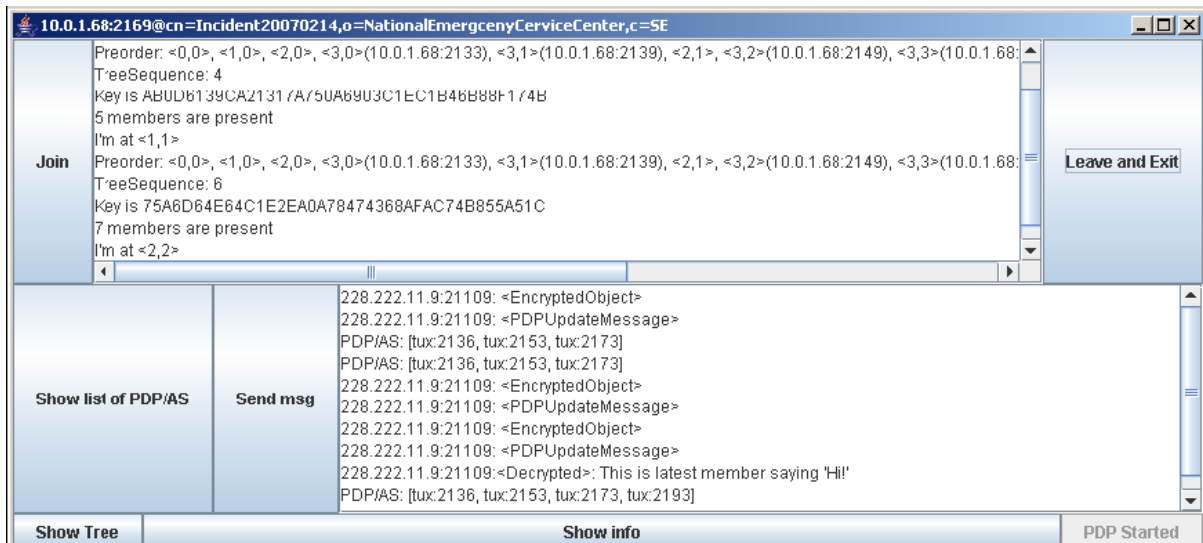


Figure 5: GUI after a few joins.

Figure 4 shows the user interface at start. The certificate has already been loaded and a master key for the group was generated. The DN of the group certificate is displayed together with some other TGDH tree information.

In figure 5 a few users have joined the group, thus expanding the TGDH tree and some encrypted messages have been sent (in this version the chat shows a lot of debug information too).

This demo exemplifies how we intend our architecture components to interact and shows some necessary steps for an implementation of the full architecture (customising a PEP, adapting a TGDH library etc). Since it was produced as part of Andreas Sjöholm's master thesis, the time frame did not allow for the extensive testing that would have been required to get meaningful performance data. Such testing will be performed on a more complete version of the architecture, including distributed attribute and policy databases and a more advanced application layer.

6. DISCUSSION

In this section we first discuss issues related to the contributory group key agreement component of our architecture. We then proceed to discuss issues related to the authorisation components.

A criticism voiced about contributory group key agreement schemes [5] claims that:

- An established communication layer is required which must support broadcast operations.
- If there are frequent changes in the group membership, due to member mobility and/or link outage, the topology of the ad-hoc group may change before the key agreement scheme has been completed.

We believe that our application area does not face these problems to a degree that would make contributory key agreement schemes problematic. In a crisis management scenario as presented in the introduction either mobile internet

access is going to be available or dedicated communication infrastructures will be provided (e.g., the Raket⁸ digital radio communication system in Sweden).

As for the second problem, we assume that the main bottleneck will be communication cost and not computation (since the computational power of even small mobile platforms is quite high, especially if they include dedicated crypto-hardware). Considering the low communication cost of the TGDH join and leave events, it would require a constant stream of such events to seriously disrupt the function of our architecture. This is quite unlikely for our application scenario, since new members will more probably join as a batch and then want to stay group members for a period of time.

Concerning the use of XACML and SAML in our authorisation architecture, common criticisms are:

- These standards are too complicated because they provide too much functionality (thus confusing the users).
- The use of XML as base syntax introduces unnecessary verbosity.

It is true that the full range of XACML and SAML functionality is not needed for addressing our problem. However, building custom made proprietary systems is not a reasonable approach either, since we then would lose interoperability through standard compliance, and make future extensions more complicated.

Therefore we claim that a reasonable approach is to create profiles for the use of XACML and SAML in specific applications, that define subsets of the full functionality. Such profiles can address the access control problems in the specific vocabulary of the application, thus making them understandable for users. Since the profile only restricts the use of policy and assertion syntax, future updates of the functionality can simply be achieved by updating the profile

⁸see http://www.krisberedskapsmyndigheten.se/default_..._176.aspx for more information

(thereby allowing the use of new features of the standard). With custom made approaches, this would often require a redesign of the whole authorisation infrastructure.

There are two possible problems with the verbosity of XML encodings: Firstly it can be a question of bandwidth use, secondly understanding and reading the documents (policies or assertions in our case) can be very difficult for humans. The former problem can be addressed by various XML compression techniques, e.g., WBXML [12]. The latter problem can be addressed by appropriate user-interfaces that support policy editing and attribute management without requiring direct interaction with the raw XML format.

For our architecture as a whole, some points should be observed: Group members are trusted, no protection against insider attacks is inherently provided by this architecture. This is due to the peer-to-peer nature of our group, as opposed to a client-server group where there would be a group leader taking certain security responsibilities. At the TgdhXacml middleware layer, the damage an insider attack could do is the same as an active attacker can do. Such insider attacks must instead be prevented at application layer.

We are convinced that the benefits of having a flat group hierarchy (no single point of failure, distribution of workload between the group members) outweigh the drawbacks in our application scenario.

Our architecture relies on no single point of control or command since group key establishment (TGDH) is self healing and resources access information (policies) are distributed to reach an acceptable degree of redundancy. What will happen in case of network disruption is inability to propagate policy updates (database synchronization, see Figure 2) and revocation information, but group functionality will not be affected.

Another important property that one needs to be aware of when planning to use our architecture is that it requires authentication frameworks already to be in place with all potential group participants. Such frameworks often exist in organisations we aim at in our applications scenarios, however some identity federation efforts could be required to have them work together in a heterogeneous group. The presence of such authentication structures within organizations makes distributed authentication possible and makes TTPs such as a Kerberos server unnecessary (which would have been hard to implement across multiple organizations). This by letting only one authoritative representative per organization mutually authenticate with the crisis coordinator and thereafter take advantage of the Assertion Server and delegations.

7. CONCLUSION AND FUTURE WORK

In this paper we have presented an architecture supporting secure communication for ad-hoc, federated groups. Our main contribution lies in combining advanced access control mechanisms, federated identity management and contributory key agreement into an integrated architecture.

This architecture allows decentralised management of group admission control and therefore makes it possible to spread the administrative workload to the most competent parties, while not exposing the full set of rights to all delegates.

In the future we plan to investigate how to provide a better administration interface for policy and attribute management. We also consider to replace the chat application

in the demo with a more realistic service. A suitable application could be Helping Hand⁹, a graphical utility for cross actor support in emergency response.

APPENDIX

A. EXAMPLE POLICIES AND ASSERTIONS

In this section we present the full policies from figure 1 and the related SAML attribute assertions. In order to reduce the verbosity, we have defined the following abbreviations:

```
nsp = "urn:oasis:names:tc:xacml" and
string = "http://www.w3.org/2001/XMLSchema#string"
```

Please note that both policies are contained in a single PolicySet and follow the XACML 3.0 syntax, which is quite different from the XACML 2.0 syntax. The first policy (*RootPolicy*) delegates to the Police board the authority to create policies for accessing resources from the group *Incident23*, provided the subjects of these policies are members of the police force.

The second policy (*Alice'sPolicy*) is issued by Alice, and gives Bob access to the resources from the group *Incident23*.

```
<PolicySet xmlns="nsp:3.0:schema:os"
  PolicySetId="Incident23Policies"
  PolicyCombiningAlgId="nsp:1.0:policy-combining-
    algorithm:permit-overrides">
  <Target/>
  <Policy PolicyId="RootPolicy"
    RuleCombiningAlgId="nsp:1.0:rule-combining-
      algorithm:permit-overrides">
    <Target>
      <AnyOf>
        <AllOf>
          <Match MatchId="nsp:1.0:function:
            string-equal">
            <AttributeValue DataType="string"
              >Police</AttributeValue>
            <AttributeDesignator
              Category="nsp:3.0:attribute-
                category:delegated:Subject"
              AttributeId="organisation"
              DataType="string"/>
          </Match>
        </AllOf>
      </AnyOf>
      <AnyOf>
        <AllOf>
          <Match MatchId="nsp:1.0:function:
            string-equal">
            <AttributeValue DataType="string"
              >Incident23</AttributeValue>
            <AttributeDesignator
              Category="nsp:3.0:attribute-category:
                delegated:Resource"
              AttributeId="group"
              DataType="string"/>
          </Match>
        </AllOf>
      </AnyOf>
    </AnyOf>
  </AllOf>
</PolicySet>
```

⁹developed at the Viktoria Institute, see <http://www.viktoria.se/~landgren/crossactorsupport>

```

    <Match
      MatchId="nsp:1.0:function:
        string-equal">
      <AttributeValue DataType="string"
        >Police board</AttributeValue>
      <AttributeDesignator
        Category="nsp:3.0:attribute-category:
          delegate"
        AttributeId="role"
        DataType="string"/>
    </Match>
  </AllOf>
</AnyOf>
</Target>
<Rule RuleId="Rule1" Effect="Permit">
  <Target/>
</Rule>
</Policy>
</PolicySet>

<Policy PolicyId="Alice'sPolicy"
  RuleCombiningAlgId="nsp:1.0:rule-combining-
    algorithm:permit-overrides">
  <PolicyIssuer>
    <Attribute AttributeId="nsp:1.0:subject:
      subject-id">
      <AttributeValue DataType="string"
        >Alice</AttributeValue>
    </Attribute>
  </PolicyIssuer>
  <Target>
    <AnyOf>
      <AllOf>
        <Match MatchId="nsp:1.0:function:
          string-equal">
          <AttributeValue DataType="string"
            >Bob</AttributeValue>
          <AttributeDesignator Category="Subject"
            AttributeId="nsp:1.0:subject:
              subject-id"
            DataType="string"/>
        </Match>
      </AllOf>
    </AnyOf>
  </AnyOf>
  <AllOf>
    <Match MatchId="nsp:1.0:function:
      string-equal">
      <AttributeValue DataType="string"
        >Incident23</AttributeValue>
      <AttributeDesignator
        Category="Resource"
        AttributeId="group"
        DataType="string"/>
    </Match>
  </AllOf>
</AnyOf>
</Target>
<Rule RuleId="Rule1" Effect="Permit">
  <Target/>
</Rule>
</Policy>
</PolicySet>

```

The SAML assertions would look like this (slightly abbre-

viated):

```

<saml:Assertion ID="58e1c517"
  IssueInstant="2007-11-16T13:27:15Z"
  Version="2.0">
  <saml:Issuer Format="string">asServer</saml:Issuer>
  <ds:Signature>
    ...
  </ds:Signature>
  <saml:Subject>
    <saml:NameID Format="string"
      NameQualifier="nsp:1.0:subject
        :subject-id">Alice</saml:NameID>
  </saml:Subject>
  <saml:AttributeStatement>
    <saml:Attribute Name="role"
      DataType="string">
      <saml:AttributeValue
        >Police board</saml:AttributeValue>
    </saml:Attribute>
  </saml:AttributeStatement>
</saml:Assertion>

```

This one asserts that Alice has indeed the role *Police board*.

```

<saml:Assertion ID="7df9c595"
  IssueInstant="2007-11-16T13:33:28Z"
  Version="2.0">
  <saml:Issuer Format="string">asServer</saml:Issuer>
  <ds:Signature>
    ...
  </ds:Signature>
  <saml:Subject>
    <saml:NameID Format="string"
      NameQualifier="nsp:1.0:subject
        :subject-id">Bob</saml:NameID>
  </saml:Subject>
  <saml:AttributeStatement>
    <saml:Attribute Name="organisation"
      DataType="string">
      <saml:AttributeValue
        >Police</saml:AttributeValue>
    </saml:Attribute>
  </saml:AttributeStatement>
</saml:Assertion>

```

This would assert that Bob is indeed in the organisation *Police*. The combination of these two assertions validate the second policy with respect to the first one, during evaluation by a PDP. Without these assertions, the second policy would be discarded as invalid.

B. REFERENCES

- [1] A. Sjöholm. Secure Group Management in Dynamic Networks. Master Thesis at Department of Computer and System Sciences, Royal Institute of Technology, Stockholm, Sweden, 2008.
- [2] D. Agarwal, O. Chevassut, M. Thompson, and G. Tsudik. An Integrated Solution for Secure Group Communication in Wide-Area Networks. In *Proceedings of the Sixth IEEE Symposium on Computers and Communications (ISCC'01)*, Hammamet, Tunisia, July 2001. IEEE Computer Society.

- [3] A. Anderson. Xacml References and Products, Version 1.83, January 2007. <http://docs.oasis-open.org/xacml/xacmlRefs.html>.
- [4] S. Cantor, J. Kemp, R. Philpott, and E. Maler Eds. Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) v2.0. Standard, Organization for the Advancement of Structured Information Standards (OASIS), March 2005. <http://www.oasis-open.org>.
- [5] A. Chan and E. Rogers. Distributed Symmetric Key Management for Mobile Ad hoc Networks. In *Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 4, pages 2414–2424, Hong Kong, China, March 2004. IEEE Computer Society.
- [6] S. Godik and T. Moses Eds. eXtensible Access Control Markup Language (XACML). Standard, Organization for the Advancement of Structured Information Standards (OASIS), February 2003. <http://www.oasis-open.org/committees/xacml>.
- [7] P. Judge and M. Ammar. GOTHIC: A Group Access Control Architecture for Secure Multicast and Anycast. In *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications INFOCOM*, volume 3, pages 1547–1556, New York, USA, June 2002. IEEE Computer Society.
- [8] Y. Kim, D. Mazzocchi, and G. Tsudik. Admission Control in Peer Groups. In *Proceedings of the Second IEEE International Symposium on Network Computing and Applications*, pages 131–139, Cambridge, MA, USA, April 2003. IEEE Computer Society.
- [9] Y. Kim, A. Perrig, and G. Tsudik. Tree-based Group Key Agreement. *ACM Trans. Inf. Syst. Secur.*, 7(1):60–96, 2004.
- [10] J. Kohl and C. Neuman. The Kerberos Network Authentication Service (V5). Technical report, The Internet Engineering Task Force IETF, 1993. <http://www.ietf.org/rfc/rfc1510.txt>.
- [11] Y. Mao, Y. Sun, M. Wu, and R. Liu. Dynamic Join-Exit Amortization and Scheduling for Time-Efficient Group Key Agreement. In *Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 4, pages 2617–2627, Hong Kong, China, March 2004. IEEE Computer Society.
- [12] B. Martin and B. Jano Eds. Wap binary xml content format. W3c recommendation, World Wide Web Consortium, June 1999. <http://www.w3.org/TR/wbxml/>.
- [13] E. Rissanen, H. Lockhart, and T. Moses Eds. XACML v3.0 administrative policy. Standard, Organization for the Advancement of Structured Information Standards (OASIS), June 2006. <http://www.oasis-open.org/committees/xacml>.
- [14] J. Vollbrecht, P. Calhoun, S. Farrell, L. Gommans, G. Gross, B. de Bruijn, C. de Laat, M. Holdrege, and D. Spence. AAA Authorization Framework. Request For Comments (RFC) 2904, Internet Engineering Task Force (IETF), August 2000. <http://www.ietf.org/rfc/rfc2904.txt>.
- [15] W. Wang and B. Bhargava. Key Distribution and Update for Secure Inter-group Multicast Communication. In *Proceedings of the 3rd ACM workshop on Security of ad hoc and sensor networks (SASN)*, pages 43–52, Alexandria, VA, USA, 2005. ACM.
- [16] W. Yu, Y. Sun, and R. Liu. Minimization of Rekeying Cost for Contributory Group Communications. In *Proceedings of Global Telecommunications Conference GLOBECOM*, volume 3, pages 1716–1720, St. Louis, MO, USA, November 2005. IEEE Computer Society.