AN IDENTITY BASED FRAMEWORK FOR SECURITY AND PRIVACY IN PERVASIVE NETWORKS

PARIJAT MISHRA

(B.Eng. (Hons.) NUS)

A THESIS SUBMITTED
FOR THE DEGREE OF MASTER OF ENGINEERING
DEPARTMENT OF ELECTRICAL AND COMPUTER
ENGINEERING
NATIONAL UNIVERSITY OF SINGAPORE
2005

ACKNOWLEDGEMENTS

This thesis is dedicated to the Open Source community.

I would like to thank colleagues in the Daidalos consortium for all the feedback, and especially Pedro Brandão for all the help he gave me; the long-suffering Sukanta for listening to my rants; Jaya Shankar for the trust and encouragement; and Dr. Winston Seah for the freedom to explore.

Table of Contents

Summary

Management of digital identities in current systems is an increasingly important tool to achieve integration and increase efficiency. It is even more essential in pervasive networks. This thesis presents the results in the analysis and design of a conceptual model for management of identities and their inter-relationships for a pervasive computing platform in a future all-IPv6 integrated network. The relevant characteristics of these networks, and the challenges of a multi-provider service-offer and composition architecture, are described. In particular, the security and privacy requirements of such an architecture are examined. A model of stakeholder identities is then developed, showing how it meets privacy requirements, enables the management of identities, and leverages them to make deploying and composing services in such networks easier. Special consideration is given to federated architectures. We balance the need to limit access to private user information, with the conflicting need to have such information to enable personalized service delivery. The model's usage is described in the context of a flexible authentication and authorization framework. The framework's use and implmentation in order to achieve privacy is described. We conclude with a discussion of related efforts, and their comparison with our framework.

List of Tables

List of Figures

Chapter 1

Introduction

Two phenomena emerging lately have changed the lives of millions of people and affected the way people, organizations and governments conduct their work. They are, respectively, mobile telephony, and the Internet. The effect of the former has been, largely, that people expect to be able to communicate anywhere, anytime. The effect of the latter has been to enable people to access a vast amount of information at the click of a button.

Interaction between different network types is becoming quite common. The POTS network, the cellular network, and the Internet now are connected to each other, with information flowing both ways between them. The Internet—a network of networks—has itself always been heterogenous. From the end user's point of view, technologies like WiFi, 100Mbit and Gigabit Ethernet, and GPRS are jostling side-by-side as candidate access technologies, and offering a variety of choices in terms of cost, ubiquity, and bandwidth.

These developments have given rise to a vision of a global *converged network*, offering users *pervasive services* that combine various kinds of network connectivity, over several modalities: voice and text messaging; email and instant messaging; real time location and presence information; and, web based services.

1.1 Background: FP6 and Daidalos

The research work described in this thesis was undertaken as a part of a project, Daidalos, funded partially by the European Commission under the *Sixth European Framework Programme for Research and Technological Development (FP6).* FP6 provides funding of more than € 17 billion for various activities, including projects and testbeds, during 2002–2006 in order to promote research activities and strengthen the scientific and technological bases of industry and encourage its international competitiveness. About € 12 billion are devoted to research projects, which are organized into various Thematic Areas. One such area is the *Information Society Technologies (IST)* "priority".[1]

Daidalos (IST-2002-506997) is an Integrated Project funded under the IST priority within FP6. The projected budget of Daidalos amounts to € 25.7 million, out of which € 14.7 million are funded by the European Commission. Daidalos officially started on Nov 1, 2003 and has a duration of 30 months. Forty-six partners from academia and industry participate in Daidalos.[2]

Daidalos is a very large project. We shall be unable to discuss the complete architecture of Daidalos, and in this thesis shall focus on mainly one aspect. However, our research problem and focus area, as well as the specific solutions sought, to a great extent, are motivated and influenced by the totality of the project's goals. Hence it will be appropriate to have a brief look at Daidalos's objectives and vision.

> **Motivation.** Mobility has become a central aspect of the lives of European citizens - in business, education, and leisure. Due to rapid technological and societal changes, there has been a bewildering proliferation of technologies and services for mobile users. This has created a complex and confusing communications environment for both users and network operators. Further development of existing technologies, and the addition of new ones in Beyond 3G (B3G) systems, will necessitate a rethinking of fundamental technological issues in order to create user-centred and manageable communication infrastructures for the future.
>
> **Vision.** The vision of Daidalos is of a world in which:
>
> - Mobile users can enjoy a diverse range of personalized services - seamlessly supported by the underlying technology and transparently provided through a pervasive interface;
> - Mobility has been fully established through open, scalable and seamless integration of a complementary range of heterogeneous network technologies;
> - Network and service operators are able to develop new business activities and provide profitable services in such an integrated mobile world.

[1]The FP6 official website is at `fp6.cordis.lu/fp6/home.cfm`.
[2]The Daidalos official web-site is at `www.ist-daidalos.org`.

Objectives. The objective of Daidalos is to develop and demonstrate an open architecture based on a common network protocol (IPv6), that becomes a significant step towards approaching the Daidalos vision. The overall Daidalos objectives are to:

- Design, prototype and validate the necessary infrastructure and components for efficient distribution of services over diverse network technologies beyond 3G;
- Integrate complementary network technologies to provide pervasive and user-centred access to these services;
- Develop an optimized signalling system for communication and management support in these networks;
- Demonstrate the results of the work through strong focus on user-centered and scenario-based development of technology.

As can be seen above, Daidalos approaches the area of pervasive networking from two different points-of-view: *operators* want to develop new business models and create innovative services; and *users'* want to have personalized, ubiquitous services.

In the next section we shall look deeper into Daidalos's technical aspects.

1.2 A Pervasive Network Beyond 3G

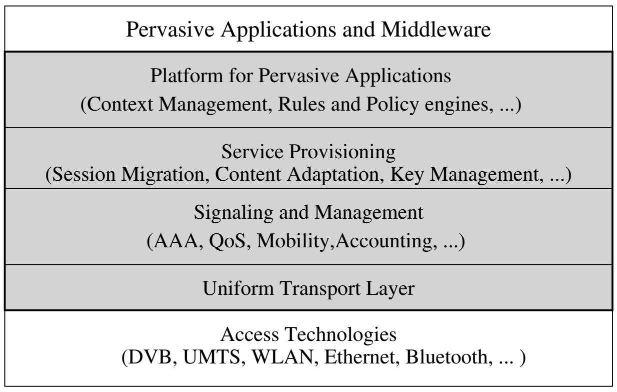

Pervasive Applications and Middleware
Platform for Pervasive Applications
(Context Management, Rules and Policy engines, ...)
Service Provisioning
(Session Migration, Content Adaptation, Key Management, ...)
Signaling and Management
(AAA, QoS, Mobility,Accounting, ...)
Uniform Transport Layer
Access Technologies
(DVB, UMTS, WLAN, Ethernet, Bluetooth, ...)

Figure 1.1: Daidalos Work Scope

Daidalos works to integrate heterogeneous access technologies to create a seamless all-IPv6 infrastructure [24]. Over this infrastructure, a platform for provisioning of services is deployed, containing necessary network management components and protocols. Utilizing this as a base, a pervasive computing architecture is created, exposing an Application Programming Interface (API) and framework over which pervasive, context-aware applications may be developed. This is illustrated in Figure 1.1; the shaded blocks indicates work within the scope of Daidalos. Daidalos does not create new access technologies, or (apart from proofs-of-concept) implement applcations.

Daidalos proposes a canonical platform for service provisioning, including traditional components such as:

- Authentication, Authorization and Accounting (AAA) [2] servers;
- QoS Brokers and related components;
- A Key Management system, such as a Public Key Infrastructure (PKI).

The platform also includes some non-traditional components, such as:

- A Policy Based Network Management System (PBNMS);
- Mobility Management support through Mobile-IPv6 Home Agent (HA)s;
- Multimedia signaling support infrastructure through Session Initiation Protocol (SIP).

On the user side, Daidalos proposes a end-user terminal software environment that will enable seamlessly mobile, secure communication and access to services.

1.3 Research Overview

In this thesis, we shall examine how to provide security and privacy for pervasive services. As an example of a pervasive service, imagine that you are viewing a news video on a home desktop computer. Then you get up to leave for the airport, but you can continue to watch the news on the way in the taxi, and while waiting in the air-port lounge because the news video has switched seamlessy to your 3G and WiFi enabled PDA. Meanwhile, in the background, you are charged by the airport the 3G service provider and the airport WiFi operator for streaming the news to you.

This simple scenario raises several questions. How does the system know *when* you left home or entered the airport? How does the system know that you would like to switch the video to your handheld? How do the WiFi/3G operators know what video you were watching, and how do they charge you?

1.3.1 Challenges in Pervasive Networks

As far as access to the Internet is concerned, there are a range of technologies available. The primary means are dial-up, Ethernet and WiFi. With the advent of 3G cellular systems, fast internet access on mobile phones is becoming a reality, as well. However, the user is acutely aware of the differences between these technologies. One may not, today, switch from using Ethernet to WiFi without interrupting the operations of networked applications. The heterogenous nature of the Internet, therefore, is painfully obvious.

Many initiatives, such as the project Moby Dick [10], have sought to integrate heterogeneous networks into an all-IP infrastructure. Not only are multiple access technologies accessible using IP or IPv6, it is possible to move network connections from one access technology to another seamlessly without dropping (many) packets. In addition, security and QoS management are added to the infrastructure, such that integrity and confidentiality of data, and service guarantees for multimedia, can be assured.

Another trend is that services available over the Internet are proliferating, giving the user a bewildering range of choices. A user typically "signs-up" with the provider of each service he wishes to consume, creating an "account" at each provider. The number of "accounts" an average user of the internet has is growing rapidly, making it more and more difficult for users to cope with the management overhead. *How may one reduce the exploding number of accounts and simplify their management?*

In our example of a pervasive service, above, you would like to be able to use the airport WiFi service, with QoS guarantees (you are watching a video, and it should not be too jerky!) without having to explicitly open an account with the service provider. To truly enjoy pervasive services, users would need either automated ways to "sign-up" for a service on the fly without much rigamarole, or be able to enjoy services without needing to have an account at all. Yet, the WiFi service provider, and providers in general, would like to charge you for their service. *How may providers charge for services that may be discovered and used dynamically, perhaps only once, without a prior business relationship with the consumer?*

Such paid services can become popular only if the charges are affordable. Thus, micropayments—on the order of cents rather than dollars—will have to become possible. However, the system needed to enable micropayments is quite complex and burdensome and it is unlikely that every new and small operator would invest in such a system. Micropayment solutions, historically, have failed to take off simply because for an average merchant to process the micropayment imposes more cost than the payment itself. *How may we offer providers an easy way of deploying services and charge for them? How may we assure the user that a provider he has had no contact with before is trustworthy—that the provider will charge him only so much and not empty his bank account?*

The true promise of pervasive services will be delivered when the user may compose novel services from many small ones. In our example above, when you were watching the news video on your PDA in the airport lounge, you could be combining the news video streaming service with a content adaptation service. The latter scales down the video resolution to a level that is useful for the PDA, saving you the bandwidth costs. *How may be create services that can be put together in different ways to create new and richer services? How can the provider of, say, content adaptation services, negotiate with the provider of news videos on your behalf, to scale the video?*

These are some of the broad questions we have looked at during the course of the project, and lead directly to our thesis statement:

> **Thesis statement:** *It is possible to use Identity Management as a basic building block around which a comprehensive security and privacy architecture can be built for pervasive systems; the architecture provides flexible services—such as authentication and authorization—to a diverse set of clients, including but not limited to, web services, and network authentication protocols.*

1.3.2 Research Goals and Achievements

Addressing the problems mentioned in the previous section has been one of the focuses of the Daidalos project. The author has been responsible for the design and specification of a Key Management framework for providing advanced security services to other components in the project. The author was also personally involved in the development of the Identity Model that serves to tie together disparate areas of the project. Both of these areas are part of the larger solution to the challenges of pervasive networks.

In this thesis we shall not address the problems identified above in full generality, but shall limit ourselves to questions of security and privacy, and shall briefly touch on charging.

We have specified a terminology and a conceptual model, that we call the Identity Model, to discuss the above problems and possible solutions in what we believe to be a coherent manner. We have built a security framework around this model. We have designed a flexible authentication and authorization system that can solve several of the problems mentioned in the last section and achive security and privacy goals. We have implemented prototype software to test and demonstrate our design.

Our work builds heavily on identity management principles, and we draw inspiration from other identity management projects, such as Liberty Alliance [22], Shibboleth [8], and Web Services Federation [5].

However, we believe our model is more general than these projects. The authentication and authorization framework that we have designed are conceptually very flexible. By building on the model, our authorization framework can treat different kinds of services in a uniform manner. In fact, even basic network access can be treated like any other service (albeit, it has to be the first service to be accessed). In Chapter 2 we shall take a look at these projects and compare their goals to our own.

We believe that our model could be applied to situations other than the one we have envisaged: it could be useful, perhaps, for analyzing any collection of heterogeneous networks where users interact with multiple providers to consume personalized services.

We defend our claims by giving representative scenarios and describing how the model applies, rather than quantitative measurements. First, it is not clear what measurements would be valid and useful, and secondly, the measured values would depend heavily on the implementation of software and protocols.

We have published some results described in this thesis as [26] and [30] in the IST Mobile Summit 2005. The papers were invited for poster sessions.

Chapter 2

Related Work and Comparison

Identity management is a growing concern in many industries, including solution vendors and service providers. This has led to some initiatives to define a standard for identity management. The most prominent ones are the WS-Federation [5] suite of specifications, and the ones from the Liberty Alliance [22][1] group.

2.1 Web-Services Federation and Liberty Alliance

WS-Federation is a part of the WS-* [29] initiative led by IBM and Microsoft. This standard bases its functionalities in other WS specifications to support a Federated Identity. Policy and privacy issues are tackled by the WS-Policy and WS-Privacy specifications.

Liberty Alliance (LA) is a consortium of over 150 companies that aim to produce standards to allow identity federation. Privacy is also one of the major concerns of LA. A comparison stufy [23] from LA compares the two approaches with some technical details. Although it is a little outdated due to recent changes in WS-*, it is a good source for the interested reader. Despite having some differences on technical details these two groups have similar conceptual models for the identity and its federation. So we describe them here jointly, pointing the differences.

In both models identities are only attributed to users. Users can possess several identities, that may (by user choice) be federated. The objectives are to provide: Single-Sign-On (SSO) and Single-Log-Out (SLO); information sharing between

[1]The specifications can be seen at http://www.projectliberty.org/resources/specifications.php.

organizations; access to attributes from identities (to provide ease-of-use to customers); and anonymous yet authorized access to resources. Both use pseudonyms to prevent correlation of identities by the service providers. A pseudonym relates to an identity and is specific to a service provider. They define a special entity that is responsible for validating the identities and keeping track of their relationships: the Identity Provider (IdP). The IdP must certify that a user presents the correct credentials for a given identity, and knows the other identities to which this one is related to. It will then issue tokens to allow service providers to ascertain the identity that pertains to them, along with its current authentication/authorization status. These tokens are the pseudonyms that the IdP relates to the correct identity for the service. When a specific identity is not required by the service, the IdP can just vouch for the authentication of the federated identity, providing anonymous access.

It will be easily perceived that our federation approach draws some ideas from these models. However, our focus extends to some different scenarios related to accounting and charging issues (as described in 1.3.1). Our basic identity model is more complex as it tries to cope with different requirements (again, business concepts related to accounting) than those of these groups. We also introduce the VID concept as a new layer to add to user experience and privacy.

2.2 Shibboleth

Shibboleth [8][2] is another federated administration that is being developed by Internet2 and MACE. Basically, the aim is to leave identity management in the user's origin site. That means that users do not have accounts in foreign sites, which rely on the user's home site (where he/she has an account) to provide the user's attributes to make authorization decisions. With Shibboleth, each site only administers its users and authorizes access based on the attributes (usually group-membership, such as "CS-Teacher@some-university.edu") that a user's home site provides. The user can control which attributes can be shared with each foreign site. Pure attributes based authorization makes some site personalization impossible but does provide privacy by default. Again the identity model basics are simpler than ours, given that

[2]An older draft [7] describes the architecture better, in our opinion.

business aspects are absent and privacy is dealt with controlled attribute release.

2.3 Other Perspectives

2.3.1 The Open Group Identity Management

The Open Group has produced a whitepaper [28] describing concepts, steps, technologies, rules of best practice for identity management. The document is a thorough description of the current issues related to identity management, providing guidelines and pointing current dangers for its deployment. It has some proposals on identity management issues[3], but maintains the conceptual identity model of the current specifications.

2.3.2 The PingIdentity Model

The PingIdentity model [9] addresses the concept of identity in layers, too. In this model, layer 4 identities are called “inferred identities” and are abstracted from attributes of underlying layers; examples would be “blue-eyes citizens” and “drivers from Utah”.

We consider them (a) application specific abstractions, requiring no more support from the underlying system than to expose the relevant attributes; and (b) not under the user’s control and not addressing privacy and security issues.

2.4 Comparison

A summary of the identity management projects discussed above, and differences from our goals, is presented in Table 2.1.

2.5 Limitations of Current Frameworks

2.5.1 Web-Centrism

The Identity Frameworks described above all suffer from the same problem: they are too web-centric. Of these, the WS-Federation specifications are the most flexi-

[3]A proposal for a unique identifier for identities is one good example.

Aspect	WS-Federation	Liberty Alliance	Shibboleth
Identity Provider	Distributed	Distributed	Must be User's Home Provider
Per-site Authentication	Available	Available	No, Only at Home Domain
Login Flexibility	Not mandated	Not mandated	Restricted to userid/password
Authorization Mechanism	Yes	Yes	No
Pseudonomity and Anonymity	Pseudonyms may be opaque	Pseudonyms are opaque	Only possible if attributes are generic (shared with a group)
Openness	Open Standard; implementation issues not defined	Open-Standard but may contain patented technologies	Open-Standard
Driver	Microsoft, IBM and BEA	Consortium	Internet2 and MACE

Table 2.1: Identity Management Frameworks

ble: they use technologies like HTTP and SOAP, but otherwise they can be used by other applications. Shibboleth and Liberty Alliance both assume the client application is a browser; their signaling protocol uses HTTP mechanisms and takes a lot of roundtrips.

2.5.2 Openness

The Liberty Alliance specifications are the most complete, and a significant number of products implementing these specifications are available. However, the copyright of the specifications says:

> "...certain elements of this Specification may require licenses under third party intellectual property rights, including without limitation, patent rights."

The identification of these elements is not the responsibility of the Liberty Alliance consortium. Also, derivative work is subject to licensing.

2.5.3 Authentication Mechanisms

One may assume, in general, that domains that are to be federated in an identity management system would likely have different authentication mechanisms. However, the standards above lack the flexibility to adopt new authentication mechanisms, or do not make explicit provisions for it.

2.5.4 Consumers versus Subscribers

In Daidalos, we consider the possibility that a person consuming the services need not be the one paying for the services. Two examples could be:

- A family may purchase multiple video-on-demand service accounts; but the children only consume the services, and the head of the family pays for the total consumption.
- A corporation may purchase internet connectivity and email service for all its employees. The corporation pays for the service usage. Perhaps different employees get mail-boxes of different sizes.

We would like to distinguish between these two different entities, and encompass them as part of the model. The frameworks mentioned above do not have a representation for the paying entity and in fact do not address billing and charging aspects in any significant way.

2.5.5 Services and Providers

As we mentioned earlier, one of the features of a pervasive network platform would be to allow providers and users to put together various existing services to create new ones. This requires that providers be able to authenticate each other in some way. In the current Internet, relationships between providers are established on a peer-to-peer basis by long-term legal contracts and service-level agreements. In particular, dynamic discovery and composition is not catered for. When the number of providers grows large, they will start facing problems similar to the ones we are attributing to the relationships between users and providrs. It is possible that services may themselves need to be identified in some manner (though likely not in the same way as users are).

Some of the frameworks mentioned above support dynamic discovery of services. For e.g., the WS-* specifications incorporate this using UDDI. However, we believe that a more uniform treatment of identities may achieve this in an elegant manner.

None of the identity frameworks we have described (and several others as well) attempt to ascribe identities to providers or services.

Chapter 3

System Architecture

The architecture diagram in Figure 3.1 shows the various functional subsystems that ought to be present in a Beyond-3G network. The diagram is a highly simplified version of the Daidalos architecture. We remove all components that are not related to AAA and Key Management. However, the representation of various kinds of operator networks is faithful to the original.

3.1 Functional Subsystems

The various functional blocks could be operated by different business entities, or the same entity. The architecture is meant to be policy-neutral.

In the former case, the different business entities would have an agreement between themselves to inter-operate. For example, one operator could be running an Access Network (AN), while another could be operating a Service Provider Network. There could be even yet another operator providing "just" content. They would like to interoperate in order to sell services to users. In that case, the functions in the various blocks would enable them to do so.

3.1.1 Terminals

The terminal is an "end-user" device. Common terminals can be laptops, PDAs, desktop computers. Other computing devices may also be used. When the terminal is "mobile", i.e., portable, then we refer to it as MT. For our purposes, there is no functional difference between the mobile and non-mobile terminal and we will use

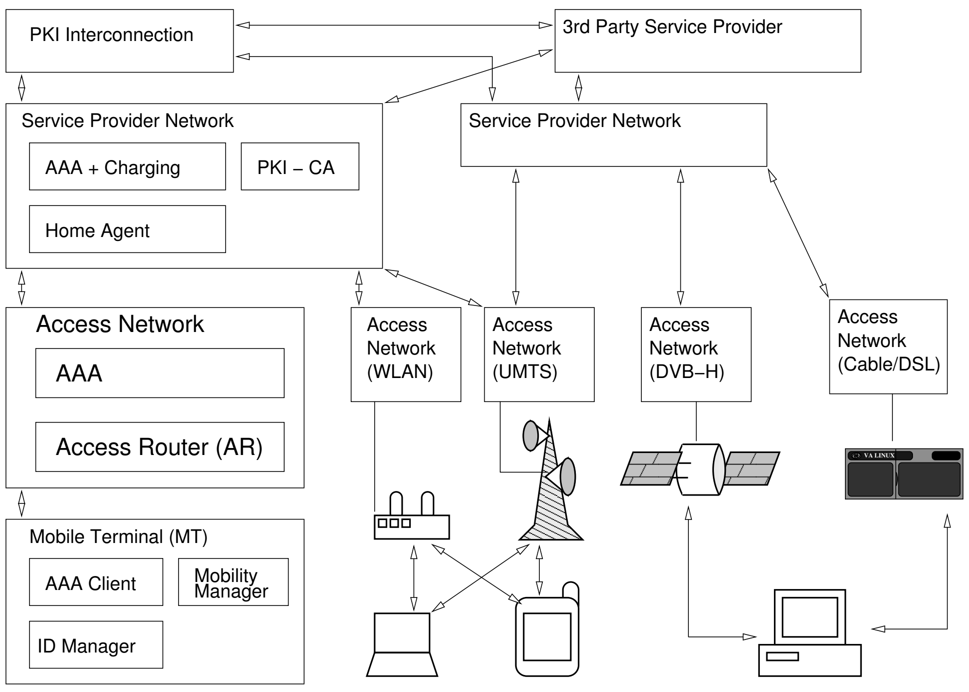


Figure 3.1: Logical Architecture

the terms interchangeably.

The terminal can connect to multiple ANs and make use of services in the AN, Service Provider Network, as well as from 3rd Party Service Providers.

The terminal contains functional blocks to authenticate to the network and services, and to manage mobility. It will also have components for managing QoS, for initiating and maintain multimedia sessions, and for monitoring/metering, but we omit them for brevity. We shall discuss the components related to authentication and authorization in greater detail below.

3.1.2 Access Networks

This is the wired or wireless network through which terminals get IP connectivity. A number of different technologies may be used, such as Wireless Local Area Network (WLAN), Universal Mobile Telecommunications System (UMTS), etc. An AN contains components important for managing network level operations, as opposed to provisioning of services.

The AAA component takes care of authentication, authorization, and accounting. The authentication and authorization will typically be related to network access and not services. It need not do the charging, which is delegated to a related Service

Provider operating a Service Platform.

The Access Router (AR) is not a functional block but a physical node. The AR is the first IP-level device that terminals see when connecting to the network. It is important because network access control is enforced at this node. Security over the "last hop" is also enabled by this node.

3.1.3 Service Provider Network

In this part, functional blocks related to provisiong of services are provided. We show a few of them: the PKI component takes care of inter-domain trust management; the AAA and Charging component takes care of authentication, authorization, accounting and charging for services. It may also take care of metering and monitoring, or there may be a separate component for that. The Home Agent components takes care os user mobility.

Other components like QoS Brokers, Multimedia Services Provisioning (for e.g., a SIP server or proxy), and PBNMS would also exist here, but we omit them for brevity.

3.1.4 Third Party Service Provider

Third Party Service Providers provide applications and content to the end-users. They could be part of a Service Provider Network, or outside it. They can use the facilities in a Service Provider Network to enable authentication, arrange QoS for content, to charge the user, and to use network information like the location of the user in order to enhance their services.

3.1.5 PKI Interconnection

PKI based key management architectures are designed to be highly scalable and secure. As such, they are the preferred choice for providing key management between domains. PKI interconnection can be based on three distinct PKI architectures.

Hierarchical Interconnection

In this architecture, interconnection between domains is made possible due to a root Certificate Authority (CA) that must be trusted by all users in all federated domains.

The advantages of this method are: (a) scalability – new domains are easily added; (b) certification paths are easy to develop; (c) certification paths are short.

However, this kind of interconnection would normally not work when the domains to be interconnected are managed by distinct administrations and have their own PKI. The choice of a trusted third party managing the root CA would be a difficult one. Even when such a choice has been made, the new root CA would have to become the trust anchor for all users in all domains, and everyone would have to update their software to replace the old trust anchors with the root CA's certificate.

Cross Certification, without Bridge

This mechanism for providing PKI interconnection does not require new entities to be added to the existing PKI infrastructure. It assumes that a specific CA in each domain (usually called the principal CA) be inter-connected with all other principal CAs of other domains through bi-directional, peer-to-peer relationships.

The main advantage of this architecture is that PKI users do not need to trust a new entity, but continue to rely on their respective CAs. Dispensing with a new entitiy also is an advantage in itself.

However, scalability issues arise very quickly with rising number of domains. Another disadvantage is that certification paths are not easy to build, and may even lead to infinite loops, due to the complex topology of the resulting architecture.

Cross Certification, with Bridge

In this method, all principal CAs in all domains establish a single peer-to-peer trust relationship with a new CA called the Bridge CA. The Bridge does not issue certificates to users, and the users continue to rely on their respective CAs. It is also more scalable that the approach without a Bridge. The only disadvantage is the creation of this new entity.

3.2 Roaming

Figure 3.2 shows a typical roaming scenario. The dashed lines show network level authentication/authorization flows. The solid lines show service level authentica-

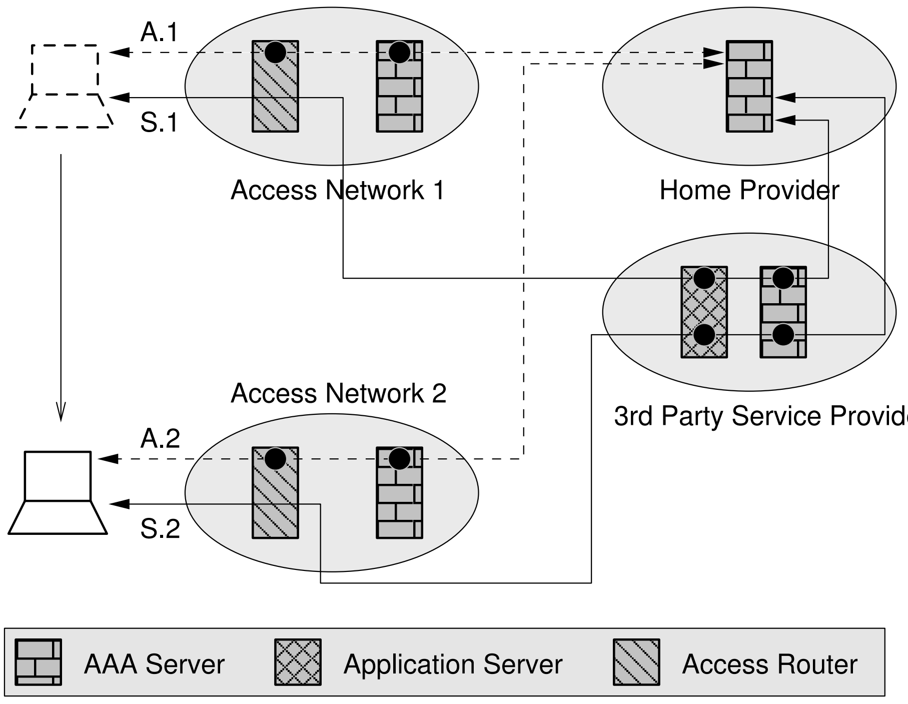


Figure 3.2: Simple Roaming Example

tion/authorization flows.

First, we assume that each MT is equipped with a set of credentials corresponding to the user of the terminal. The credentials could be a user-id and password pair, a digital certificate, or a One-Time-Password generating device.

The credentials are "issued" to the user by a service provider or operator, commonly called the Home Provider. It is the Home Provider that stores the information necessary to authenticate the user in its AAA servers and has the ability to authenticate the user. Commonly it is also the one that can determine the user's authorization to use various services, but other models are possible.

3.2.1 Network Access Control

Assume that at a certain point of time, a user's MT was getting internet access through AN 1. To get basic network access, the MT would have to authenticate to AN 1. The line A.1 shows which components would be involved in such a flow. The authenticating process on the MT would converse with a peer on the AR. The peer would likely not do the authentication itself, but forward requests to its local AAA server, and return responses from the AAA server to the MT.

The local AAA may or may not be the user's Home Provider. If it is, then it can immediately perform an authentication and return the result—Success or Failure—to the AR and MT.

In Figure 3.2 AN 1 is shown not to belong to the user's Home Provider. Such a network is commonly called a Visited Network or Foriegn Network. In this case, the AAA server of the Visited Network would determine the user's Home Provider from the authentication request.

If the Visited Network and Home Provider do not have a roaming agreement, then the user's authentiation request must be rejected. Assuming they do, the Visited Network's AAA server would forward the authentication request to the Home Provider's AAA server (over a secure channel). The Home Provider's AAA server would authenticate the user and return the authentication response to the Visited Network's AAA server, which would forward the response to its AR , and thence to the MT.

Figure 3.2 shows that the MT moved from AN 1 to AN 2. Assuming these two networks belong to two different administrative domains, the procedure outlined above must be repeated for the MT to get network access in AN 2.

3.2.2 Service Access Control

Similar to network authentication, lines S.1 and S.2 show the flow of messages for authentication and authorization for accessing a service (e.g., access to a secure web-site). Note that typically the Visited Networks' AR and AAA server are not involved in the decision-making and do not intercept the messages related to service authentication and authorization.

The application on MT requiring access to a service hosted on an Application Server operated by the Third Party Serivce Provider (TPSP) would connect directly to it and present authentication information. The Application Server would contact its local AAA server to authenticate the user. The local AAA server would determine the user's Home Provider, and contact it to authenticate the user.

Authorization information (as opposed to authentication information) could be sent in the same flow from the MT to the Application Server. Alternatively, the MT may not send any authorization information; such information could be determined by the TPSP's AAA server, the Home Provider's AAA server, or a combination of both. It may even be the case that no explicit authorization takes place, since correct authentication implies authorization.

3.2.3 Authentication Mechanisms

It is clear that the credentials for network authentication and service authentication need not be the same. In fact, it is quite possible that for network authentication and service authentication the MT may be authenticating to different Home Providers (although this situation is not shown in the figure). This imposes on the user the burden of maintaining several sets of credentials. The alternative—always having a single home provider for all services—is not palatable either.

Chapter 4

Identity Model

During the course of designing a Key Management and Authentication-Authorization framework for a B3G system a model for representing the various stakeholders in the system is useful. These entities would be directly or indirectly using the framework, and it is important to identify them, and the relationships between them, as a pre-requisite to capturing their interests in the system.

4.1 Requirements Identification

Characteristics of the network for which the Key Management and Authentication-Authorization framework is being designed necessarily have some impact on the specification of the model we have set out to develop. Three ways in which the network characteristics affect our model are:

4.1.1 Security

Many pervasive computing initiatives make an implicit assumption that both the source and consumer of context information are under the control of the same authority, and therefore may trust each other.

In a multi-provider architecture, where a user's private information resides in one administrative domain and is consumed in some other domain, it is not possible to make this assumption. One must consider how trust relationships may be established between providers, and how secure channels set up between providers for the flow of sensitive information.

Also, it is the user who should dictate what private information may a provider disclose to other entities. In practice, there are a lot of such decisions, and without the use of some intelligent agents to make such decisions for the user, the user would be quickly overwhelmed.

In order to enable the “privacy agents” to do their work—which involves not only releasing private information in appropriate situations, but also to withhold private information from untrusted requestors—the security framework must provide the agents the tools needed to enforce their decisions. At the minimum, the security framework needs to identity unambigusouly the requestor of information.

4.1.2 Privacy

Researchers in pervasive computing platforms *do* address the issue of user privacy even when they are dealing with a single administrative domein. In some cases, they also consider the possibility of anonymity of the user in a pervasive computing environment. In [31] the authors make a case for privacy enhancing services in ubiquitous computing, and mention some possible services. In [21], six principles of security and privacy in pervasive computing are laid down.

According to [21], the goal of privacy in a pervasivce computing system ought not to be a *total* clampdown on information. To quote:

> What we can and will be able to achieve is prevent unwanted accidents—data spills of highly personal information that people who have never asked for it suddenly find at their doorstep. What we can do is allow people who want to respect our privacy to behave in such a way, so that we will eventually be able to build a long lasting relationship based on mutual trust and respect. And what should also be within our reach is achieving a good balance of convenience and control when interacting with ubiquitous, invisible devices and infrastructures.

The principles of privacy are:

- Notice. Most legal systems require that data collection systems be open about the fact that such data is being collected. It is the right of a person whose data is being collected to know that information about him has been collected, and in most cases he is able to examine the data and protest against inaccuracies. Pervasive computing platforms, which would make heavy use of sensors, should be open in the same sense, especially when the sensors collect data that identifies a person uniquely as opposed to a generic detection.

- Choice and Consent. Some legal systems not only require data collectors to give notice about their data collection practices, but also to obtain *explicit consent* from the data subject.
- Anonymity and Pseudonymity. Some applications require no personal information about the user and it becomes feasible to hide such information. For example, a surfer on the WWW may go through an anonymizing proxy and visit web sites without letting the web server know his IP address. On the other hand, in a pervasive computing environment most applications would be tailored to the user and would need some way of tracking the user. Pseudonymity is a good substitute for anonymity in most such cases. It is not necessary for the application to know exactly who you are—it just wants to know what your *preferences* are. A pseudonym would serve just as well as your real name for the purpose of identifying yourself to the application.
- Proximity and Locality. Location information is a side-channel, so to speak, that applications could use to track you even if you were anonymous or pseudonymous. The granularity of such information should be revealed in a controlled manner to external applications. There should also be limit in time and space to how far the information is disseminated. Prescence information collected within a building for applications within the building should not be revealed to applications outside the building, and this information should be deleted when you have left the building, to prevent abuse.
- Adequate Security. Security mechanisms which run fine on desktop computers may be too comlpex for sensors and small devices that help us build pervasive networks. It is false to believe that pure cryptographic protection of data like sensor readings would be enough to protect one's privacy, since the sensor's security mechanisms cannot comptete with a well heeled attacker's attack capabilities. Thus, instead of overwhelming users with a bunch of se-

curity mechanisms, the principle of *adequate* security should be followed. Sensor data should simply not be transmitted to remote locations. The other principles mentioned above should be applied to limit the damage that an attacker could do and make it unrewarding to attack the weak links in the security chain.

- Access and Recourse. Trusting a system, and especially a system as far reaching as a ubiquitous one, requires a set of regulations that separate acceptable from unacceptable behavior, together with a reasonable mechanism for detecting violations and enforcing the penalties set forth in the rules. Both topics belong more into the realm of legal practice. Technology can help in implementing specific legal requirements such as use limitation, access, or repudiation. Augmenting a P3P-like protocol with something like digital signatures would allow for non-repudiation mechanisms, where parties could actually prove that a certain communication took place in case of a dispute. Database technology could provide data collectors with privacy-aware storage technology that would keep data and its associated usage practices as a single unit, simplifying the process of using the collected data in full compliance with the declared privacy practices. Sophisticated XML linking technology could enable the data subject direct access to his or her recorded information in order to enable the required access rights.

In [18] the authors expand on these principles and then introduce the concept of Identity Management as a conceptial tool to think about users' privacy and security needs.

4.1.3 Identity Management

A growing body of industrial whitepapers and product descriptions address the isse of management of customers and employee identities for a business. However, few of them address the issue from a user's perspective.

We propose that a conceptual model that views identities not only within the context of a user's needs, but also in relationship to other stakeholders in the network is critical for identity management to make any practical impact. Hence we introduce a model that takes all these factors into consideration.

4.2 Meanings of Identity

Prior to identifying the stakeholders, we should clarify the meaning of digital identity, as used in this research. The term "identity" is used in several subtly different ways. Here are some possible definitions (adapted from [1]):

identity *n. pl.* **identities**

1. The collective aspect of the set of characteristics by which a thing is definitely recognizable or known.
2. The set of behavioural or personal characteristics by which an individual is recognizable as a member ofa group.
3. The quality of condition of being the same as something else.
4. The distinct personality of an individual regarded as a persistent entity; individuality.

Note how some definitions emphazise *uniqueness*, while others *membership* of a group, and yet others *sameness*.

This ambiguity is present even when discourse is limited to software systems. Following the lead of [9], the various meanings of the term "identity" in software and network systems may be classified into several classes.

At first, it would seem that at an individual level, our identities should be owned and controlled by ourselves. However, in the digital world, is is not as simple. Depending on what we actually mean by identity, we shall see that the identity is actually controlled and owned by the user along with every principal the user interacts with. We find that largely, the various meanings of the term identity may be put into layers, in order of flexibility and authonomy. Referring to Figure 4.1, these meaning are:

1. Layer 1. This may be called Physical Identity. This is the person, *without* the need for any reference or "handle" to another entity. Attributes of physical identity are characteristics such as facial features, DNA, fingerprints, retinal

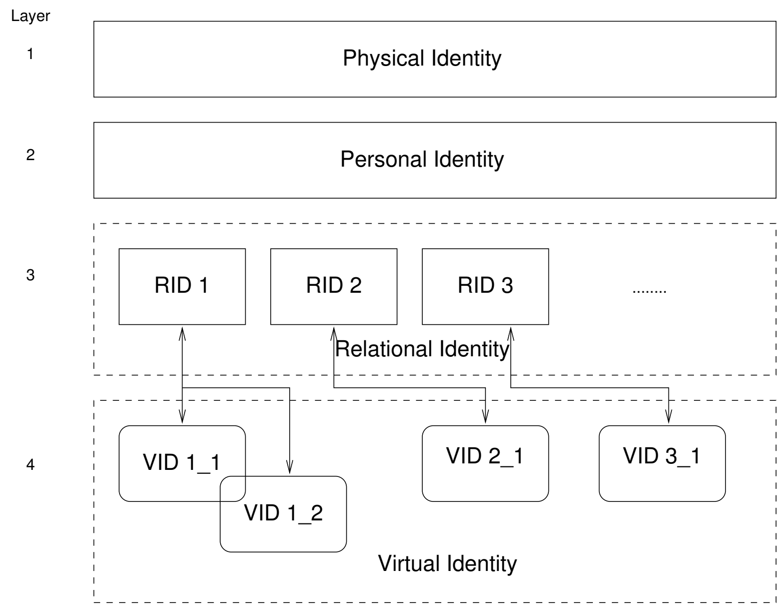


Figure 4.1: Layers of Identity

patterns: the things associated with a person's physical existence. These characteristics may be captured as data in a software system and attached to an identity from a higher layer, and will seldom be used by itself. In other words, it serves to describe another software entity, and is meaningless in a software system without reference to a "container" identity. With regards to the containing identity, these data are usually referred to as *attributes*.

2. Layer 2. This may be called Personal Identity. This is the what humans refer to as "me". Some attributes of this identity are: mood, location, preferences, and current status. While these attributes change with time, the sum total is—from a human perspective—a persistent entity. These characteristics are seldom captured in traditional software systems. Recently, pervasive, or human-centric, computing efforts have attempted to capture this information. Like Physical Identity, these data are meaningless without a referencing "container" identity. Unlike Physical Identity, these data are frequently varying, and partially under the user's control. Because of these factors, collectively, they are often referred to as the user's *context*. This serves to distinguish them from attributes, mentioned above.

3. Layer 3. This could be called Relational Identity. This layer contains identities that are purely or largely implemented in software. These identities are

assigned to the user by an interacting party and exists as long as a relationship is maintained. Moreover, it is *only partially, if at all, under the user's control*. It is a software representation of a user's relationship with the entity owning or operating the software system. Examples of such identities are: driver's license, passport, social security number, telephone number, company network login accounts, etc. Note here, that there may be various fields on a driver's license that emanate from Physical Identities, as we mentioned earlier. A Layer 3 identity can act as a container of Layer 1 identities.

4. Layer 4. This layer may be called a Virtual Identity. It is our modeling contruct in order to fulfil some goals, described below. Like Relational Identity, it is meant to be a software representation of information about a user. Unlike Relational Identity, it is meant to allow *users* to control their representations in software systems almost in the same way as they control their Personal Identity. It is an approach meant to achieve sometimes conflicting privacy and security goals of users and businesses. Virtual identities are seldom supported by business systems.

It is important to note that in the above discussion, Layer 1 and 2 identities are tradionally *not* considered as identities in software systems, but may be treated as attributes of them when appropriate. We shall proceed to build our model using only Layer 3 and 4 identities. But first, we need to clarify why we make the disctinction between these two layers.

When a service provider and a service consumer execute a contract, the provider obtains some information about the user and the particular services. They fall into three categories:

1. Authentication information. This is used so that the user, at the time of obtaining a service, may identify himself to the service provider: "I am so-and-so, and here is proof that I am who I say I am." For example, when logging into an email system, the user would provide a user-id ("I am Robert"), and a password ("this proves that I am indeed Robert").
2. Authorization information. This is used by the service provider to determine if the user is indeed authorized to use the service.

Only in very simple situations may one assume that a user who is authenticated is automatically authorized to use a service. One such situation is, for example, when there is a only one service, and is available to authenticated users all the time.

If the service were available only within a time interval, and this time interval varied by user, then the service provider would have to determine whether the authenticated user may access the service at that particualar time. If there were more that one service, with different users accessing different sets of services, then again, the service provider would have to determine whether the authenticated user is allowed to have access to the service he is requesting.

Thus, in general, authorization information is distinct from authentication information.

3. Accounting Information. For paid services, the service provider would need to keep track of service usage. Even in flat-fee models, the service provider may like or be required to have per-user usage statistics. Finally, a bill would have to be created and sent.

When this information is captured in the business software system of the provider, effectively, a Layer 3 identity is created. This information is indispensable for the provider in order to fulfill the contract. We shall refer to this as **REGID** (for REGistration IDentity).

On the other hand, users would like to reveal as little personal information as possible, when accessing a service, due to security and privacy reasons.

For these purposes, we propose the concept of **VIDs** (for Virtual IDentities). A VID may be derived from a REDID by removing some parts and faking other parts of the information originally specified in the REGID. A VID could be made persisten if the user wants to use it repeatedly, or it could be generated, used for a single service serssion, and then discarded.

Figure 4.1 is derived from one described in in [9]. The lowest 3 layers are identical. In Durand's model the 4th layer [1] is one inferred from attributes of the lower layers, thus making it a group vision ("blue eyed citizens", "light drink lovers", "driver's from Utah"). We conclude that the VID layer (as described in section 4.2)

[1]The article starts at layer 0, so this would be layer 3 in the article, to be precise.

was more suitable to the current needs. It is abstracted from the lower one, deriving another layer of an identity that although used by a person could be completely unrelated to any of its attributes. Nonetheless it is the identity that a provider sees when interacting with the user, even if with bogus information. We do not consider the inferred identity as relevant in our framework as a layer. Although it is part of one's identity it is not distinct from others and thus not unique. A famous family therapist said: *"The human experience of identity has two elements: a sense of belonging and a sense of being separate."* We emphasize the "sense of being separate" in our layers and leave "the sense of belonging" to a recollection of the identity's attributes, thus upholding *"... the individual characteristics by which a person or thing can be identified"* from the dictionary. As discussed below, the VID layer enables privacy protection to the user as it restricts the knowledge of the user behind a VID to a single point that does not need to be the service provider.

4.3 Stakeholders and Inter-Relationships

In a software system creating a platform for Beyond-3G networks, there are a large number of entities, and some terms are used in different contexts with different meanings, while others refer to the same entitiy. To prevent confusion, we introduce the terms we intend to use and give their definitions for the purposes of this document.

The objectives of this section are to define some terms related to identity and their inter-relationships, such that: (a) The terms can be used with the same meaning for discussions of profiles and context; identity and federation; privacy and confidentiality; and services. (b) The number of such terms is the minimum required for a coherent discussion. (c) The static inter-relationships are extremely generic, so as to accommodate different business models. (d) An *instantiation* of these relationships can be used by any reasonable scenario.

User. A person that attempts to access a system, whether authorized to do so or not. If the attempt is allowed, then the person becomes a service consumer. If the attempt is denied, the person may be an attacker. In our discussion, we assume strong authentication is always being used, and a successful operation in the system emanates from an authenticated user (that is, a service consumer).

The user is a domain concept, and not meant to designate a system entity. But since it is used very often, we define it here, to clarify that it does not refer to any system entity. System entities are described below.

Provider. Legal body that provides some benefit to users (in our context, this benefit is a suite of services). This provision is defined in a *contract*, which is a legal agreement that is established between a *subscriber* and a *provider*.

Subscriber. An entity that executes a contract with a provider, to create an account.

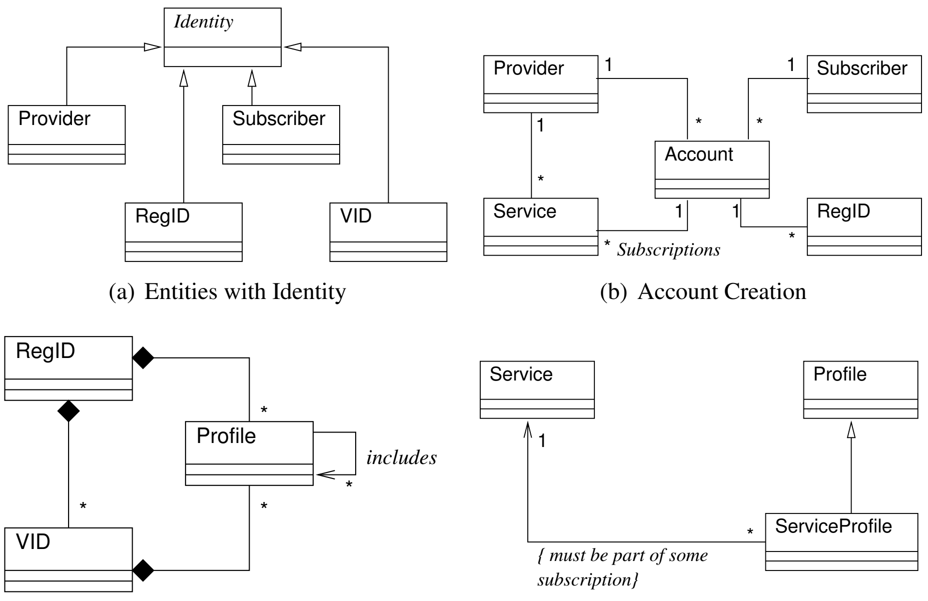
Account. Result of the contract established between a *subscriber* and *provider*. It captures the billing details, list of authorized users, limits on service usage and service-level agreements, and billing details.

Identity. A common characteristic of Layer 3 and 4 identities discussed in Section 4.2. We extend the concept of identity to capture information not only about users, but also subscribers, providers, and services.

Registration Identity (REGID). It is a container for various user and service related data, such as userid-password pairs, certificates, and *profiles*. It is a system level object that encapsulates the beneficiary of a specific set of services form those available in an account. The contents of a REGID are controlled largely by the subsciber and the provider.

Virtual Identity (VID). It is a user-controlled representation of the user's attributes within the system. It contains an identifier along with additional information like a profile, credentials, usage trace etc. A VID can be regarded as a view somebody in the system has of the user (more precisely, the REGID). In our proposal, a REGID can "have" several VIDs, created by the user. The VID can inherit attributes from the REGID, as well as over-ride them, and allows the user to introduce new attributes.

Profile. It is s a group of attributes. A profile is associated to only one identity. Profiles may be private or shared, depending on user-preferences and provider capabilities, and a particular profile may include other profiles. The profile is principally a mechanism to group attributes conveniently.



(a) Entities with Identity (b) Account Creation (c) Virtual IDs and Profiles (d) Service Profiles

Figure 4.2: Identity Model as UML diagrams

4.4 Modeling the Relationships

Figure 4.2 shows the relationships as UML diagrams. Here we explain the relationships, and the design decisions behind them in greater detail.

4.4.1 Entities with Identity

Figure 4.2(a) shows that the Provider, Subscriber, REGID and VID objects implement the interface Identity. An Identity interface guarantees that objects will have an identifier and some way of proving that they are valid holders of the identifier (i.e., a credential).

An Identity is an interface that guarantees that these entities have:

1. An Identifier, unique within the scope of the model. Local Identifiers can be made unique by appending domains. This will be further discussed below.
2. Some credentials to prove claims from entities that they indeed are authorized holders of a given Identity.

It is clear that REGIDs and VIDs have an Identity.

Providers also have an Identity since they need to authenticate to each other to validate their trust relationships and set up secure channels. Software agents working on behalf of the users need to be able to negotiate with the providers and need a automated way of verifying the identity of the provider they are conversing with. The This will be discussed below. This conclusion follows from the requirements mentioned in 4.1.1.

Subscribers have Identity so that the common authentication infrastructure may be used to authenticate them, just like REGIDs and VIDs. One may imagine a subscriber accessing a web-based account management tool. If a subscriber could be handled like any other user, the same authentication framework could be used for users and subscribers.

4.4.2 Accounts

Figure 4.2(b) shows that for services to be consumed, an Account—usually backed by a legal contract—must be set up between a Subscriber and a Provider. A given subscriber and provider may have more than one account (i.e. a 1:* relationship). There can be various reasons for this: the subcriber may wish to separate bills according to some criteria (one bill for services X and Y, another for Z and W). Within an account, the subscriber may specify: (a) all the REGIDs on whose behalf he is operating the account; (b) all services he is willing to avail; (c) a mapping from each REGID to the services it is allowed to used; (d) the limits on these services, on a per-REGID basis.

Each REGID stands for a user. Some of these REGIDs are created when the account is created, and others added later to support additional users; REGIDs may be deleted, too, when the users they represent leave the system permanently.

For e.g., a company may outsource its email system. The company pays for the system and its maintenance, while the employees of the company use the email system. Employess would need to authenticate, so that the system may deliver their individual mailboxes and not reveal one individual's emails to another. In this scenario, the company is the *subscriber* while the employess are *users*.

In one degenerate case, the subscriber and the provider may be the same entity. This is the case when a set of services is being run by an organization for its own employees, for example. Let us call this entity as, simply, the Owner. This entails that the Owner—Account relationship is also 1:1. Thus the Account object may actually be absorbed into the Owner object.

In another degenerate case, the subscriber and user may be the one and the same. For e.g., when a person opens a paid online email account for his sole use he both uses the account and pays for it.

As mentioned above, the Account object stores, for each REGID, the set of services it is authorized to use. An alternative is that the REGID can store references to the same set of services. The difference is one of implementation.

4.4.3 Customization and Personalization

Figure 4.2(c) shows how users may customize their apperance to services by deriving a VID from a REGID. REGIDs contain a set of Profiles. Profiles are groups of attributes. For example, all attributes related to an email service could be grouped into an EmailProfile, and all attributes related to a Calendar service could go into a CalendarProfile. These attributes are repositories of personal and personalizable information. Attributes common to several services ("color of web-page background", for example) can be put in a separate profile and included by other profiles, as shown bu the *includes* association.

Figure 4.2(d) shows that a Profile can be designated for and thus refers to a Service. This reference is indirect, however, which is shown by the contraint remark within braces.

A VID is a object derived from REGID by copying the Profiles and changing, removing or adding some attributes.

A trivial example would be: A user is assigned by the Subscriber a REGID x that comes with an EmailProfile that shows unread emails in red. The user generates two VIDs, y and z. VID y's EmailProfile shows unread emails in blue. VIDs z's EmailProfile adds a new attribute, that not only highlights unread emails, but also highlights them differently according to whether they were received within the last twenty-four hours.

In this way, VIDs are a higher level collection of personalization content. Colloquially, they embody a "personality". The user may change his REGID's personality by switching VIDs. We shall discuss how this is done below.

A user may create VIDs as and when needed, and either make them persistent for frequent use, or keep them around for a single service session and then discard them.

Chapter 5

Achieving Privacy

In this chapter we shall examine how the Identity Model we created in Chapter 4 can be used within a network with architecture similar to that explained in Figure 3.1 in Chapter 3, in order to achieve security and privacy goals.

5.1 Federated Operator Scenarios

In general, provider deployments can and will be based on diverse settings.

- Some providers will not have a AAA framework and rely on the Service Provider's platform to provide them with the infrastructure for hosting the sercice; the infrastructure will provide components such as accounting, policy decision points, policy enforcement points, etc.;
- On the other extreme, there will be providers, likely telco operators, with a full blown AAA and security infrastructure, where policies are defined and enforced in their own domain;
- In between there will be operators with partial infrastructures, such as only accounting servers but not charging mechanisms, with policy enforcement points but not policy definition and decision points, etc.

These scenarios (in particular the first two) motivate us to use concepts of federation. Even the all-in-one provider will require federation in order to allow access to users from other domains.

Federation allows one admnistrative domain to trust another for authentication and authorization decisions.

5.1.1 Security Requirements

In a federated infrastructure, such as that shown in Figure 3.1, the following assumptions can be made:

- Domains with their own AAA infrastructures will likely have their own authentication mechanisms;
- Authorization will be distributed; i.e. policies located at different administrative domains may be combined to perform the authorization decision;
- To save the user from the burden of authenticating again and again, at each provider whose services he is using, Single-Sign-On will be desirable.

On the other hand, from a pervasive networks point of view, the following additional assumptions can be made:

- One user may be accessing services at different providers using different identities;
- Users will initiate multiple sessions on multiple devices, with some devices simultaneously supporting multiple such users.

5.1.2 Privacy Requirements

We assume that the authentication and authorization protocols are chosen such that they are robust against passive and active attacks against integrity and confidentiality. This still leaves the user with some privacy concerns. We consider the following:

1. A user may wish that a passive attacker snooping on the network is unable to (a) find his "real" identity; (b) correlate different service sessions or invocations, to build a usage profile.
2. A user may wish to achieve the same privacy levels (a), (b) as above, but with respect to the service provider whose service he is accessing. He may, for some reason, want that successive accesses to the same service not be correlated by the service provider.

5.2 Identity Management

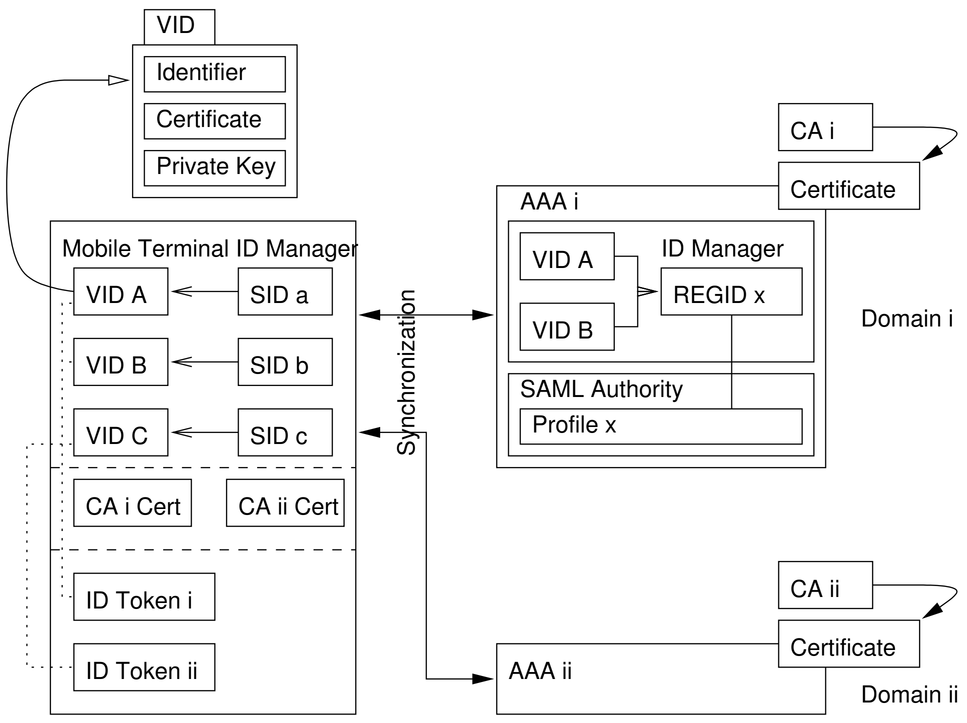


Figure 5.1: ID Managers in MT and SP

Identity Management is enabled by components called ID Managers on the MT and within the AAA servers of Service Providers. Figure 5.1 illustrates the situation.

Identity Management works this way:

- Each Home Provider issues REGIDs to its users. There is at least one REGID per user. A user with two different REGIDs is effectively two different users as far as the Home Provider is concerned. A REGID is unique in the operator's domain.
- For each REGID, a number of VIDs are defined. These VIDs may share all or none of the REGIDs attributes. The VIDs can be created and deleted at any time. At least one VID must be present for the system to bootstrap. This VID could be created at the time of creation of the REGID. The user may later add more VIDs as he wishes and remove them. by asking the operator (or using a management interface that the operator may provide for such purposes).
- VIDs have an attribute called Identifier that is globally unique. A VID Iden-

tifier consists of two parts: (a) the first part is a random alphanumeric string chosen by the user, that is locally unique to the Home Provider; (b) the second part is the Home Operator's globally unique domain name. Combining these two parts we have an identifier that looks very much like an email address. (In fact, an email address can be used as VID for our purposes.) From the VID Identifier, one may find out the Home Provider. However, since the first part is a string with no correlation to the REGID, it will not be possible to deduce the user's identity from the VID alone.

- The VIDs at the AAA ID Manager must be synchronized with the VIDs present in the MT ID Manager. A protocol like SACRED may be used for this purpose.[1]
- Each REGID and VID is associated with a set of credentials issued by the operator. These credentials could be an RSA key-pair issued by the Home Provider, where the public key is enclosed in a Certificate signed by the Home Provider's CA.
- Services are accessed using a VID's identifier and credentials.
- Applications installed on the MT "know" only their Service Identifier (SID). The SID depends on what the application is accessing. For e.g., an email client may use the SID "XYZ-Email" when accessing the XYZ email service, and "ABC-Email" when accessing the ABC email service. The exact definition is left to applications and user.
- The ID Manager on the MT maps the SID to a VID using an internal table. This table is configured by the user. Every time an application is installed on the Mobile Terminal, it "registers" itself with the ID Manager, and the user is offered a chance to configure the mapping to a VID. The user can change the mapping at any time. In fact, the ID Manager offers an interface whereby advanced privacy management systems may dynamically alter the VIDs for an SID, or even invent VIDs for specific uses.[2]
- The ID Manager delivers the VID to the application requesting. The Application then accesses the attributes of the VID to get its credentials.

[1]This is part of on-going work in the Daidalos project.
[2]Integration with such systems is part of on-going work in Daidalos.

- The application may use the credentials in an application specific authentication mechanism to authenticate the user and secure the session. In case the application does not have such a protocol, a generic protocol such as SSL, SASL or IKE may be used to authenticate the user and bootstrap a secure channel.
- The application talks to a "service". Typically, the service process will use a AAA framework to actually process the credentials, rather than using local information.
- The VID Identifier and credentials are transported, via AAA mechanisms, to the AAA server of the Home Provider. The AAA server first authenticates the VIDs credentials. Then it maps the VID to the underlying REGID and using information from the VIDs profiles, REGIDs profiles and the Account object, compute an authorization decision. This is then returned, again using AAA mechanisms, to the service which requested the authentication and authorization.

In this way, authentication and authorization can be combined with our identity model to achieve privacy. Note that the user may change VIDs per service at any desired level of granularity: one VID for all services; one VID per service; or one VID per service session. The latter, of course, inplies a larger overhead in generating and synchronizing VIDs. Paranoia, thus, needs to be balanced by performance considerations. However, the model is quite general and can be applied within existing application protocols. The applications and services themselves would need to be modified to be ID Manager "aware" but in most cases could continue to use their existing protocols.

Notice how the requirements from the discussion above on privacy requirements are met:

- The user's real identity is not transmitted over the network and thus an attacker or the service is unable to determine the user's real identity.
- In many situations, it is not necessary to determine a user's exact identity; it is sufficient to correlate two different sessions, perhaps of different services, to disrupt privacy. However, by using different VIDs, and/or changing VIDs

often, the user can prevent such correlation.

5.3 Common Authorization Framework

The research work presented in this section was originally published in [30].

In a federated architecture, it may be assumed that different AAA domains will use different authentication and authorization mechanisms. Yet, results of authentication and authorization decisions must be passed from the decision making domain to federated domains.

For the purposes of transmitting authentication decisions, which are binary in nature (i.e., Success or Failure), the de facto standard protocol on the Internet is Diameter [6]. Standards based AAA servers support inter-communication using Diameter.

However, authorization information is more complex. One may need to communicate, for example, that a service has a certain QoS level, rather than merely if it is available to the requestor.

Within Daidalos, we have chosen to use the Security Assertion Markup Language (SAML) [25] for transporting authorization information. SAML standardizes the exchange of information about the user's authenticationn status, attributes and authorization decisions. It makes the security infrastructure independent of the specific mechanisms used for the authentication of users, and also aupports Single-Sign-On across administrative domains.

When using SAML, a component called SAML Authority is introduced into the AAA server. It is responsible for generating and parsing SAML messages. It is also possible, though not necessary, to incorporate the authorization logic into the SAML Authority, as we have.

We can improve on the mechanism described in the previous section, in the following manner.

5.3.1 Enabling Single-Sign-On

Enabling Single-Sign-On requires us to decouple the authentication and authorization phases. A service no longer uses a user's credentials to authenticate and authorize him. Instead, there are two steps.

Authentication Step. This step works much like that described in the previous section, except that a dedicateed authentication protocol capable of carrying some extra information is employed (we shall discuss this in greater detail below). For now we assume that authentication is done between a process on the MT and the AAA server. As before, the user may choose what VID the should be used by the process running the authentication protocol on the MT.

When the VID Identifier and its credentials arrive at the AAA server, the AAA server, as before, authenticates the credentials, and generates the authentication decision. If the authentication was successful (i.e., the decision is Success), it then invokes the SAML Authority, which uses the VID and REGIDs profiles, and information in the conceptual Account object, to generate a SAML assertion and artefact. A SAML artefact is an opaque alphanumeric string uniquely corresponding to the assertion. It has meaning only to the SAML authority, since no other entity can translate the artefact into the assertion or any other useful information. The artefact is returned to the MT along with the authentication decision.

Authorization Step. When an application needs to authenticate and authorize to a service, it takes the artefact delivered in the previous step, and presents its VID Identifier and the artefact to the service. The service, as before, hands both to the AAA infrastructure. Via AAA mechanisms, the artefact arrives at the SAML authority at the Home Provider. The SAML authority uses the artefact to recover the assertion. It then uses the assertion, VID and REGID profiles, information about the requesting service, etc. to make the authorization decision. The decision is then returned as a SAML document to the requesting service.

The advantage of this two-step solution is that any application may use the artefact once it has been obtained during the authentication step. Since authentication protocols usually require several round trips, while an authorization decision using the artefact only takes one trip, this saves time. Also, since authentication is done only once, this seamlessly enables Single-Sign-On.

5.3.2 Protecting the SAML artefact

Sending a plain SAML artefact over the network has some disadvantages:

1. An attacked may capture an application's authorization request message (which contains the artefact) and use it in replay attacks unless the application protocol has protection against such attacks;
2. A passive attacker may observe that several service requests were authorized using the same artefact and conclude that they were initiated by the same user; this defeats the purpose of using different VIDs for different services (or service sessions).
3. The same consideration as above applies to the service provider, which can use the artefact to correlate different service sessions and be able to attribute them to the same user.

This motivates the introduction of the ID-Token structure. It is a structure safe to use over the network, even over an insecure channel.

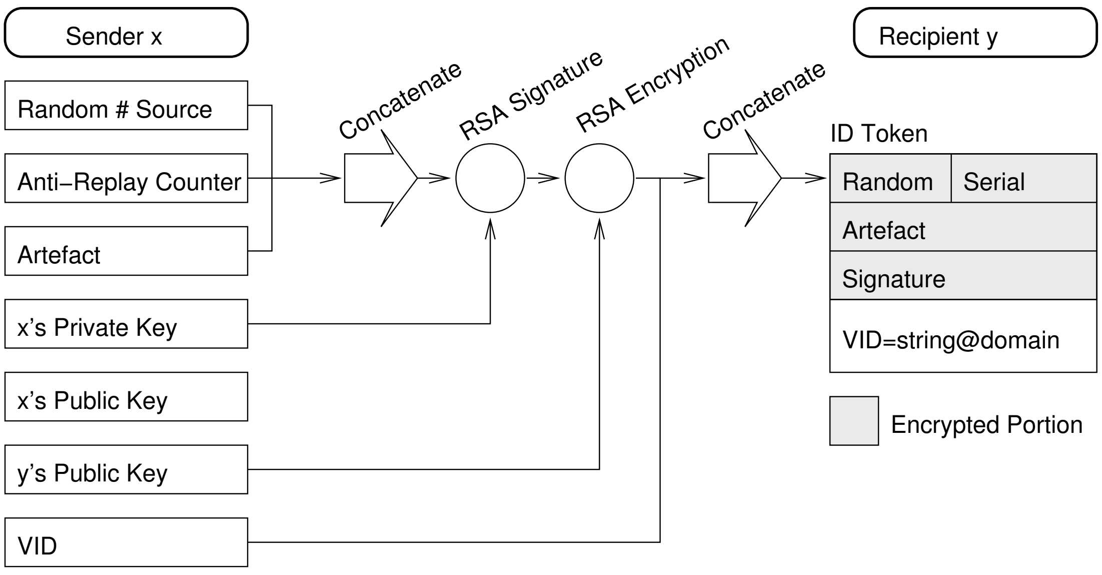


Figure 5.2: ID Token Generation

As shown in Figure 5.2, the ID Token has these main components:

- The VID for which the service request (or authentication request) was initiated.
- A random number, that makes the ID-Token different every time it is generated from the artefact.
- A serial number (from the Anti-Replay counter), that helps avoid replay at-

tacks. The serial number is maintained by the ID Managers. The serial number of an incoming ID-Token must be greater than the last one seen.

- The SAML artefact, itself, which references the appropriate assertion.
- A digital signature computed over the concatenated form of the previous items.

The ID-Token is used in the manner described below.

Authentication. When the AAA server has authenticated a user during the authentication step, and received an artefact from the SAML authority, it generates the ID-Token using the requesting VIDs public key and its own private key, anti-replay counter and a random number. (In other words, substitute 'AAA' for x and 'VID' for y in the figure.) The process of ID-Token generation is shown in Figure 5.2.

Verification. Figure 5.3 shows the process of ID-Token verification. When the MT's ID Manager receives the ID-Token, it uses a process opposite to the generation process in order to verify the ID-Token and then stores the artefact and updated the Anti-Replay Counter to the serial number.

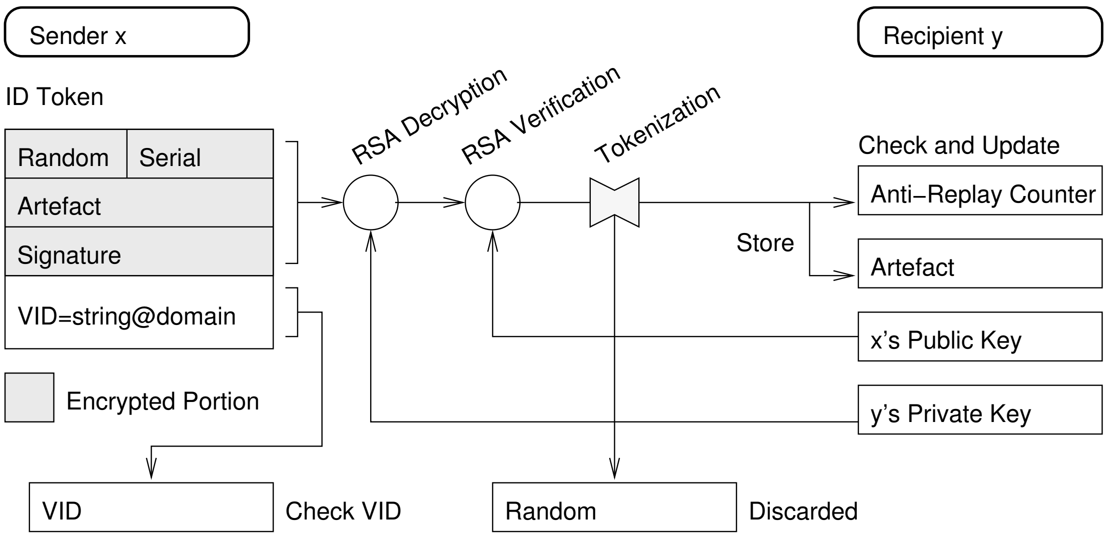


Figure 5.3: ID-Token Verification

Re-generation. When an application wishes to authorize to a service, it requests the MT ID Manager for a VID and an ID-Token. The ID Manager generates the ID-Token in the same manner as shown in Figure 5.2, using the VIDs private key for signature and the AAA server's public key for encryption. (In other words, substitute 'VID' for x and 'AAA' for y in the figure.) Note that it also increments the Anti-Replay Counter and generates a new Random, before constructing the ID-Token from the stored artefact.

5.3.3 Revisting Privacy

When an ID-Token, instead of a bald artefact, is transmitted over the network, a passive attacker sees only a VID, and an opaque, encrypted chunk of data (the encrypted portion of the ID-Token). Due to the random number component, the encrypted portion is not going to be the same between two successive authorization invocations, even if all other material remains the same. Combined with a dynamic VID, the passive attacker has no way to correlate service usages. The same applies to the service.

Note that the VID used for authentication and retrival of ID-Token need not be the same as the VID used for service authorization. The assertion can be recovered from the artefact alone and remains the same regardless of which VID is used in the service authorization request.

5.4 A Complete Authentication and Authorization System

In this section, we shall put together the pieces from the previous sections, and describe a system for flexible authentication and authorization using VIDs and ID-Tokens for network and application layer services. (This work was presented originally in [26].)

Obviously, the first network service that is required by a user is basic network connectivity—the right to send and receive data packets, even with a limited scope, over the network. Access to this connectivity is granted through a network access control procedure that depends on the link. Most specifications for this procedure place it at link layer (e.g. 802.1X port-based authentication [16] for Ethernet or 802.11 links). Recently, a working group has been created at the IETF to develop a protocol above IP called PANA that will carry authentication messaging independently of the underlying link technology.

PANA [13] aims at offering a single authentication method at the IP layer, above

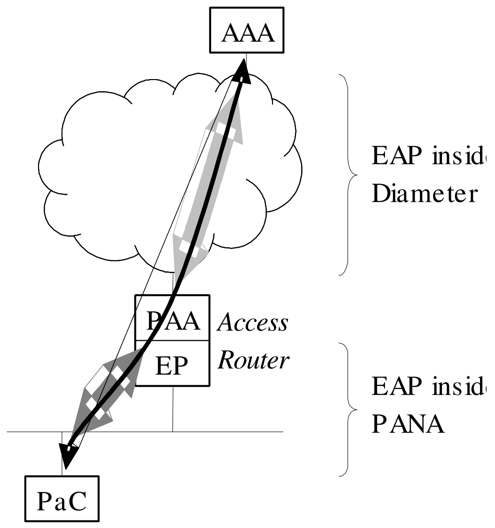


Figure 5.4: PANA Deployment

different link technologies for multi-access and point-to-point links. PANA defines how a PANA Client (PaC) authenticates to a PANA Authentication Agent (PAA), which may rely on an Authentication Server (AS) to perform credentials verification. PANA's design supports various types of deployments; PaC is normally placed in the user terminal whereas PAA is by definition to be placed a single (IP) hop away from PaC, typically in a Network Access Server (NAS). For our purposes, the NAS is the same as the AR. See Figure 5.4.

The PANA protocol runs between the PaC and the PAA and carries an EAP (Extensible Authentication Protocol [3]) method, using UDP as the transport layer protocol. In most cases, PANA authentication involves a distant AAA server that communicates with the PAA using an AAA protocol. PANA access control procedure then fits into a larger AAA-based access control framework. A AAA server with enhanced Auditing and Charging features, as it is used is Daidalos, is designated as "“A4C server”.

PANA does not provide traffic confidentiality by itself. Yet, PANA is able to bootstrap a confidentiality protocol at link (e.g. 802.11i [15]) or IP (e.g. IPsec [20]) layer. See [17], [27] for descriptions of how this may be achived using PANA. In either case, the secure channel is established between the PaC and the PANA Enforcement Point (EP), which is dynamically configured by the PAA upon successful authentication.

PANA is able to carry information by using Attribute Value Pairs (AVPs); the base protocol defines the ones required for operation. The protocol supports the definition of new AVPs to contain new values, this allowing application specific AVPs. We use this feature to carry authorization information between access networks and users (discussed below).

5.4.1 PANA-based Authorization for Network Access

An ID-token is only provided when the user has been successfully authenticated by an entity trusted by the resource owner. In this case, the resource is network access, the owner is the AN operator, and the trusted entity is the Home Provier's AAA server. When the token is already present at the user's device, the authentication phase can be bypassed.

Authentication Phase. In the authentication phase (see Figure 5.5), the ID-token must be delivered from A4C server to the user's device in two steps: first A4C sends the ID-token to AR/PAA after EAP authentication using Diameter; then, the AR/PAA sends the ID-token to MT using PANA. New defined AVPs for authorization are used in both steps to transport the ID-token generated by the A4C server. First, an ID-token AVP is encapsulated in Diameter EAP application protocol [11] messages. Then, PANA (specifically PANA-Binding-Request) transports this AVP to the MT with authorization parameters: ID-token.

Registration Phase. In the registration phase, the user must deliver the ID-token to the network for obtaining access. A similar procedure as described above is employed for transporting the ID-token AVP using also PANA and Diameter. Note, that the PANA messages for the registration phase are different when (a) it is sent on the PANA session built on the authentication procedure, or (b) it is sent on a new session. This is related to PANA's state machine. Figure 5.5 shows an example where authentication (using EAP-TLS [4]) and registration phase are executed in the same PANA session.

In this approach, the PAA and PaC must be modified to understand the new AVPs. This is because the EAP-TLS authentiation method is unable to transport the ID-Token on its own.

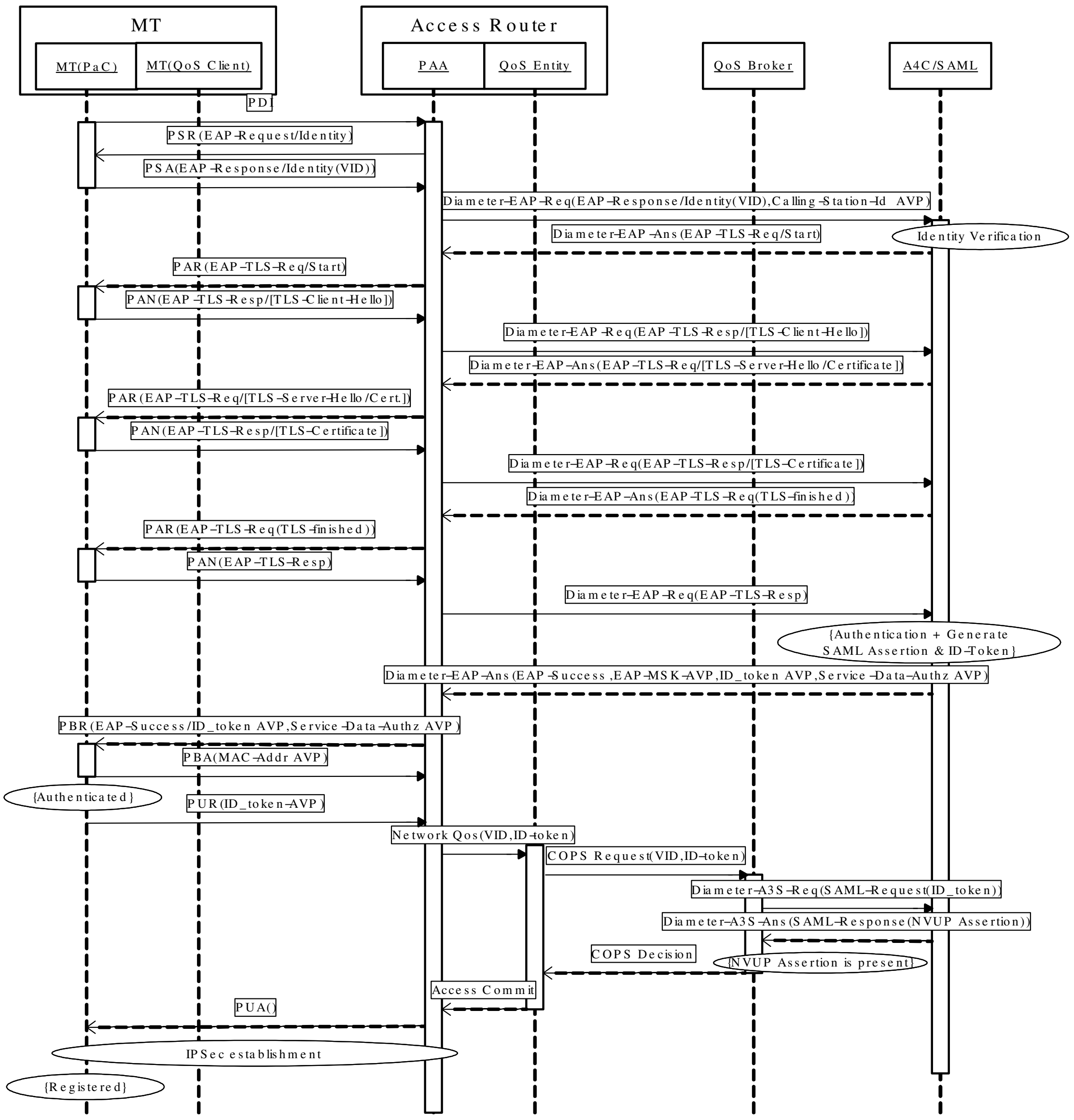

MT
MT(PaC)
MT(QoS Client)
Access Router
PAA
QoS Entity
QoS Broker
A4C/SAML
PDI
PSR(EAP-Request/Identity)
PSA(EAP-Response/Identity(VID))
Diameter-EAP-Req(EAP-Response/Identity(VID),Calling-Station-Id AVP)
Diameter-EAP-Ans(EAP-TLS-Req/Start)
Identity Verification
PAR(EAP-TLS-Req/Start)
PAN(EAP-TLS-Resp/[TLS-Client-Hello])
Diameter-EAP-Req(EAP-TLS-Resp/[TLS-Client-Hello])
Diameter-EAP-Ans(EAP-TLS-Req/[TLS-Server-Hello/Certificate])
PAR(EAP-TLS-Req/[TLS-Server-Hello/Cert.])
PAN(EAP-TLS-Resp/[TLS-Certificate])
Diameter-EAP-Req(EAP-TLS-Resp/[TLS-Certificate])
Diameter-EAP-Ans(EAP-TLS-Req(TLS-finished))
PAR(EAP-TLS-Req(TLS-finished))
PAN(EAP-TLS-Resp)
Diameter-EAP-Req(EAP-TLS-Resp)
{Authentication + Generate SAML Assertion & ID-Token}
Diameter-EAP-Ans(EAP-Success,EAP-MSK-AVP,ID_token AVP,Service-Data-Authz AVP)
PBR(EAP-Success/ID_token AVP,Service-Data-Authz AVP)
PBA(MAC-Addr AVP)
{Authenticated}
PUR(ID_token-AVP)
Network Qos(VID,ID-token)
COPS Request(VID,ID-token)
Diameter-A3S-Req(SAML-Request(ID_token))
Diameter-A3S-Ans(SAML-Response(NVUP Assertion))
COPS Decision
{NVUP Assertion is present}
Access Commit
PUA()
IPSec establishment
{Registered}

Figure 5.5: Authentication and Authorization done by PANA

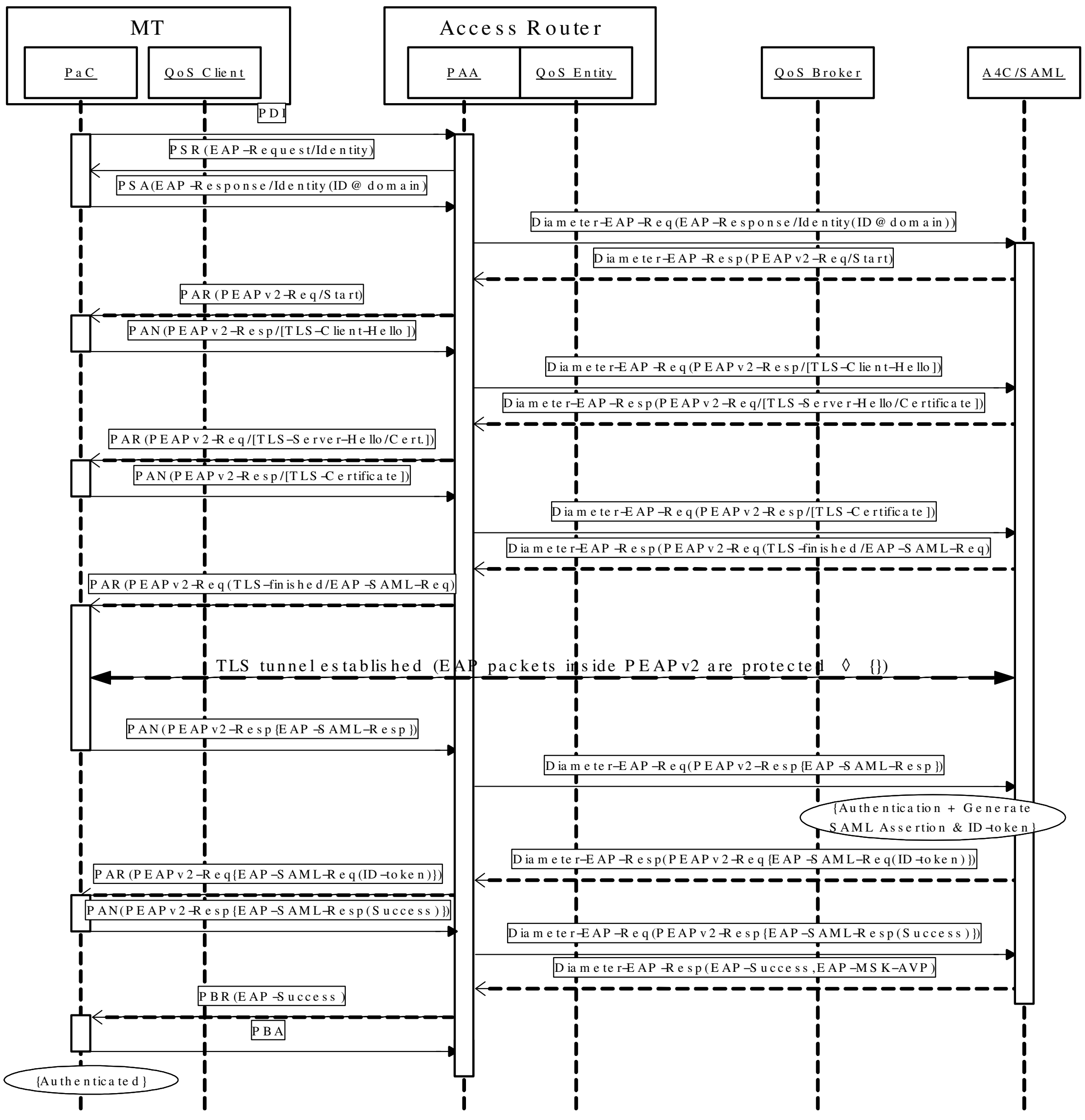

MT
PaC
QoS Client
PDI
Access Router
PAA
QoS Entity
QoS Broker
A4C/SAML
PSR(EAP-Request/Identity)
PSA(EAP-Response/Identity(ID@domain))
Diameter-EAP-Req(EAP-Response/Identity(ID@domain))
Diameter-EAP-Resp(PEAPv2-Req/Start)
PAR(PEAPv2-Req/Start)
PAN(PEAPv2-Resp/[TLS-Client-Hello])
Diameter-EAP-Req(PEAPv2-Resp/[TLS-Client-Hello])
Diameter-EAP-Resp(PEAPv2-Req/[TLS-Server-Hello/Certificate])
PAR(PEAPv2-Req/[TLS-Server-Hello/Cert.])
PAN(PEAPv2-Resp/[TLS-Certificate])
Diameter-EAP-Req(PEAPv2-Resp/[TLS-Certificate])
Diameter-EAP-Resp(PEAPv2-Req(TLS-finished/EAP-SAML-Req))
PAR(PEAPv2-Req(TLS-finished/EAP-SAML-Req))
TLS tunnel established (EAP packets inside PEAPv2 are protected ◊ {})
PAN(PEAPv2-Resp{EAP-SAML-Resp})
Diameter-EAP-Req(PEAPv2-Resp{EAP-SAML-Resp})
{Authentication + Generate SAML Assertion & ID-token}
Diameter-EAP-Resp(PEAPv2-Req{EAP-SAML-Req(ID-token)})
PAR(PEAPv2-Req{EAP-SAML-Req(ID-token)})
PAN(PEAPv2-Resp{EAP-SAML-Resp(Success)})
Diameter-EAP-Req(PEAPv2-Resp{EAP-SAML-Resp(Success)})
Diameter-EAP-Resp(EAP-Success,EAP-MSK-AVP)
PBR(EAP-Success)
PBA
{Authenticated}

Figure 5.6: Authentication and Authorization done by EAP

5.4.2 EAP-based Authorization for Network Access

As mentioned earlier, using PANA with the EAP-TLS authentication mechanism to transport the ID-Token requires that the PaC and PAA be modified. We discuss a second approach that avoids this requirement.

EAP provides a flexible way to authenticate to entities (in particular ad-hoc nodes) because it supports multiple authentication methods. Some EAP methods have the capability to carry generic information apart from authentication information.

This alternative was demonstrated in [14] which describes how some kinds of

EAP methods can carry Mobile-IPv6 bootstrapping information to MT during EAP-based authentication process.

In this approach, we use the PEAPv2 [19] authentication method because it provides the flexibility needed to achieve our objectives. It allows the definition of new EAP methods that are encapsulated and carried inside a TLS secure tunnel. This channel is generated during a TLS handshake in the first phase of the protocol. The new EAP method is used to transport the authorization information in the second phase of PEAPv2. A new method called the EAP-SAML method is used as a carrier for ID-token assertions and authorization information. There is no need to invent new AVPs for PANA and Diamteter.

Note that the PANA protocol is used as a lower layer to transport EAP packets from MT to AR . The authentication sequence and ID-token delivery to MT is shown in Figure 5.6.

5.4.3 Registration with Existing ID-Token

Figure 5.7 shows the registration process when the user already has an ID-token. As we can see a new PANA session is executed. The PEAPv2 TLS tunneled phase 2 is used to deliver the ID-token. In this case only the A4C is authenticated by the MT because the user does not need to be authenticated again as he already owns the ID-token.

In the second phase, the A4C requests the ID-token from the user by using the new EAP method (EAP-SAML request/response).

After A4C verifies that the ID-token is correct, it informs the AR/PAA that this user is authorized to access the network. Then AR/PAA requests the QoS Broker to obtain quality of service parameters associated to this user and to know if it is possible to get access. Note that AR has to recover both VID and ID-token to carry out the registration process. However, it cannot access the EAP messages because they are encrypted inside a TLS tunnel. Thus, A4C sends both parameters to AR/PAA by using new Diameter AVPs: VID AVP and ID-token AVP that are added to Diameter EAP Application.

This approach has a clear advantage: access equipment does not need to be modified to support this solution because usually they act as simple EAP mes-

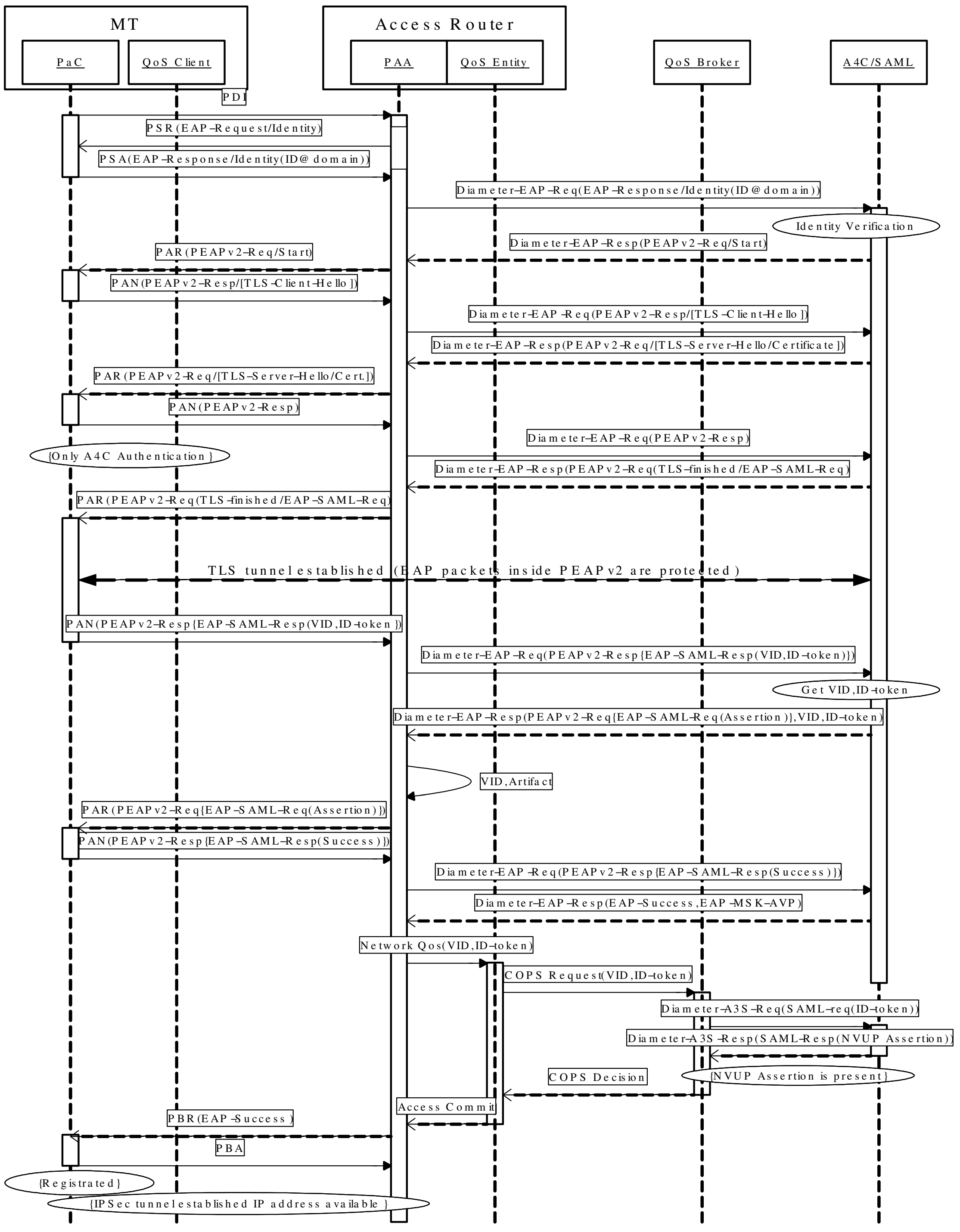

MT
PaC
QoS Client
Access Router
PAA
QoS Entity
QoS Broker
A4C/SAML
PDI
PSR(EAP-Request/Identity)
PSA(EAP-Response/Identity(ID@domain))
Diameter-EAP-Req(EAP-Response/Identity(ID@domain))
Identity Verification
Diameter-EAP-Resp(PEAPv2-Req/Start)
PAR(PEAPv2-Req/Start)
PAN(PEAPv2-Resp/[TLS-Client-Hello])
Diameter-EAP-Req(PEAPv2-Resp/[TLS-Client-Hello])
Diameter-EAP-Resp(PEAPv2-Req/[TLS-Server-Hello/Certificate])
PAR(PEAPv2-Req/[TLS-Server-Hello/Cert.])
PAN(PEAPv2-Resp)
Diameter-EAP-Req(PEAPv2-Resp)
{Only A4C Authentication}
Diameter-EAP-Resp(PEAPv2-Req(TLS-finished/EAP-SAML-Req)
PAR(PEAPv2-Req(TLS-finished/EAP-SAML-Req)
TLS tunnel established (EAP packets inside PEAPv2 are protected)
PAN(PEAPv2-Resp{EAP-SAML-Resp(VID,ID-token})
Diameter-EAP-Req(PEAPv2-Resp{EAP-SAML-Resp(VID,ID-token)})
Get VID,ID-token
Diameter-EAP-Resp(PEAPv2-Req{EAP-SAML-Req(Assertion)},VID,ID-token)
VID,Artifact
PAR(PEAPv2-Req{EAP-SAML-Req(Assertion)})
PAN(PEAPv2-Resp{EAP-SAML-Resp(Success)})
Diameter-EAP-Req(PEAPv2-Resp{EAP-SAML-Resp(Success)})
Diameter-EAP-Resp(EAP-Success,EAP-MSK-AVP)
Network Qos(VID,ID-token)
COPS Request(VID,ID-token)
Diameter-A3S-Req(SAML-req(ID-token))
Diameter-A3S-Resp(SAML-Resp(NVUP Assertion))
{NVUP Assertion is present}
COPS Decision
Access Commit
PBR(EAP-Success)
PBA
{Registrated}
{IPSec tunnel established IP address available}

Figure 5.7: Registration using ID-Token

sages pass-through. Furthermore, any EAP lower-layer (PANA, IEEE 802.1X) can be used. Additionally, depending on the EAP method used privacy can also be achieved. However, normally EAP methods that are able to carry additional information consume many round trips and it induces performance degradation. On the contrary, PANA allows a big reduction of roundtrips and the whole process is very much faster in terms of messages than an EAP based approach. Thus, we are mandating to use PANA in the MT.

5.4.4 Authorization for Network Depenedent Services

From an operator's point of view network access control is not sufficient to unlock access to *all* specific network-level features. The use of some optional network features (designated hereafter as "network-dependent services") could be conditioned to certain rights in the user profile (and relevant charging model as well). Also, some of these network-dependent services may require use of software (e.g. specific protocol stack) or hardware (e.g. computing power, memory or bandwidth) resources on some entities in the network. Uncontrolled use of such resources may easily lead to Denial-of-Service attacks against these entities.

For these reasons, we describe a dedicated authorization phase for accessing network-dependent services in addition to the initial authentication/authorization phases.

PANA was historically defined to carry only authentication information, with binary results (either access to the network is accepted, or it is refused). After a PANA session had been established between the PaC and the PAA, the only PANA messages that the PAA could have accepted from the PaC were PANA-Reauthentication and the PANA-Termination. Yet, some new PANA messages have been defined recently (the protocol is still in draft stages and is evolving) that allow updating a PANA context in a secure way (taking advantage of the existing PANA Security Association). These new messages are PANA-Update-Request and PANA-Update-Answer, and they can be used to carry customized AVPs.

The network-dependent service example we have chosen to depict in this paper concerns multicast receiver access control. In a nutshell, the problem is the following: a multicast group, even if secured through the use of an encryption key, must

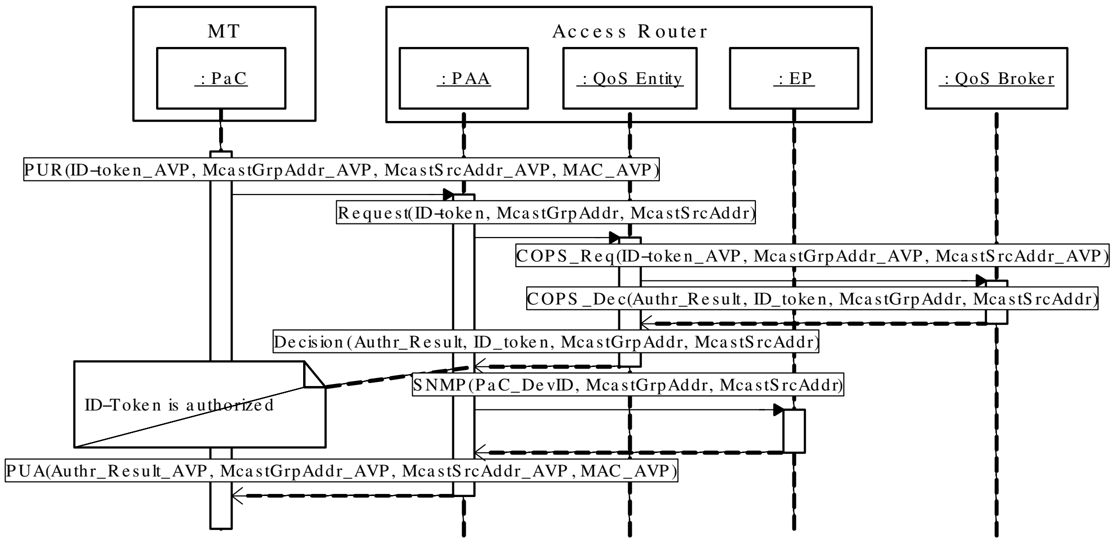

MT
: PaC
Access Router
: PAA
: QoS Entity
: EP
: QoS Broker
PUR(ID-token_AVP, McastGrpAddr_AVP, McastSrcAddr_AVP, MAC_AVP)
Request(ID-token, McastGrpAddr, McastSrcAddr)
COPS_Req(ID-token_AVP, McastGrpAddr_AVP, McastSrcAddr_AVP)
COPS_Dec(Authr_Result, ID_token, McastGrpAddr, McastSrcAddr)
Decision(Authr_Result, ID_token, McastGrpAddr, McastSrcAddr)
ID-Token is authorized
SNMP(PaC_DevID, McastGrpAddr, McastSrcAddr)
PUA(Authr_Result_AVP, McastGrpAddr_AVP, McastSrcAddr_AVP, MAC_AVP)

Figure 5.8: Multicast Receiver Access Control using PANA

actively control which members subscribe to this group, so that malicious nodes do not join and launch DoS attacks against their local access network. Hence the default behavior for an AR (at least in Daidalos) is to silently discard Multicast Listener Discovery [12] (MLD) Report messages as long as the node wishing to receive multicast traffic has not been authorized for doing so.

Figure 5.8 shows how PANA and the ID-Token can be used to authorize a authorize the MT to join a multicast group. This method uses a custom AVP to transport the ID-Token, but this is not a hardship: the PANA-Update-Request and PANA-Update-Answer messages already support carrying custom AVPs.

Chapter 6

Implementation

The concepts discussed in the previous chapters have been successfully developed and deployed in the Daidalos testbed. Integration has been achieved to a degree: it is feasible to run the complete authentication and authorization protocol, using VIDs, ID-Tokens, and PANA with EAP-TLS and EAP-PEAPv2.

The author was responsible for the implementation of the ID Manager on the MT. Some design decisions had to be made, and considerable software engineering had to be done to deliver the implementation in a manner suitable for integration in a ready-to-demonstrate testbed. We discuss some implementation details below.

Figure 6.1 recaptures the software components that we have been discussing in the previous chapters.

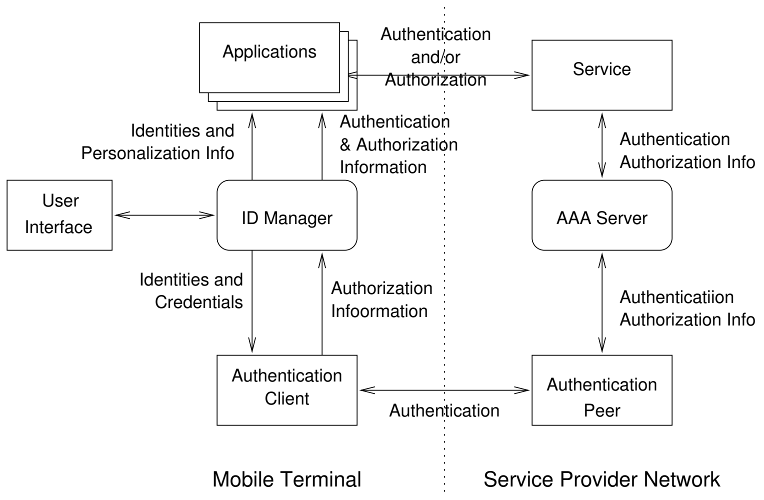

Applications
Authentication and/or Authorization
Service
Identities and Personalization Info
Authentication & Authorization Information
Authentication Authorization Info
User Interface
ID Manager
AAA Server
Identities and Credentials
Authorization Infoormation
Authenticatiion Authorization Info
Authentication Client
Authentication
Authentication Peer
Mobile Terminal
Service Provider Network

Figure 6.1: The ID Manager in the Security Framework

6.1 The Big Picture

As can be seen, on the MT, the ID Manager is accessed by a number of applications to get authentication and authorization information, which they present to services. The authentication client also interacts with the ID Manager, since it is the ID Manager that actually holds the credentials.

The ID Manager must:

- Allow various applications means to request and receive identity related information; some special "management" applications will insert this application into the ID Manager, as well. The interface presented to the applications needs to be programmatic, rather than interactive. Data stored within the ID Manager needs to be protected against inconsistencies that may be introduced due to paralllel insertions by different applications.
- Allow the user some basic control over the contents of the ID Manager and provide an interactive, as opposed to programmatic, interface to inspect the ID Manager.
- Keep the data safely stored on persistent storage between invocations. The data is sensitive, so the process must also ensure as far as possible that the data is not corrupted due to crashes and sudden termination.
- Keep data confidential and authenticate the user prior to releasing the data to processes launched by the user (the user's applications and the management interface).

These requirements led to the decision that the ID Manager should be a daemon process that is started by an external trigger. It could be started at system boot time by the init daemon, or be triggered by a script that awaits for the insertion of some hardware token into the mobile terminal.

6.2 The ID Manager

The ID Manager is implemented in Python. Python provides a high-level, object oriented environment that makes it easier to manage complex data structures.

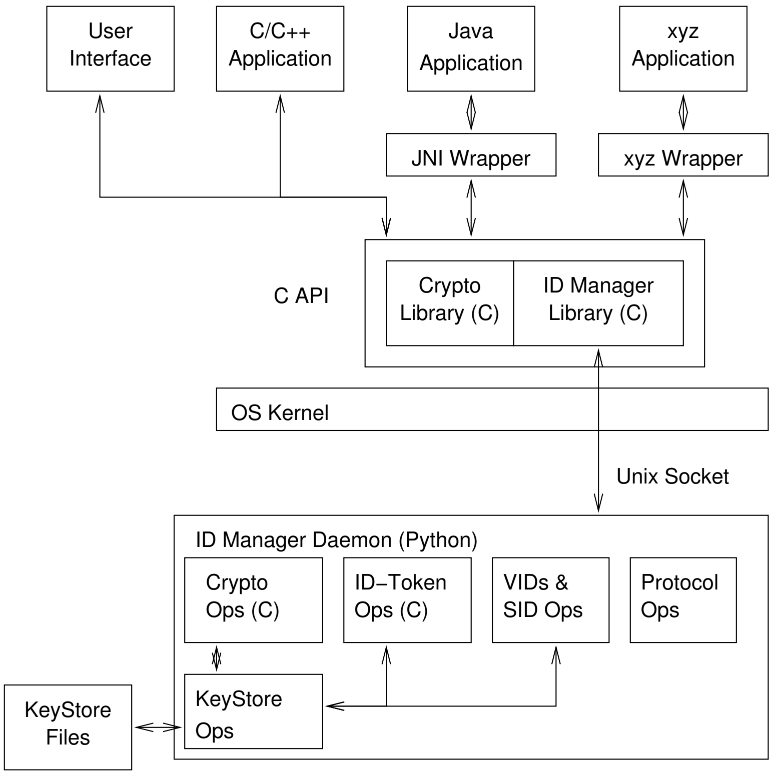


Figure 6.2: Software Component Architecture

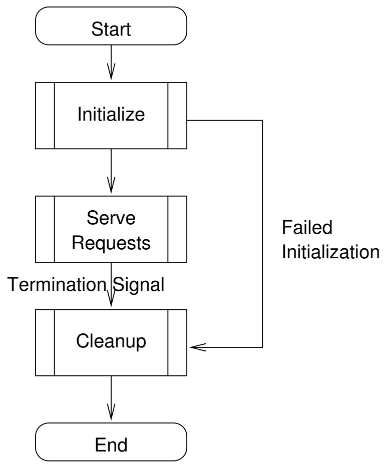


Figure 6.3: ID Manager Process Flow

A file-based representation of an ID Manager's state is called a KeyStore. The ID Manager saves its internal state to KeyStores every now and then. We shall see the usage of KeyStores below.

Figure 6.3 shows the process flow. We shall explain the various processes below.

6.2.1 Initialization

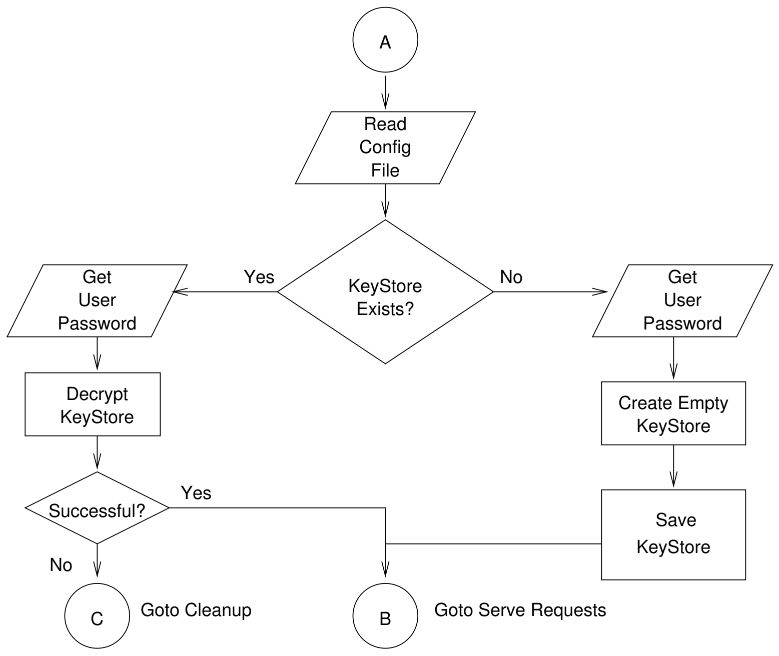


Figure 6.4: Initialization Process Flow

Figure 6.4 shows the actions taken by the ID Manager upon startup.

When the ID Manager is launched, it reads a configuration file, which specifies, amongst other things, which KeyStore to use. The user can change KeyStores between invocations by modifying the configuration file. The user can copy a KeyStore created and stored on one machine to another machine, invoke the ID Manager there, and expect to have access to all information in the KeyStore. In fact, this is essential to bootstrapping the system. The user's Home Provider is expected to pre-provision the user with at least one VID, which is needed to obtain access to basic network connectivity. (Once such connectivity has been obtained, the user may download other VIDs.) This VID and related attributes are supplied as a KeyStore, which the user may copy into some location on his MT. The user then can configure the ID Manager to use this KeyStore.

KeyStores contain sensitive information. If the file containing the KeyStore were to fall in the wrong hands, this information would be compromised. Thus KeyStores are kept in an encrypted form. A password is used to encrypt the KeyStores.

Upon startup, if the ID Manager finds that the KeyStore mentioned in its configuration file does not exist, it attempts to create it. At this point, it prompts the user to enter a password. After that, the ID Manager creates a skeletal KeyStore, encrypts it with the password, and finally stores it on the disk. The password is cached in the ID Manager's memory for use in future writes.

On the other hand, upon startup, if the ID Manager finds that the KeyStore mentioned in the configuration file *does* exist, it will again prompt the user for a password, to decrypt the KeyStore. If the decryption is successful, it means the password is valid (but see 6.2.4), and the ID Manager will proceed to parse the decrypted data and assemble its internal state.

6.2.2 Processing

The ID Manager daemon is a server that listens and responds to queries over a unix socket. Clients may connect to this socket, which has a well-defined path, and invoke operations. Client processes and the ID Manager converse using a query-response protocol, with one message per query and response.

The ID Manager waits until it receives a processing request from a client. When it does, it validates the query for proper information, formatting, etc. If this stage fails, it will drop the request. If the request if formatted properly, the ID Manager will attempt to process the request. It may not always be able to successfully process the request. For e.g., if a client asks for an attribute for a particular VID, and that VID does not exist, then the ID Manager will be unable to supply the client with the requested attribute. The ID Manager responds to such error situations by returning a message indicating failure. In other situations, the ID Manager will return an appropriate response message containing the information requested by the client. Some requests request the ID Manager to perform an operation rather than return information. For e.g., a client could ask the ID Manager to delete a VID. The ID Manager responds to such requests with a message indicating successful operation.

Whenever there is an operation done by the ID Manager that results in a change in its internal state, it will synchronize its state to the KeyStore on disk, using the password to encrypt the information.

Exposing an interface over sockets has the advantage that (a) clients may be implemented in any programming language; (b) new messages may be added to the protocol without invalidating older clients or requiring them to be recompiled; (c) in future, the ID Manager may switch to a TCP/IP socket and reside on a machine different from the clients, effectively implementing a remote, centralized, KeyStore.

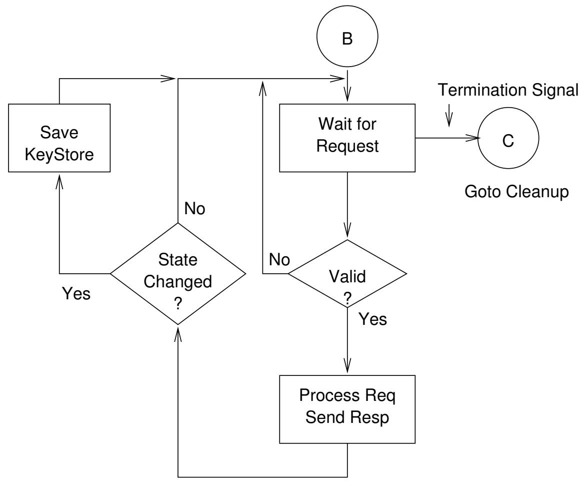


Figure 6.5: Processing Loop Process Flow

Figure 6.5 captures the process flow during this stage. We have glossed over the details of the kinds of requests the ID Manager actually serves.

6.2.3 Cleaning Up

The ID Manager runs until it gets a termination signal. The termination signal is either the `SIGINT` or `SIGTERM` unix signals. The ID Manager intercepts these signals and goes into the cleanup stage. Doing things this way means that abnormal conditions (a system shutdown, for example) is also handled in the same manner as a normal termination and the cleanup stage is always invoked. This ensures that KeyStores are never left in an inconsistent state.

The cleanup stage does little except to check if the KeyStore needs saving, and saves it if necessary. Figure 6.6 shows the process flow of this simple stage.

6.2.4 Reading and Writing Key Stores

As mentioned above, the ID Manager is immplemented in Python. Using Python has the additional benefit that its `pickle` module can be used to read and write data structures (indeed, whole object hierarchies) to/from an encoded format very

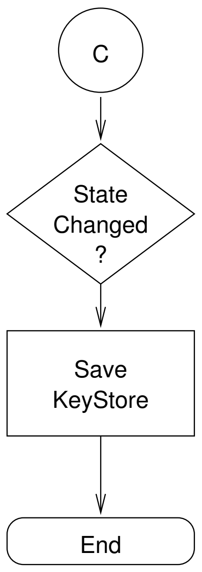


Figure 6.6: Cleanup Stage Process Flow

easily. The ID Manager employes this module to convert its internal state into a byte array that is then written to a KeyStore.

When using an existing KeyStore, the byte array is read off the disk and then converted to the internal object hirarchy, again using `pickle`. The format of the binary data in the KeyStore is independent of machines, endian-ness and operating systems, as long as the `pickle` module is being used. Luckily, Python is quite portable.

Actually, the output from the `pickling` process is first encrypted before writing to a KeyStore; and the contents of a KeyStore are decrypted before `unpickling`.

The program `openssl` is used to encrypt and decrypt the KeyStores. The cipher used is 3DES in CBC mode. Any other block cipher may be used. As mentioned above, a user supplied password is used to encrypt and decrypt KeyStores.

Normally, if a user supplies a wrong password, or an attacker who has got hold of a KeyStore is trying to guess the password, the decryption routines will fail completely, signaling that something is wrong.

In rare cases, it is still possible that when a wrong password is supplied by the user, decryption is successful. The data, however, will likely be gibberish. Worse, it may be a slightly modified version of the original data. If we were to use this data, in the best scenario the ID Manager would crash while assembling its internal state from the data, and in the worst scenario the internal state would be a modified version, crafted by an attacker, of the original data. This means we could compromise the user's privacy and security.

To prevent even this rare situation from compromising security, the KeyStores have a header that the ID Manager checks for integrity information after decryption. If the KeyStore has been modified in any way, or the password is wrong, this header will not decrypt properly and the ID Manager will then discard the rest of the data as suspect. The header is a combination of a random string followed by a checksum over the string, such that the total length of the header is larger than one cipher block. The random string ensures that when using a chaining mode such as CBC, the cipher will produce a cipherstream that will look different every time, preventing an attacker from launching a subsitution attack. The checksum is the data element that serves to assure that the KeyStore's integrity has been maintained. This header is prepended to the output of `pickle` before encryption; and after a successful decryption it is checked and stripped before giving the rest as input to `pickle`.

6.3 The Client Library

Notwithstanding the advantages of a socket-based interface, implementing the ID Manager protocol is quite some work. A lot of messages (there are over thirty at the moment) need to be encoded and decoded precisely. Programmers of client applications prefer being presented with an API. For their benefit, a client library has been implemented that hides the details of managing sockets, and encoding, writing, reading and decoding messages. Programmers simply call functions, suppying data to them as arguments, and getting information as results. The function convert their arguments to an encoded format and then send them over the socket; the functions block on the socket, waiting for the ID Manager's response; upon receiving the response message, the functions decode the message and return an appropriate result value to the caller, including error codes. Thus the library presents client application authors a vastly simplified interface.

The client library is implemented in C, which makes the library usable to applications written in C and C++. Most modern programming languages provide a mechanism to extend functionality in C. Thus if necessary, the library can be wrapped in another language. For example, some client applications are written in Java. A JNI wrapper around the client library is under development to support those applications.

6.4 Software Development Issues

The ID Manager daemon does some cryptographic operations, particularly those involving ID-Tokens, that must be programmed in C. Other functions—wrappers around OpenSSL—already had implementations in C and would be hard to implement in Python.

Fortunately, Python provides an extension mechanism to implement functionality in C and use it from within Python, and this mechanism was used to create a module creating a Python interface to all functions that need to be implemented in C or were implemented in C.

The software's build system (using GNU autoconf and automake tools) is capable of building a ready-to-install RPM file. The software is distributed as source code (in CVS) for developers as well as binary RPMs for testers.

Chapter 7

Conclusion

In this thesis we described an authentication and authorization framework for security in a pervasive network, based on work done within an EU research project, Daidalos. We extracted those aspects of Daidalos that were relevant to our discussion and elaborated on them in Chapter 1. We said that our research goals within Daidalos were to discover how to:

- Reduce the impact of a growing number of different accounts and services on the user and make it easier for users to manage their digital avatars.
- Make it easier for providers to roll out services and charge for them.
- Let users discover new services and providers, and use services from providers with whom they have not had previous experience or an existing business relationhip.
- Let users and providers compose new services on the fly.

Then we asserted that within this thesis we shall concentrate on security and privacy issues related to the above goals.

7.1 Summary

We proposed that an identity management framework was the right way of thinking about security and privacy management in pervasive networks. In Chapter 2 we described prominent identity management frameworks in existence. We identified several deficiencies in them:

- Web-Centrism;
- Burden of patents or other intellectual property rights;
- Inability of incorporate multiple authentication mechanisms and protocols;
- No focus on providers' need to distinguish between a consumer and a subscriber;
- and, no focus on inter-provider trust models.

In Chapter 3 we descibe a subset of components in a pervasive network that are relevant to security and privacy, to create the context for further discussions.

7.1.1 Identities: Uniform Treatment of Users and Providers

We then proposed our own Identity Model in Chapter 4. As can be seen, most of the current developments elide particular aspects in our conceptual identity model. The identities managed in existing specifications are at our 3^{rd} layer (see Figure 4.1) where some identification of the user is present. The 4^{th} VID layer that we introduce does not exist in present specifications. The current trends are more focused on providing frameworks for managing the federation aspects than to define an identity model that correlates to the environment's needs. Our model tries to bring identities to a pervasive network, encompassing business semantics related to the management of the entities in such networks.

We asserted that in our model providers have identity as well as users and subscribers. In the work we have demonstrated, we did not explicitly describe how provider identities can be used. On the other hand, it is perhaps clear from our description of the authentication framework that providers may dynamically build trust between each other using the same means as users build trust between themselves and providers. This is made possible due to the introduction of the ID Manager entity into every providers' network. The ID Manager manages identities and rights, and these identities (according to our model) can refer to users, subscribers and providers. In business jargon, we have demonstrated a system that can be used to build trust relationships in B2C as well as B2B environments.

7.1.2 Privacy: Federations, Authentication and Authorization

Pervasive environments and their associated networks trigger concerns about privacy and security of user information. Ease of use, reduction of management overhead, and enhancement of users' services require sharing of user information; users' need for privacy forbids such sharing. Networks like the one envisioned in the Daidalos project introduce issues regarding accounting and charging.

We introduce a system of ensuring a truce, enabling a give-and-take approach, between these conflicting requirements. We have crystallized the user's manifestation in systems as objects to be managed: namely RegID and VID. We have given a short overview of where responsibilities lie within this model. Federation, based on current practices, was incorprated so to allow information sharing between identities thus enabling innovative, composed and more user friendly services. In Chapter 5 we showed:

- The kinds of providers that may exist in a federation domain and their differing needs and capabilities;
- How to use our Identity Model in federations, and how the privacy requirements elaborated in the same chapter are achieved;
- A framework for performing authorization using our model and standard technologies like AAA networks and SAML;
- Some enhancements to the protocols in the above technologies;
- and finally, specification of a complete authentication and authorization protocols.

7.1.3 Dissemination of Results

The model has been implemented and deployed in a prototype security framework. It is being used for authentication and authorization within Daidalos and will be part of the demonstration system for the project's audit. We describe the implementation of the ID Manager on the MT in Chapter 6, with process flows and software engineering issues.

We have published the Common Authorization Framework described in Section 5.3 in a paper [26] and a poster in the IST Mobile Summit 2005; We have presented the work on PANA/EAP based authorization described in Section 5.4 for network access in a paper [30] and poster in the same conference.

7.2 Future Work

Work is underway to expand the usage of the Identity Model to context management in pervasive applications, and to accouting and charging.

The authentication and authorization framework built is being incorporated in various applications within the project.

In the future, a protocol to actually allow the user to securely create VIDs, and synchronize them with the Home Provider's AAA servers, will be specified.

Also underway is work on an automated agent-like system that can understand security policies of providers and choose/create VIDs for users depending on their privacy preferences. This will use the *profiles* in our model to store such preferences.

We have not mentioned some pre-liminary work done on modeling federation itself. In fact, the federation model is implicit in the specification of the authentication and authorization framework. We would like to model this is more detail and verify the models, to be able to explicitly capture in the system the various possibilities within a federation: how can we model how much is a provider willing to trust other providers; *how* does a provider contrain the amount of information that it reveals about its users, taking into account user preferences, etc. As we mentioned above, the Identity Model enables users and providers to trade levels of privacy and ease-of-use, but actually a model of federation is necessary to be able to allow systems to determine the various trade-off points in an automated manner.

Bibliography

[1] *American Heritage Dictionary of The English Language*. Houghton Mifflin Company, 4th edition, 2000.

[2] B. Aboba, J. Arkko, and D. Harrington. Introduction to Accounting Management. IETF Request for Comments, Oct. 2000. RFC2975.

[3] B. Aboba, L. J. Blunk, J. R. Bolbrecht, J. Carlson, and H. Levkowetz. Extensible Authentication Protocol (EAP). IETF Request for Comments, June 2004. RFC3748.

[4] B. Aboba and D. Simon. PPP EAP Tls Authentication Protocol. IETF Request for Comments, Oct. 1999. RFC2716.

[5] S. Bajaj, G. Della-Libera, B. Dixon, M. Dusche, M. Hondo, M. Hur, H. Lockhart, H. Maruyama, N. Nagaratnam, A. Nash, H. Prafullchandra, and J. Shewchuk. Web Services Federation Language (WS-Federation). Online. DeveloperNetworks., July 2003. Available: www-106.ibm.com/developerworks/webservices/library/ws-fed/.

[6] P. R. Calhoun, J. Loughney, J. Arkko, E. Guttman, and G. Zorn. Diameter Base Protocol. IETF Request for Comments, Sept. 2003. RFC3588.

[7] S. Cantor and M. Erdos. Shibboleth-Architecture DRAFT v05, May 2002.

[8] S. Cantor, M. Erdos, R. B. Morgan, W. H. Steven Carmody, K. Hazelton, and D. Wasley. Shibboleth Architecture. Protocols and Profiles, working draft, May 2004.

[9] A. Durand. How the Nature of Identity Will Shape Its Deployment, Nov. 2003. Available: www.digitalidworld.com/misc/LayersofIdentityArticle.pdf.

[10] H. Einsiedler, R. Aguiar, J. Jähnert, K. Jonas, M. Liebsch, R. Schmitz, P. Pacyna, J. Gozdecki, Z. Papir, J. I. Moreno, and I. Soto. The Moby Dick Project:

A Mobile Heterogeneous ALL-IP Architecture. In *Advanced Technologies, Applications and Market Strategies for 3G (ATAMS)*, pages 164–171, Kraków, Poland, June 2001.

[11] P. Eronen, T. Hiller, and G. Zorn. Diameter Extensible Authentication Protocol (EAP) Application. IETF Internet Draft (work in progress), June 2004. draft-ietf-aaa-eap-08.

[12] R. V. et al. Multicast Listener Discovery version 2 (MLDv2) for IPv6. IETF Request for Comments, June 2004. RFC3810.

[13] D. Forsberg, Y. O. B. Patil, H. Tschofenig, and A. E. Yegin. Protocol for Carrying Authentication for Network Access (PANA). IETF Internet Draft (work in progress), July 2005. draft-ietf-pana-pana-10.

[14] G. Giaretta. MIPv6 Authorizationa and Configuratiion based on EAP. IETF Internet Draft (work in progress), Oct. 2004. draft-giaretta-mip6-authorization-eap-02.

[15] IEEE. 802.11i-2004 Amendment to IEEE Std 802.11, 1999 Edition (Reaff 2003). IEEE Standard for Information technology–Telecommunications and information exchange between system–Local and metropolitan area networks?Specific requirements–Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications–Amendment 6: Medium Access Control (MAC) Security Enhancements, 2004.

[16] IEEE. 802.1X-2004 IEEE Standards for Local and Metropolitan Area Networks–Port-Based Network Access Control, Dec. 2004.

[17] P. Jayaraman, R. Lopez, , Y. Ohba, M. Parthasarthy, and A. Yegin. PANA Framework. IETF Internet-Draft (work in progress), Dec. 2004. draft-ietf-pana-framework-03.

[18] U. Jendricke, M. Kreutzer, and A. Zugenmaier. Mobile Identity Management. Technical Report 178, Institut für Informatik, Universität Freiburg, October 2002. Workshop on Security in Ubiquitous Computing, UBICOMP 2002.

[19] S. Josefsson, A. Palekar, D. Simon, and G. Zorn. Protected EAP Protocol (PEAP) version 2. IETF Internet Draft (work in progress), Oct. 2004. draft-josefsson-pppext-eap-tls-eap-10.

[20] S. Kent and R. Atkinson. RFC 2401: Security Architecture for the Internet Protocol, Nov. 1998. Obsoletes RFC1825. Status: PROPOSED STANDARD.

[21] M. Langheinrich. A Privacy Awareness System for Ubiquitous Computing Environments. In L. H. G. Borriello, editor, *4th International Conference on Ubiquitous Computing (UbiComp2002)*, pages 237–245, Springer-Verlag LNCS 2498, Sept. 2002.

[22] Liberty Alliance Project. Introduction to the Liberty Alliance Identity Architecture. Online., Mar. 2003. Available: www.projectliberty.org/resources/whitepapers/LAP Identity Architecture Whitepaper Final.pdf.

[23] Liberty Alliance Project. Liberty Alliance & WS-Federation: A Comparative Overview, Oct. 2003.

[24] M. Liebsch, T. Melia, and A. Sarma. DAIDALOS. Online., Aug. 2004. Available: www.ist-daidalos.org/publications/Daidalos-overview-2004-03-30.pdf.

[25] E. Maler, P. Mishra, and R. Philpott. Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) v1.1. OASIS Standard, Sept. 2003. www.oasis-open.org.

[26] A. Olivereau, A. F. G. Skarmeta, R. M. Lopez, B. Weyl, P. Brandao, P. Mishra, and C. Hauser. An Advanced Authorization Framework for IP-based B3G Systems. In *IST Mobile Summit 2005 Proceedings*, 2005.

[27] M. Parthasarthy. PANA Enabling IPsec based Access Control. IETF Internet-Draft (work in progress), Dec. 2004. draft-ietf-pana-ipsec-05.

[28] S. Slone and T. O. G. I. M. W. Area. Identity Management, Mar. 2004.

[29] Various. Web Services Specifications. Online: MSDN Library. Available: msdn.microsoft.com/default.asp.

[30] B. Weyl, P. Brandao, A. F. G. Skarmeta, R. M. Lopez, P. Mishra, C. Hauser, and H. Ziemek. Protecting Privacy of Identities in Federated Operator Environments. In *IST Mobile Summit 2005 Proceedings*, 2005.

[31] M. Wu and A. Friday. Integrating Privacy Enhancing Services in Ubiquitous Computing Environments. In L. H. G. Borriello, editor, *4th International Con-*

ference on Ubiquitous Computing (UbiComp2002), pages 237–245, Springer-Verlag LNCS 2498, Sept. 2002.

Appendix A

List of Abbreviations

AAA Authentication, Authorization and Accounting

A4C AAA with Auditing and Charging

AN Access Network

AR Access Router

API Application Programming Interface

CA Certificate Authority

EAP Extensible Authentication Protocol

GPRS General Packet Radio Service

GSM Global System/Standard for Mobile (Communications)

HA Home Agent

IETF Internet Engineering Task Force (`www.ietf.org`)

MT Mobile Terminal

NAS Network Access Server

PANA Protocol for carrying Authentication for Network Access

PaC PANA Client

PAA PANA Authentication Agent

PBNMS Policy Based Network Management System

PKI Public Key Infrastructure

POTS Plain Old Telephone Service

REGID Registration Identity

SAML Security Assertion Markup Language

SIP Session Initiation Protocol

TPSP Third Party Service Provider

VID Virtual Identity

WiFi Wireless Fidelity (Chiefly American term used for the IEEE 802.11 suite of wireless standards and products based on them.)

WLAN Wireless Local Area Network

UMTS Universal Mobile Telecommunications System