Pseudonymous Authentication and Authorization enhancing ubiquitous Identity Management

Thomas Hildmann, Thomas J. Wilke

Forschungszentrum für Netzwerktechnologien und Multimedia-Anwendungen (FSP-PV), Sekr. MA073, Strasse des 17. Juni 136, 10623 Berlin

{hildmann | tjw}@prz.tu-berlin.de

Abstract

Security management for complex IT-infrastructures with heterogeneous components is a demanding challenge. One aspect of this undertaking is the ubiquitous identity management ensuring homogeneous security quality of the authentication and the authorization. Different to common praxis employing an implicit authorization within identity management systems, this contribution will present and discuss an identity management approach with an explicit authorization mechanism.

1 Introduction

Granting or denying access to a digital resource would not be an issue when we would have to deal with granting or denying only. The real problem is giving access to authorized persons but denying it to all others [Schn00].

In the early days of computers access control could be managed by knowing each other and having access to the computer console. Nowadays access control is hard to manage. More and more organizations are working on identity management systems. This can cover the problem of identification of persons in a distributed environment, but is no solution for authorization. Managing the authorization implicitly or with lean structures (e.g. with attributes in the identity management system, IDMS) can be a solution for some application fields. We strongly recommend an explicit authorization using an ubiquitous role-based access control model. Such an approach does not need a big-brother component which is able to track all users in a system. Pseudonymity can be realized during authentication and authorization.

This article is structured in four parts: In a first step the security related problems of an ubiquitous identity management in terms of transparency, data protection, privacy and support of multi administrative authorities are analyzed and requirements are rolled out. Mechanisms of new security paradigms like multi lateral security, enhanced data protection, privacy and multi administrative domains are introduced, resolving the conflicting security requirements. In a second step several architectures are presented which cover the discussed security requirements within various qualities. The impact and appropriate scenarios for the different architectures are discussed afterwards. Finally the applied strategies, implemented mechanisms and presented architectures are summarized and compared to the common approaches.

2 Requirements and Mechanisms

We examine a client-server based network system with different services (e.g. Web-based services). Many authorization systems do so by considering the requirements of the system maintainers only. Nowadays more and more the requirements of the system's users come to the fore, this includes also the privacy of users. Coming from the classical model where authorization follows authentication, the first can be implemented using a smart-card based pseudonymous authentication system (described later) or achieved with any other authentication fitting the requirements. The following sections will describe the used mechanisms and the requirements we are interested in.

2.1 RBAC-Model

The NIST standard for role-based access control [FSG+01] defines a hierarchical RBAC model. The model discussed in this paper is derived from this. The UML representation of this model is presented in [ShAh00], here we are using a simplified version of the class diagram.

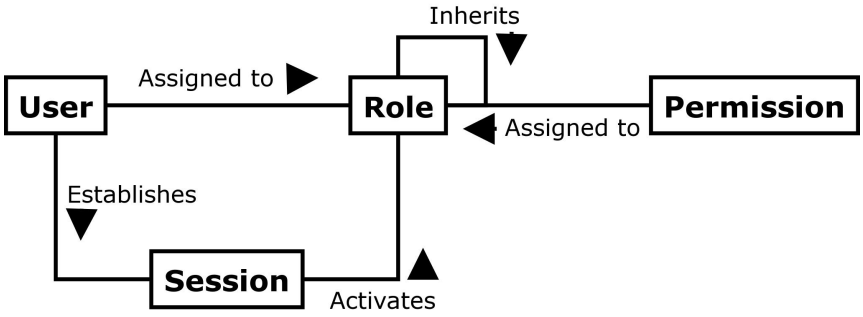

Fig. 1: UML Class Diagram: Hierarchical RBAC Model

A user is assigned to one or more roles which can inherit other roles. With this construct a role hierarchy can be build. Permissions (objects and operations on these objects) can be assigned to these roles. When a user establishes a session this activates one or more roles for this user.

This model works perfectly for a single application. To realize a ubiquitous model for an environment with multiple applications, it is necessary to separate a common and an application specific part of the model.

Figure 2 shows an extended model where "role" is separated into three role-types:

- SBR: Structure Business Role
- ABR: Application Business Role
- Access Role

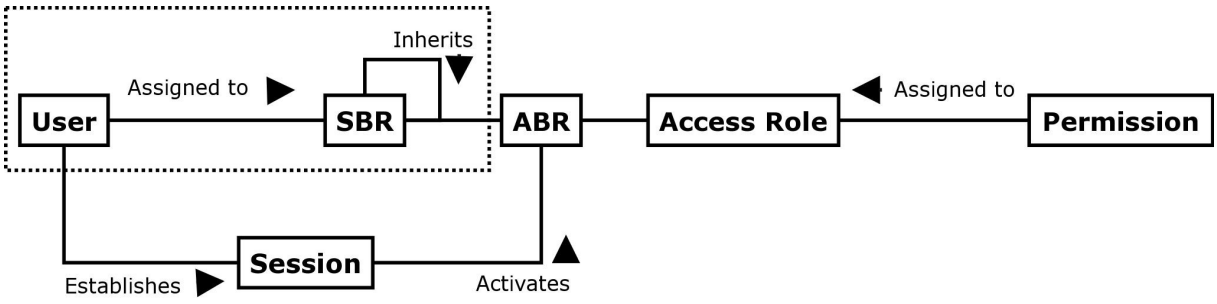

Fig. 2: UML Class Diagram: Distributed RBAC Model

We distinguish between access and business roles, where business roles are roles from a business point of view (like “project leader”, “programmer”, etc.) and access roles are roles from a more technical point of view (like “database administrator”, “report generator”, etc.). Business roles are divided into structure business roles SBR and application business roles ABR where the first ones are representing the organisational structure and the second ones the application relevant business roles. Typically some ABRs have a 1:1 mapping to corresponding SBRs, sometimes with different vocabulary depending on application’s context.

The dotted box contains the ubiquitous, application independent model. This is typically the faster changing part. Users can be administered using a classical identity management system (IDM). Further the structure of the organisation or community must be modelled. This part normally does not change as much as the users. At least the ABR model is only changing if there are changes within the application. The ABRs must be mapped to the SBRs in an appropriate way. Details will be given later.

2.2 ADF/AEF Authorization

The authorization can be separated into two different components: The access enforcement facility (AEF) and the access decision facility (ADF). A subject requests access to an object to the AEF. The AEF confirms this query on the ADF. Then this query is answered by the ADF and access is granted or denied. Application, AEF and ADF can be implemented in different modules, programs or even on different machines. The AEF component can be part of an application level firewall for example. The ADF component can be centralised to implement a common security policy, to base access decisions on up-to-date access control information or to reduce administration efforts.

Please note that parts of the RBAC model can be administrated by different users (e.g. in parts by the users themselves). Administrative roles can be defined to secure the model itself. This separation of duty increases the security of the system, distributes the effort, can make the security policy more transparent for the users and can prevent latency in case of changes. The ADF itself can be distributed again and can be used to interact with different institutes, locations or even between different organisations or companies [HiBa99].

2.3 User-Tracking: Identity vs. Privacy

One of the main issues of a centralized authorization is tracking information which can be gathered on the ADF server. Assuming every application queries the ADF to check the au-

thorization of a user, the ADF is able to gather the user-id, the used application, eventually every accessed object, date and time. This issue can be solved in different ways. Some examples are:

1. The ADF component can be distributed and AEFs can randomly change their used ADF component. In this case the data the access decisions are based on must be mirrored or must be accessible by each ADF instance.
2. AEF can randomly generate “fake queries” to mask the real access queries. But this causes traffic and raises the load of the ADF.
3. Knowledge of the requests can be distributed in a way that gives incomplete information to every party.

It is a usual requirement that a complete log can be generated in case of an incident. This is possible by synchronizing the logs of all evolved systems and gathers the information on demand. The requirement needs it to be a great effort in time and personal engagement when demanded. The implementation has to be balanced so that user tracking is too difficult for every day work but easy enough for forensic analysis for example.

2.4 Smart-card-based pseudonymous Authentication

The pseudonymous authentication with smart-cards is described in [Hild01]. The basic idea of this authentication process is: A smart-card initiates a public-key three way authentication with one authentication server (randomly chosen). The authentication server can check a card-revocation list by using a card-identifier which can not be mapped to a user. The user-identifier is encrypted by the smart-card and can be decrypted by the AEF of the application. After this authentication the validity of the smart-card is checked, the user is identified. But only the application has a user-identifier. The authentication servers just get parts of the information.

3 Architectures

Covering authentication and authorization processes comprises that there are always parties knowing at least some parts of the facts. Distributing the knowledge to a number of parties having divergent interests in the best case is the basic idea. Information distribution and avoidance are principles of multilateral security. The advantages of the implementation of these principles are fairly discussed [MüRa99]. With the given requirements and principles in mind there are still several decisions to make. A general question is: Which data represents the user-session? The second question is: How to deploy a system to cover the requirements and realize the principles?

3.1 Session Representation Method

The RBAC model typically activates a role for a user during a session. Many authentication systems simply bind all relevant data of the user to a session (e.g. after logging in on a Unix system the username, real-name eventually phone number etc. pp. can be discovered). A unique identifier, a kind of pseudonym is enough for authentication. In a role-based environment the activated role for the current session is enough to check the subject's permission. It is possible to use the pseudonymous identity for role activation only. Supposing the role activation is done by a separated instance an AEF is able to handle the access decision requests without any knowledge of the user's identity.

In a Web-based environment the session is typically represented by a browser cookie or a session identifier linked to the URL. This session identifier can hold the activated role itself or a reference to the role or role list. It is also imaginable to hold the role or role list in the server application and to use an application-specific identifier for the session.

From a dataflow point of view the role-information can be stored at the client (cookie or URL with role-information), at the server (application session id) or at the ADF instance (ADF session id). Please notice that in case of a server or ADF storage of the role-information setting a cookie or adding a session id on client-side is still necessary to handle the Web-session.

In all cases the role-information can be encrypted in a way that only the ADF is able to read it. On the other hand it seems to look like Voodoo for application developers not to know the role of the user working with the application although this knowledge is not necessary to check the permissions.

Storing the role-information on the client side seems to be a good way in most cases. It prevents other components from handling with timeouts, with great amounts of session data or orphaned sessions. A client has to store single session information only and closing the browser or surfing to other websites will be handled correct for the session.

3.2 Knowledge Deployment Methods for Access Queries

Most of the current applications are designed in a way that integrates AEF and ADF within the application itself. Often AEF and ADF are mixed up and permissions are handled implicitly. Applications that are designed this way are not able to share the knowledge of the user/role structures or easily implement a common policy. A good component design is able to share the user/structure data with other applications and manages the application specific data on its own. Therefore we suggest to separate the ADF component in two parts: An ubiquitous user/structure part and an application specific part. Each application has its own point of view. This point of view depends on the application's requirements. On the other hand the application's role-model will fit into the overall role-model. This is where our application business roles (ABR) are mapped to the structure business roles (SBR). This division seems to be a good place for distributing the user-tracking knowledge. Assuming an application specific ADF and an organisation specific ADF it is possible to separate information about the user's actions in different ways.

1.1.1 ADF-Blinding

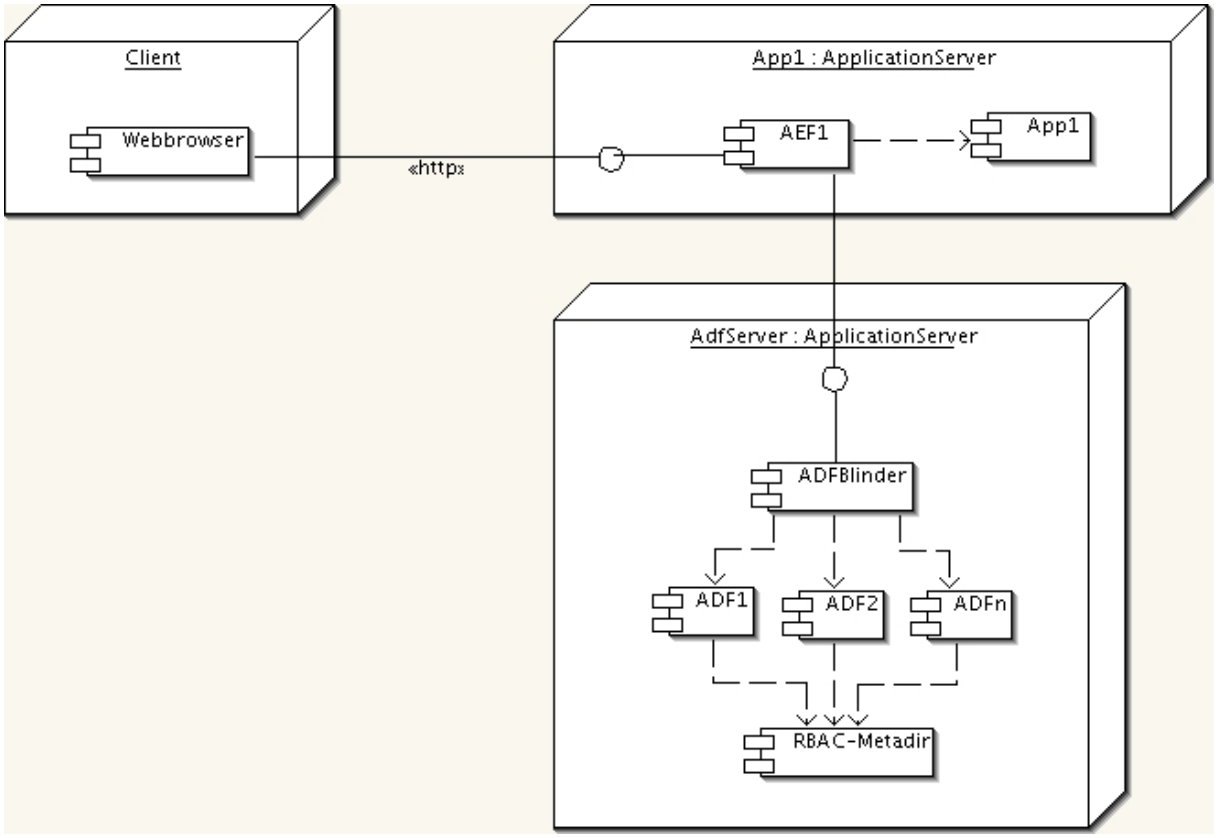

Fig. 3: UML Deployment Diagram: ADFBlinder Architecture

Figure 3 shows a system architecture using a blinding mechanism to distribute the knowledge about the user behaviour. Each application (App_1 .. App_n) holds two symmetric cryptographic keys ($k_{1..n}$ and $l_{1..n}$). The *AdfServer* is able to isolate processes. Depending on the used operating system there are techniques like jails, compartments or software partitions to realize this. There are *ADF*-instances (ADF_1 .. ADF_n) for each application (App_1 .. App_n). The ADF_n instance holds the key k_n for the corresponding application App_n. The *ADFBlinder* holds a list of the keys $l_{1..n}$.

To do an authorization query the following protocol can be used:

- The authorization query for App_1 is encrypted with k_1.
- AEF_1 encrypts the application id of App_1 with l_1. Typically a random value must be encrypted together with this id to prevent the ciphers from looking the same after encryption and make comparisons of the ciphers impossible.
- The encrypted query and the encrypted application id are sent to the *ADF*-interface.
- *ADFBlinder* is able to decrypt the application id using l_1.
- The query is forwarded to the corresponding ADF_1.
- ADF_1 is able to decrypt the query.
- Again the answer is encrypted with k_1 and is send back to AEF_1 the same way.

1.1.2 Hiding of Structure-Application-Mapping

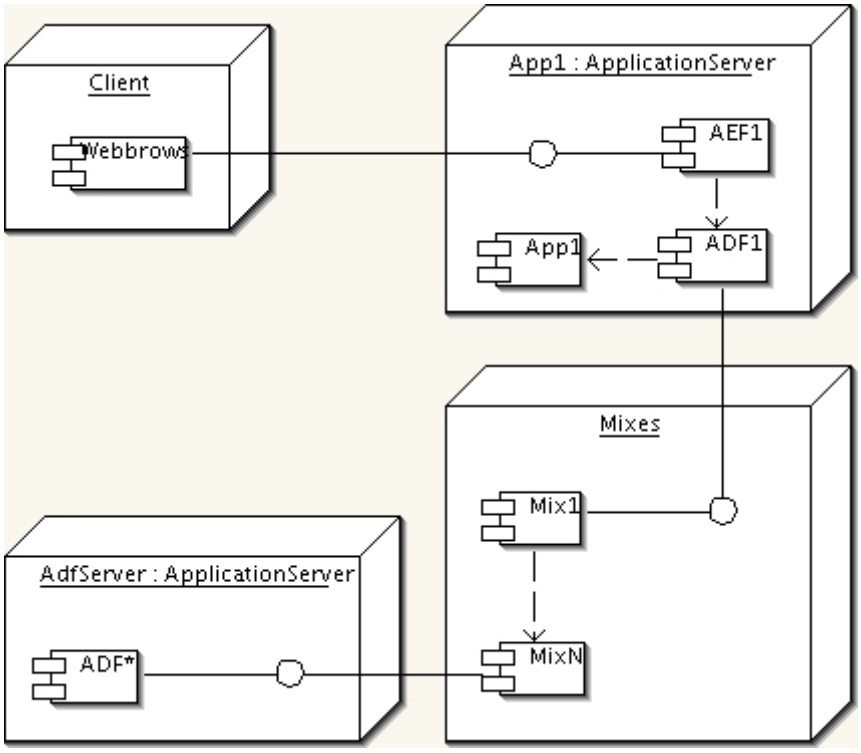

Fig. 4: UML Deployment Diagram: Hiding of Structure-Application-Mapping

The classical solution using Chaume's Mixes [Chau81] is shown in figure 4. This architecture shows that the ADF_n-Part can also be settled on the application's machine. The centralized *ADF* component (ADF_*) is used for the *user* → structure business role(s), *SBR* → application business role, *ABR* mapping. ADF_n is able to compose queries about the user and *SBR*. This query is randomly sent through a cascade of mixes. This makes it impossible for ADF_* to re-construct which application is used by the user.

A protocol example:

- AEF_1 sends an authorization request to ADF_1.
- ADF_1 composes a list of possible *SBR* mapping-roles.
- The *SBR*-list and the session id is send via mixes to ADF_*.
- ADF_* is able to check the membership of the user according to the session id.
- The membership-list is send back via mixes to ADF_1.

The mixes are used to hide the identity of the ADF_n initiating the mapping process from the ADF_*. This enables ADF_* to record the user-role-relationships but prohibits knowledge of the application the user utilised.

1.1.3 Isolated ADF-Components

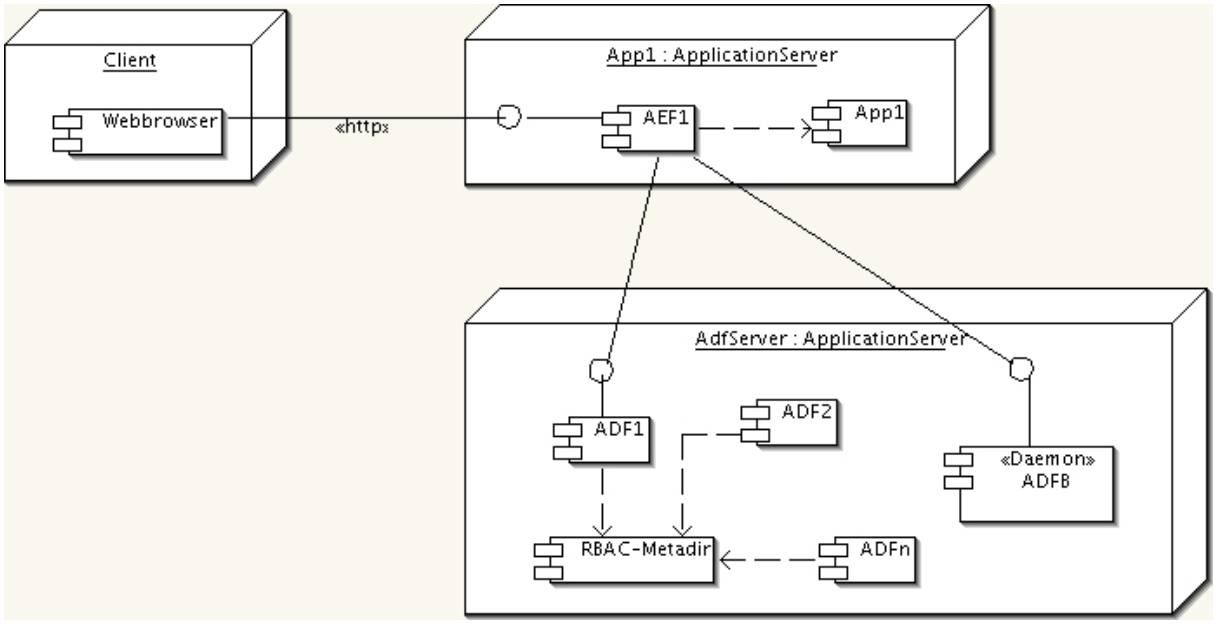

Fig. 5: UML Deployment Diagram: Isolated ADF-Components

Another possible implementation is using a broker architecture (see figure 5). This scenario needs no further encryption. It depends on a cryptographic end-to-end connection only.

- AEF_1 connects to the broker *ADFB* (Access Decision Facility Broker) and sends its application id.
- The *ADFB* returns the pointer to the ADF-instance ADF_1.
- AEF_1 connects to ADF_1 and sends the query.
- The following communication is *AEF-ADF*-bilateral.

The privacy enhancement results from the separation of ADF-components for each application. Only application-dependent user tracking data is gained by the application corresponding ADF instances (i.e. ADF_1 just has knowledge about user behaviour on App_1). This architecture assumes a passive *RBAC-Metadir* (meta-directory) component which is unable to log accesses by the $ADF_{1..n}$ components. If not all relevant data could be gathered by the meta-directory.

4 Discussion of the Scenarios

The ADFBlinder-approach is a common cryptographic driven architecture. The architecture just needs a simple network connection and jailed processes. Weakness of this architecture is the RBAC-meta-directory. This component is able to identify the ADF-instances and can possibly map them to the according application. The meta-directory also holds the data to map users to roles etc. This means it must be ensured that the meta-directory is not able to exploit

the queries. This may be the fact if the meta-directory is a simple filesystem and tracking can only be done by the kernel. The meta-directory can also be distributed, masked with mixes etc. It seems that this architecture just shifts the user-tracking problem.

Hiding the mapping between user-structure and application-permission has got several advantages: No additional cryptography is needed and mixes are a well-investigated area. On the other hand driving mixes raises the effort of the system. To make mixes work properly they should run on different machines and be run by administrators with different interests. But this aims for the ADF-instances, too. Depending on the security requirements it would be a much better idea running the different application depending ADF-instances on different machines.

The isolated ADF-components approach is the simplest one. It needs no additional encryption and no mixes. On the other hand it has also the known meta-directory issue. To make this architecture even more simple the according ADF-instance can be hard-coded (or configured) into the AEF.

Pseudonymous authorization can also be realized using credentials [Bisk02]. Credential-based authorization uncouples the access and the authorization requests. The problem of such an approach is the limit of the fine grain of the access control model. One strategy could be to have a great key-ring of credentials on client-side or to make ADF-requests more often. Getting the right credential just in time re-couples the process again. Combining the discussed architectures with credential based authorization looks promising. Credentials could be used to solve the meta-directory issue when used between the $ADF_{1..n}$- and *RBAC-Metadir*-Components. On the other hand the mentioned session-identifiers are already a kind of credentials depending on their content.

5 Summary

Employing a ubiquitous identity management has several advantages. Some advantages are based upon the implicated centralized structure which enables homogeneous policy enforcement for credential management and assignment. But there are also some undesired effects which have to be considered seriously in terms ensuring the security quality of the authentication and authorization processes, the access availability to services for legitimated individuals, illegitimate traceability and so on. As shown there are different possibilities to address parts of the known issues. Every architecture has its own implication. It depends on the specific requirements of the project which architecture is suited best. The shown architectures are surely not the only possible ones. Discussing this work we found an uncountable number of derivates and modifications. Completely different solutions are also possible. These are just examples to show how simple pseudonymous authentication can be implemented and to initiate a discussion about such enhanced identity management solutions with privacy aiding technologies.

6 Outlook

We are currently implementing a distributed role-based identity and authorization management system at the Technical University of Berlin. We decided to use the third architecture because its straightforwardness. The discussed meta-directory issue does not affect our implementation because of parts of this component works on the filesystem and the kernel is assumed to be secure. After finishing the first release we think about improvements, such as non-repudiation features to have processes transparent to all parties. It is of interest to research

the correlation and implication of fine grained access models and used architectures to ensure finding the best possible architecture to fit the demands. To lower administrative efforts more work about administration of security models and administrative roles has to be done first. Every idea and feedback is welcome.

7 Acknowledgment

We want to thank the whole FSP-PV/PRZ TUB IT security team for their implementation-work on parts of the presented ideas, for many fertile discussions and useful hinds. Our special thanks go to Klaus Hamann and Thomas Gebhardt for critical reviewing of pre-versions of this article.

8 Literature

[Bisk02] Joachim Biskup: Credential-basierte Zugriffskontrolle: Wurzeln und ein Ausblick; GI Jahrestagung 2002

[Chau81] David Chaum: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms; Communications of the ACM 24/2 (1981) 84-88.

[FSG+01] David F. Ferraiolo, Ravi Sandhu, Serban Gavrila, D. Richard Kuhn, Ramaswamy Chandramouli: Proposed NIST Standard for Role-Based Access Control; ACM Transactions on Information and System Security, Vol. 4, No. 3, August 2001, Pages 224-274.

[HiBa99] Thomas Hildmann, Jörg Bartholdt: Managing trust between collaborating companies using outsourced role based access control: Proceedings of the fourth ACM workshop on Role-based access control, Fairfax, Virginia, United States 1999, Pages: 105 – 111.

[Hild01] Thomas Hildmann: Vermeidung von Datenspuren bei smartcard-basierten Authentisierungssystemen: Proceedings der GI-Fachtagung VIS 2001 12.-14.09.2011 in Kiel, Vieweg Verlag, Seiten: 163-178.

[MüRa99] Günter Müller, Kai Rannenberg: Multilateral Security in Communications – Technology, Infrastructure, Economy. Addison-Wesley-Longman, München, Reading (Massachusetts) u.a. 1999.

[Schn00] Bruce Schneier: Secret & Lies – Digital Security in a Networked World. John Wiley & Sons, Inc. 2000.

[ShAh00] Michael E. Shin, Gail-John Ahn: UML-Based Representation of Role-Based Access Control; Proceedings of the 9th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises 2000, Pages: 195 - 200