Universidade Nova de Lisboa
Faculdade de Ciências e Tecnologia
Departamento de Informática

Dissertação de Mestrado

Context-aware multi-factor authentication

Luís Henrique Fernandes Moura Miranda (aluno nº 27169)

Orientador: Prof. Doutor Henrique João Lemos Domingos

Trabalho apresentado no âmbito do Mestrado em Engenharia Informática, como requisito parcial para obtenção do grau de Mestre em Engenharia Informática.

2º Semestre de 2008/09

29 de Julho de 2009

Acknowledgements

I would like to thank Professor Henrique João L. Domingos for his supervision as teacher and adviser, to the department of informatics of Faculdade de Ciências e Tecnologia at Universidade Nova de Lisboa (DI-FCT-UNL) for conceding a scholarship for introduction to investigation during the past year, to my colleagues at DI-FCT-UNL for their friendship and availability to help me during the last five years, to my friends who were always there despite my absence and most of all to Andreia and to my parents for their patience and unconditional support.

Abstract

Authentication systems, as available today, are inappropriate for the requirements of ubiquitous, heterogeneous and large scale distributed systems. Some important limitations are: (i) the use of weak or rigid authentication factors as principal's identity proofs, (ii) non flexibility to combine different authentication modes for dynamic and context-aware interaction criteria, (iii) not being extensible models to integrate new or emergent pervasive authentication factors and (iv) difficulty to manage the coexistence of multi-factor authentication proofs in a unified single sign-on solution. The objective of this dissertation is the design, implementation and experimental evaluation of a platform supporting multi-factor authentication services, as a contribution to overcome the above limitations. The devised platform will provide a uniform and flexible authentication base for multi-factor authentication requirements and context-aware authentication modes for ubiquitous applications and services. The main contribution is focused on the design and implementation of an extensible authentication framework model, integrating classic as well as new pervasive authentication factors that can be composed for different context-aware dynamic requirements. Flexibility criteria are addressed by the establishment of a unified authentication back-end, supporting authentication modes as defined processes and rules expressed in a SAML based declarative markup language. The authentication base supports an extended single sign-on system that can be dynamically tailored for multi-factor authentication policies, considering large scale distributed applications and according with ubiquitous interaction needs.

Keywords: authentication, context, multi-modal, multi-factor, SSO, ubiquity

Resumo

Tal como são conhecidos nos dias de hoje, os sistemas de autenticação são inapropriados, considerando os requisitos dos serviços e aplicações de sistemas distribuídos de grande escala, heterogéneos e ubíquos. Algumas destas limitações são: (i) o uso de factores de autenticação demasiado fracos ou rígidos como provas de identidade de principais, (ii) a falta de flexibilidade para combinação de diferentes modos de autenticação que considerem interacções dinâmicas e dependentes do contexto, (iii) o facto de não terem por base modelos extensíveis a factores de autenticação novos e emergentes e (iv) a dificuldade em gerir provas de autenticação multi-factor com base em sistemas *single sign-on* para autenticação uniforme e normalizada. O objectivo desta dissertação considera o desenho, a implementação e a avaliação experimental de uma plataforma que suporte autenticação multi-factor, como uma contribuição para superar as limitações mencionadas. As principais contribuições passam pela implementação de um modelo de autenticação extensível, que integre simultaneamente factores de autenticação clássicos e emergentes, podendo estes ser compostos de acordo com requisitos dinâmicos e dependentes do contexto de utilização. O critério de flexibilidade é conseguido através da concretização de uma base uniforme de autenticação, que suporte diferentes modos de autenticação segundo um conjunto de regras e processos expressos numa linguagem declarativa baseada na especificação SAML. A base de autenticação suporta e estende um sistema *single sign-on* para que este permita a configuração dinâmica de regras e politicas de autenticação multi-factor, bem como as necessidades de interacção ubíqua em serviços e aplicações usados em ambientes de grande escala.

Palavras-chave: Autenticação, contexto, multi-modal, multi-factor, SSO, ubiquidade

Contents

List of Figures

1. Introduction

1.1 Motivation

In the nowadays society, services and resources are shared between entities trough large scale and ubiquitous distributed information systems. Many times these systems are based on heterogeneous and ubiquitous computer systems and devices, interconnected by different communication infrastructures including mobile and stationary communication settings. Large scale ubiquitous systems and applications are used as services available with different context-aware interaction requirements and specific access conditions. Today, ubiquitous and large scale systems are composed by multi-application environments or services composed by different application components, more or less specialized for different devices, ranging from personal and convenient devices (as mobile phones, handhelds, etc), to well-managed computers with supervised security conditions in enterprises and institutions. Authentication methods became mandatory as part of any resource sharing system. The above conditions demand new context-aware authentication requirements that will be discussed during this dissertation.

Today, authentication systems are typically based on one of three classic factors: *something the user knows* (as passwords, PINs, shared secrets, etc.) *something the user has* (dynamic one-time-password tokens, smart cards, etc.) and *something the user is or does* (biometric factors). However, each one of those factors has limitations. Passwords are easy to guess and users tend to misuse them, tokens can be lost, stolen or reproduced and are difficult to manage and biometry is an expensive solution, with possible accuracy limitations and personal intrusion issues. Multi-factor and multi-modal authentication comes as better solution since it combines multiple authentication factors, therefore hiding their flaws while enforcing the overall accuracy, convenience and assurance levels. This motivates the objective of the current dissertation that is focused in the design, implementation and experimental evaluation of a platform supporting multi-factor authentication services, as a contribution to overcome the introduced limitations. The devised middleware will provide a uniform and flexible authentication extension for multi-factor authentication requirements and context-aware authentication modes for the ubiquitous applications and services currently available.

1.2 Context

1.2.1 Authentication and distributed systems

Making ubiquitous environments secure essentially boils down to different security properties and issues. As addressed in the classical distributed systems security vision, these properties appear as categorizations of fundamental concepts, such as: authentication, confidentiality, integrity, availability, access-control and auditing [4].

Security properties, adversary models, attacks typology and security services (used as counter measures against attacks to establish security properties) are extensively described in the classical approach and literature of distributed systems security [18],[50] and computer networks security services, protocols and standards [9],[40]. Adversary models, the typology of attacks and adversary hypothesis are usually defined, discussed and formalized in security frameworks [47] inspiring the design of security services and standards.

Security standards also evolve according with new security conditions, new technology, considering possible limitations and context-aware conditions, new adversary models and new attack hypothesis.

For authentication purposes, principals are primarily defined as abstract entities, associated to digital unique identifiers, at different abstraction levels of distributed systems. From this viewpoint, a principal can be an interface, a device, a data-link address, a network address, a software component or a user. Depending on the abstraction level, authentication must be established as an essential property of secure interaction channels, from point-to-point communication (establishing authenticated data-link or network-level endpoints) to end-to-end user level (involving user's identity authentication proofs).

New technologies require the vision of new adversary models, new adversary hypotheses and new metrics of risk. Using the same security services with the same security assumptions for different scenarios sometimes imply on a continuous vulnerability state or inappropriate settings [8],[10]. New ubiquitous systems - as multi-application scenarios supported by different devices - present new challenges associated to different adversary model conditions and different concerns associated to interaction modes supported. There are two important issues directly related with these concerns. The first issue, directly related with the establishment of secure communication channels, needs authentication services to authenticate principals associated to end-points of communication protocols. Verifiable authentication evidences can be

also used as seed values to generate cryptographic keys or other shared secret values or parameters to establish security associations in secure protocols protected by cryptographic algorithms [1],[47],[21]. The second issue is concerned with access-control (based on the previous authentication and evaluation of proofs or evidences of each principal identity). Access-control addresses the problem of controlling the access to resources, objects and operations over those resources and objects). Authentication and access-control are different security properties, even if can be closely related and sometimes are two faces of the same coin.

Authorization is the process of granting access to a resource based on the identification of the interested entity [47]. Access control is achieved by verifying if authentic subjects satisfy one or more permission policies. For that, it is necessary to validate in advance the identities of the subjects involved on the process.

1.2.2 Access control models

There are three classic models for access control: Discretionary Access Control (), Mandatory Access Control () [35] and Role Based Access Control (RBAC) [44]. Access control models may rely on multiple mechanisms. Access control lists and capabilities are the most common among them.

1.2.2.1 Discretionary access control (DAC)

DAC Is an object centric model which, consists on setting permission to objects in a decentralized way. A system based on this model doesn't define any hierarchy between subjects and objects. In this model, it is the subject who sets the access policies to his own objects. As an example there is the UNIX's file permission system, where an administrator can set different policies for three groups of users: the subject himself, other users and for the work-group. A DAC model requires that the identifiers used to map subjects to its resources must be previously authenticated in order to warrant the security concerns related to access control.

1.2.2.2 Mandatory access control (MAC)

MAC comes in opposition to DAC. In this model, all objects are visible and controlled by one or more policy administrators, who grant permission to all users in a centralized way. With this

model it is possible to apply security policies to the whole system at once, which is much more difficult to make in systems based on DAC model. A typical example where the MAC model can be seen is on a Database Management System. In a MAC model, the identifiers associated to subjects representing users and policy administrators must previously authenticated in order to warrant the security concerns associated to access control model.

1.2.2.3 Role based access control (RBAC)

Some systems must deal with different levels of granularity. While in file systems the grain blocks are users and objects, the same may not be appropriated for large systems with large amounts of objects and users. Considering this scenario, a system using DAC or MAC based models is limited in what concerns to scale, since all entities must be managed independently. An alternative is the use of Role Based Access Control Model [44]. RBAC is a neutral and flexible access control model sufficiently powerful to simulate Discretionary Access Control (DAC) and Mandatory Access Control (MAC). With RBAC the permissions to perform certain operations are assigned to specific roles. Subjects are assigned particular roles, and through those role assignments acquire the permissions to perform particular system functions. Since subjects are not assigned permissions directly, but only acquire them through their role (or roles), the management of individual user rights becomes a matter of simply setting the appropriate roles to those subjects, which simplifies common operations such as adding a user, or changing the user's department.

1.2.2.4 Access control lists

ACLs are data structures that store information about objects and the operations that users are able to perform on them [4]. Systems based on DAC model attach ACLs to objects that must be protected, that way user centric policing is achieved. MAC based systems hold centralized ACLs that are crosscutting to all objects.

1.2.2.5 Capabilities

In opposition to ACLs, capability based systems are user centric [4]. Each user has a set of capabilities that may be stated on a credential. Those credentials have information about objects

and respective permissions for a specified user. Every time the user wants to access an object, he must provide his capabilities, which are used by the resource holder in order to perform access control.

1.2.3 Authentication mechanisms, factors and services

1.2.3.1 Classic authentication factors

Authentication processes require digital identities and proofs in order to validate subjects. Those proofs are usually classified in three classes: *something the subject has*, *something the subject knows*, or *something the subject is or does* [4]. The first factor relies on proofs as objects that must be kept by subjects, such as smart cards, SIM cards or cell phones. The second factor is related to information that must be memorized by subjects, like passwords, PINs or passphrases. Finally the third factor depends on biometric traits such as fingertips, face, hand geometry, iris, or voice patterns and rhythm recognition [17].

1.2.3.2 Emerging authentication factors

The widespread usage of mobile devices brings the opportunity for new authentication factors using richer interaction models, convenience criteria and ubiquitous as well as pervasive authentication requirements. Spatial location, relying in positioning identification devices such transceivers, may be used to verify if subjects are who they claim to be assuming they have to be present at some place [11],[13]. Context-aware information can be used as an authentication factor. For example, if subjects have information about the same context, they can mutually authenticate themselves by making the same gestures [26] or by seeing the same objects [27]. Recently, EEG analysis came to be used as another authentication factor. Although it can be considered a biometric factor, EEG analysis can be used to identify brain patterns associated with the subject's brain conditions. Some, tasks when idealized, origin brain patterns that can be used to identify uniquely one entity [25]. RFID technology can also be used to promote authentication proofs associated to secure authentication protocols using RFID tags as subject identifiers[7], [54], [5].

1.2.3.3 Authentication mechanisms

Authentication mechanisms are defined as abstractions which can be used as trustable base building block components. They can be combined in different ways to implement authentication services with different security properties. Examples of authentication mechanisms are: cryptographic algorithms or primitives, authentication protocols or authentication auditing tools. Authentication mechanisms can be categorized as specific mechanisms (corresponding to mechanisms with verifiable formal properties that map specific security properties) and pervasive security mechanisms (no specific to a well-defined security property) [40]. Following the definition existing on related literature, examples of specific security mechanisms are: encipherment based on computational cryptography, digital signatures or public-key certification or notarization or authentication exchange protocols. Examples of pervasive mechanisms are: event-detection evidences, security logging and audit-trails, trust-management or reputation control. According to these notions, authentication factors embrace both authenticated proofs used as input parameters to specific authentication mechanisms as well as complementary evidences to set up pervasive mechanisms.

1.2.4 Uni-factor drawbacks

Authentication schemes uniquely based on *something the user knows* authentication factor have the advantage of uniquely relying on information that it's supposed to be known only by the participating entities, thus being difficult to steal or to duplicate. Nevertheless what really happens is that users tend to use guessable passwords, share them with other users or even write them down, giving origin to security vulnerabilities [4].

Systems based on security tokens can prevent most of the disadvantages previously shown, but the utilization of objects tends to be less practical or non convenient, since they can be lost, stolen, or reproduced. On the other hand, solutions based on security tokens in a large scale system environment are in general difficult to manage and have high operational costs. Some devices used as security tokens can be expensive according with their functionality.

The third factor, overcoming some of these disadvantages, is related to biometric authentication. Human traits are hard to steal and physically intrinsic to users. Besides the advantages, biometrics have some concerns like, trait collectability, noise in sensed data, physical and social

intrusion and lack of accuracy that is determined by their False Rejection Rate(FRR) and False Accepting Rate (FAR)[19, 18].

Emergent authentication factors, as introduced before, are in general application specific and cannot be used by themselves as uni-factors for the majority of the existing scenarios.

1.2.5 Multi-factor and multi-modal authentication systems

One approach to solve the problem of uni-factor authentication is to combine multiple factors [12], [23], [55]. The multi-factor approach improves the assurance level of authentication processes, by combining the advantages of all authentication factors and covering the drawbacks of the remaining. For instance, one classic type of multi-factor authentication, known as two factor authentication, is present on ATMs. Here, users introduce a card (first factor) and a PIN code (second factor). It is considered that any access control system that uses more than one factor supports strong authentication.

Factors can even be used in a multi-modal fashion. In this case different types of information related to the same factor are combined to get uni-factor authentication. A typical example of this approach is present on biometric multi-modal systems that combine two biometric modes like iris and fingertip recognition. The usage of multi-factor and multi-modal authentication improves the level of assurance granted by the validation processes, since the disadvantages of each factor is overlapped by the advantages of the remaining types of proofs.

1.2.6 Ubiquitous systems

The usage of cell phones and other mobile devices is common in the nowadays society. These devices are rapidly becoming smaller, cheaper and more powerful, acting as mini personal computers with complex operating systems providing richer development primitives. Many of the mobile phones available today have features for interaction with users such as cameras, accelerometers, Bluetooth, WI-FI or GPS transceivers. All these specifications had allowed the usage of mobile devices to access complex applications (and sometimes critic like home banking solutions) with high assurance requirements. The usage of ubiquitous devices on these kinds of applications introduces security issues. This is due to the fact that accessing the same application from a fixed desktop at home and do the same at a public place using a cell phone may require different assurance levels. On the other hand, the features available on these devices allow to performing multi-factor and multi-modal authentication processes based on richer

authentication processes. The works described on [27], [45] and [26] show alternative authentication factors, namely *something the user sees*, *somewhere the user is* and *some gesture the user makes.*

1.2.7 Single sign-on systems

Considering an organization where multiple authentication systems are used, the management and validation of digital identities becomes a hard process both for users and system administrators. Users must keep multiple credentials (typically passwords) and log-in at least as many times as the number of resources they want to access. On the other hand, administrators must manage the identities and respective authentication data from multiple subjects.

Single sign-on systems (SSO) address these problems by supplying a centrally managed identity provider accessed by all resources. With a single sign-on system, users only have to pass through the login process once. Therefore, SSO systems act as a unique and uniform authentication base used by all components in the organization. Systems like, OpenSSO [32], Enterprise SignOn Engine (ESOE) [36] and Kerberos protocol, with a new vision using public cryptography [56], are examples of *state-of-art* single sign-on based approaches. Some of these authentication platforms support reliable extension mechanisms, such as OSID [16] and JAAS, [24] and provide interfaces for interoperability with other systems relying on open standards like such SAML [2] and OpenID [41].

1.3 Identified limitations

Authentication services as systems combining different authentication mechanisms - as available today - are inappropriate for the requirements of ubiquitous and heterogeneous large scale distributed systems, applications and services.

This limitation is particularly visible in current application environments supporting ubiquitous, context-aware and multi-channel interaction platforms. In these environments (like an Internet banking and financing services platform) users are constantly accessing a wide range of different devices based on different access criteria, such as: availability, convenience, cost,

functionality specialization, dynamic profiling and other context-aware conditions. These criteria impose different security concerns and tradeoffs with a relevant impact on the need of different and new authentication mechanisms.

The major drawbacks of the current authentication services are the following:

- In general, authentication services make use of weak or rigid authentication factors used as proofs of identity of principals. Most of them are based on secret-sharing, which are very vulnerable such as password-attacks, and social engineering [49].

- Non flexibility for multi-factor authentication purposes adapted to dynamic context-aware interaction criteria. Available technology allows the installation of authentication platforms based on the multi-factor authentication principle to overcome the limitations or risks associated to specific factors stronger than passwords. However, these are proprietary systems; with a limited vision on very specific factors for specific applications and environments (examples include two-factor schemes merging passwords and specific devices such as: tokens or smart cards or biometric authentication servers including support for a specific biometric factor, such as fingerprints). Furthermore, these systems configure authentication factors in a rigid and static way in a user-centered perspective and for a specific application. The available systems don't have a combined vision of user and ubiquitous context-aware centered perspective oriented for different applications and heterogeneous devices. Finally, due to the rigid behavior of available technology, these kinds of systems aren't extensible to emerging pervasive factors such as *something the user sees, somewhere the user is* or *something the user makes*.

- Difficulty to manage the coexistence of different authentication information bases when different authentication factors need to be combined. Taking in account current single sign-on solutions and their specifications, there is no support for the combination of multiple independent authentication systems in a uniform authentication process.

1.4 Contributions

Given the limitations identified on the last subsection, the objective of this dissertation is the design, implementation and experimental evaluation of a middleware platform supporting multi-factor and multi-modal authentication services, as a contribution to overcome the limitations discussed on the previous section. The devised platform must provide a uniform and flexible authentication base for multi-factor authentication requirements and context-aware authentication modes that can be used in ubiquitous applications and services. The main goal is to implement an extensible authentication framework model integrating classic as well as new and emergent authentication factors. These factors must be composed according to context-aware dynamic requirements and user interaction conditions. The flexibility criteria will be addressed by the establishment of a unified authentication back-end, supporting authentication modes as processes and rules expressed in a context-aware multi-factor and multi-modal markup language (CAM2ML).

The management of multi-factor and multi-modal identities and their authentication data is assured by the utilization of *state-of-art* single sign-on authentication platforms, which already support extension mechanisms for new and emergent authentication factors as well as primitives for interoperability with other systems, relying on open standards. Therefore, the objective of the proposed solution is accomplished reusing the services provided by single sign-on systems, which can be dynamically tailored to combine multi-factor authentication rules and policies for multi-device supported applications and different interaction needs.

1.5 Document structure

The remaining parts of this document are organized in the following way: chapter 2 presents the related work and its analysis, considering the requirements identified and the dissertation objectives; chapter 3 is dedicated to the description of the model of the proposed authentication platform; chapter 4 describes the architecture and components of such system; chapter 6 summarizes the implementation issues; chapter 7 presents and discusses experimental result and the evaluation of the implemented platform and finally, chapter 8 concludes the dissertation and identifies future work directions.

2. Related Work

This chapter presents a related work overview according with the motivations and objectives of the current dissertation. The related work presented covers different issues and topics as follow: first a brief overview on authentication factors and classic authentication systems is presented, as well as a discussion on the drawbacks of these systems for the requirements of ubiquitous context-aware and multi-channel interaction platforms; as an approach to overcome those drawbacks, multi-factor and multi-modal authentication systems are introduced. Standard framework specifications for integration of authentication systems are also discussed, as well as the approach of single sign-on based systems and protocols. Finally the chapter presents a critical analysis concerning the design of a multi-factor and multi-modal authentication platform providing uniform and flexible authentication services for context-aware and ubiquitous applications and services.

2.1 Authentication in classic modes

In this section the classic authentication factors will be discussed. Despite their individual drawbacks, they are still used *as is* in the majority of authentication systems.

2.1.1 Static passwords

Most of the current authentication systems use static passwords as authentication proofs. In these systems users must provide passwords in order to get authorization to access resources. Validating the authenticity of an electronic component is a trivial process. However the same is not applicable for humans. The utilization of passwords brings a set of drawbacks and vulnerabilities associated to: user's psychological factors, problems while entering passwords, password design and generation, and password management. In the following paragraphs, these drawbacks are discussed.

- **User's psychological factor** - The main issues concerning static passwords are related

to user's psychology. Users can supply their passwords to a third party on purpose, accidentally or by deception. Users usually share their passwords with other people they trust. This compromises any authentication system by granting access to users that are eventually unknown. An unconscious way of giving a password to an undesired entity is by being victim *of Phishing*, also known as social engineering [4]. *Phishing* is the act of extracting secret information from an authentic user by telling a plausible untruth, for example, a user replying his credentials to a malicious sender who pretends to be a system administrator.

- **Problems while entering passwords** - Another issue that has to be considered is the difficulty that users may experience while entering passwords that are too long or too complex. In those cases, authentication could be a confusing and error prone process creating many unnecessary requests to be treated by the authentication server. In critic systems, where some operations that require authentication must be urgently executed, this issue can bring safety implications.

- **Password design and generation** - Password design must be aware of the fact that user's memorizing capacities are limited. It is hard to remember long and strong passwords and this is one of the main reasons for user's complaints [38], [57]. Organizations where strict policies are adopted ensuring that each password must respect an extensive and rigorous list of rules, see their users either choosing "simple to guess" passwords or writing them down making them easy to steal. On the other hand, passwords with predefined creation rules are more difficult to memorize. With the widespread of systems that require authentication, users have to manage a large number of passwords, which tempts the user to define the same password for different systems. This can lead to vulnerabilities in all systems at the same time if the password becomes compromised. At the same time, this practice exposes passwords in different systems and also causes a lack on user's privacy control when private information is maintained by those different systems.

- **Password management systems** - Authentication systems are subject to attacks at password entry and storage [4]. In order to prevent eavesdropping and man-in-the-middle attacks [47], the interfaces must be aware of some design concerns. A terminal prompting for a password might protect the user by providing the possible means to hide it from other people while the secret is being inserted. Additionally, authentication systems must ensure that passwords are not sent in clear throughout the communication channels. At

last, even the login interfaces should be authenticated in some sort of way that a user knows he is dealing with the right system.

Other kinds of issues refer to how the password repository must be protected. Password files must only store one-way representations of the passwords - such as message digests - instead of maintaining them in clear. Otherwise an attacker that has access to the file-system would have instant access to all user accounts. In fact even if the password file has the passwords protected by a one-way-function, a dictionary attack [47] could be executed by generating random passwords and comparing its one-way representations with the ones present on the password file. Therefore, password files must be somehow protected, for instance by using cryptography or cryptographic tamper-proof devices or modules.

2.1.2 Token based systems

A security token is an electronically represented set of claims about an entity that should be presented when certain types of authentication are requested. Representing *something the user has* authentication factor, they do not force users to memorize any secret information. Instead, this information is stored or processed by software or hardware. Both variations are presented below.

2.1.2.1 Hardware security tokens

Hardware tokens [4] are electronic devices which can store authentication information such as digital signatures, cryptographic keys or passwords. Relatively to their behavior, hardware tokens can supply credentials to authentication systems by providing information such as securely stored static passwords, dynamic passwords - also known as onetime Passwords (OTP) which are regenerated periodically and must be synchronized with the authentication server - or by responding to a challenge thrown by an authentication server. They are typically connected through USB interfaces, Bluetooth or RFID technology, on other cases they are totally disconnected, working as a isolated trust computer bases. In those cases, a built-in display such as in RSA's SecurID solution [42], shows the OTP that must be supplied to the authentication system. Nowadays, this kind of tokens is designed to be small and portable objects, thus easily accommodated by users as seen on Figure 2.1. The main disadvantages related to these devices

Figure 2.1 Hardware security tokens

are the deployment costs, since tokens have to be distributed among all users. On other hand, their usage implies that users have to carry an extra object, which is suitable for being stolen or being lost. Finally, fully disconnected tokens are limited by the batteries life time. The usage of Smart cards is also solution that consumes less energy however it leads to poorer processing capacities and security limitations due to the weaker structure of paper/plastic based cards [4].

2.1.2.2 Software security tokens

Software tokens [4] are stored on electronic devices such as desktop computers, laptops, PDAs or mobile phones. As advantages we can point their lower price, human independence

- since users don't need to memorize any information - and the fact that they don't need independent batteries - as they are inherent to the system holding them. Despite that, authentication systems using software tokens have to deal with the common storage vulnerabilities that passwords do. On the other hand, these tokens are more exposed to virus and other kind of software attacks, which makes them a weaker solution comparing with hardware tokens.

2.1.3 Multi-modal authentication with biometrics

Biometrics can be used as an authentication factor. They can identify and authenticate users given their physiological and behavioral traits. Traditional systems rely on the evidence of fingertips, hand geometry, iris, retina, face, hand veins, facial thermogram, signature, voice [17] and recently EEG pattern recognition [25]. Combined with classic factors they can be used to build stronger authentication mechanisms, however, they are not an efficient solution when used as part of unifactor authentication. The following paragraphs will tackle the main issues concerning authentication trough biometry.

Biometrics overcomes most of the problems noticed on systems based on the first two factors due to their intrinsic nature. Users do not have to remember any passwords. A brute force attack to the feature space is more difficult to perform than in a password based system. That is because it is easier to generate valid passwords than to create or capture biometric samples. On the other hand, it's always possible to increase security by extracting more information from users' traits. The same is not applicable to passwords where making them more complex makes them harder to remember. Finally, users do not have to carry any object as they are forced by token based authentication systems. Other great advantage over the two first factors is that biometry is one of the few available techniques that can be used to prove negative recognition. Negative recognition is the ability of ensuring that one person only uses one identity that is not shared with anyone [18].

2.1.3.1 Biometric authentication system architecture

Biometric authentication systems are typically composed by four main modules: the sensor module, the feature extraction module, the matcher module and the decision module. The sensor is responsible for collecting raw data from the user. That data is supplied to the second module which extracts meaningful features and store them on a normalized representation. The matcher compares extracted features with stored templates and generates the comparison score.

Biometric identifier	Universality	Distinctiveness	Permanence	Collectability	Performance	Acceptability	Circumvention
DNA	H	H	H	L	H	L	L
Ear	M	M	H	M	M	H	M
Face	H	L	M	H	L	H	H
Facial thermogram	H	H	L	H	M	H	L
Fingerprint	M	H	H	M	H	M	M
Gait	M	L	L	H	L	H	M
Hand geometry	M	M	M	H	M	M	M
Hand vein	M	M	M	M	M	M	L
Iris	H	H	H	M	H	L	L
Keystroke	L	L	L	M	L	M	M
Odor	H	H	H	L	L	M	L
Palmprint	M	H	H	M	H	M	M
Retina	H	H	M	L	H	L	L
Signature	L	L	L	H	L	H	H
Voice	M	L	L	M	L	H	H

H= high, M=medium, L=low

Figure 2.2 Biometric traits comparison

Finally the decision module uses the obtained score to accept or reject the user as authentic. Thus, it's necessary that biometric systems decide when to accept an user as authentic given the extracted features. For scores that are higher than a specific threshold, authentication is accepted.

2.1.3.2 Issues concerning authentication trough biometrics

Each of the mentioned biometrics has to deal with a set of concerns, which are noise in sensed data, intra-class variations, performance, acceptability, distinctiveness, non universality, collectability, permanence and circumvention [18].

- *Noisy data* - Noise in sensed data can be noticed when a fingerprint image with a scar is collected, when the user's voice is altered due to a cold or finally when the place where the sensor acquire data is poorly illuminated.
- *Intra-class variation* - By intra-class variation are understood the cases when by some reason there is the need of comparing features extracted by sensors that have the same purpose but achieve it using different techniques.
- *Performance* - Performance is related to the set of issues related to trait extraction and processing response time. Biometrics have a weaker performance comparing to the other

factors.

- *Privacy and acceptability* - Due to intrinsic nature relation between users and their biometric traits, authentication data can be logged and used to gather personal information that users don't want to share. Acceptability defines whether users are comfortable using biometrics. Some of them need physical contact, a fact that raises hygienic questions. On the other hand the lesser collaboration is required, easier will be to forge authentication.

- *Distinctiveness* - It is the capacity that one trait has to identify uniquely one person. For two of the most commonly used representations of hand geometry and face the total recognized patterns that can be mapped on user identification are respectively $10^5$and 10^3 [14].This can be critical on a system with many users.

- *Universality and collectability* - Is the likeliness that a specific trait has of being extracted with the necessary quality from a single user, while collectability measures the effort needed to extract it. For example, quality fingertip features cannot be always extracted, the reason is that some people have low quality skin ridges.

- *Permanence* - Permanence is known as the likeliness of a certain trait still can be extracted regardless the users age. A user may loose access to a certain system if the biometric trait that is being evaluated disappears while he becomes older. On other hand, once a user sees one of its traits compromised, it cannot be used anymore.

- *Circumvention* - Finally, circumvention is one of the most important issues related to biometrics. If somehow user's traits are reproduced by an attacker, it can be used to fool the authentication system through spoofing attacks.

Figure 2.2 [18], shows some biometric systems and their level of resistance to the issues presented on the last paragraphs.

Due to the issues mentioned above, the biometric authentication process is limited by the quality of the extractions and by the variability of its accuracy. Establishing the authentication threshold is a trade-off. By setting a lower value, the likeliness of accepting false users as authentic is bigger, on the other hand by increasing the threshold, some users that are actually authentic may fail to enroll to the system. The first case is measured by a *False Matching Rate* (FMR) and the second by a *False Non-Matching Rate* (FNMR) [18]. These issue are directly

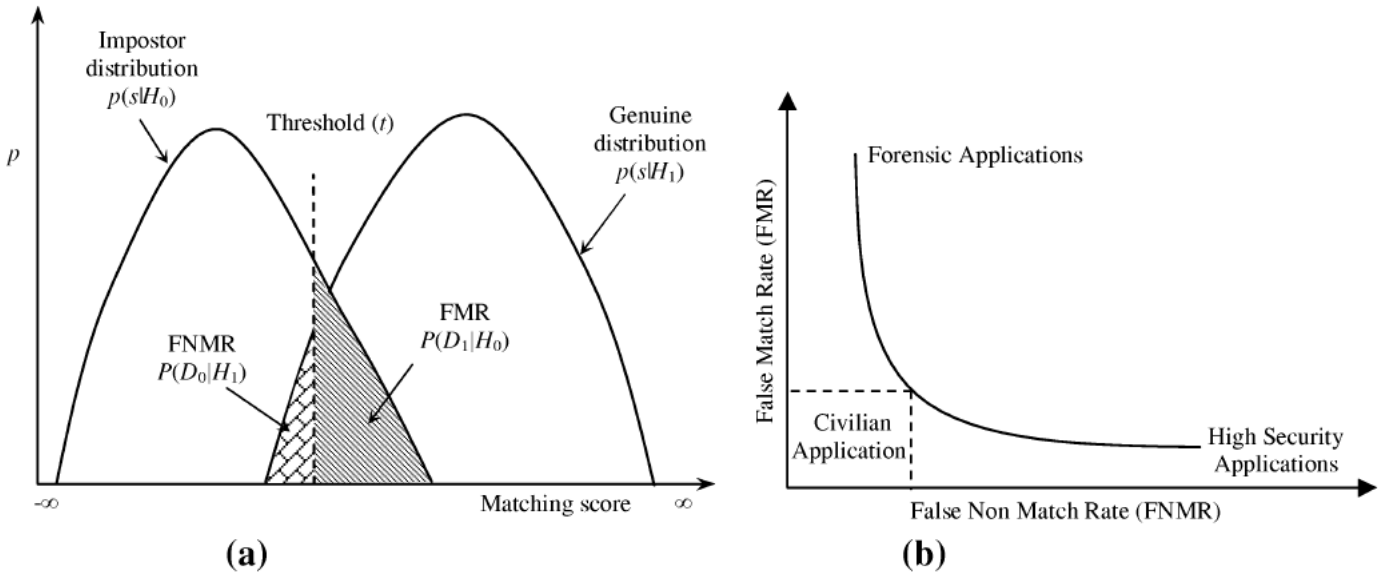

Figure 2.3 Relation between *FNMR* and *FMR*

related concerned with the previous discussed trade-off between False Rejection Rate (FRR) and False Accepting Rate (FAR) for each factor in current biometric technology. The optimization of this balance is specific to each biometric factor. On the other hand, the optimization of this trade-off for fine grained accuracy systems imply on very expensive devices that are not possible to adopt in widespread ubiquitous applications, Figure 2.3 [18]shows how FNMR and FMR are correlated.

2.1.3.3 Multi-modal biometric authentication systems

Most of the existent biometric systems are unimodal [19] thus suffering from the disadvantages introduced above. One technique to reduce the disadvantages while maintaining to some extent the advantages is by combining biometric extractions from different traits. Doing this makes the system more accurate since the likelihood of two trait extractions lead to a *False Non-Matching* or *False Matching* authentication is smaller. Universality, intra-class variation and circumvention are easier handled with the multi-modal approach. If an user can't provide data from one of the requested traits, the system has the possibility of presenting an alternative. Spoofing can be prevented by requesting a random subset of biometrics each time a user tries

to authenticate himself through a challenge-response based approach. It would be harder to an attacker to spoof many traits at the same time. The design issues to take in consideration during the conception of multi-modal biometric systems are the choice and number of traits to extract, the methodology adopted to extract them, the level of trait integration and the cost-performance trade-off. Traits to be available are a decision that is application driven, since it depends on the available sensors and the application requirements. For instance, it is plausible to build a multi-modal biometric system with voice, fingertip and face recognition to be used on a last generation Smart Phone.

Although the accuracy is improved, multi-modal biometric systems are even more expensive solutions as they have to combine many sensors.

2.2 Authentication factors for ubiquitous environments

Ubiquitous computing is concerned with the integration of processing capacity on objects that people daily use. The vision that Mark Weiser had in 1991 [53] was ahead from his time, however nowadays it is already available the required technology for building a word like the one he has predicted. Wireless LANs, cell phones, Smart Phones and PDAs are already widespread among nowadays society.

As referred before, ubiquity brings challenges and opportunities to authentication systems. Classic mechanisms aren't sufficient anymore as they can't deal with context variations imposed by users mobility. The following paragraphs illustrate alternative authentication factors that are suited for context-aware authentication.

2.2.1 Something the user sees

In [27] the authors propose the utilization of the cameras existent on mobile phones as a new visual channel to achieve demonstrative identification of communicating devices formerly unattainable in an intuitive way. on

With Seeing-is-Believing (SiB) protocol, bidirectional Authentication can be accomplished by the following procedure based on data matrices used as two dimensional bar codes.

- each user has a cell phone, which displays a data matrix that can be generated from their public key;

Figure 2.4 Mobile phones performing SiB protocol

- when user A wants to communicate with user B he takes a picture from B's data matrix;
- then user B makes the same from A's data matrix.
- based on the obtained pictures, both users know each others public key and therefore they can build a secure channel trough Bluetooth or WI-FI.

An example of two devices performing SiB protocol is illustrated on figure 2.4 [27].

Unidirectional authentication can also be accomplished between a cell phone and a displayless device with a sticker containing the bar code. Users wanting to communicate with the displayless device start by taking a picture from the sticker, then a secure channel is negotiated trough the wireless channel.

Besides the security of the underlying cryptographic primitives, the security of SiB is based on the assumption that an attacker is unable to perform an active attack on the visual channel, and is unable to compromise the mobile device itself.

2.2.2 Something the user makes

A study made by Rene Mayrhofer [26] addresses the problem of spontaneous interactions using mobile devices. The author shows an alternative to the classic authentication schemes,

which doesn't force the principals to share any initial information.

Imagine two devices, *A* and *B*, trying to communicate with each other. The candidate key protocol describes a key generation method to be used on a secure channel:

- assuming that *A* and *B* share the same environment, they use the same sensors to collect data streams;
- the digests of the features extracted from the streams, called candidate parts, are exchanged and then compared, in order to select only those which are in common to both devices.
- *A* key is generated from the concatenation of common parts and then their digest is exchanged to confirm that they are equal. Man-in-the-middle attacks are then mitigated if a third device, say it *E*, tries to guess the generated key, considering that it doesn't have access to the context shared between *A* and *B*.

As demonstration, the author materialized the idea of sharing contexts by making two devices shaking together. The data collected by their accelerometers was then used to generate the secret key. *E* would not be able to generate the same key because it would have to make exactly the same movements at the same time that *A* and *B* do. The experiments made on this type of attack shown that imitate the exactly movements is almost impossible even when there is cooperation between the attacker and the victim [26]. The security of the algorithm is based on the entropy achieved trough the process of data collection and feature extraction.

2.2.3 Somewhere the user is

Location can be used as an authentication factor. Resources may have different access control policies depending on the place from where the requests are being made. For instance, a doctor is allowed to see the health history of his patients while he is working on the hospital, but it would be desirable that the same was not true when he tries to do it from home. On other hand, location based authentication is natural to the sort of applications where it is not reasonable to distribute security keys among all the users. For those cases the system could only provide access to users that are within a specific area, without the need of having to share secrets.

	Accuracy	Usability	Assurance	TCO	Performance	Pervasive usage	Applicable domains
Passwords, PINS	+	++	- -	++	++	+	++
Security Tokens	+	- -	+	-	-	-	++
Biometry	- -	++	++	- -	- -	- -	-
Location	-	++	-	-	~	- -	- -
Ubiquitous technology	~	++	-	+	~	++	++

++ very good behavior / ~ neutral / - - very bad behavior

Table 2.1 Comparison between authentication factors

All the work made about mobile and pervasive computing brought many improvements on location sensing techniques. They can be divided in three categories, triangulation, scene analysis and proximity [6].

In [11], signed geodetic information is used in combination with the classic factors to associate a request to a physic place, then preventing an attacker of being untraceable and constraining the access to some places in the globe. For that purpose any user would have to possess a transceiver in order to obtain his signed position.

In opposition to world wide location's claims verification, a different approach can be taken by setting a region where any user has access to the protected resources. For that purpose, in [45] the authors, suggested the utilization of a validation module that only authorizes the access to resources for users that are surrounding it at a maximum distance of R. The protocol developed by the authors starts with the user sending a request to the validation module trough RF communication, then the validation module challenges the user with a nonce and finally the user replies trough ultrasound communication. The distance between the user and the verification module is obtained by calculating the time elapsed between the sending of the challenge and the reception of the reply. Ultrasound is preferably used due to the higher latency of sound traveling comparing with light or RF which would require clocks with a much higher frequency.

2.3 Multi-factor and multi-modal authentication

The last section have provided a detailed description for the authentication factors typically used nowadays. We can easily conclude that the utilization of any of those factors introduce disadvantages, which can be more or less critical considering multiple criterions. Table 2.1 makes the comparison between the authentication factors discussed throughout the current section in terms of the following criterions:

- *Accuracy* - Relation between false positives and false negatives assured by the authentication factor.
- *Usability* - Commodity felt by user during the authentication process.
- Assurance - Confidence and level of assurance granted by the authentication factor.
- *TCO* - Total cost of ownership and maintenance.
- *Performance* - Time consumed by the authentication process as well as resource management.
- *Pervasive usage* - applicability to ubiquitous environments.
- *Applicable domains* - Possible situations where the authentication factor can be used as valid proof category.

Composing authentication factors and modes is one solution for achieving higher assurance levels. Combining multiple authentication modes enables to benefit from the advantages of all involved factors while hiding their individual drawbacks. On the other hand, and similarly to what happens in multi-modal biometric authentication, supplying multiple authentication alternatives for the same operations allows to establish richer interaction models considering security and convenience.

2.4 Single sign-on mechanisms and protocols

Single sign-on (SSO) systems are solutions that allow users to authenticate only once and access multiple applications without reauthenthication [52]. Large organizations must deal with multiple distributed applications and domains. The motivations for the design of SSO systems are related to the following situations:

- Each application manages authentication and authorization issues independently from the others using their own directories. The study presented [52] on shows that the in average, the Fortune 1000 top American companies manage more than 150 directories. Each application requires its own authentication proofs, typically based on shared secrets, therefore forcing users to memorize multiple passwords. On section 2.1 the disadvantages of forcing users to keep multiple passwords were already discussed.

- If the attributes of an user changes that must be replicated on all systems, which is a considerable burden to system administrators. The operation costs are high. It's estimated that a great portion of help desk calls are related to password recovery and other authentication issues [52]. The same is applied to the extension to new authentication modules based on different factors. All directories must be synchronized and the applications readapted.

- The notion of roles cannot be applied globally to the organization, since the identities for the same subject are different and widespread among the multiple directories.

Single sign-on systems address these limitations acting as a middleware service reused by different applications, providing centralized management and authentication data access. Applications rely on the platform to retrieve user credentials, which are submitted only once for each session, in order to perform context-aware multi-factor and multi-modal authentication. Figure 2.5 shows the comparison between an organization using traditional logon procedures and other using SSO. This method allows to centrally monitoring all authentication processes and data repositories reducing the number of different passwords. Users only authenticate once and the credentials are reused by all applications. Operation costs are decreased since changes, configurations and fixes are made at only one point.

As mentioned before, authentication systems are traditionally based on pairs of usernames and passwords, however, due to the evolution of ubiquitous computation. Other authentication factors and modes are now available. Different devices such as security tokens, one-time passwords, cell phones, smart cards and biometry can be included on authentication processes, as well as different applications such as mobile applications, firewalls, virtual private networks or extranets. Extending multiple authentication systems to accommodate these changes may be chaotic, on the other hand, strong extension mechanisms can be included on a SSO and therefore all changes and additions become uniform and easier. The same is applicable to interoperability with other organizations. SSO allows uniform interaction with other systems using

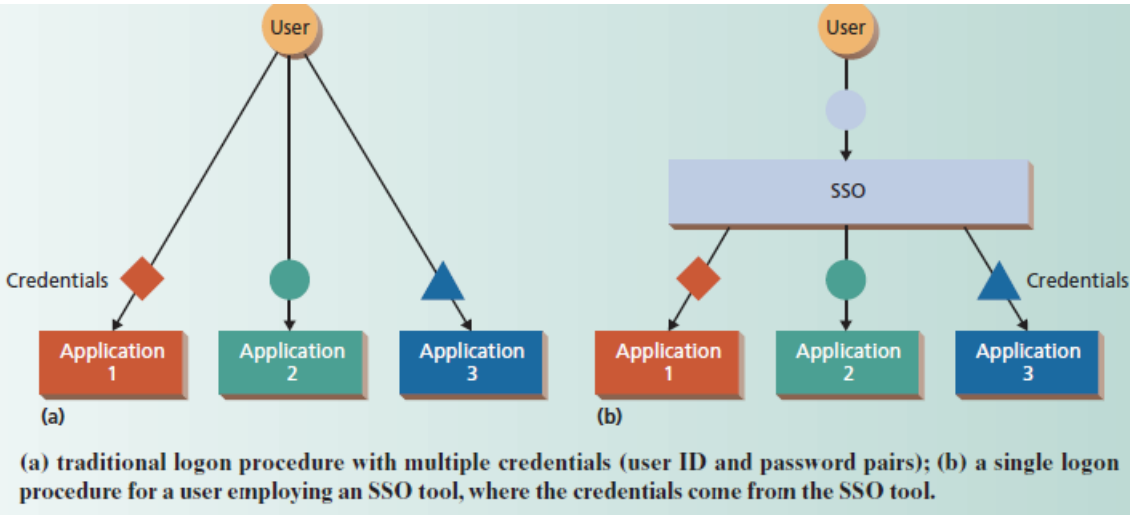

Figure 2.5 Single sign-on middleware architecture

a unique interface relying on well established standards.

Java Authentication and Authorization API [24] is an example of a strong extension mechanism for the inclusion of multiple authentication methods. Security Assertion Markup Language [2] and OpenID [41] are examples of interoperability standards for single sign-on authentication.

Among all single sign-on solutions, OpenSSO[32] and Enterprise Single Sign-On [36] were chosen as fully integrated and tested platform and Kerberos V5[22] as a SSO compliant protocol.

These systems were chosen due to their openness. Therefore it is possible and easier to integrate them on the authentication middleware framework devised by the current dissertation.

2.4.1 Java authentication and authorization API

Since their earlier versions, Java 2 Platforms provided code security mechanisms. Initially they were code-centric in the sense that they only validate the code origin and whether it was signed and by whom. This approach is suitable for the cases where there is the need to decide if the code is legitimate and thus if it should be executed. However, most of the systems are user-centric since they are accessed by many users that must be authenticated before they have access to resources [15]. For that cases the latest Java versions, more precisely the J2EE platform, supply ways of prompting users for login and grant access to resources based on realms

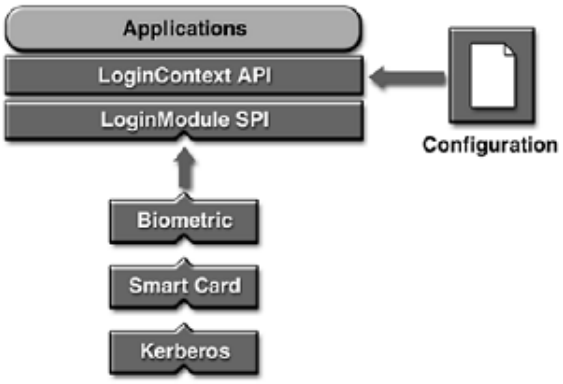

Figure 2.6 JAAS architecture

and user-roles which are defined either by programing or through the setting of rules on a specific file following the recommendations of Enterprise Java Beans Specification (EJB) [15]. Although these mechanisms allow the development of specific authentication modules to java applications, they are not able to integrate existent authentication systems, which don't have access to Java's security layer, neither support easily reconfiguration of the mechanisms used in the authentication processes. This led Sun to design Java Authentication and Authorization API (JAAS). It is an interface that provides authentication and assigns privileges to users in an abstract way so that the applications are unaware of what authentication mechanisms are being used.

JAAS is based on Pluggable Authentication Modules [43], which means that any kind of authentication system can be plugged in simply by defining it on a configuration file. The binding between the JAAS framework and other systems is made trough the definition of *Login Modules*, each one is responsible for the interaction with the external authentication system. The authentication results are returned to JAAS which then pass them to the application modules. Every time one java module wants to authenticate an user, it instantiates a Login Context on JAAS framework, which is responsible for determine which Login Modules must be called. This is set on the configuration file that not only defines what modules must be used but also the order they must be executed. In order to handle the requests prompted by the Login Modules, applications must pass callback handlers during the Login Context Instantiation. A brief description of this architecture is illustrated on Figure 2.6 [24].

Authentication is a two phase process, first all the Login Modules are called and only if all of them succeed the authentication credentials are returned to the applications.

JAAS is then a good solution for integrating different authentication providers in Java systems. Extensibility is achieved by simply changing a configuration file, without the awareness of the top applications.

2.4.2 SAML

Security Assertion Markup Language (SAML) [2] is an emergent standard specification developed by the OASIS Security Services Technical Committee. It is currently on version 2.0 and it defines a set of protocols, bindings and profiles for single sign-on authentication using XML based assertions. An assertion is a message that contains statements about subjects and other information that may be used by applications in order to recognize them as authentic.Assertions can be categorized as the following three types:

- *Authentication assertions* - which carry user credentials and information that concerns whether the user was successfully authenticated or not by a identification provider.
- *Attribute assertions* - which contain information about an entity that is used in the authorization decision process.
- *Authorization assertions* - which contain basic information about users permission to access specified resources.

Assertions are signed by an issuer that must be trusted both by subjects and applications. By itself, SAML is not associated to any transport protocol or any specific technology. It only defines the format for the messages exchanged between parties. However the specification introduces a set of well-defined bindings between SAML and communication technologies such as HTTP or SOAP. Finally, standard profiles are mappings between SAML protocols and real application domains and can be easily integrated, for example, with PKI or X509v3 management solutions. SAML Protocols define three roles executed by specific principals: the Identity Provider (*IdP*), the Subject (*S*) and the Service Provider (*SP*). The interactions between roles are illustrated on figure 2.7 ,which describes the authentication protocol considering a SAML Web Browser SSO profile.

$1.S \rightarrow SP : S$
$2.SP \rightarrow S : AuthReq(ID,S,SP),IdP$
$3.S \rightarrow IdP : AuthReq(ID,S,SP)$
$4.IdP \rightarrow S : [AuthenticationAssert(ID,S,SP)]_{Kidp}$
$5.S \rightarrow SP : [AuthenticationAssert(ID,S,SP,IdP)]_{Kidp}$

R: resource identifier
LOA: level of assurance identifier
C: Context info
AM: authentication modes
AD: authentication proofs
V: validity period,

$[...]_{Kidp}$: *message signed by the identity provider principal*

Figure 2.7 SAML protocol

First versions of SAML did not have support for information sharing between organizations, in such a way that SSO could be spanned to more than one authentication realm. Liberty Alliance proposed a federation based framework, named Liberty Identification Federation Framework (ID-FF) [3], as a standard reference for communication between authentication domains. It is a super-set of the early versions of SAML and describes a circle of trust where the organizations plug their service and identification providers. Each organization it is responsible for managing their access control data as described on the standard. This issue was overcome by the version 2.0, which brings integrated support for federation relying on ID-FF.

Despite being a standard protocol that improves single sign-on and interoperability between organizations, SAML does not consider the description of dynamic mappings between mutable authentication contexts and multi-factor as well as associated multi-modal authentication credentials and proofs.

2.4.3 OpenID

OpenID [19] is a open user-centric authentication framework that permits the utilization of just one identifier to login on many sites.Typically, users have to keep multiple identifiers and respective credentials for each one of the websites where they must be authenticated. This brings convinience and security problems. Convinience is an issue since each website has its own authentication process, forcing users to pass through multiple enrollment processes. This

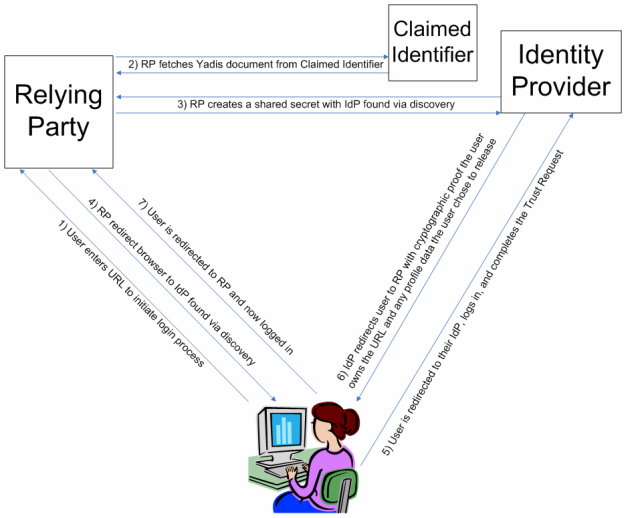

Figure 2.8 OpenID interaction model.

can be inconvenient for users, since there is no guaranty that the identification they normally use is not already taken. On the other hand, multiple authentication systems mean multiple login processes. Security issues are related to the way websites manage users and passwords, some can do it more securely than others. Assuming the fact that sometimes users take the same passwords to different services, if an user does it at a website that has poor or malicious credentials management, he will compromise all his accounts on the other websites.

Open ID relies on global identifiers which may be an URL or an Extensible Resource Identifier (XRI), a specification defining the structure and protocol for abstract identifiers [34]. Users, also known as end-users, may obtain identifiers from two different ways, giving the address of their own blog website or after enrollment process on a XRI compliant identity provider. In the first case, the web server hosting the blog provides a dedicated identity provider. Let's consider a subject *S*, a relying party *RP (*service provider) and an identification provider denoted as *IdP*.

The process of authentication starts with the user accessing a website and entering its identifier on the OpenID form. Then the Relying Party contacts the specified URL to get the Identity Provider that can authenticate the user. In Open ID 1.0 this is accomplished by retrieving directly the address of the IdP. In version 2.0 a XRDS [34] file is retrieved instead, with information about what services can be used to authenticate the user. After knowing what provider to contact, the relying party issues a authentication request, and establish a secret with the user agent related to that session. The user-agent, normally a browser, is redirected to the identification provider site which prompts the user with a login page. The user enters the requested data

and if it is correct the provider returns the secret shared with the relying party as an authentication credential. Finally the user-agent forwards the secret to the relying party which compares the generated secret with the one given by the user who now can decide whether to authorize him or not. Figure 2.8 [19] demonstrates the interaction model mentioned above.

OpenID is focused on Web authentication, namely for blog users. Despite of being a multi-platform authentication standard, it is not suited for the representation of context-aware multi-factor and multi-modal abstractions.

2.4.4 OpenSSO

OpenSSO [32] is an open source single sign-on framework developed by Sun Microsystems. It can be used as a support system for web servers and other kind of systems that share multiple resources and therefore need centralized authentication and access control mechanisms.

As it can be seen on Figure 2.9 [32], OpenSSO provides authentication, authorization, session management, federation, and logging as base services. Its architecture is divided in six layers, but its functionality is distributed by three tiers, the client side, the core services and the plug-in modules or data integration. The client side is composed by policy agents. These components are located near to the shared resources, every time a subject requests a resource, agents intercept it. If the agent hasn't cached any credential for that subject it prompts the subject to login.

The core services integrate the main functionality of the system. Each service is made available to the client side trough web services. Policy agents access those services using Open SSO Client SDK, an API relying on OpenSSO web services. Internally, Authentication and Authorization services interact with specific components, which in conjunction with the OpenSSO framework core decide when to grant access to a specified object.

Single sign-on is managed by the Session component, which maintains information about authenticated sessions. The policy agents get the users session state through this service, this way users only need to authenticate once.

The Federation Component manages the information about the participation of an OpenSSO instance in a circle of trust that is composed by multiple IdPs and service providers. This mechanism allows users to maintain authenticated even when accessing a resource placed on other organization than the one where they did log in. For that, OpenSSO relies on SAML 2.0.

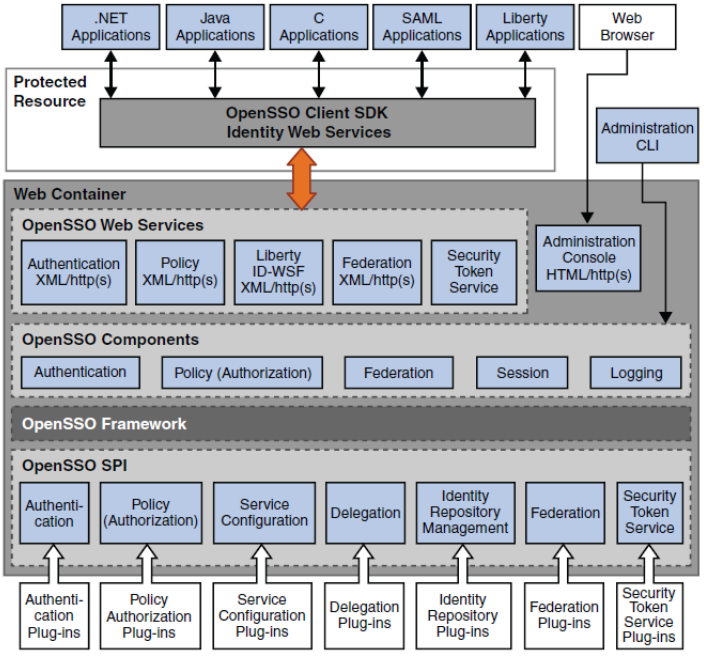

Figure 2.9 OpenSSO architecture

Finally the plug-in modules are systems external to the OpenSSO framework that provide user data, policies, configuration information and autonomous authentication systems. This way, authentication can be delegated to external systems using Service Program Interfaces (SPI) which are made available through a JAAS back-end.

2.4.5 Enterprise sign-on engine

Enterprise sign-on engine (ESOE) [36] is an open source authentication technology for access and data transfer management. It is currently developed by Queensland University of Technology and its main concerns are supporting user authentication at the web tier, federation with business partners, application single sign-on, multi-factor authentication, identity data transfer to applications in real time, access control policies enforcement, services management and platform extensibility. ESOE relies on open standards such as SAML 2.0 for secure data transfer and XACML [33] for standard and fine grained policy definition.

Its architecture relies on a pipe-line trough where all core modules interact. This allows to easily change and substitute core modules and makes the system very extensible. In the same manner as OpenSSO, ESOE enforces authentication by associating protected resources

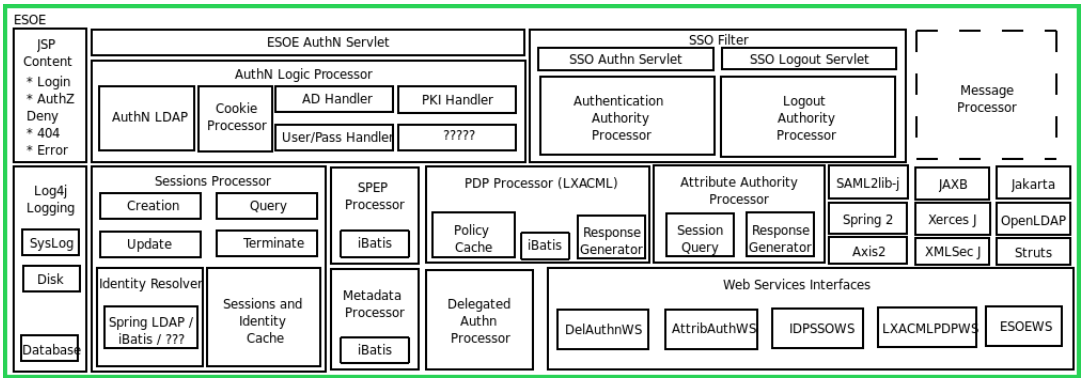

Figure 2.10 ESOE architecture

to special components, which interact directly with the protected resources. These elements are called Service Policy Enforcer Points (SPEPs) and they perform authentication requests to the ESOE, as well as they perform caching and policy enforcement. Since all SPEPs are connected to the same ESOE system, authentication is centrally managed. The components of the architecture are shown on Figure 2.10 [36].

ESOE supports off-the-shelf module connection in order to plug in custom data repositories. Multiple identity resolution sources can be combined and presented to core modules which use them in a transparent way. Federation is limited to the integration with systems that use OpenID and SAML. However ESOE permits the integration of custom modules that deal with different external authentication systems.

ESOE suffers from the same limitations that OpenSSO does. It is most suited for web authentication and doesn't provide mechanisms for dynamic combination of authentication factors. On the other hand, it does not have native support for the integration of ubiquitous systems and devices with specific requirements and interaction models.

2.4.6 Kerberos and the PKINIT approach

Kerberos [22] is an authentication protocol developed at the Massachusetts Institute of Technology (MIT). It is now on its 5th version and it was created in order to address the common authentication problems on a organization with many protected resources. To some extent, Kerberos can be seen as single sign-on system, since authentication services are centrally managed.

Kerberos introduced a new vision over authentication mechanisms by separating users authentication from service authentication, while allowing centrally management. The original

specification defines two main components, the authentication server (AS) and the ticket granting server (TGS). The authentication server is responsible for users initial authentication. The ticket granting server supplies credentials for specific service providers. Separating the authentication in two steps allows users to login only once on the authentication server. The access to different services is made using different credentials created on demand.

In the first step of the protocol both the client *C* and the *AS* generate the same cryptographic key. The authentication server uses that key to encrypt a message containing a new key for communication between the *client* and the *TGS*. Then the client securely communicates with the TGS retrieving a third key, for communication with the service provider *V*. Finally both *C* and *V* negotiate a session key. The authentication protocol runs as follows:

$1.C \rightarrow AS \ : \ Options||IDc||Realm_c||ID_{tgs}||Times||Nonce1||$
$2.AS \rightarrow C \ : \ Realm_c||ID_c||Ticket_{tgs}||\{IDtgs||TS_2||Ticket_{tgs}\}_{Kgc}$
$Ticket_{tgs} = \{Flags||K_{c,tgs}||Realm_c||ID_c||AD_c||Times\}_{Ktgs}$
$3.C \rightarrow TGS: Options||ID_v||Times||Nonce_2||Ticket_{tgs}||Authenticator_c$
$Authenticator_c = \{ID_c||AD_c||TS_1\}_{Kc,tgs}$
$4.TGS \rightarrow C: Realm_c||ID_c||Ticket_v||\{K_{c,v}||Nonce_2||Realm_v||ID_v\}$
$Ticket_v = \{Flags||K_{c,v}||Realm_c||ID_c||AD_c||Times\}_{Kc,tgs}$
$5.C \rightarrow V \ : \ Options||Ticket_v||Authenticator_c$
$Authenticator_c = \{ID_c||Realm_c||TS_2||Subkey||Seq\#\}K_{c,v}$
$6.V \rightarrow C \ : \ \{TS_2||Subkey||Seq\#\}_{Kc,v}$

The client and the authentication server share a secret from what they generate the cryptographic key. This secret, typically a password, is never exchanged between them. However, passwords have the disadvantages also described on section 2.1. Thus, despite all the security assured by the protocol, attackers would attack the secret instead. L. Zhu ET AL, have created a mechanism for initial authentication in Kerberos using Public Key Cryptography (PKINIT) published as RFC 4556 [56]. In this approach, the *AS* doesn't share secrets with users, instead it accepts public key certificates. If a certificate is valid, then the AS returns the first message of the protocol encrypted with the public key that came encoded on the certificate. This way, only the owner of the private key associated to the given public key can successfully decrypt the message. The authentication process continues without any changes comparing with the earlier versions. This approach requires the utilization of a public key infrastructure on the server-side and the utilization of smart-cards and respective readers on the client-side. This method implies

high costs of ownership, which may not be reasonable for all situations where authentication is required.

2.5 Summary and contributions

2.5.1 Overview

In this section, multiple authentication issues were presented. It was shown that each of the three classic authentication factors has its own drawbacks. Systems based on shared secrets are vulnerable due to password misusage by humans, who make them easy to guess and share them with third parties either voluntarily or by social engineering. The utilization of security tokens may not be convenient for users, as they are forced to carry an extra object that can be easily lost, stolen or reproduced. Biometrics seam to overcome the problems mentioned above due to the fact that human traits are difficult to steal and don't force users to memorize any information. However their considerable *False Match* and *False Non-Match* rates allied to their intrusive nature, makes biometrics inefficient and poorly accepted. Other factors such as *somewhere the user is, something the user makes* or *something the user sees* can be used, but they are very application specific and typically can't be used by themselves.

Considering the above drawbacks, a possible solution is to combine authentication factors in a multi-factor and multi-modal way, so that it is possible to take benefit from all the advantages of each factor while its flaws are overcame by the remaining factors. With this approach, authentication processes are more reliable and higher levels of assurance can be achieved. At the same time, this approach can introduce a high degree of flexibility for authentication management according to different systems needs, providing a more appropriate environment to manage different authentication policies for ubiquitous and pervasive systems and applications.

Organizations with multiple users and distributed resources have special needs. Managing the authentication of each resource independently is not practical neither for users nor system administrators. Password synchronization, user information maintenance and integration of multiple authentication modules and data directories significantly increases the total cost of ownership and management. Standard single sign-on systems and frameworks already address these issues by providing centralized authentication platforms reusable by multiple applications. Data storage, user information management and authentication module configuration is made at one single point and it is instantly available for all applications. Extending these systems is easier since they rely on extensible architectures using standard frameworks like OSID or

JAAS. Finally, integration with applications, storage bases and other authentication platforms is improved due to the usage of standard protocols like SAML and OpenID. On the other hand, users don't have to memorize multiple authentication information for each application. They can log-on once and their credentials are accepted by all applications.

Despite of being a complete and integrated solution for most of the actual scenarios, single sign-on systems and protocols as we know today are limited when dealing with multi-modal and multi-factor authentication. Typically, these kind of systems adopt a *one size fits all* philosophy, where the same factors, or combination of factors are used to user authentication regardless the context of the authentication request. SSO systems allow defining static authentication process involving the validation of multiple factors and modes, however they are not flexible to adapt to different context criteria such as device type, location, period of the day, used network and protocols or users and device capabilities.

Considering the new opportunities brought by ubiquitous computation, it is possible to identify a gap that it is not addressed yet. The dynamic adaptation of authentication processes to different context conditions is not addressed by the systems and protocols that are available today. This dissertation has the objective of designing and developing a context-aware multi-factor and multi-modal authentication framework, acting as a middleware platform between single sign-on systems and multi-type and multi-device applications.

2.5.2 Context based authentication and multi-factor authentication

According to the previous considerations, the approach of context based multi-factor authentication seems to be an interesting direction that can be implemented in universal authentication platforms.

Context is intended as the state of a set of properties that can be retrieved from each authentication request, such as the device from where the request is being made, the location of the user, the history of recent authentications and the period of the day. Those attributes are important to help deciding the combination of factors required for a certain assurance level. As example, consider an user living at London that accesses the account balance from is desktop computer on a daily basis. The operation is not critical and the user makes it every day probably from the same fixed device. It is reasonable to consider that the issuer is the same user that did it before, thus a smaller subset of factors needs to be used in order to authenticate him. Now let's consider the same request made on the behalf of the same user located at New York, using a

mobile phone, in a period of the day that is not typical. Probably it was made by the correct user, however the authentication context is different. The required assurance level doesn't change, since the operation is the same, however the necessary proofs to attain the same assurance level must be different. This scenario can be assumed as more suspicious, then a larger combination of factors must be requested.

2.5.3 Contribution

This dissertation presents an integrated authentication framework entitled Context-Aware Multi-factor Multi-modal authentication framework (CAM2). The authentication platform devised by the proposed framework is sufficiently generic to integrate any kind of application that needs authentication services, supporting classic as well as new and emergent authentication modes. For the interaction with the platform using

Considering multiple types of devices, contexts and authentication modes, it is necessary to express uniform context-aware multi-factor authentication protocols and new interaction models. For this purpose, CAM2 specifies a markup language (CAM2ML) supporting SAML abstractions. CAM2ML define the structure of authentication policies describing authentication processes given different authentication contexts.

This dissertation presents the architecture and the implementation of a context-aware multi-factor and multi-modal authentication platform, as a middleware extending the functionalities of underlying authentication systems and protocols such as the ones presented on this chapter. The main contribution is focused on leveraging a base SSO system to provide context-aware multi-factor authentication abstractions and services. This can be achieved by an extensible middleware layer approach that uses SSO base services to provide those abstractions.

The remaining parts of the document will present the model and the architecture for the current solution, followed by details about the implementation and concluding with validations and discussion over the results obtained.

3 . A context-aware multi-factor authentication system

3.1 Scope and requirements

This chapter proposes and presents a context aware multi–factor authentication framework (CAM2) that introduces new interaction models and single sign-on authentication architectures. The issues mentioned above are addressed by developing a middleware that extends the functionality of the current single sign-on solutions. It introduces dynamic authentication processes that adapt the number and type of required proofs to authentication contexts. The design of this framework is focused on the following requirements:

- *Multi-factor and multi-modal authentication* - The framework must provide mechanisms for the implementation of multi-factor and multi-modal authentication, improving the assurance level on authentication processes.

- *Dynamic and context-aware proof requirement* - The framework must consider the detection and evaluation of context states, relying on that information for choosing the appropriate combination of authentication modes.

- *Extensibility for new authentication models* - The framework must provide extension mechanisms that allow the inclusion of new authentication modes, without having to change the authentication platform. The modes to be supported must represent the classic factors as well as the ubiquitous and emergent ones.

- *Generic usage* - The framework must be usable by multiple types of base single sign-on systems. It must be used by multiple types of applications using different interaction models and identities.

- *Acceptable performance* - The logic added by the new services must have low impact on the latency of the authentication processes, when compared with the existent solutions.

3.2 Core concepts

With the aim of implementing context aware multi-factor authentication processes, we have to consider three key concepts: contexts, assurance levels and authentication policies.

- *Context* - By context, we refer to a set of generic attributes that can provide information about the environment during the authentication events. These attributes may be the time of the authentication request, the channel from where the request had arrived, or the type of device that is being used as well as its features. Environment state evaluation must be made in real time before the initiation of an authentication process. Therefore, it is possible to dynamically choose a combination of authentication factors that is adequate for the context given as input.

- *Level of assurance* - The level of assurance is the confidence on authentication that is required by a specific operation. It may vary for different operations where security exploits have different impacts, i.e checking the balance of the bank account vs. making a payment. The introduction of context-awareness leads to the requirement of different combinations of authentication modes, depending on the given context. Authentication events may occur under different environments which can be considered more or less suspicious. Therefore, for the same operation, with the same required assurance level, the system must dynamically adapt the number of authentication modes to the context state that is observed.

- *Authentication policies* - In order to implement context-aware multi-factor authentication, there is the need of defining mappings between the concepts that were presented on the last two paragraphs. CAM2 define those mappings through authentication policies that specifies assurance levels, authentication context instances and combinations of authentication proofs. This way, every operations with specific assurance levels are associated to multiple authentication policies, one for each different context that is expected.

CAM2 framework specifies a generic interaction model for context-aware multi-factor authentication. For that interaction model, it also describes the behavior of CAM2 compliant identity providers and specifies the data exchanged during context-aware multi-factor authentication processes. This last item is accomplished through the definition of a context-aware multi-factor and multi-modal authentication markup language (CAM2ML). Next sections will discuss in detail each one of these aspects. Figure 3.1 illustrates an example of how can CAM2 be used.

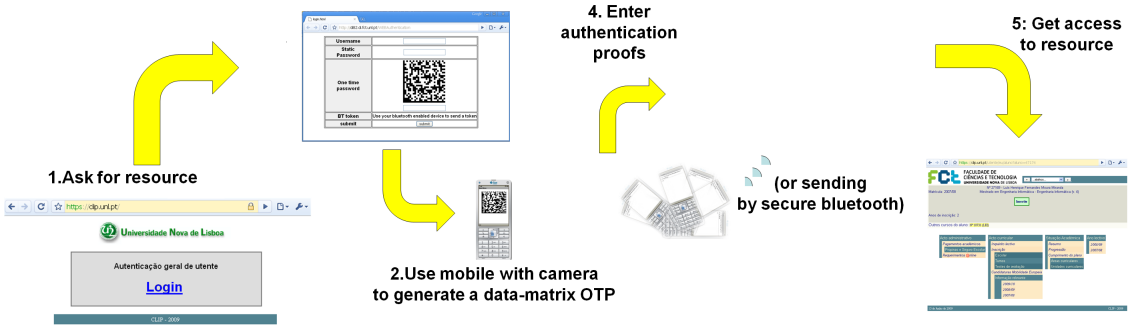

Figure 3.1 CAM2 Utilization

3.3 Interaction model

CAM2 framework defines a generic interaction model based on single sign-on models like the ones mentioned on section 2, namely SAML and OpenID. It is composed by three participants: a subject (*S*) a service provider (*SP*) and a CAM2 identity provider (*IdP*). CAM2 interaction model defines two events, the policy registration and the authentication process. The full interaction model is illustrated on Figure 3.2.

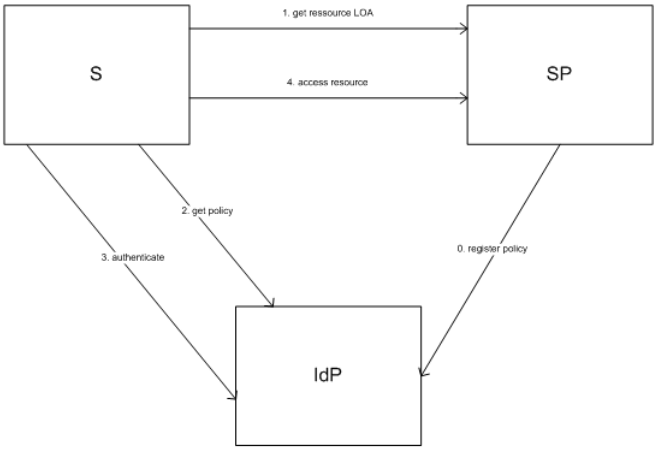

Figure 3.2 CAM2 interaction model

- *Policy registration* - For each combination of required assurance level and possible context state, service providers must submit a specific authentication policy on the CAM2 identity provider(Step 0 on Figure 3.2). Later, the *IdP* will rely on these policies to enforce context-aware multi-factor and multi-modal authentication.

- *Authentication process* - A subject that wants to access a resource on the service provider must obtain the assurance level required for that operation (Step 1). Then *S* interacts with CAM2 *IDP,* which evaluates the authentication context and returns the appropriated authentication policy given the required assurance level (Step 2). Finally *S* gathers the necessary proofs and try authentication on the *IdP* (Step 3), in case of success the subject obtains an authentication credential that can be presented to the *SP*(Step 4).

3.4 CAM2ML

Context aware multi-factor and multi–modal markup language is as XML based language for describing the contents exchanged during CAM2 authentication. Like SAML it relies on assertions to declare statements that involve subjects and states of the authentication process.However it differs from SAML since it has support for describing contexts and their respective mappings to combinations of authentication proofs. Assertions are composed by generic and well structured fields,which allow transparent interaction between heterogeneous parties. This allow multiple types of systems to use the same CAM2 authentication platform as a single sign-on system.

3.4.1 Assertions

All CAM2ML assertions keep a common set of identifiers, namely for the issuer entity, the service to be accessed and the required assurance level. The issuer entity identifies the CAM2ML identity provider, the application identifier binds the assertion to a service provider and finally, the assurance level points the confidence associated to that assertion. CAM2ML assertions are signed by the issuer that identifies them. The issuer must be trusted by all parties and can be discarded both by the subject and by the service provider. In each step of CAM2

authentication interaction, two types of assertions can be exchanged, namely policy assertions and authentication assertions.

- *Policy assertions* - These statements express authentication processes. Each policy assertion contains a set of context attributes that defines a possible environment state for an authentication event. The assertion also carries a combination of authentication modes, which define the proofs required for accessing the identified service provider with the specified assurance level and context attributes. Considering the interaction model already shown on the earlier subsection, policy assertions are present during the policy registration phase (Step 0) and on policy retrieval phase (Step 2).
- *Authentication assertions* - Are credentials stating that a specified subject has successfully achieved authentication. To be more precise, they state that a specified subject successfully obtained authentication on a specified CAM2 identity provider for a specified assurance level. Service providers must map that assurance level identifier into access control operations and decide whether to grant access to subjects. Additionally to the common identifiers carried by all assertions, they also contain information about the validity period of the assertion. Authentication assertions are exchanged between the subjects and the identity providers (Step 3) and between subjects and the service providers (Step 4).

Notice that CAM2ML does not define the message format for the retrieval of the assurance level identifier during Step 1. This is due to the fact that the semantic of these identifiers is managed only by the service providers and is used to perform access control decisions. CAM2ML only specifies that assertions must carry a field that identifies the assurance level.

3.4.2 Requests

For each assertion a request message is also defined. A party involved on the authentication process uses them to retrieve assertions from other. Requests have common field identifiers for the subject, for the service provider and for the required assurance level. As expected, there are two types of requests, policy requests and authentication requests.

```
<cam2ml:assertion id="ASSER-0010"
    type="Policy"
    xmlns:cam2ml="http://CAM2ML.di.fct.unl.pt/Basic">
 <cam2ml:issuer>
  <cam2ml:cn>CAM2_DI_IDP<cam2ml:cn>
   <cam2ml:signature>
    <cam2ml:algorithm>SHA1withDSA</cam2ml:algorithm>
    <cam2ml:value>
     302c021471b9b1a654b6f818628fc112738ee5e3c92e0de
     3021457d9193b7385e59adaa0e182f916e0d345b23ba3
    </cam2ml:value>
   </cam2ml:signature>
 </cam2ml:issuer>
  <cam2ml:application>http://di164.di.fct.unl.pt/SecureClip</cam2ml:application>
<cam2ml:levelOfAssurance type="level2"/>
 <cam2ml:policy>
  <cam2ml:authentication_modes>
   <cam2ml:mode type="PASSWORD"/>
   <cam2ml:mode type="HW_TOKEN"/>
  </cam2ml:authentication_modes>
 </cam2ml:policy>
</cam2ml:assertion>
```

Figure 3.3 Example of a policy assertion

- *Policy requests* - Policy requests carry a combination of context attributes that describe the authentication environment. The CAM2ML identity provider evaluates the context stated on the request message and returns a policy for the specified service and required assurance level. Policy requests are present on the second step of the interaction model.

- *Authentication requests* - Authentication requests store authentication data gathered by the subject. This data is the representation of multi-factor and multi-modal proofs in conformity to an authentication policy obtained earlier. Each proof has an identifier for the authentication mode it represents (passwords, security tokens, fingertip recognition, etc.). Along with the authentication proofs, authentication requests contain the context description. The identity provider relies on authentication requests firstly to evaluate the context and the assurance level identifier, secondly to confirm if they correspond to any valid authentication policy and finally to evaluate the authentication proofs. Authentication requests are present on the third step of the interaction model.

```
<?xml version="1.0" enconding="UTF-8"?>
 <cam2ml:request type="policy_request"
  xmlns:cam2ml="http://CAM2ML.di.fct.unl.pt/Basic">
  <cam2ml:application>http://di164.di.fct.unl.pt/SecureClip</cam2ml:application>
  <cam2ml:levelOfAssurance type="level2"/>
   <cam2ml:policy_request>
    <cam2ml:contex>
     <cam2ml:protocol type ="HTTP"/>
     <cam2ml:device type ="MOBILE"/>
     <cam2ml:time>
      <cam2ml:interval time ="02:12:15.267"
     </cam2ml:time>
    </cam2ml:context>
   </cam2ml:policy_request>
  </cam2ml:application>
</cam2ml:request>
```

Figure 3.4 Example of policy request

3.4.3 Reference protocol

CAM2ML can be applied to every possible instance of the interaction model, however it is possible to describe a generic CAM2ML based protocol for CAM2 interaction model:

$$
\begin{aligned}
&1.S \rightarrow SP && : R, S \\
&2.SP \rightarrow S && : LOA \\
&3.S \rightarrow IdP && : PolicyReq(LOA, C, S, SP) \\
&4.IdP \rightarrow S && : [PolicyAssert(LOA, C, SP, AM, TS)]_{Kidp} \\
&5.S \rightarrow Idp && : AuthenticationReq(LOA, C, S, SP, AP) \\
&6.IdP \rightarrow S && : [AuthenticationAssert(LOA, S, SP, V, TS)]_{Kidp} \\
&7.S \rightarrow SP && : [AuthenticationAssert(LOA, S, SP, V, TS)]_{Kidp}, R, S
\end{aligned}
$$

R: resource identifier, LOA: level of assurance identifier, C: Context info,
AM: authentication modes AD: authentication proofs, V: validity period,

TS:time stamp, $[...]_{Kidp}$*: message signed by the identity provider principal*

The protocol runs as follows:

1. the subject starts by requesting access to a service;

2. the service provider then returns the identifier of the level of assurance needed to access that service;
3. the subject makes a policy request to CAM2 identity provider, which evaluates the assurance level identifier and the authentication context of the request;
4. then it returns the appropriate policy;
5. the subject forwards an authentication request to the identity provider with the required proofs.
6. finally, if authentication succeeds, the identity provider returns an authentication assertion that can be presented as a credential to the service provider.

This protocol can be extended or re-adapted to be in conformity to specific instances of the basic interaction model. For example, in some systems, the subject may store policy assertions gathered in the past and re-utilize them without having to pass through the first four steps of the generic protocol. Even the content of messages may be changed depending on the application scenarios and their specific requirements.

3.4.3.1 Security analysis

CAM2ML protocol supports authentication and integrity properties for policy and authentication assertions through cryptographic signatures. Those signatures involve all the data carried by assertions, then assuring that relying parties are able to trustfully accept or reject them. Every assertion includes a time-stamp that must be used against replay attacks. Confidentiality, integrity and bilateral authentication at the endpoint level is not defined by CAM2ML. Instead those properties must be assured by transport level mechanisms such as TLS/SSL or message level for example using WS-Secure.

3.4.4 CAM2ML vs SAML

CAM2ML and SAML have the same purpose, define standard messages to implement standard and domain independent authentication processes. The reason for creating a new language is related the lack of expressivity of authentication contexts, multi-factor and multi-modal authentication processes and dynamic levels of assurance that languages such as SAML can offer nowadays. Instead of developing a new language from scratch, SAML could eventually

be extended to the requirements of context-aware multi-factor and multi-modal authentication, however, one of the objectives of this work is to provide simple management and configuration for authentication processes, which is done by definition of authentication policies through the edition of XML documents. The simplicity of CAM2ML allows users with low technical capabilities to use it. On the other hand, abstracting all the details of SAML enables focusing this work only on context-aware multi-factor and multi-modal issues. The strengths and capabilities of SAML are not ignored, CAM2 compliant identity providers rely on it to implement the basic authentication mechanisms, as will be discussed on the next subsection.

3.5 CAM2 identity platform

Single sign-on systems as we know today have the limitations already described. CAM2 framework describes a reference architecture for a middleware that extends those single sign-on systems enabling them to support context-aware multi-factor and multi-modal authentication. Its architecture respects a three-tier architectural style. The 3rd tier (or data integration and management tier) corresponds to the base identity management support and authentication processing facilities that can be provided by a base single sign-on system from which CAM2 is being extended, in order to support the integration of multi-factor validation factories. The 2nd tier (or authentication logic tier) is represented by an authentication engine layer that evaluates the context of each request and retrieves the combination of factors and modes required to perform authentication. It is also responsible for the management of context-aware authentication policies storage and authentication delegation. Finally the 1st tier (or client integration tier) is implemented by a unified front-end that externalizes different authentication protocols and captures the authentication requests from the most variable kind of applications, constructing a uniform authentication request that is forwarded to the authentication logic layer. Figure 3.5 illustrates this model. Each tier is fully described on the next subsections.

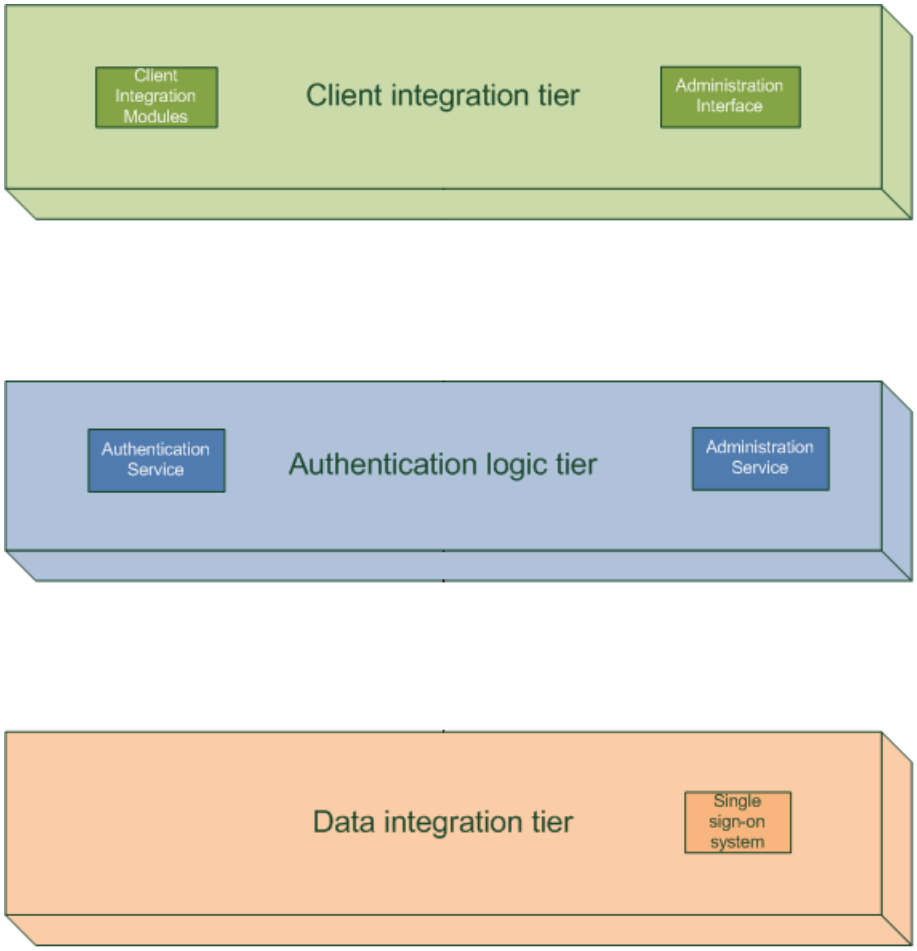

Figure 3.5 CAM2 Identity Provider: Reference architecture

3.5.1 Client integration tier

CAM2 framework has the objective of providing single sign-on authentication for multiple types of systems and devices. Considering that, this tier supports different interaction types, which are specific to the applications accessing CAM2 compliant identity providers. The client integration tier is composed by integration modules that are responsible for adapting the client specific authentication mechanisms to CAM2ML uniform protocol. This enables the integration of CAM2 based authentication systems on already existing applications with a minimum of changes to their code (applications still have to implement interfaces for gathering multi-modal

and multi-factor validity proofs).

The registration and management of authentication policies is externalized by this tier. Service providers interact with this layer which keeps a set of application specific integration modules. Both for authentication and for management, the logic of the processes is implemented by the lower layers. The communication between the client integration tier and the lower tiers is done using a CAM2ML based protocol.

This model beneficiates the applications and devices with limited resources or specific interaction requirements, however, other kind of applications with more capabilities may access directly the authentication logic layer using the messages and the protocol defined by CAM2ML.

3.5.2 Authentication logic tier

Context-aware multi-factor and multi-modal authentication is implemented at this tier. CAM2ML requests are handled at this level by three services which are responsible for policy retrieval, validation of authentication proofs and policy management and registration.

CAM2ML authentication policies map contexts, applications and assurance levels in a combination of authentication modes. For each policy request that the authentication logic tier receives, it evaluates the context attributes contained on the message and collects an authentication policy from some repository, according with the application and assurance level that the request caries.

Policy management and registration logic is made through an administration interface that allows the insertion of new policies and edition of existing ones.

This tier handles CAM2ML authentication requests. It relies on the lower tiers and on the inner policy repository to perform context-aware multi-factor and multi-modal authentication. The process is divided in the following steps:

- Firstly the context attributes contained on the request must be evaluated. If there is any applicable authentication policy then the process follows to the next step.
- If the first step succeeds, the authentication proofs are evaluated. Each factor is delegated to a specific authentication module resident on the data integration tier (will be discussed later). If, and only if, all individual authentication processes succeed, the whole multi-factor process also succeed and a new authentication assertion is generated.

The evaluation of the authentication environment both during the policy retrieval and on the authentication process is redundant, however it is necessary in order to maintain the statelessness of the interaction model. This property gives subjects the possibility of retrieving a policy only once and using it for multiple authentication events. On the other hand, online context evaluation allows real-time impact after changes on authentication policies without the need of any configuration.

3.5.3 Data integration tier

The first tier maintains access to a set of specific authentication modules. These modules may perform static unifactor or multi-factor validation implemented by instances of typical single sign-on authentication bases. Each of those systems may keep one or more authentication methods such as LDAP[20], Biometry or UNIX's password files. Therefore, the data integration tier works as wrapper for multiple authentication systems externalizing an abstracted view over them. The upper tiers only have to provide identifiers for the authentication modes and the principal to be validated, along with the proofs he has supplied. The integration of all platforms is centrally managed and it may rely on standard authentication specifications such as SAML or OpenID, then taking advantage of all the benefits brought by these *state-of-art* technologies.

The authentication logic tier relies on data integration tier to delegate the validation of individual proofs. This process is transparent for the upper tier, which is unaware of the details of integration running on the background. With that abstraction is possible to change, extend and reconfigure the underlying authentication systems without interfering with the context-aware multi-factor authentication combined process.

3.6 Application scenarios

The methodology introduced by CAM2 framework can be instantiated in multiple domains that has the common authentication requirements. The next subsections will introduce some examples of the generality and applicability of context-aware multi-factor and multi-modal authentication.

3.6.1 Web authentication

Long gone are the days when websites were only accessed from desktop computers kept on a room. Today the web is accessible from every kind of devices such as cell phones, PDAs or public access points. Therefore web servers are exposed to multiple interaction contexts due to the mobile nature of ubiquitous environments. The number of validation proofs required to achieve the same assurance level on authentication processes may vary for each state of the environment.

The implementation of context-aware multi-factor and multi-modal authentication for the Web can be done using CAM2 framework. Basing on the principles of specifications like SAML and OpenID, CAM2 supports web authentication through HTTP redirection. When a user tries to login on a website supporting CAM2 authentication, his browser redirects him to a web page belonging to a CAM2 compliant IdP. This web page can extract some context information from the request, such as the time it was issued, the region and the network type from where it was invoked or the device type. After context examination, the CAM2 IdP chooses the adequate authentication policy and the required authentication proofs are prompted through CAM2 web page. If authentication succeeds the browser is again redirected to the service provider using an URL that contains an authentication assertion. This assertion can be used as an authentication credential.

3.6.2 Mobile authentication

Mobile phones are one of the greatest examples of ubiquity. The evolution of pervasive computation is strongly connected to the development of new cell phones with enhanced features. Nowadays these devices support a vast sort of technologies as for example Bluetooth, WI-FI, UMTS , capture of video and audio, biometric recognition or movement detection. These features bring the opportunity of implementing richer interaction models during the authentication steps, allowing to extract multiple identity proofs from different authentication factors.

Let's consider the example of a bank that supplies a payment service using mobile phones. A simple application installed on the device asks the user for the payment details and next send them to the service front-end located somewhere on the Internet. Along with that information, the user must prove that he is the owner of the account from where the money is going to be taken. The model described by CAM2 framework can be applied to this scenario. The

application installed on the mobile device can detect the context conditions and request the adequate policy to a CAM2 IdP. After getting it, the application prompts the user for authentication proofs, which can combine classic factors with the ubiquitous methods described on section 3 (*Something the user sees*, *something the user makes*, *somewhere the user is*). These last type of factors potentates the commodity on the interaction while increasing the assurance level on the authentication process.

3.6.3 Spontaneous authentication

The massive usage of mobile devices brings some issues. Users exchange digital contents with multiple users. Some of them never had any type of interaction and do not share any previous information. CAM2 model is suitable to be applied on these scenarios. Let us consider an user A holding a resource on his cell phone and another user B wanting to access it. User A needs to authenticate user B in order to authorize him to access the resource. As already have been said, the users don't share any previous information, however they can acquire the same context attributes, which can be a data matrix printed on users A phone, a gesture both users do with the device or other kind of data that can be collected. Finally, user B presents the information collected by his device to a CAM2 IdP, which use it to require a set of proofs and generate authentication credentials that can be presented do user A. Therefore, spontaneous authentication can be addressed using a third party authentication base relying on CAM2 framework.

3.6.4 Kerberos extended by CAM2

Authentication systems are not the only suffering from the limitations when dealing with context-aware multi-factor and multi-modal authentication. The authentication protocols and standards used by those applications must also be adapted. As an example there is the Kerberos protocol. As discussed on section 3, it is a protocol known by its robustness during the authentication process using multiple cryptographic symmetric keys. Nevertheless it has a major vulnerability point since the key between clients and the authentication server (AS) is generated from a secret shared between the two parties. Those secrets are usually passwords introduced by users. Violating the Kerberos protocol, both by brute force mechanisms or cryptanalysis, may be hard however, as already was discussed on section 2.1, trying to guess the password used for the generation of the first key is much more easier.

Context-aware multi-factor and multi-modal authentication is a possible solution for the vulnerability presented on the last paragraph. By requesting a dynamic combination of authentication modes, attackers find more difficult to guess the combination of validation proofs inserted by the user. CAM2 interaction model can be used on Kerberos initial process by adding one step to the protocol. Therefore, an AS working as a CAM2 compliant IdP returns an authentication policy required for a certain level of assurance and context conditions provided by the client. After gathering the authentication proofs from the user, the client application generates a secret key from the data that was collected. The CAM2 AS generates the same key and the first step of Kerberos protocol may take place.

3.6.5 Asynchronous authentication

In some situations single sign-on is not possible to achieve due to the the inability of client applications to contact the authentication service. Let's consider the example of a firm which employees work outside the headquarters. Throughout the day, employees use their computers to work, therefore they have to get authentication in order to create new work sessions. However their credentials are managed by a remote service located on the firm headquarters and sometimes network connections aren't available. In this case CAM2 model may be applied the following way: at the beginning of the day and while the employees have access to a network connection, an application installed on the computers retrieve one authentication policies for each possible expected context condition. For each policy, a secure digest over the respective validation proofs is also obtained. When employees want to get authenticated, the application evaluates the authentication context and requests the right proofs. The digest over those proofs is compared to the digest that were obtained earlier. If they are equal then the authentication succeeds.

4. Architecture for CAM2 authentication platform

The concretization of CAM2 authentication model is validated through the development of an authentication platform supplying context-aware multi-factor and multi-modal authentication services. This section presents a detailed architecture for the implementation of CAM2 compliant authentication platforms. It will start by presenting the details about mapping CAM2ML messages into a model of objects, and finally the architectural details for the implementation of a identity provider that fulfills the requirements of context-aware multi-factor and multi-modal authentication.

This architectural view includes considerations about the development of the authentication modules that were implemented during the elaboration phase of this work. They are only a few examples among the vast possibilities, however they were specially chosen to illustrate the generality of the architecture.

The section ends with the presentation of the architectural model for the implementation of some representative client integration modules, namely for web application, mobile devices, web services integration and finally an extended version of Kerberos.

4.1 CAM2ML Object model

CAM2ML specification presents the formalization of authentication protocols and respective messages for interaction between subjects, service providers and IdPs. therefore, each party must manipulate the concerns related to assertions and requests as entities that they can use to get data and generate new information.

CAM2ML object model represents CAM2ML elements, as well as their dependencies, mapped on a object hierarchy. Figure 4.1 shows an UML class diagram for CAM2ML object model.

4.1.1 CAM2MLExportable interface

XML, is the format used for the exchange of CAM2ML messages between clients, service providers and CAM2 IdPs. However, in order to manipulate them, the components using CAM2ML elements must have some translation mechanism from XML to objects and *vice-versa*. Considering those requirements, all CAM2ML objects inherit the *CAM2MLExportable*

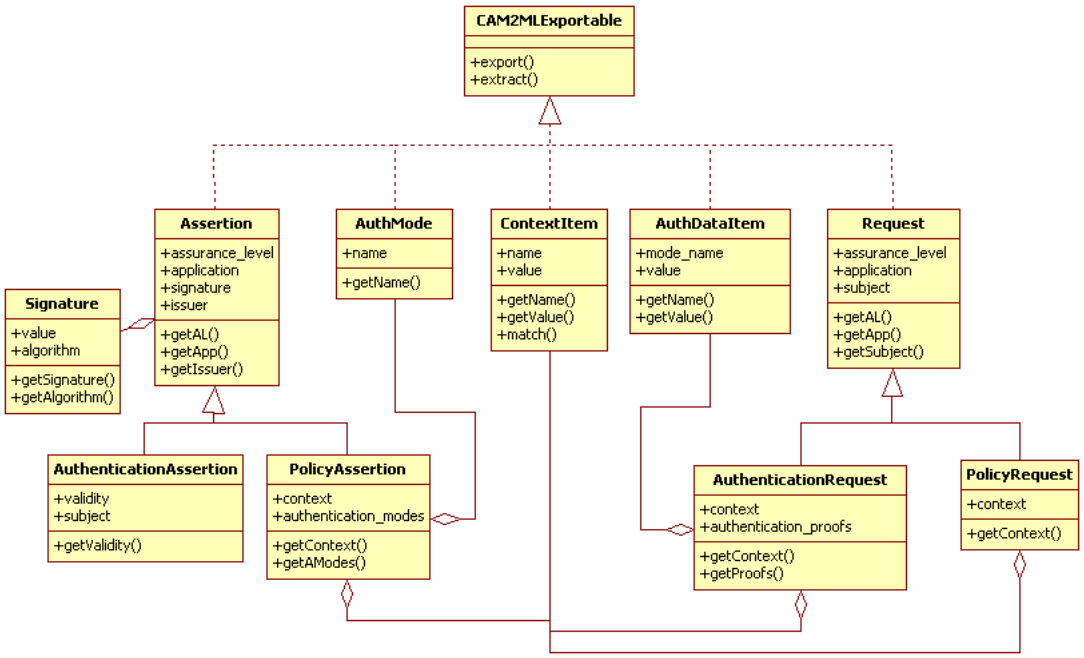

Figure 4.1 Object model for CAM2ML mapping

interface operations. This interface provides two functionalities: the export method, that converts objects into CAM2ML documents and the extract method, which imports the information contained by a document and generates an object.

4.1.2 Context Item

A context is a set of environment attributes extracted from CAM2ML assertions. Each attribute has a name and a value that can have multiple formats. Context attributes may vary during the life-cycle of an instance of CAM2 authentication platform. The abstract class *ContextItem* represents environment attributes, it has the common operations to all context attributes, namely getting its name, getting its value and verifying if the attribute matches another. Every context attributes must be represented by an object that extends the abstract class, implementing its operations differently and accordingly to their purpose.

The context manager is an object that is able to dynamically load a *ContextItem* and return it to other components. It provides an operation that, given the name of an attribute, loads the adequate class from the file system and initializes it. By combining both the dynamic load

mechanism and and the match operation, it is possible to verify if the context information, provided in a policy request, matches the context existent on a specified policy stored on CAM2 authentication platform.

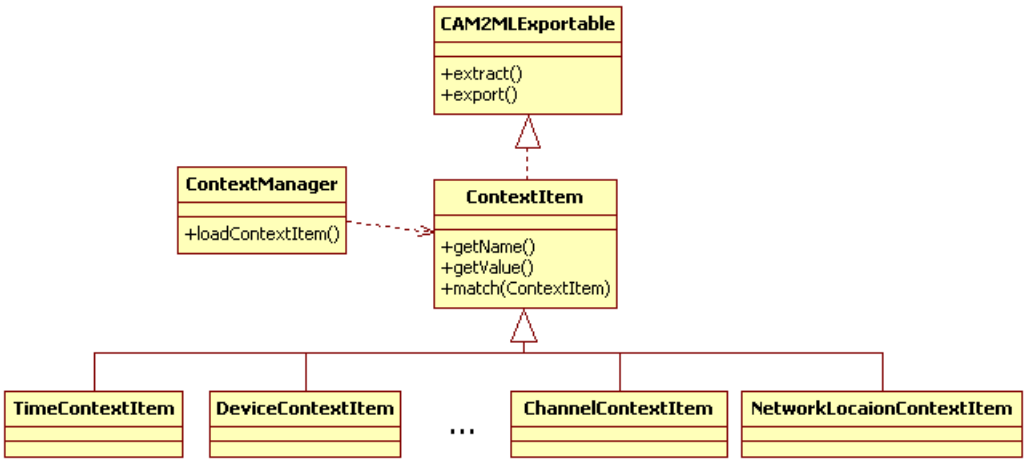

Figure 4.2 Context item and context manager class diagram

4.1.3 Authentication modes

Both CAM2ML policy assertions and authentication requests must deal with the authentication modes they contain. While policy assertions must contain the modes required for a particular authentication process, authentication requests must identify the proofs they carry. With the purpose of representing both data types, the current architecture provides two abstract classes : the *AuthDataItem* and the *AuthMode*. The *AuthMode* object identifies authentication modes and the *AuthDataItem* object represents authentication proofs identified by their mode names.

4.1.4 Assertion

All CAM2ML assertions have common fields, namely the application to which they are applied, the assurance level they grant and a signature of the statement identified by his issuer. The *Assertion* abstract class implements the CAM2Exportable interface, so all assertions may be converted into CAM2ML documents and documents into assertions. Applications and assurance levels are handled as Strings and signatures are composed by two attributes: one for the

algorithm and other for the signature bytes. This representation allows the utilization of multiple algorithms. The *Assertion* class is extended by *PolicyAssertion* and *AuthenticationAssertion* classes:

- *PolicyAssertion* - Is the class that represents CAM2ML policy assertions. Additionally to the fields inherited by the *Assertion class,* it keeps three more fields: a collection of *ContextItem* objects for description of the authentication context, a collection of AuthMode objects for definition of the required proofs and finally a Date field stating the max validity period for assertions generated from that policy. These objects are stored by CAM2 IdPs, which rely on them to choose the most appropriate authentication process. Policy management and storage will be further explained.

- *AuthenticationAssertion* - CAM2ML authentication assertions are represented by *AuthenticationAssertion* class. It only adds two fields to its super class, respectively for identification of the subject to whom the assertion is applied and for the validity period of the assertion. *AuthenticationAssertions* may be used as authentication credentials between subjects and service providers. The subject field uniquely identifies the authentication principals; the application and assurance level identifiers assure that the assertion is appropriate for a specified operation on a specified service provider. Finally, the validation field may be used to determine whether a credential is still valid, accordingly to the purposes of the authentication policy used to generate the assertion.

4.1.5 Request

Similarly to CAM2ML assertions, requests also have some common fields for identification of the subject, assurance level and application to which the request applies. Also implementing the *CAM2Exportable* interface, *Request* abstract class is used for the same purpose that *Assertion* class. It is extended by *PolicyRequest* and *AuthenticationRequest* classes.

- *PolicyRequest* - This class represents the request element described on CAM2ML specification. It extends the *Request* abstract class by adding a collection of *ContextItem* objects, which represent the authentication context.

- *AuthenticationRequest* - Additionally to *Request* abstract class*, Authentication Requests* contain both context items, for context description, and a collection of *AuthDataItems,* which carry authentication proofs gathered from the subject they represent.

4.2 CAM2 Identity platform

4.2.1 Client integration layer

This layer represents the interface for external applications and services wanting to use context-aware multi-factor and multi-modal authentication. CAM2ML front-end abstract component generates uniform information and supply it to the authentication logic layer.

Applications relying on CAM2ML front-end services are grouped in four categories: Web integration fabric, web services integration fabric, Kerberos integration fabric and Administration tools integration fabric. Each group has common functionalities that are aggregated by specific fabrics. Then, the development of new applications is easier, since it can be done by extension of the most appropriated fabrics. Figure 4.3 presents the UML component diagram for the devised architecture.

4.2.1.1 CAM2ML front end services

This component is used by all integration modules providing generic tools for the creation of CAM2ML objects and dynamic context evaluation. The evaluation of context at this point is limited, since the only way of obtaining context attributes is by observing the arrival conditions of the request. Stronger methods use application specific modules that acquire context data from the devices and send it along with the authentication requests. To enforce authenticity, context information can be signed by the client side application. CAM2ML front end services works also as the intermediate between clients and the authentication logic layer. During their interactions, the translating methods transform application requests in CAM2ML objects and objects in application specific responses. The communication between the client integration layer and the authentication logic layer is made through SOAP web services over a TLS/SSL layer. TLS/SSL provides authenticity, confidentiality, and integrity as security properties during the exchange of CAM2ML assertions and requests.

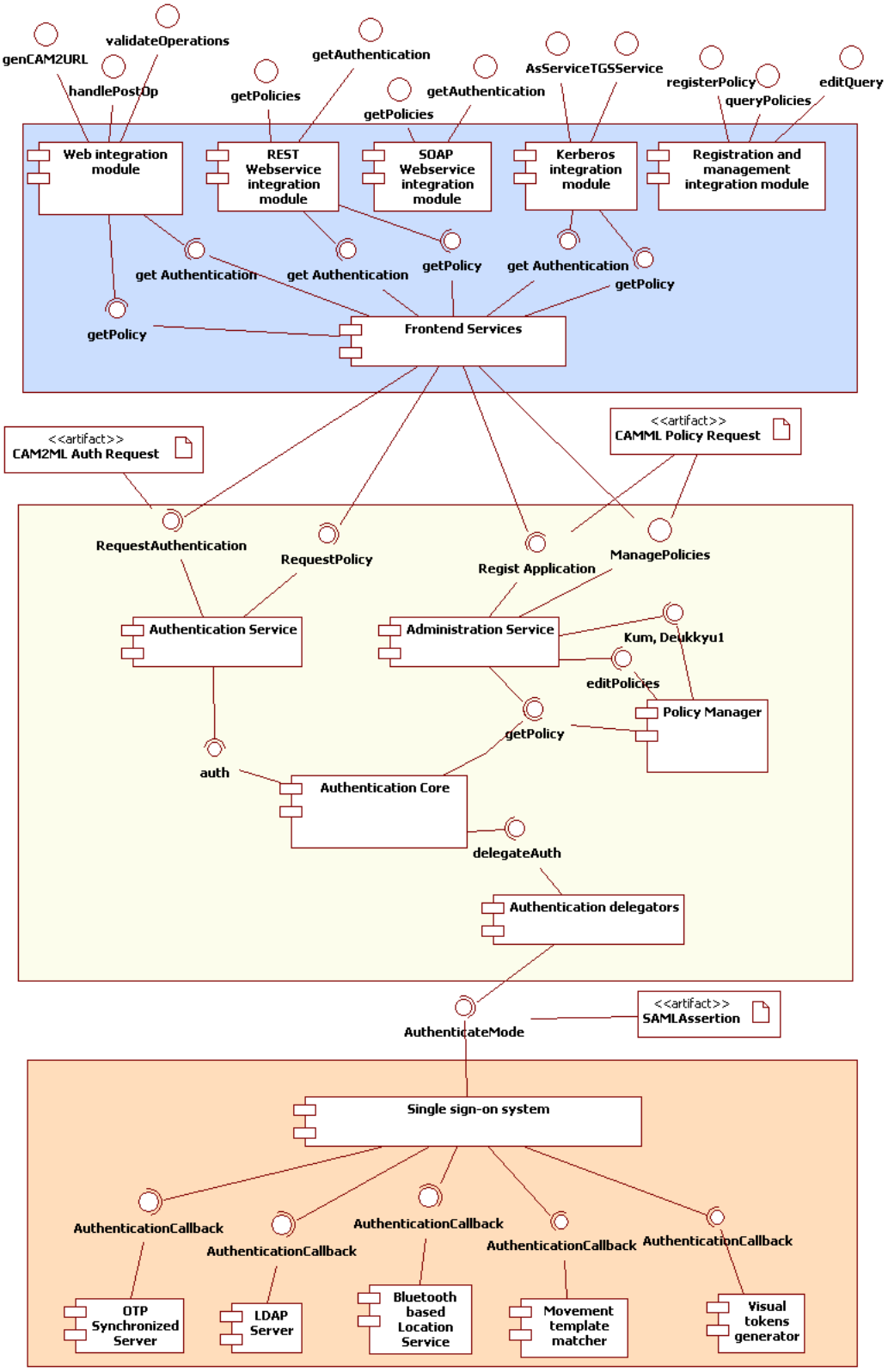

Figure 4.3 Component diagram for CAM2 IdP architecture

4.2.1.2 Web authentication integration module

The web integration module is a web application that externalizes context aware multi-factor and multi-modal single sign-on authentication. It was designed to support web based authentication. The interaction with other applications is made through browser redirection, while the interaction with users is made through the display of web pages. Considering a website that wants to validate the identity of its users, the authentication proceeds as follows:

1. The user accesses the website and chooses to be authenticated using a CAM2 authentication platform.
2. The website redirects the user to the web integration module web application which evaluates the authentication context and retrieves the right policy. Both the expected assurance level identifier and the URL of the website are encoded on the redirected URL.
3. After knowing what are the authentication proofs that must be gathered, the web integration module prompts the user for the required data through the presentation of a sequence of pages. Each page relates to one authentication mode.
4. Once all authentication proofs are collected and authentication succeeds, the browser is once more redirected through a HTTP POST request to the first website.
5. Along with the URL of the redirection, an authentication assertion is encoded and finally used as a credential. Finally, the website providing the resource to the user starts a new session and sets a cookie on the users side.

Context is evaluated by the web integration module by analyzing the data embedded on the request redirected by the user's browser. The current implementation extract the requests arrival time, the relative global location and the communication protocol (HTTP). The only non trivial context attribute is the relative global localization of the requests, which is done with the help of a public domain web service for global location using IP addresses.

Websites often has to maintain authenticated sessions after users identification. These sessions may be totally managed by the application, however, CAM2 architecture provides a *Session* object supplying operations like the management of the redirect process, the creation of

cookies and CAM2 credentials validation.

The management of the redirect process is made in two steps: firstly the *Session* object generates an URL for the CAM2 IdP containing the assurance level identifier encoded on it. Finally, when the service provider website receives a POST method, the *Session* object searches for any CAM2 assertions it may carry and if so a new cookie is created. Cookie generation is based on authentication assertions. Assertions are encoded and stored on the appropriate structure. Every time users perform a operation the *Session* object verifies if the assertion is still valid, if the assurance level is equal or greater than the required and if the application is the correct. Since assertions are signed, it is possible to assure that the information stated by the cookie is certified.

4.2.1.3 SOAP authentication integration module

Web based authentication is not suitable for all kinds of application. In some cases, there is the need of integrating authentication services on specific application modules with a thin grained set of operations and without any web interface. Due to the nature of this applications and requirements, the browser redirect method is not appropriated.

Web services are a nice solution to the integration of external services during the development of new application and extension of existing ones. At the limit, applications may only orchestrate a combination of web services to produce their results. The advantage in the utilization of a web services based solution, when compared with other architectures such as Java RMI , CORBA or COM , is that it uses HTTP as the message transport protocol. This allows web services to be used through NAT systems, Firewalls and other limitations imposed by the Internet.

A SOAP web service that exports the basic authentication services described by CAM2 framework is provided by the current architecture. Along with the RESTful authentication integration module it composes the web service integration sub-layer within the client integration layer. Two operations are provided: the policy retrieval and the authentication request. Developers wanting to implement authentication using this web service must firstly invoke the policy retrieval operation, passing a policy request CAM2ML object. The response will eventually contain a policy assertion and in that case the client application must gather the validation proofs and create a CAM2ML authentication request object. That object is passed to CAM2

authentication platform through the invocation of the authentication operation. If the authentication succeeds, the response contains an authentication assertion object which can be used as a credential.

4.2.1.4 RESTful authentication integration module

As already discussed throughout this document, it is a fact that mobile devices have limited resources. Battery life period, communication range and processing capacity are issues that must be taken in mind during the design of applications for pervasive devices. Some applications need simpler interaction models, for example, is not reasonable to compile SOAP based web service stubs in some limited devices. Even some web servers does not support server-side complex processing, thus not permitting the usage of SOAP web services or the redirect approach presented on the previous sections.

RESTfull web services are an alternative to SOAP, since they do not require the existence of stubs on the client side. Invocations are made through HTTP Post and GET requests and responses are returned through HTTP responses. Each operation supplied by a REST web service is accessed through a specific URL. Developers only have to create HTTP requests, encode the required parameters and handle the HTTP responses to obtain the results.

The current implementation provides an authentication integration module that relies on RESTfull web services to export CAM2 authentication services. It comprises a Java servlet for hosting two resources(or services), namely for policy retrieval and other for authentication request. Both resources rely on CAM2ML front-end services abstraction to perform the interaction with the authentication logic layer.

Similarly to SOAP authentication integration module, these services do not maintain any state information neither support for session management. Instead, RESTfull authentication integration module must be used as a basic tool for the development of authentication interfaces on resource limited scenarios.

4.2.1.5 Kerberos legacy integration modules

Section 2 introduced Version 5 of Kerberos as a vastly used authentication protocol. It is used by systems like Windows, Mac OS, FreeBSD and Solaris as default method for remote identification of principals. Still it has a critical point of failure, the usage of passwords as seeds for the creation of cryptographic keys used on the first iteration of the protocol. The usage of

public key certificates (PKINIT) mitigates this problem by excluding static passwords from the initiation process, however it forces the utilization of smart-cards and the respective readers, significantly increasing the total cost of ownership.

The proposed architecture introduces a new approximation to Kerberos initiation process. While presenting a different instance of CAM2 interaction model, it also shows how to mitigate attacks to the passwords relying on dynamic combinations of authentication modes instead. Kerberos integration sub-layer describes the architecture both for the authentication server (AS) and the ticket granting server (TGS). While the TGS was designed following the default protocol defined by version 5 of Kerberos specification, the AS had to suffer some changes which will be discussed on the remaining paragraphs of this subsection. In order to enable comparisons between the existent solutions and its CAM2 extended versions, three Kerberos authentication clients where created, namely one for the default specification of Kerberos V5, other for its PKINIT version and a third version compliant with CAM2 specification. Clients may choose what version of Kerberos they want to use by stating it in on the first message through the *option* field.

- *Kerberos V5 integration services* - This integration module follows the protocol defined by Kerberos specification without any changes. Client applications must generate a cryptographic symmetric key from a password which must be the same that the one generated by the authentication server. This authentication method is seen as a subset of CAM2 authentication process only using one authentication mode. Passwords are retrieved and validated through CAM2 authentication platform. This architecture does not introduce any changes to already existent *kerberized* applications, since the protocol maintains unchanged.

- *Kerberos V5 with PKINIT integration module* - Kerberos V5 PKINIT support is accomplished by changing the first interaction of the default protocol. The authentication server receives the first message from clients and extracts its public key certificate. After verifying its validity, it ciphers the TGS ticket with the extracted key. The client uses its private key to decipher the information needed to interact with the ticket granting server. The approach here described uses CAM2 authentication platform to validate the certificates sent by client applications. Once validated, their public keys are extracted and used to encrypt the first response.

4.2.1.6 CAM2ML Extended Kerberos V5

In order to show the flexibility of CAM2ML, an extended version of Kerberos protocol was designed. In addition to the steps needed for Kerberos authentication, to achieve context-aware multi-factor and multi-modal authentication, the client applications must obtain the policies required for a specific authentication context. For that, the following interaction is performed:

1. The client application adds information about its context to the first message of the protocol. The options field contains a flag that warns the authentication server to use CAM2 version of Kerberos.

2. The AS verifies the context contained on the first message and relies on CAM2 front-end services to perform two steps: getting the appropriate authentication policy and obtaining the credential that must be generated by the client application. This credential is a digest computed over the combination of authentication proofs that must be gathered by the client application.

3. The iteration follows with AS replying with the policy obtained.

4. The client application receives the authentication policy and gathers the required authentication proofs from the user.

5. Finally, both the authentication server and the client application also computes a digest over the collected data. Both the client and the AS generate the same secret key from the same computed digest.

The Kerberos protocol follows normally from here as defined on version 5 specification. Notice that the authentication proofs are never exchanged between the AS and the client. The same occurs with the default version of Kerberos, which is one of its strengths. However, the first interaction differs from the original version of Kerberos by relying on a combination of authentication proofs, instead of creating a cryptographic key from a static password. This behavior improves the assurance level, since it is harder for an attacker to guess all proofs provided by users. The extended version Kerberos is formally presented on the following protocol:

$$1.C \rightarrow AS \ : \ Options||IDc||Realm_c||ID_{tgs}||Times||Nonce1||Ctx$$
$$2.AS \rightarrow C \ : \ Realm_c||ID_c||Ticket_{tgs}||AP||\{LOA||IDtgs||TS_2||Ticket_{tgs}\}_{Kgc}$$
$$Ticket_{tgs} = \{Flags||K_{c,tgs}||Realm_c||ID_c||AD_c||Times\}_{Ktgs}$$
$$3.C \rightarrow TGS: \ Options||ID_v||Times||Nonce_2||Ticket_{tgs}||Authenticator_c$$
$$Authenticator_c = \{ID_c||AD_c||TS_1\}_{Kc,tgs}$$
$$4.TGS \rightarrow C: \ Realm_c||ID_c||Ticket_v||\{K_{c,v}||Nonce_2||Realm_v||ID_v\}$$
$$Ticket_v = \{Flags||K_{c,v}||Realm_c||ID_c||AD_c||Times\}_{Kc,tgs}$$
$$5.C \rightarrow V \ : \ Options||Ticket_v||Authenticator_c$$
$$Authenticator_c = \{ID_c||Realm_c||TS_2||Subkey||Seq\#\}K_{c,v}$$
$$6.V \rightarrow C \ : \ \{TS_2||Subkey||Seq\#\}_{Kc,v}$$

CAM2ML Extended Kerberos protocol

The extended authentication protocol is very close to the original. The only differences reside on the first two steps. In the first step the context *ctx* is passed and on the second an authentication policy *AP* is returned. Already existent kerberized applications can therefore be adapted to CAM2 with a minimum amount of changes to their structure.

CAM2 authentication platform provides an interface where Kerberized applications can request login. It supplies authentication using basic Kerberos V5 protocol authentication, Extended CAM2ML Kerberos and PKINIT Kerberos version. For basic Kerberos authentication, the CAM2 platform acts as a proxy for a password directory plugged to the base single sign-on system, using null contexts. For PKINIT version CAM2 platform interacts with a PKI plugged on the single sign-on base also using null contexts. Finally, for CAM2ML authentication the platform acts as described on Protocol 4.1.

4.2.1.7 Admin console integration services

Authentication policies must be managed by applications that use CAM2 authentication platform. On a first step, applications must register policies for each authentication context that is expected and each assurance level required by its operations. Once submitted, they will eventually need to be changed to refine the authentication process. Management and registration sub-layer provides a SOAP web service with these basic operations:

- *Get policies* - retrieval of all authentication policies registered by an user and grouped by his applications. The result for this operation is a list of records, each containing the policy ID, the assurance level identification, and the application to which it applies.
- *Get policy by ID* - retrieval of a single policy given its ID. The policy is returned in CAM2ML format.
- *Edit policy* - policy edition given its ID and changed data in CAM2ML format. The old policy is removed from the platform and the new one is inserted in its place.
- *Insert policy* - policy insertion given a policy in CAM2ML format.

These operations need to be authenticated, therefore, along with the operations parameters, a CAM2 credential must be passed to the administration web service. This credential is an authentication assertion CAM2ML object. Each operation of the administration web service must first validate the credential and then return the results.

The typical application developed over the Admin console integration module is a Web interface providing CAM2 registering and management systems. It uses CAM2 session object to manage the navigation session and relies on CAM2 Web authentication integration module to validate the identity of users wanting to view, edit or submit new policies.

4.2.2 Authentication logic layer

4.2.2.1 Policy manager

The policy manager is the component responsible for storage and retrieval of authentication policies. It has a interface that exports operations for querying the policy repository. Policies must be stored on persistent memory and for that purpose the policy manager is responsible for extracting information from some repository (for instance a database or even the file system) and provides it as CAM2ML objects to the other components. The Policy manager supports operations for querying policies by context attributes, querying policies g by application and assurance level and to perform edition and insertion of new policies.

4.2.2.2 Authentication core

The authentication core layer implements the logic of context-aware multi-factor and multi-modal authentication. It support two operations, Policy retrieval and authentication.

The policy retrieval operation receives a *PolicyRequest* as argument and extracts the information about the context. Accessing the policy manager component, it queries for any policy which application name, assurance level and context attributes match the information contained on the *PolicyRequest*. The operation returns as result a *PolicyAssertion* .

The authentication operation receives a *AuthenticationRequest* as argument and removes both the context information and the authentication proofs. As already described in the section discussing CAM2ML objects, authentication proofs are kept on *AuthDataItem* objects. For each one of those modes, the right *AuthenticationDelegator* must be loaded.

An *AuthenticationDelegator* is an object that implements the authentication logic for a specific authentication mode. Relying on JAAS, each delegator contacts the data integration layer and requests authentication for that mode. Much in the same manner that context items are dynamically loaded, *AuthenticationDelegators* also are managed by a *DelegatorManager*. This architecture enables the easy addition of new authentication modules without having to change the authentication logic. The authentication process passes through the following steps:

1. On a first step and similarly to what happens in the policy retrieval operation, the authentication core tries to get an authentication policy matching the request fields.
2. If at least one policy is returned by the policy manager, it verifies if all the required proofs are supplied on the request.
3. If all proofs are available each one is validated by the appropriated authentication delegator. Each delegator is executed on a different thread. Due to parallelization, the latency of the process is bound to the execution time of the slowest delegator.
4. If, and only if, all delegators successfully validate their particular authentication proofs the authentication succeeds and an authentication assertion is generated and returned.

The authentication component is used both by the authentication service and by the policy retrieval service. Figure 4.4 illustrates in detail the authentication process.

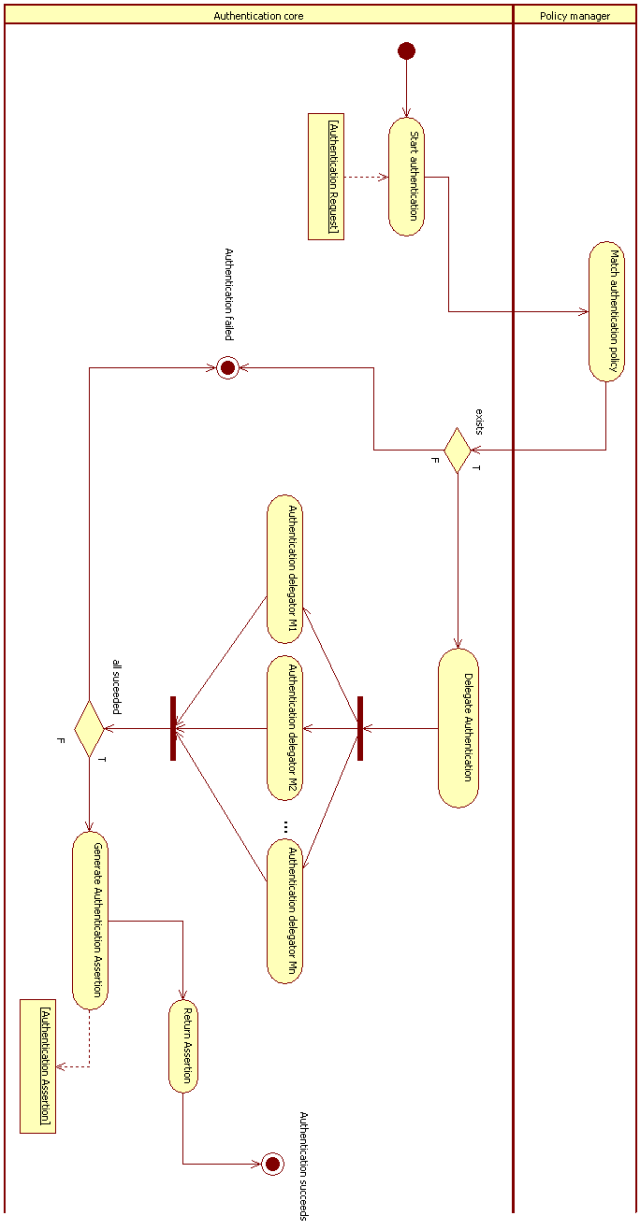

Figure 4.4 Activity diagram for the authentication process

4.2.2.3 Platform services

The platform services export the operations found in the authentication logic layer components. Three services are available, the authentication and policy retrieval services, which rely on the authentication core and the management service using the policy manager component. They provide SOAP based web services for each one of those operations. Web services are a good solution for easy integration of CAM2 authentication services, therefore platform services can be used directly by applications without passing by the client integration layer. For that, applications only have to know the steps of CAM2ML reference interaction model.

Policy requests contain sensible data that is susceptible to security attacks such as replaying[47] and eavesdropping, therefore the architecture relies on secure web services through a TLS layer between endpoints. This grants authenticity, confidentiality and integrity as security properties to CAM2ML interactions.

4.2.3 Data integration layer

Data integration layer is composed by a collection of single sign-on authentication systems, each providing multiple services for static-factor authentication and identity management. Those services are used by the authentication delegators discussed on the authentication logic layer section. Each delegator implements the specific logic to interact with the single sign-on system hosting the authentication modules. That interaction may rely on standard specifications such as SAML, benefiting from their advantages end enabling interoperability with other systems. Single sign-on systems, such as the ones discussed throughout the related work section, typically provide interfaces that allow the extension by adding off-the-shelf authentication modules. In a way or or another, these systems invoke operations over the authentication modules in order to validate factor-specific authentication proofs. A representative set of authentication modules was implemented during the elaboration phase of this dissertation. They cover all the classic authentication factors plus the new and emergent ubiquitous approaches. Their design will be presented on the next subsections.

4.2.3.1 onetime passwords synchronization server

onetime passwords (OTP) fits in the authentication factor category related to *something the user has.* The reference implementation developed during this dissertation provides a synchronization service for OTP synchronization. It connects to single sign-on systems as a custom authentication module and works as a standalone component. OTPs are represented as character sequences which can be more complex than the usual passwords, since users do not have to memorize any information.

It is assumed that users hold a device capable of reproducing time-synchronized passwords. OTPs are provided to users through a LCD display built-in on the device. New passwords are generated from a initial seed registered both on the synchronization server and on users devices. The generation process is made by calculating the hash-values over the concatenation of the initial seed with the current time. This method forces both the server clock and the user device clock to be and must be internally synchronized. The generation of the OTP by the device is a completely disconnected process.

4.2.3.2 LDAP server

Lightweight Directory Access Protocol[20] based systems are widely used as a solution for identity management and authentication proof storage. The LDAP server component, presented on Figure 4.3, is used to query for subjects information and verify if a specified password matches the attributes that are stored. due to the issues already discussed on the second chapter, this authentication method is the most common among the available single sign-on authentication systems. The single sign-on system, used as base authentication platform for the reference implementation, already has integrated this authentication module. Further explanations will be supplied on the next chapter.

4.2.3.3 Kerberos data interface

During the first step of the Kerberos authentication protocol, the authentication service needs to cipher data that only the client can decipher. This key is usually accomplished by using a key generated from the password bytes (Kerberos V5) or the clients public key gathered from a X509 certificate(Kerberos V5 w/PKINIT). In order to perform CAM2 authentication, it is required that the authentication attributes (passwords, biometric templates, OTP seeds) can be

retrieved to be used as key material. Typically, the common single sign-on authentication systems doesn't provide any interface for doing that. For that purpose, the Kerberos data interface provides methods for authentication data extraction relying on the base repositories such as LDAP server, OTP synchronization server or the Bluetooth token validation server.

4.2.3.4 Bluetooth token validation server

Bluetooth support is provided by most of the cell phones available today. This architecture describes the design of a Bluetooth based security token. Users must only have a cell phone supporting Bluetooth and CAM2 Bluetooth token application installed on it (discussed on the next chapter). The validation server has a Bluetooth interface that interacts with mobile devices sending a challenge. If the device replies correctly to the challenge, the event is registered and kept in persistent memory. During the enrollment process, users register the MAC address of their mobiles and a initial seed associated uniquely to their devices. The interaction between client applications and CAM2 authentication platform runs as follows:

1. User tries to access a service which initiates his authentication using CAM2 authentication platform.
2. One of the requested authentication modes is a Bluetooth token. The authentication platform displays a message asking the user to stay near to a Bluetooth reader certified by the validation server and use the CAM2 Bluetooth token application.
3. The validation server sends a challenge to the user device through Bluetooth. The device receives the request, concatenates it to the initial seed and computes a hash-value. The result is returned also through Bluetooth.
4. The validation server receives the reply and registers the event on persistent memory. Each event contains information that links users to the place where the Bluetooth reader is located and the time when the operation was performed.
5. During the authentication delegation phase, the authentication logic layer queries the validation server for any successful event occurred during a time period.

To some extent Bluetooth tokens may be used as location validates, since the communication range is limited. Therefore it is possible to limit the confidence in the authentication process to the area covered by the Bluetooth reader owned by the validation server. The utilization of these tokens enhance the overall assurance level by adding one extra channel to the interaction process. An attack both to the Internet connection and the Bluetooth connection is less likely.

4.2.3.5 Movement template matcher

cell phones supporting accelerometers can supply some basic information about their rotation over coordinate axis. That information can be aggregated and used to find specific movements during a specified time period. Once identified, movements can be compared to a template database, in order to verify if those movements match the information stored. Basing on the principles devised by Rene Mayrhofer on his article proposing the use of environment information for establishment of authenticated connections[26], it is possible to use the physical position variations (movements) as seed for authentication proofs.

Let's consider *rX*, *rY* and *rZ* as rotation quotients over respectively the *X*, *Y* and *Z* axis. The value of a rotation quotient for a specific axis at a specific state represents the angle between the previous state of the device and the current state. Templates store three rotation sequences, namely for *rX*, *rY* and *rZ*. Each sequence maintains *n* consecutive rotations over an axis consecutive gathered during a time period. The following matrix aggregates the three sequence in a matrix *T*.

$$T = \begin{bmatrix} r_{X1} & r_{Y1} & r_{Z1} \\ r_{X2} & r_{Y2} & r_{Z2} \\ r_{X3} & r_{Y3} & r_{Z3} \\ . & . & . \\ r_{Xn} & r_{Yn} & r_{Zn} \end{bmatrix}$$

Let's now consider the data retrieved from the mobile phone. During a time period the cell phone is able to supply consecutive values for the rotation quotient. Three rotation quotients sequences may also be aggregated:

$$S = \begin{bmatrix} r_{X1} & r_{Y1} & r_{Z1} \\ r_{X2} & r_{Y2} & r_{Z2} \\ r_{X3} & r_{Y3} & r_{Z3} \\ . & . & . \\ r_{Xn} & r_{Yn} & r_{Zn} \end{bmatrix}$$

The movement sequence captured by the mobile phone is considered valid if and only if the following happens:

$$\forall a,t \left| T_{ta} - S_{ta} \right| < k, \qquad a \in \{x,y,z\}, \in \varepsilon[1,n]$$

with k representing a threshold that is used to calibrate the acceptance rate.

This authentication module relies on information that the user must know, however it doesn't forces the users to memorize complex data, they only have to remember a few movements. On the other hand, making movements is more appropriate to some situations where the user is not able to write a password. The movement template matcher relies on the methods described above to implement the operation of template matching given a sequence of rotation quotients. It maintains templates kept on persistent memory and allows the configuration of k to allow the calibration of the acceptance rate.

4.2.3.6 Visual proof repository

Instead of combing time information with built-in seed values (like onetime passwords), it is possible to combine it with information provided by the authentication platform, which is richer. Bluetooth tokens already do it, however they forces the utilization of a second communication channel. It would be desirable that the process was completely disconnected, therefore mitigating attacks to the channel. Presenting that information to the user and forcing him to insert it on a disconnected device is not convenient. The use of data matrices can be applied to the generation of authentication data as discussed on [27]. Every time visual proofs are required as an authentication mode, data matrix is presented to user, who has to use its cell phone camera to capture the image. The data matrix is processed and the encoded value is used in combination with the seed initially stored on the device. An hash-value over that combination is calculated and the user is prompted to introduce some positions of that value.

This component stores data matrices encoding random data used to the generation of visual-proof based tokens. It has an interface that allows the authentication delegators to validate the data inserted by the user. For that, this component simulates the same behavior that the client and compares the data introduced with the information it has generated.

5 . Implementation

This chapter is dedicated to the implementation details of CAM2 authentication platform. The architecture presented on the last chapter was taken as reference for the construction of such platform. This section starts by discussing the technologies used during the implementation, followed by the description of each component construction and their *runtime* behavior. Finally the chapter ends with the presentation of some applications developed for validation purposes. The implementation blueprint is illustrated by figure 5.1.

5.1 Chosen technologies

5.1.1 J2EE Platform

J2EE oriented development is suited for the rapid integration of CAM2 components on the structure of typical data centers. The three tier based architecture allows direct division of CAM2 authentication platform components among different physical machines, the client integration layer is associated to front-end machines with great I/O capabilities, the authentication logic layer may be assured by high performance machines and finally the Data integration layer components can be installed on machines focused on database management and data retrieval. J2EE facilities, when used at the client integration tier, enable easy deployment of new authentication integrators, without time-consuming adaptations to the existent structure. The same is applicable to the integration of new authentication modules and identity repositories.

Java was used as the basic implementation language for the majority of the components available on CAM2 authentication platform. It was chosen due to many reasons:

- *Ease of development* - The Java language is simple, giving great support to developers in what concerns to memory allocation and garbage collection. It is much focused to object oriented paradigm, being an excellent solution for implementation of the component based architecture shown on the last chapter.

- *Extensibility* - Java provides strong extension mechanisms through the usage of interfaces, and abstract objects, which improves the definition of new context items, authentication modules and delegators. The extension of new components is as simple as adding items to a java properties file and deploying the required classes. Relying on Java dynamic class

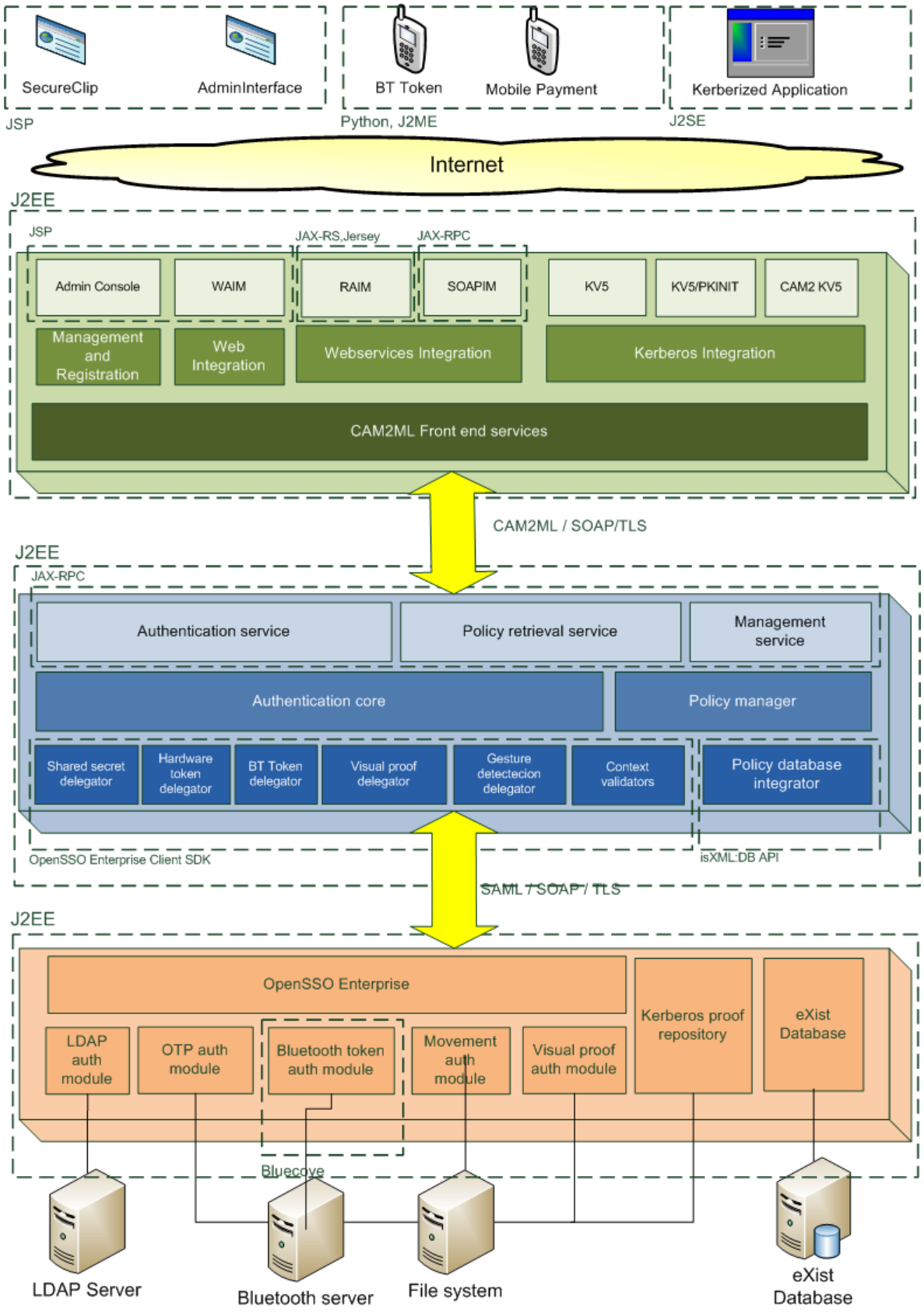

Figure 5.1 Authentication platform implementation blueprint

loading mechanisms, the components are ready to use.

- *Distributed nature* - Java enhances the development of communication between components through the network, providing native resources such as Java RMI and Web services using Java-RPC
- *XML Handling* - Java provides native support for XML handling through it XPath API and DOM representations.
- *Platform independence* - Relying on its virtual machine, Java is platform independent, improving the generality of the application of CAM2 authentication concepts.
- *Know how* - Java is the language where we feel the most convenient. Instead of using other object-oriented languages such as, using Java has enabled the development of CAM2 authentication platform only focusing on its components, without having to spend time learning how to implement them.

5.1.2 J2ME

J2ME is Java technology applied to limited devices. It has a limited version of J2SE virtual machine, supporting a subset of its operations. The current implementation relies on J2ME for the development of mobile applications used as proof of concept. It was chosen due to its strong integration ability with Java applications and ease of development.

5.1.3 Python and Symbian C++

Despite being easy to use and develop, J2ME is not a good solution when it is necessary to manage low level resources. The access to features like cameras, accelerometers and native functionalities is very limited. Due to this limitation, Python was used to gather information from the sensors of the mobile phones.

Python [28]is a dynamic interpreted object-oriented language focused on code readability and rapid development. It has a powerful standard library and provides easy mechanisms for the integration of new libraries and frameworks. Python is available on some mobile phones such as Nokia phones using Symbian operating systems.

Symbian [46] is an operating system designed for mobile systems supporting enhanced multimedia features such as video, audio, sensors and Bluetooth. Since 2008 it is owned by

Nokia, which integrates it in the majority of its top line mobile phones.

Symbian C++ primitives and Python language are used for data acquisition from the sensors of a Nokia 5800 XpressMusic[1] mobile phone applications. In the current implementation, small python components were developed to supply sensor information to the J2ME applications developed as validators.

5.1.4 JavaServer Pages

JSP is a server-side technology, mostly used in J2EE, for the construction of dynamic content web pages. It supports the integration of java code, which improves the communication with CAM2 authentication platform. The web authentication integration module discussed on the last chapter is a JSP web page connected to the authentication platform through the client front-end services provided by the client integration layer.

5.1.5 OpenSSO

OpenSSO was widely described on chapter 2. It is a single sign-on platform developed by SUN with strong extensibility mechanisms and suited for easy application integration.

The authentication delegators, used by the authentication logic layer on CAM2 authentication platform, rely on OpenSSO's client SDK for the validation of mode specific authentication proofs. OpenSSO supports LDAP for default authentication mode, therefore the remaining authentication modes had to be implemented from scratch and plugged through the Authentication Service SPI.

OpenSSO Enterprise 8.0 was chosen due to multiple factors, from such we can distinguish the great availability of documentation, the integrability with Java code, an active community that is constantly improving the capacities of this state of-art system and finally the easy of development and extension of new authentication modules.

5.1.6 GlassFish

Glassfish [30] is an open source reference implementation of a full integrated J2EE platform. It provides support for multiple technologies such as JSP, web services support and application server. CAM2 components are deployed on glassfish as EAR and WAR files. Glassfish was

[1] http://www.nokiausa.com/find-products/phones/nokia-5800-xpressmusic

chosen due to its ability to deal with all required technologies at the same time and to its easy integration with OpenSSO, the single sign-on authentication system used as base for CAM2 authentication platform.

5.1.7 eXist

eXist [29] is an open source database management system based on XML data model. It stores XML documents and allows efficient querying relying on a dedicated XQuery index system. This solution beneficiates CAM2 authentication platform, since the CAM2ML policies stored on the Platform and manipulated by system administrators are XML documents. Therefore, policies can be efficiently retrieved as XML documents and be converted to CAM2ML Objects. eXist provides an API, known as *isXML:db* for connection to the database and integration in Java applications. CAM2 Policy manager relies on this API to provide Policy retrieval services.

5.1.8 Web services and security

As described throughout the chapter about the CAM2 platform architecture, web services were used both due to their integration in SOA based applications and to the security properties that secure web services can provide. Security for web services is achieved through simple configuration on Glassfish. A key pair and its respective X.509 certificate was generated using Java Keytool. The keystore is used by Glassfish to implement server-side authentication for REST web services, SOAP web services and HTTPS web access. Web services were generated and deployed on Glassfish with AXIS [37] toolkit from Apache foundation. CAM2ML assertions are signed using a combination of DSA and SHA1 as algorithm . The key, with 1024 bytes of size, is associated to a X.509 certificate and respective public key generated using Java Keytool.

5.2 Runtime

At runtime, the components of CAM2 authentication platform are distributed among the physical infrastructure as illustrated on figure 5.2.

The front end components, namely the web authentication interface, the web services, the Kerberos servers and the web admin interface, are represented by independent applications

deployed on Glassfish application server.

The web services providing authentication and administration operations also are deployed on the application server, however both the policy manager and the authentication core component are deployed as stand-alone Java RMI servers.

Finally the instances of OpenSSO Enterprise 8.0 and eXist database are deployed as independent GlassFish applications, while the Kerberos data repository and the Bluetooth token server runs as stand-alone Java RMI servers. Bluetooth token server also depends on Bluecove [51], a Java library that simulates Bluetooth JSR-82 implementation, for the communication with cell phones.

In order to provide transparent and easy extensibility, the platform relies on Java properties files to perform dynamic configuration. The following properties are allowed to be configured:

- *Server names and ports* - The server names and listening ports for all Servers.
- *Context items* - The file names of the classes implementing the context items.
- *Authentication delegators* - The file names of the classes implementing the authentication delegators.

This mechanism, in combination with dynamic policy administration, makes it possible to change the behavior of the authentication platform without having to stop any component.

5.3 Protocol Integration modules

For demonstration, CAM2 authentication platform supplies three types of authentication protocol integration modules, namely for Web pages, web services integration and Kerberos based client applications. The following paragraphs present some details considering the implementation of such modules.

5.3.1 Web integration authentication module

The web integration module provides an interface for the extended version of WEB POST/REDIRECT protocol as the one used by SAML. It was implemented as a JSP page supporting the

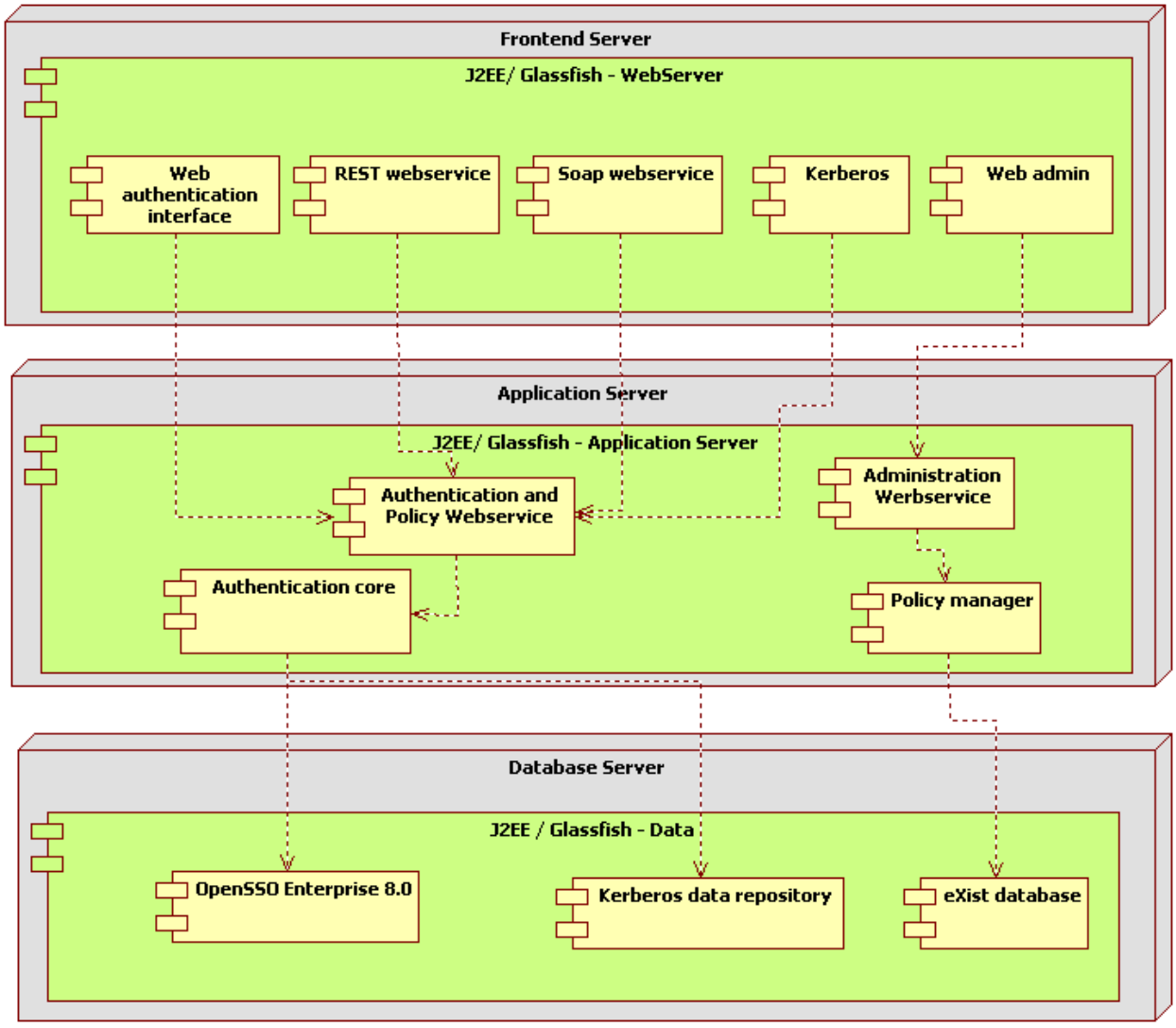

Figure 5.2 Deployment diagram

redirect based mechanism presented on the earlier chapters. It supports context evaluation, authentication proof collection and management of the authentication process. Context evaluation is achieved by collecting the time when the request arrived to the web server; the device type, which is embedded on the HTTP request sent by the client; and the clients IP addresses, which are used to give an initial hint about the relative location of the user. Authentication proofs are gathered through a sequence of HTML pages, each collecting data relative to one authentication factor.Finally this module handles the authentication process by performing the CAM2ML protocol with the authentication logic layer.The following listing shows the JSP code required to integrate CAM2 authentication on a web page.

```
<%@page import="CAM2.UniformFrontend.WebAuthentication.Session"%>

Session s = Session.getSession(session);
        if(s==null)
s=Session.createSession(session,
    "https://di164.di.fct.unl.pt/CAM2SecureClip/index.jsp",
    "https://di164.di.fct.unl.pt/WEBAuthentication/index.jsp");

Cookie c =s.validateSession(request,"level1");
        if(c!=null)
                response.addCookie(c);
        else
                s.getAuthentication(request,"level1")
%>
```

The *Session* object, discussed on the last chapter, manages the authenticated session. For each new access to the website, a new *Session* object is created with information about the CAM2 IDP and the current application identifier. After assuring that a *Session* already exists, the web page verifies if the request caries any valid cookie or if the request contains any CAM2 credential (redirected from Web authentication integration website). If none of these cases succeed, the authentication process is started.

5.3.2 Web services integration authentication modules

SOAP and REST were used as architectures for the providence of web services as CAM2 authentication integration mechanisms. They provide the operations for authentication and policy retrieval.

The SOAP based service was developed using the API implementing JAX-RPC specification included on Java 1.5.

The RESTful service relies on Jersey [31], which is the reference implementation for JAX-RS. JAX-RS is a specification describing the integration and development of RESTfull web services in Java. Jersey's API is not available on Java 1.6 however it is open source and is available at jersey's web page [2].

5.3.3 Kerberos integration authentication module

Kerberos integration module provides interfaces both for the authentication server and for the ticket granting server. Communication is implemented directly over TCP Sockets.

Despite the fact of Kerberos V5 supporting multiple cipher suites, the current implementation only uses chain based cipher Triple DES with PKCS7 padding for data encryption.

5.4 Authentication modules

Due to its extensibility capabilities, CAM2 framework can integrate all types of authentication modules for multi-factor and multi-modal authentication. The current implementation of CAM2 platform considers some authentication modules for demonstration purposes. They represent the classic and ubiquitous authentication factors. The following table shows the relation between the implemented modules and the existent authentication factors:

[2] https://jersey.dev.java.net/

	Something the user knows	*Something the user has*	*Something the user is or does*	*Something the user sees*
Passwords	X			
Hardware Security token		X		
Bluetooth OTP	X	X		
Visual based				X
Gesture Identification			X	

- *Static passwords* -The validation of static passwords is granted indirectly by the default LDAP module provided by the single sign-on base system (The current implementation uses OpenSSO). For this authentication module any additional development was needed on the single sign-on side, therefore showing that the usage of CAM2 middleware with the current authentication platforms is easy and reuses their functionalities in a transparent manner.

- *Hardware security tokens* - For the simulation of hardware security tokens, the current implementation relies on MobileOTP [39], an open source project providing a two-factor authentication solution for java capable mobile devices like phones or PDAs. The MobileOTP application requires a PIN code and generates a time synchronized onetime password, which is displayed on the device. Then, the generated code can be introduced on a HTML form, or reused by other mobile application. This type of security tokens are completely disconnected from any network or application.

- *Bluetooth onetime passwords* - Additionally to hardware tokens, the current implementation also provides an application connected OTP generator. It is also a mobile application that we have specially developed for this purpose. The communication with applications is made through Bluetooth as described on 4.2.3.4. Other modules, such as the visual based authentication module and the gesture identification module, may reuse it to send information to CAM2 platform.

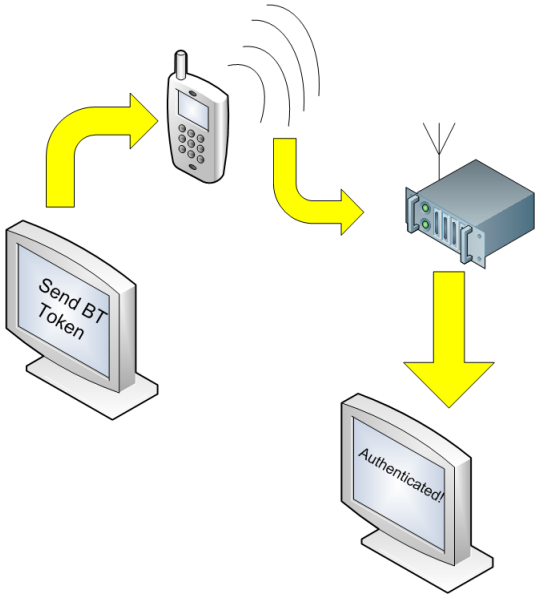

Figure 5.3 Bluetooth OTP authentication

Figure 5.3 illustrates the authentication process using the Bluetooth OTP mobile application. The user starts by being asked for sending a Bluetooth onetime password, then he uses the mobile application installed on his device and the generated code is sent through Bluetooth to the authentication server, which finally validates it.

- *Visual based authentication* - Data matrices were used to implement a visual based authentication module. We have developed a mobile application that generates a onetime password which can either be sent through trough the Internet (UMTS, GSM or WI-FI) or using the Bluetooth onetime password module. For acquisition and processing of the data matrices, the mobile application relies on QuickMark[48], which is a multi-device mobile bar code reader that provides a simple API for application reutilization. As can be seen at the Figure 5.4, users start by taking a picture of a Data Matrix. Relying on QuickMark, the mobile application processes the image and generates a onetime password, which finally is inserted using the authentication server interface. Notice that this OTP can also be sent using Bluetooth OTP.

- *Gesture identification* - Authentication based on *some gesture the user makes* is assured by the Gesture Identification module. It is installed as a mobile application and is reused by

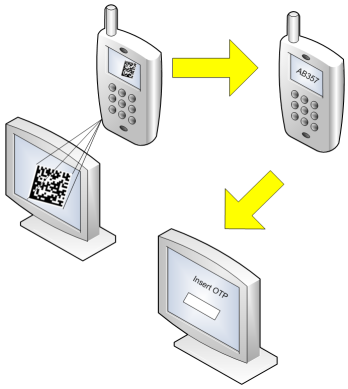

Figure 5.4 Visual based authentication

other applications wanting to use CAM2 authentication. The application has two blocks: a Python component and a J2ME component. The first uses the Symbian OS primitives to retrieve information about the state of the sensors available on the device. For this implementation, we have only used the accelerometer sensor, which provides information about the positioning of the device. The second provides the authentication service to other applications. When gestural authentication is requested by some application, the Gesture Identification module collects the values provided by the accelerometer and sends it to CAM2 Authentication Platform, which validates the authentication proof and returns the decision to the device. Figure 5.5 shows this interaction.

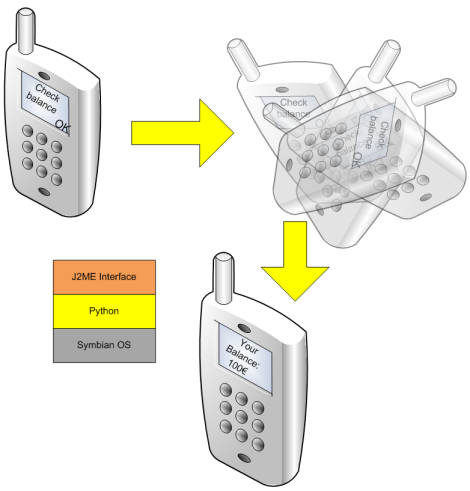

Figure 5.5 Gesture Identification

5.5 Validators

5.5.1 CAM2 Secure CLIP

As proof of concept, a web page supporting CAM2 authentication was implemented. It is inspired on CLIP, a web application for management of courses and students used by Universidade Nova de Lisboa[3]. This application is interesting, since it is accessed by all students, teachers and employees which access the system from multiple contexts. On the other hand, Clip supports multiple operations which require different assurance levels.

However, CLIP only uses a static password for authentication purposes. The implementation of the version of clip website relies on CAM2 web authentication integration module to add context-aware multi-factor and multi-modal authentication. Figure 6.1 shows an example of the authentication process using CAM2 authentication platform.

1. The user accesses *CAM2 Secure CLIP* website and chooses CAM2 system for authentication by clicking the link for that purpose.

2. the page is redirected to the web authentication integration website which evaluates the context attributes, the required assurance level and the application identifier to choose the appropriate policy. For that example and required assurance level (subject accessing the website using a laptop), only a password is requested.

3. The users browser is redirected once more to CAM2 Secure Clip web page, which concede access to the user.

4. The user sees the index page and try to access the information relative to student *27169*. This operation requires other assurance level, then the authentication process starts again. However this time a password is not enough, therefore a Bluetooth token is requested.

5. Finally the user gets access to the information about the student.

[3]https://clip.unl.pt

The conception of such application demonstrates that it is possible to easily adapt current working websites to the usage of context-aware multi-factor and multi-modal authentication. This was accomplished without changing the structure of the web page. This adaptation is simply made by assigning assurance level identifiers to the operations and registering authentication policies on CAM2 authentication platform.

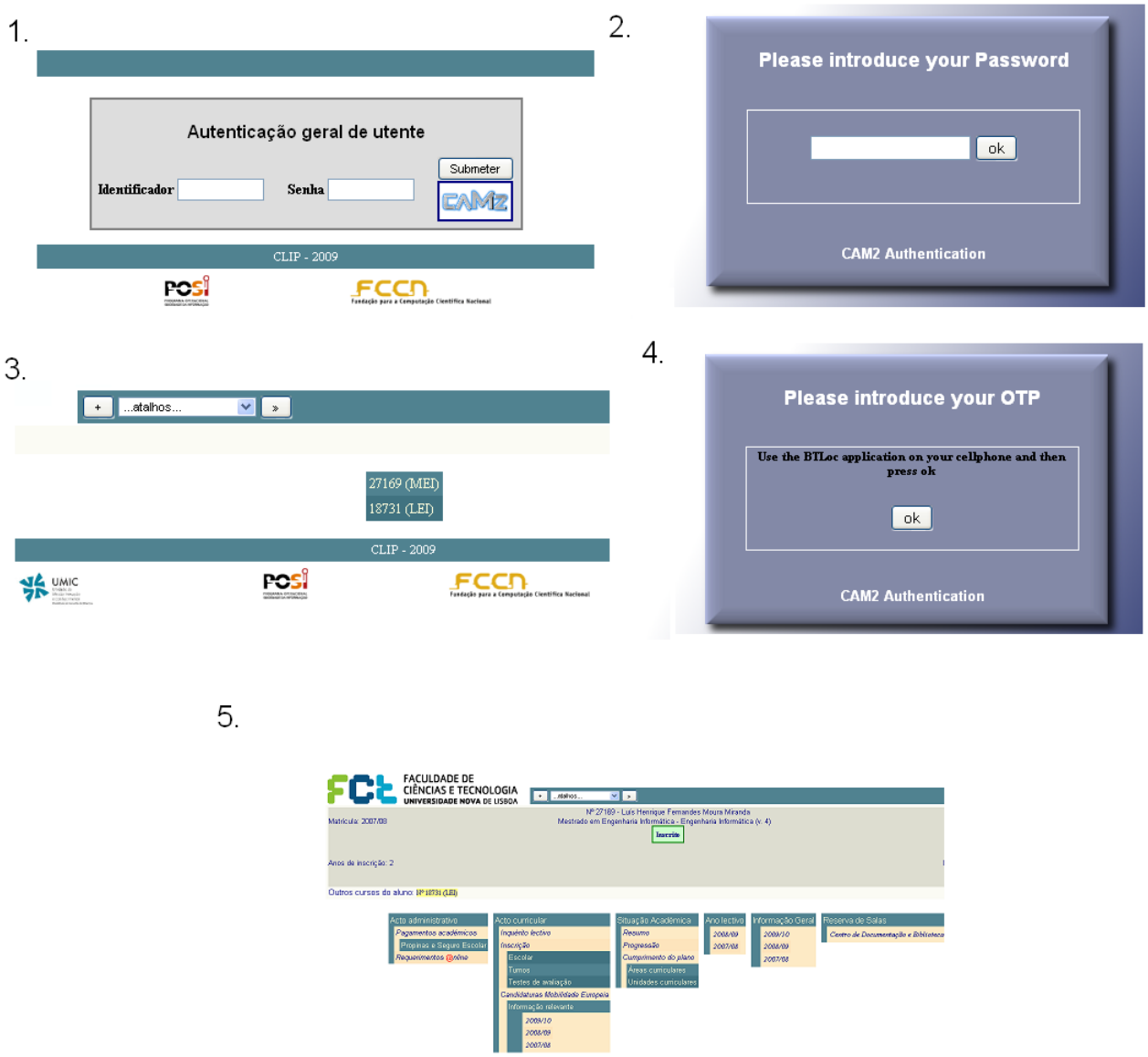

Figure 5.6 Example of an authentication event on CAM2 version of CLIP system

5.5.2 CAM2 Mobile payment application

Relying on REST authentication integration module, mobile payment application is a J2ME application including context-aware multi-factor multi-modal authentication.

The purpose of the application is very simple: provide an interface for electronic service payment given an entity, a service reference and a money amount. Transactions must be submitted through the Bank service interface, available as a REST web services secured by SSL/TLS. Before the submission of the transaction, the application must get authentication credentials from a CAM2 authentication platform. For the cell phone using during the implementation (Nokia 5800 Xpress Music), the application supports passwords, one-time-passwords, Bluetooth tokens, movement identification, and visual tokens as authentication modules.

The same application may be accessed through a web interface that also uses CAM2 authentication. See Figure 6.2.

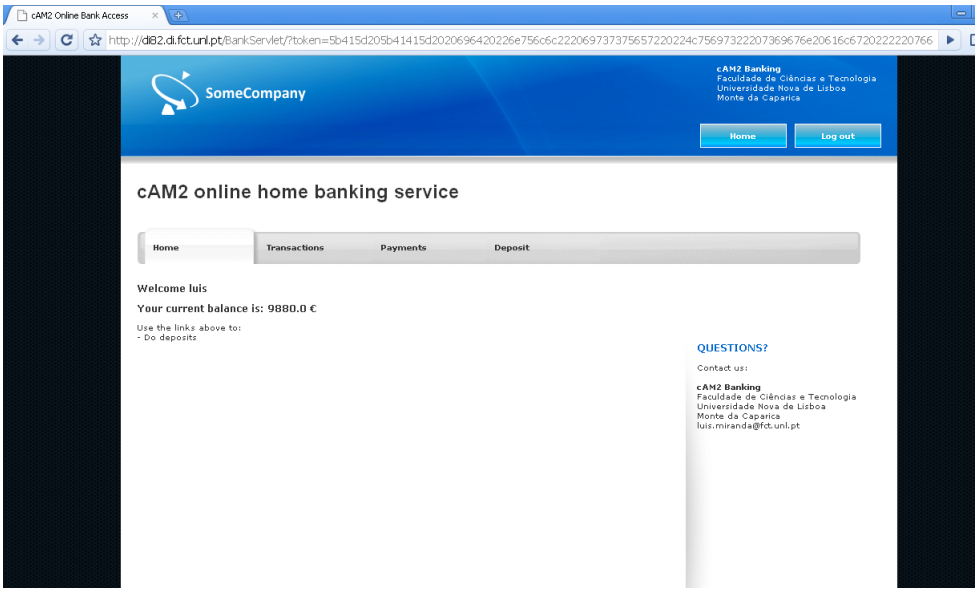

Figure 5.7 Bank application WEB interface

5.5.3 CAM2 Kerberos

A simple Swing based client application was developed to test the three variants of Kerberos protocol provided by CAM2 authentication platform. After successfully getting valid credentials, the client connects to a Time server, which returns the current time. Figure 6.3 illustrates a snapshot of the referred application.

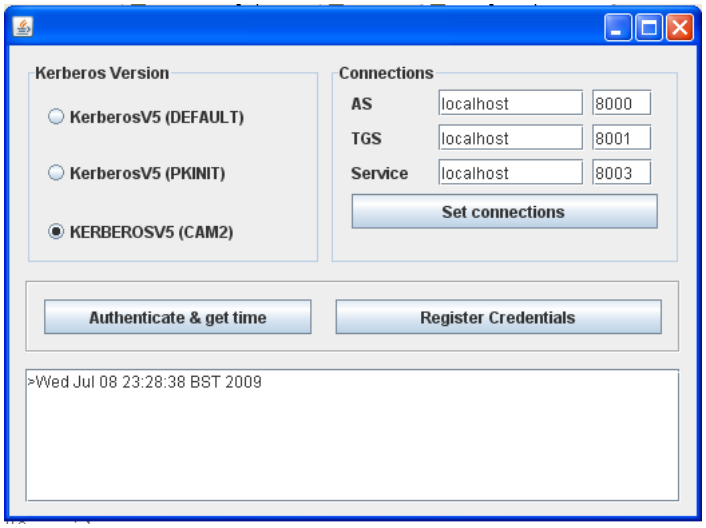

Figure 5.8 Kerberos client GUI

5.5.4 CAM2 Administration interface

The Web interface uses the administration web service to obtain results and present them to the user through a web page. It uses CAM2 session object to manage the navigation session and relies on CAM2 Web authentication integration module to validate the identity of users wanting to view, edit or submit new policies. Figure 6.4 shows the list of policies managed by user *admin*.

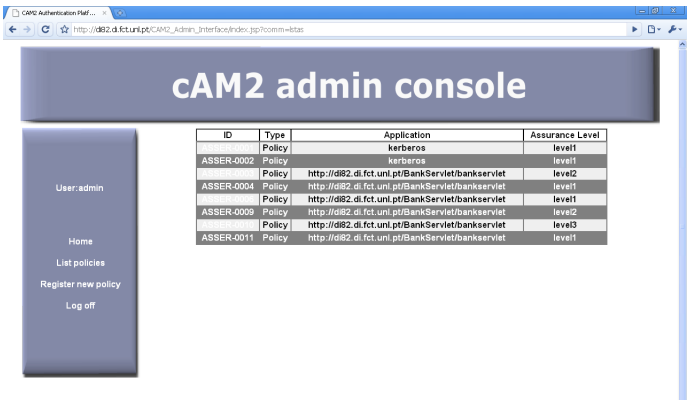

Figure 5.9 CAM2 Web Administration console

6. Performance evaluations

CAM2 authentication platform acts like a middleware over common single sign-on authentication systems, extending their functionalities to support flexible context-aware multi-factor and multi-modal authentication. The additional logic introduces some overhead in terms of exchanged messages and processing time. This section analyzes the performance of CAM2 authentication processes comparing it with OpenSSO Enterprise 8.0. The aim of this analysis is to evaluate the impact of the middleware in multiple scenarios. In order to be possible to compare results between both systems, the authentication process simulated during the tests only considers the utilization of a password as authentication proof. OpenSSO was chosen as comparison base since it is a stable and production ready implementation of a SSO system.

This section starts by presenting the comparison between benchmarks tests performed directly to an OpenSSO instance and over the same system extended by CAM2 platform. Finally a discussion is presented, providing considerations about the relations between the results obtained under two distinct environments, in a local area network and in a LAN

6.1 Workbench

Large organizations must handle multiple applications used by many people, demanding heavy request processing capacity from the infrastructure. On the following benchmarks we have tried to simulate those environments by installing CAM2 platform on a dedicated server, which capacity is comparable to the machines available on typical data centers. Table 7.1 and Table 7.2 show both software and hardware specifications.

Host Hardware	
Model Info	HP ProLiant Rack Server
CPU	Dual Core Intel Xeon @ 2.5 GHz 8 MB Cache
RAM Memory	16 GB
Network	4 Ethernet interfaces @ 1Gbps / 100 Mbps
Virtual Environment	
CPU	Dual Core Intel Xeon @ 2.5 GHz 8 MB Cache
RAM Memory	2 GB

Table 6.1 Hardware specification

Host Software	
Operating System	VMware v. 4.0 ESXI v.4.
Guest Software	
Operating System	Linux version 2.6.26-2-amd64 Debian 2.6.26-19lenny1 (Debian 4.1.2-25)
Application Server	Glassfish - Sun Java System Application Server 9.1_02 (build b04-fcs)
Database	eXist-1.2.6-rev9165
SSO base	OpenSSO Enterprise 8.0 Build 6

Table 6.2 Software specification

As we can deduce from Table 6.1, the machine used for performance tests is a rack server used for virtualization. The virtual machine hypervisor acts directly over the hardware performing bare metal virtualization. CAM2 was installed on a virtual server, while the single sign-on base is assured by an instance of OpenSSO and policy storage is handled by eXist database.

6.2 Benchmarks

The goal of these benchmarks is to evaluate the performance impact of CAM2ML protocol and CAM2 platform over current authentication systems and standards. For that, we have exposed the OpenSSO instance to load and stress tests, both directly and using CAM2 platform. The tests consist on performing multiple authentication requests and measuring the latency until response is obtained. The direct test over OpenSSO considers the sending of a user/password combination. For the test using CAM2 platform, we have defined a CAM2ML authentication policy that requests a password given a specified context and required assurance level.

In order to perform latency measurements, we have used JMeter, a tool developed under the Apache Jakarta project to collect and analyze the impact of load and stress over WEB applications. The following charts were obtained with JMeter from a client computer connected to the glassfish application server considering two scenarios:

- Local tests - where the client computer was on the same network that the server.
- Internet tests - where the client computer relied on the Internet to connect to the server.

For both types of tests we have measured latency for 1, 50 and 100 concurrent clients. For later evaluation of the platform behavior, a comparison factor was established between the average latencies measured from the two systems. Let k be the ratio between CAM2 platform and OpenSSO latencies:

$$k = \frac{L_{OpenSSO}}{L_{CAM2}}$$

Where $L_{OpenSSO}$ is the average process latency obtained with OpenSSO and L_{CAM2} the average process latency measured with CAM2 authentication platform. When K takes values near to 1 it means that the two systems have similar performance behaviors.

6.3 Local tests

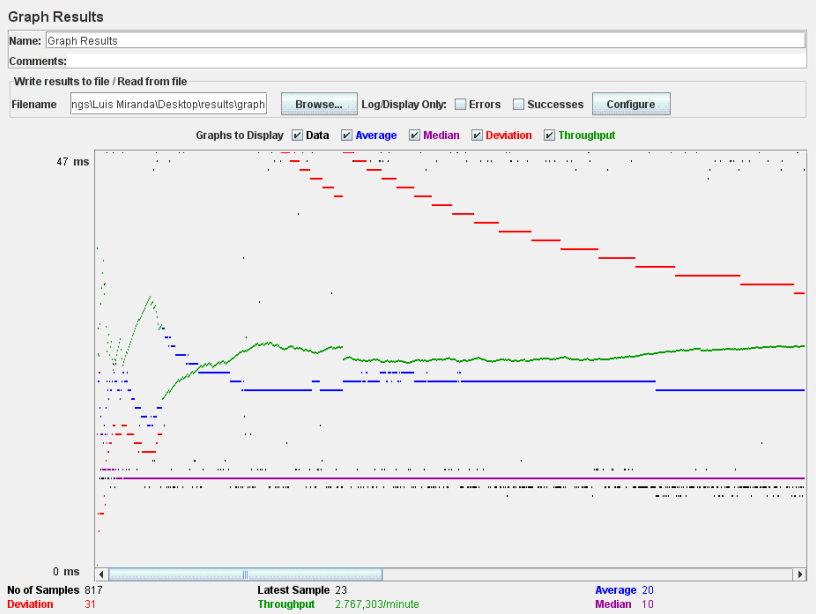

Figure 6.1 OpenSSO load test with 1 client

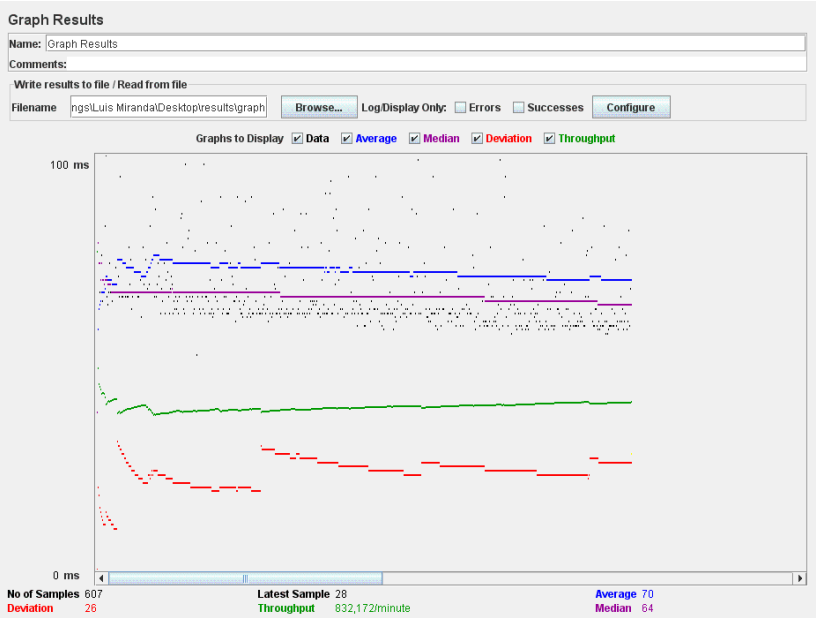

Figure 6.2 CAM2 load test with 1 client

Figure 7.1 and Figure 7.2 shows the results obtained for the latency between CAM2 and OpenSSO platforms using one client making multiple sequential requests. Here, we can conclude that authentication using CAM2 platform is 3.5 times slower than when OpenSSO is used as an unique platform. Nevertheless, the average latency using CAM2 is only 70 milliseconds.

Benchmark comparison factor: $k = 0,29$

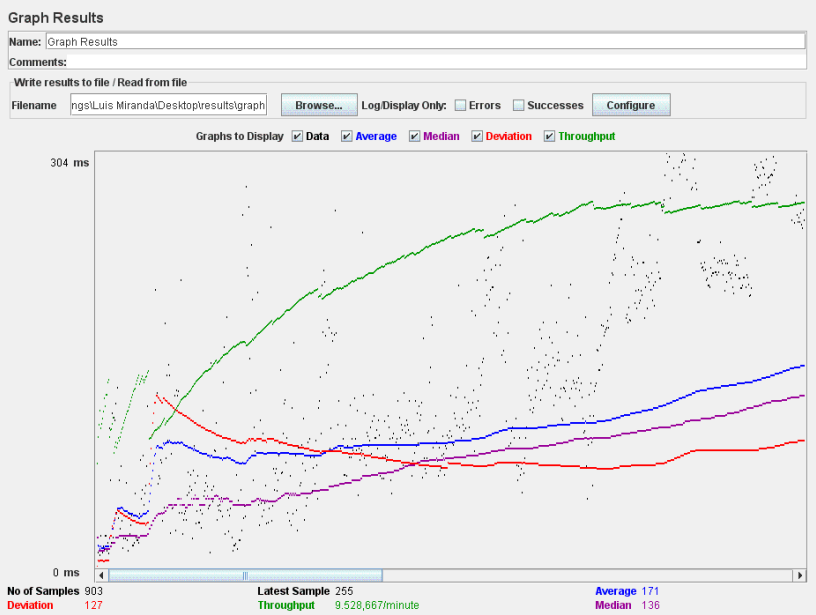

Figure 6.3 OpenSSO load test with 50 concurrent clients

The second test is illustrated at Figure 7.3 and Figure 7.4. It introduces concurrency by using 50 clients at the same time. The latency increases both for CAM2 platform and OpenSSO. Despite the values for CAM2 platform being close to 500 milliseconds, the value of k remains nearly the same (5% greater)

Benchmark comparison factor: $k = 0,34$

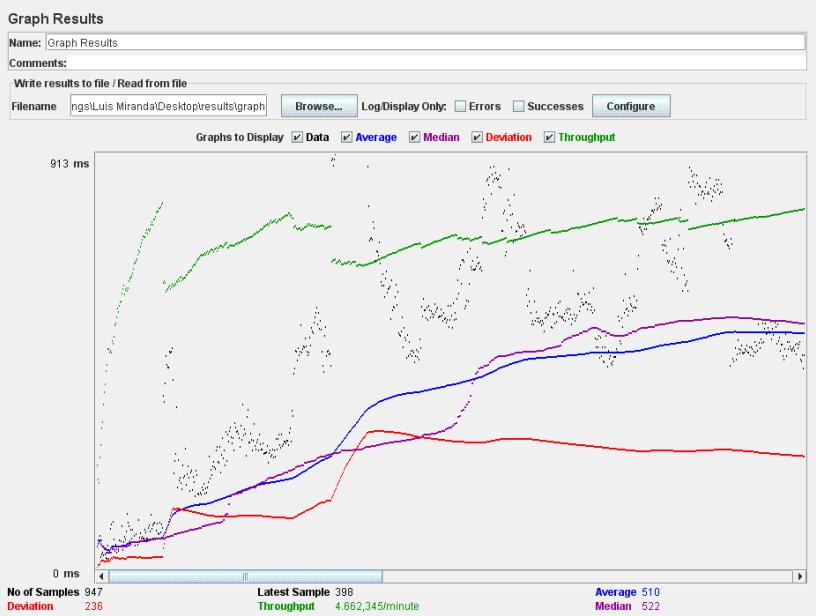

Figure 6.4 CAM2 load test with 50 concurrent clients

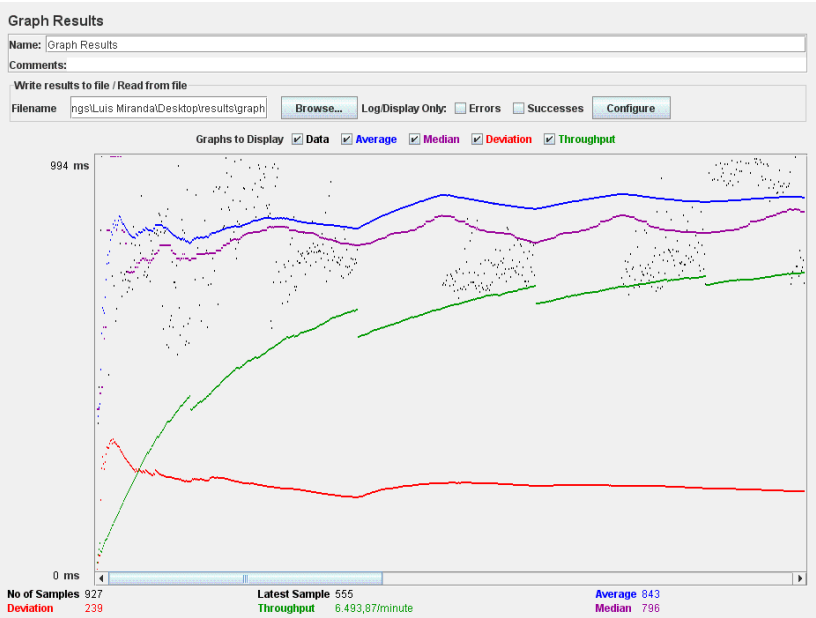

Figure 6.6 CAM2 load test with 100 concurrent clients

Figure 7.5 and Figure 7.6 shows that for 100 concurrent clients, the server is exposed to a great amount of load and both platforms have a similar behavior.

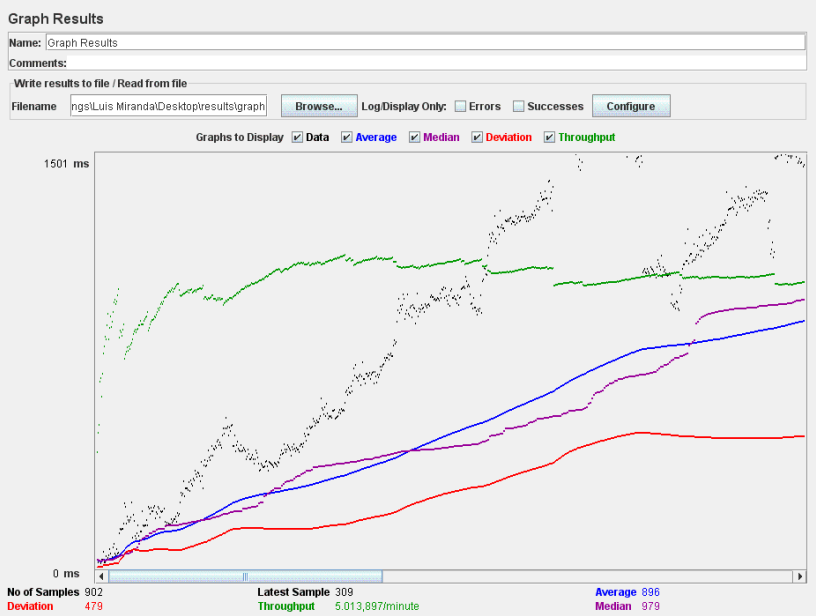

Figure 6.5 OpenSSO load test with 100 concurrent clients

Benchmark comparison factor: $k = 0,94$

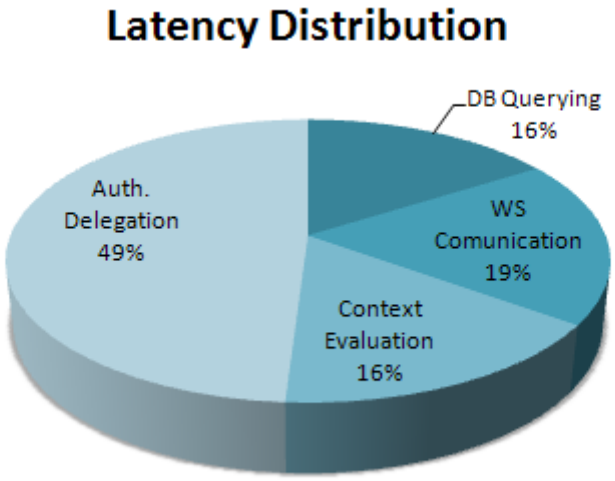

Figure 6.7 CAM2 latency distribution

Finally on Figure 7.7 the charge distribution among CAM2 authentication platform components is illustrated . Approximately 50% of the time consumed by the authentication process is due to authentication delegation. For this request, CAM2 platform only evaluates one password as authentication proof, relying on OpenSSO. Recall that the delegation process is parallelized, therefore, if we consider more than one authentication module, the total time consumed by delegation is bounded to the slowest authentication module. Querying the policy database is the second task consuming more time (16%). Both the context evaluation and the signature of authentication assertions consume 8% of the time while the time used for communications between the client and the authentication platform is below 6% of the total consumed time.

6.4 Internet tests

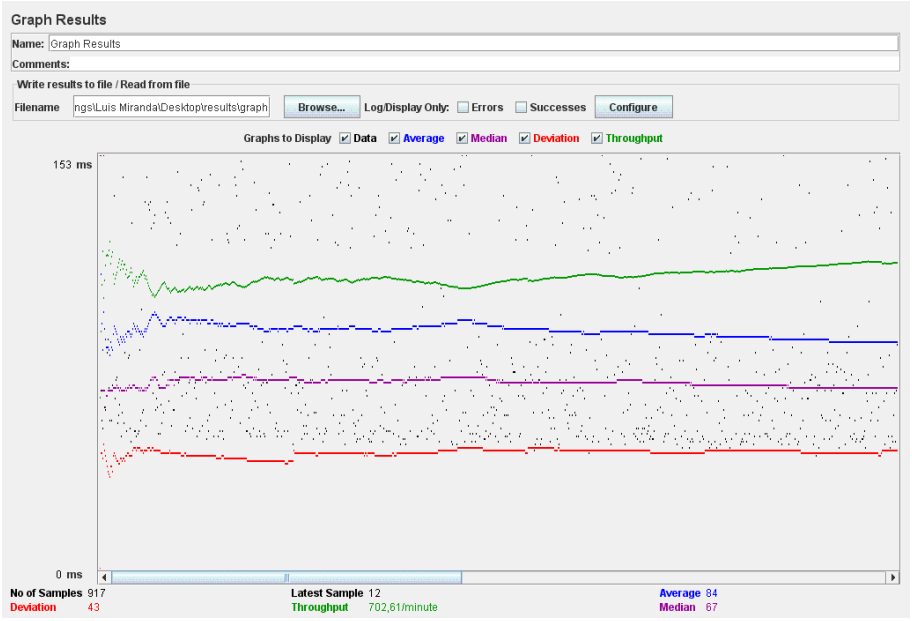

Figure 6.8 OpenSSO load test with 1 client

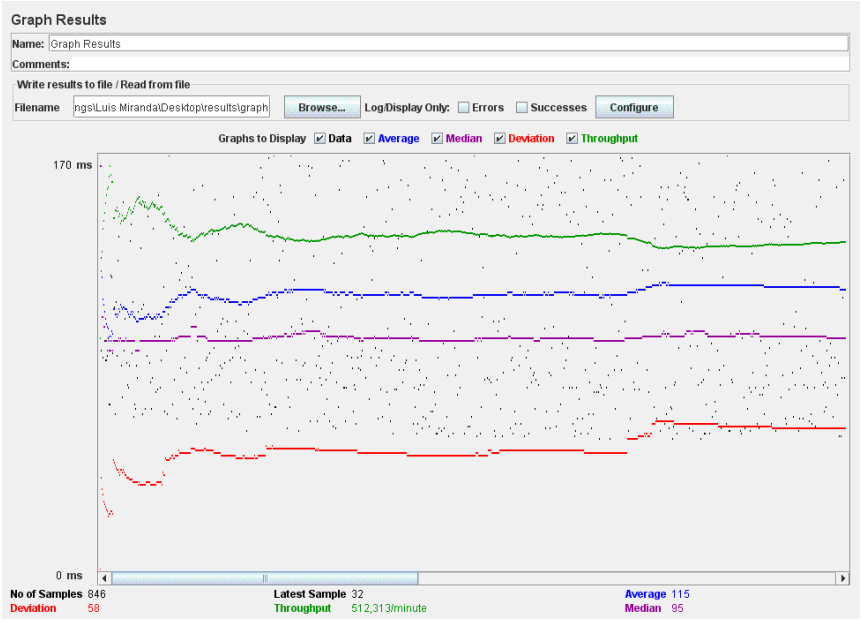

Figure 6.9 CAM2 load test with 1 client

The results illustrated at Figure 7.8 and Figure 7.9 show the impact of the introduction of communication latency, after connecting clients and servers through the Internet. Here we can see that the latency measured between requests and answers for CAM2 platform is only 1.3 times slower than with OpenSSO.

Benchmark comparison factor: $k = 0,73$

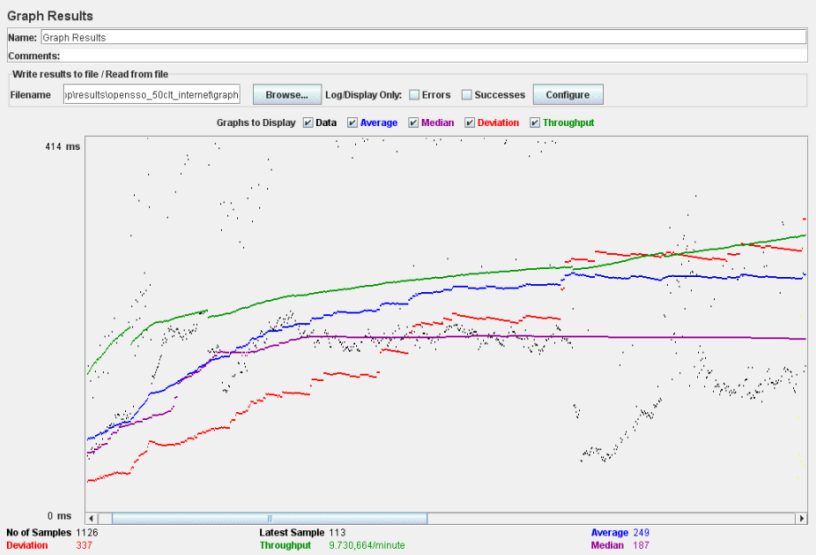

Figure 6.10 OpenSSO load test with 50 concurrent clients

On Figures 7.10 and 7.11 50 concurrent clients communicate through Internet. The latency increases both for CAM2 platform and OpenSSO. When compared with the same benchmark configuration on a local network, the measures obtained on this test are significantly more approximated than the first.

Benchmark comparison factor: $k = 0,49$

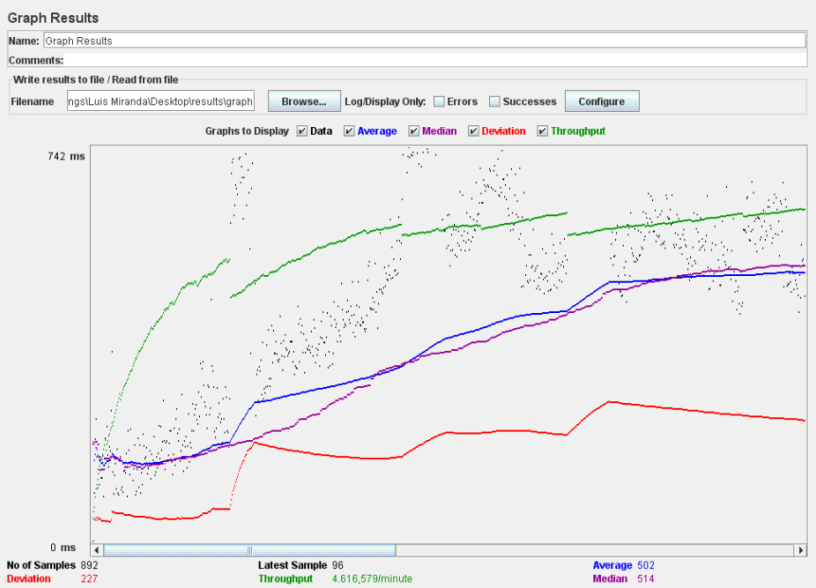

Figure 6.11 CAM2 load test with 50 concurrent clients

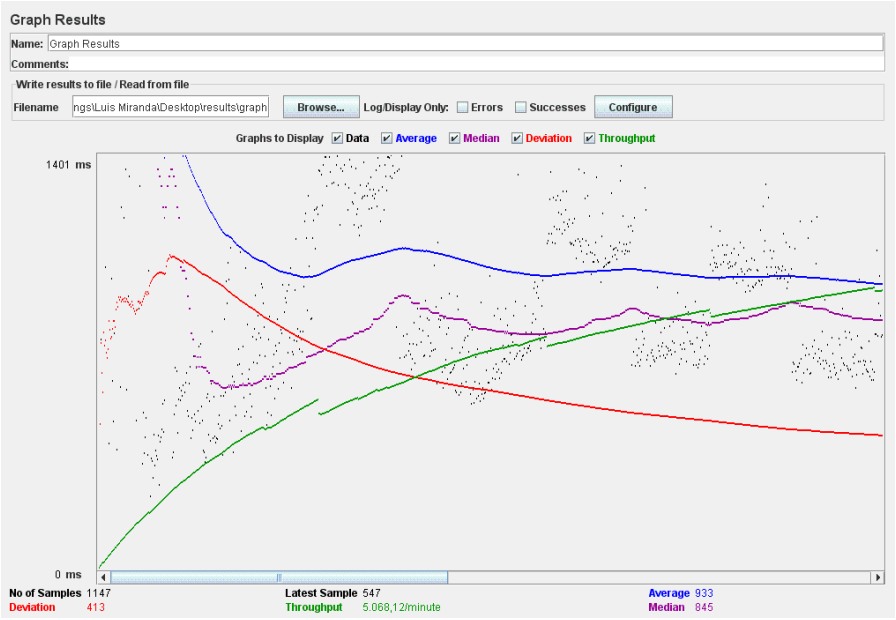

Figure 6.12 OpenSSO load test with 100 concurrent clients

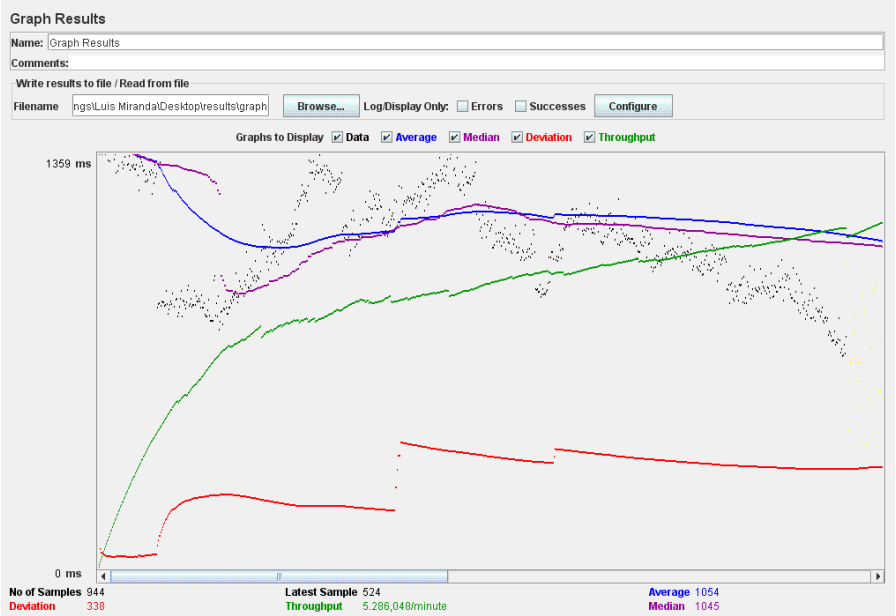

Figure 6.13 CAM2 load test with 100 concurrent clients

On Figure 7.12 and 7.13 with 100 concurrent clients, the measures are identical to the first test on these conditions on a local network.

Benchmark comparison factor: $k = 0,84$

Finally on Figure 7.14 the impact of communication latency is well noticed on the distribution chart, where 57% of the time consumed by all the process is dedicated to the communication between the client ant the application through web services (policy request , policy response, authentication request, authentication response). Authentication delegation, that was the most consuming task on the first benchmark, only consumes 28% of the overall time for these tests. Finally the overhead imposed by the CAM2 authentication platform is reduced to 15%.

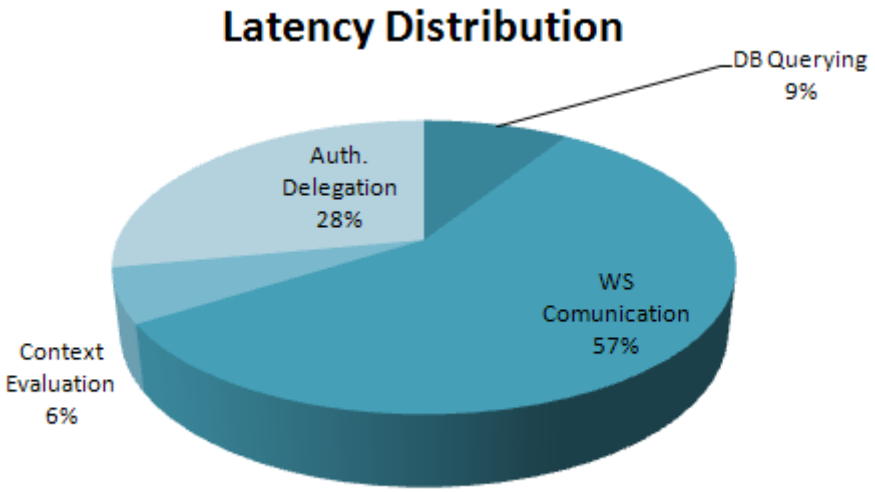

Figure 6.14 CAM2 latency distribution

6.5 Results overview

The results obtained on each benchmark are summarized on Table 7.3. As we can see, the values of k are closer to 1 for the measures taken using the Internet. When load is increased, the latency introduced by CAM2 platform processing rises slowly, making k even *closer* to 1. When submitted to stress conditions, both systems have the same behavior with k taking values near to 1. Considering the usage over the Internet CAM2 Platform increases 38% of latency, while in stress conditions with multiple clients only introducing 6%.

	1 client	50 clients	100 clients
Local Network			
OpenSSO (ms)	20	171	843
CAM2 (ms)	70	510	896
k	0,29	0,34	0,94
Internet			
OpenSSO (ms)	84	249	933
CAM2 (ms)	115	502	1054
k	0,73	0,49	0,88

Table 6.3 Summary of results

Therefore we can conclude that typical conditions found on the Internet, such as high latencies and multi-access by multiple clients, reduce the performance gap between the authentication systems as we know today and intelligent middleware such as CAM2 platform. These conditions are also typical to the emergent architectures that are being implemented today, based on distributed components placed *on the cloud* as in SOA based architectures and cloud computing.

7 . Requirement validations

Throughout the document, the following requirements were proposed: multi-factor and multi-modal authentication, expression and performing of dynamic context-aware proof requirement, extensibility for new authentication models, generic usage and performance. In order to address those requirements, CAM2 framework was designed, comprising new context-aware dynamic interaction models. To validate the model described by the framework, a CAM2 compliant authentication system was implemented and used to collect experimental results. The next paragraphs will discuss the objectives that were achieved by this dissertation.

Expression of dynamic context aware authentication processes

The definition of a context aware multi-factor and multi-modal authentication markup language address this issue. CAM2ML is a markup language for the definition of authentication processes. It support SAML abstractions by providing two types of statements, policy and authentication assertions. Policy assertions are used to state information about the authentication methods required for a context given a required assurance level. Authentication assertions contain statements about authentication events handled by a CAM2 compliant authentication platform. CAM2ML protocol, when applied to the basic interaction model provided by CAM2 specification, assures uniform context aware authentication between a circle of trust with service providers, clients and CAM2 identity providers.

Multi-factor and multi-modal authentication

Multiple authentication modules were developed in order to prove the functionalities of CAM2 platform, namely for the validation of LDAP passwords, time-synchronized onetime passwords, Bluetooth tokens, users gestures and visual proofs. Passwords and one-time-passwords represent the classic authentication factors(1st and 2nd factors). The validation of Bluetooth tokens, visual proofs and gesture pattern recognition represent the new and emergent ubiquitous factors. To some extent, the Bluetooth tokens can be used to implement localization aware authentication, since base stations have a limited range. The recognition of gestures and the utilization of visual proofs symbolize alternatives, on one hand they are convenient to users and on the other they allow to extract data with more entropy. As explained on the early chapters

the combination of multiple factors contributes to the increasing of the assurance level of authentication processes.

Dynamic and context-aware proof requirement

CAM2 authentication platform supports dynamic evaluation of authentication contexts. Authentication policies map context attributes, and assurance level identifiers, in authentication modes. Relying on those attributes, CAM2 authentication platform dynamically retrieves the appropriate policies from the database for each request. Due to dynamic context evaluation, any change to an authentication policy is instantly assumed by the platform, without having to change its structure or even causing any downtime.

Extensibility for new authentication models

CAM2ML authentication policies, which are stored on the CAM2 authentication platform, contain a combination of generic identifiers for the authentication modes, such as Hardware security tokens, static passwords, Bluetooth tokens, Gesture identification or Data Matrix spontaneous authentication. The platform is completely independent from these identifiers, relying on them only to dynamically load the appropriate authentication module. This architecture allows the addition and reutilization of authentication modules without interfering with running instances of the authentication platform.

The implementation of a extended version of Kerberos V5 and the adaptation of a WEB Post/Redirect protocol also have shown that is possible to adapt CAM2 current authentication protocols. CAM2ML is as flexible language that can be expressed on multiple ways, allowing the same platform to interact with multiple systems, requiring minimal adaptations.

Generic usage

CAM2 framework must deal with multiple types of client applications, base single sign-on systems, authentication modes and context attributes. The generality of context attributes and authentication modes is dealt by the extensibility mechanisms provided by the authentication platform.

Generality applied to client applications is supported by the Client Integration Tier services defined by CAM2 framework. Various types of applications may be uniformly integrated with

CAM2 platform with a minimum effort amount. During this dissertation and for validation purposes, three application domains were considered: Web applications, through the development of a prototype for the integration of CLIP Web application, mobile applications through the implementation of a simple application for bank account management, and finally a Kerberized application, representing that is possible to integrate CAM2 authentication in a well known ans vastly used protocol, mitigating some of its problems.

Generality applied to base single sign-on systems is addressed by relying on authentication delegators as connector modules, which hide the integration details with the base system from the CAM2 authentication logic. In the current implementation, the communication with OpenSSO is made through OpenSSO Client SDK, however it could easily be adapted to SAML including the authentication platform on a federation circle of trust.

Performance

Performance is a critical issue on centric authentication systems, such as single sign-on platforms ,since there is the need of managing requests from multiple clients and from multiple system at one point. Typically, these systems are available through the Internet and then must be available 24 hours per day, handling requests from every part of the globe. Data centers have an important role on this field, assuring availability to systems like the one we propose. However, due to the number of accesses expected, performance must be taken in consideration. If the overhead introduced by CAM2 middleware was large enough to cause performance impact, the model would not be viable.

The performance tests executed on this chapter had shown that on local environments, and comparing with OpenSSO Enterprise 8.0, the CAM2 platform doubles the authentication process time. Nevertheless, the latency added to the base single sign-on still is reduced and fairly below one second. The same tests, now using the Internet to connect clients to CAM2 platform, had shown that the process latency imposed by the middleware is much smaller than the time spent due to communication latency, then proving that, when used on the Internet, CAM2 authentication processes overall latency (communication and processing) is bound to the communication latency noticed on the Internet, minimizing the impact of the introduction of context-aware authentication mechanisms.

8. Conclusions and future work

Nowadays, centrally managed authentication architectures present some limitations when applied to ubiquitous, heterogeneous or large scale environments. Current authentication systems use specific authentication factors. To overcome the identified drawbacks associated to these factors, the implementation of multi-factor authentication platforms is an interesting direction. These platforms present interesting advantages such as:

- mitigation of drawbacks of individual authentication factors combining them as complementary authentication proofs;
- flexibility to use the combination of factors according to the specific access control needs of different services and applications.
- centralized management promoting the coexistence of different authentication information bases, when different factors are combined.
- support for different authentication processes according with the required security criteria, balanced with other requirements such as high availability, convenience principles, operational costs, functionality specialization, risk or dynamic subjects profiling and context-aware conditions

A context-aware authentication platform can be implemented leveraging the base functions that can be provided by *state-of-art* single sign-on systems.

Typical single sign-on solutions maintain rigid authentication methods, without taking in consideration multiple contexts from where the authentication can be invoked. Authentication involving different devices, channels and usage interactions provides different assurance levels even for the same principals and operations. Setting a fixed combination of authentication factors is not the best solution. For instance, a small number of factors may be insufficient to grant the required assurance levels in some situations, while uniquely requiring a large amount of authentication proofs is not convenient for users and brings costs and performance issues to service providers. Systems like OpenSSO and ESOE provide extension mechanisms for the integration of multiple authentication modes, however they don't supply dynamic adaptation of authentication factors to mutable contexts and assurance levels.

This dissertation has addressed the limitations presented above through the specification of a context aware multi-factor and multi-modal authentication framework that can sit as an extensible middleware atop of a SSO base solution. The contribution provides an uniform and flexible authentication platform for multi-factor authentication requirements and context-aware authentication modes. The platform and its abstractions can be used by different ubiquitous applications and services.

The proposed platform follows a context-aware authentication model based on authentication policies expressed by a defined context-aware multi-factor and multi-modal markup language - CAM2ML - as well as a new interaction model specially tailored for the identified requirements. Authentication processes involving those contexts, factor combinations and assurance levels can be defined with the expressiveness provided by CAM2ML. The protocol defined by CAM2ML also enables interoperability between clients, service providers and identity providers , in the same way SAML already does to currently available authentication systems using federation mechanisms. However, SAML do not support extension mechanisms for mapping authentication contexts and factors given different assurance levels, then limiting their interaction models to static combinations of authentication proofs.

The implementation of the platform demonstrates that it is possible to develop dynamic context-aware multi-factor and multi-modal authentication systems suitable for ubiquitous requirements and flexible enough to support the addition of new and emergent authentication factor without interfering with the system runtime.

The platform implementation was validated under three metrics: generality to ubiquitous support, context-aware multi-factor combination and performance evaluation.

- Generality and ubiquitous support was tested and validated through the integration of CAM2 concepts on Web applications, mobile applications and by the creation of a CAM2 version of Kerberos protocol.

- Dynamic and context-aware multi-factor combination of modes were achieved and validated by the implementation of five authentication modules, representing classic and ubiquitous authentication factors.

- Performance was validated by testing system load and authentication latency in two distinct environments: a Local Area Network environment and Internet environment. The

evaluation was compared with the performance metrics obtained with the OpenSSO 8.0 Enterprise in the same conditions. The overhead introduced by CAM2 middleware doubles the latency of a single authentication process in OpenSSO. However the latency introduced by the Internet attenuates the overhead, only increasing latencies from 6% to 36%, depending on the load conditions. These results demonstrate that the proposed architecture does not compromise the the overall latency seen by end users.

According with the contributions proposed on the beginning of this dissertation, the objectives were achieved, demonstrating that it is possible and viable to develop flexible context-aware multi-factor and multi-modal authentication systems, leveraging SSO systems to the challenges, requirements and opportunities of ubiquitous computing devices and applications.

8.1 Future work

The work done around CAM2 elements is not finished and may be continued through multiple directions.

- The development and inclusion of new authentication modes is an endless process. Opportunities for the creation of new authentication factors, and exploration of the existing ones, are constantly emerging due to the fast growth of ubiquitous computation. Additionally to the authentication modes implemented during this dissertation, some others where thought, namely the recognition of fingertips, faces, voices and EEG patterns and the addition of spontaneous authentication interaction models.

- SAML is increasingly being adopted as the standard framework for single sign-on authentication when there is the need of supporting interoperability through federation. Besides the fact that CAM2ML supports some SAML abstractions, the authentication platforms relying on it can not be included on the circle of trust defined by SAML specification. Two direction can be taken from here. In a first approach, CAM2ML objects could be translated to SAML assertions and SAML assertions to CAM2ML objects. For that it only would be necessary to create another integration module on the client integration

tier of CAM2 authentication platform. All the remaining process would be straight forward. An alternative would be extending the SAML specification to support the concepts introduced by this dissertation, substituting CAM2ML.

- Registering subjects and their authentication data is not considered by CAM2 specification. Instead, it is assumed that the enrollment process is previously and coherently performed on each base single sign-on system in order to perform authentication delegation. The creation of a single-enrollment process could be integrated including users devices information and features. That way identity management could also be made centrally, reducing the complexity of maintaining the same data in multiple bases.

- Despite of being a simple markup language CAM2ML may not be straightforward for users with low technical capacities. Managing policies may then become an error-prone process. To overcome this problem a graphical programing language could be developed on top of CAM2ML, simplifying the understanding and changeability of authentication policies.

- CAM2ML does not have explicit support for the expression of identities representing roles. In order to use the concepts proposed by CAM2 on systems relying on RBAC mechanisms, it would be necessary to introduce the required extensions to CAM2ML.

- Subjects and their authentication data are intrinsically connected. This can result in privacy issues if we consider that CAM2ML assertions can be used by service providers to delegate authentication over multiple sub-systems. The work described on [10] presents a way of using combinations of virtual identifiers and use them on cross-layer single sign-on. CAM2 interaction model and CAM2ML could be extended considering these concepts.

CAM2 is far away of reaching its maturity. Multiple aspects related to ubiquitous computation are emerging, bringing new challenges and opportunities for the enrichment of CAM2 framework and authentication systems in general. The development of this work is continuous and can always be refined to adapt to the requirements of the computational world.

Nomenclature

ACL Access Control List

API Application Programming Interface

AS Authentication Server

CAM2 Context-Aware Multi-Factor Multi-Modal Authentication Framework

CAM2ML Context-Aware Multi-Factor and Multi-Modal Markup Language

COM Component Object Model

CORBA Common Object Request Broker Architecture

DAC Discretionary Acess Control

DSA Digital Signature Algorithm

EAR Enterprise Archive. A file format used by J2EE

EEG Electroencephalography

EJB Enterprise Java Beans

FAR False Accepting Rate

FMR False Matching Rate

FNMR False Non-Matching Rate

FRR False Rejection Rate

GPS Global Positioning System (GPS)

HTTP HyperText Transfer Protocol

HTTP POST A type of HTTP request message.

HTTPS Hypertext Transfer Protocol Secure

IdP Identity Provider

J2EE Java Platform Enterprise Edition

J2ME Java Plataform Micro Edition

J2SE Java 2 Standard Edition

Java RMI Java Remote Method Invocation API

JSP JavaServer Pages

JSR-82 Java APIs for Bluetooth. Is a Java ME specification for APIs that allow Java midlets to use Bluetooth on supporting devices

LAN Local Area Network

MAC Mandatory Access Control

OTP One Time Password

PDA Personal Digital Assistant. It is also known as a palmtop computer

PIN Personal Identification Number

PKI Public Key Infrasctructure

RBAC Role Based Access Control

RESTful Representational state transfer

RF Radio Frequency radiation. Is a subset of electromagnetic radiation with a wavelength of 100km to 1mm

RFID Radio-frequency identification

SAML Security Assertion Markup Language

SDK Software Development Kit

SHA1 The SHA hash functions are a set of cryptographic hash functions designed by the National Security Agency (NSA)

SIM Subscriber Identity Module. It securely stores the service-subscriber key (IMSI) used to identify a subscriber on mobile telephony devices (such as mobile phones and computers)

SOAP Simple Object Access Protocol

SP Service Provider

SPEP Service Policy Enforcer Points. Used By Enterprise Sign-on Engine as authentication agents.

SPI Service Program Interface

SSL Secure Sockets Layer

SSO Single Sign-On

TCO Total Cost of Ownership

TGS Ticket Granting Server

TLS Transport Layer Security

UML Unified Modeling Language

UMTS Universal Mobile Telecommunications System

URL Universal Resource Locator

USB Universal Serial Bus

WAR Web application Archive. A file format used by J2EE

WI-FI Is a term for certain types of wireless local area network (WLAN) that use specifications in the 802.11 family

X.509 An ITU-T standard for a public key infrastructure (PKI) for single sign-on (SSO) and Privilege Management Infrastructure (PMI)

XML Extensible Markup Language

XPath XML Path Language

XQuery A query and functional programming language that is designed to query collections of XML data

XRDS Extensible Resource Descriptor Sequence

XRI Extensible Resource Identifier

Bibliography

[1] M. Abadi. Access control based on execution history. In *Proceedings of the 10th Annual Network and Distributed System Security Symposium*, 2003.

[2] C Adams, N. Edwards, P. Hallam-Baker, J Hodges, C Knouse, C. McLaren, P. Mishra, R. Morgan, E. Maler, T. Moses, D Orchard, I. Reid, M. Erdos, J. Oblix, C. Oblix, C. Netegrity, P. Netegrity, T. Entrust, and D. Bea. Assertions and protocol for the oasis security assertion markup language (saml), 2001. `http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf`.

[3] Liberty Aliance. Liberty id-ff architecture overview, December 2008. `http://www.projectliberty.org/`.

[4] J. Anderson and R Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley, 2001.

[5] P. Bernardi, F. Gandino, B. Montrucchio, M. Rebaudengo, and E. Sanchez. Design of an uhf rfid transponder for secure authentication. In *GLSVLSI '07: Proceedings of the 17th ACM Great Lakes symposium on VLSI*, 2007.

[6] G Borriello. A survey and taxonomy of location systems for ubiquitous computing. Technical report, IEEE Computer, 2001.

[7] M. Burmester and J. Munilla. A Flyweight RFID Authentication Protocol. In *Workshop on RFID Security*, 2009.

[8] Bruce Christianson, Bruno Crispo, James A. Malcolm, and Michael Roe, editors. *Security Protocols, 13th International Workshop, Cambridge, UK, April 20-22, 2005, Revised Selected Papers*. Springer, 2007.

[9] G. Coulouris and J Dollimore. *Distributed systems: concepts and design*. Addison-Wesley Longman Publishing Co., Inc., 1988.

[10] Cristiano Andre da Costa, Adenauer Correa Yamin, and Claudio Fernando Resin Geyer. Toward a general software infrastructure for ubiquitous computing. *IEEE Pervasive Computing*, 2008.

[11] D Denning and P MacDoran. Location-based authentication: grounding cyberspace for better security. *Internet besieged: countering cyberspace scofflaws*, 1998.

[12] P. Doyle, M. Deegan, C. O'Driscoll, M. Gleeson, and B. Gillespie. Ubiquitous desktops with multi-factor authentication. In *ICDIM*. IEEE, 2008.

[13] A. Durresi, V. Paruchuri, M Durresi, and L. Barolli. Secure spatial authentication using cell phones. In *ARES '07: Proceedings of the The Second International Conference on Availability, Reliability and Security*, pages 543–549. IEEE Computer Society, 2007.

[14] M. Golfarelli, D. Maio, and D. Malton. On the error-reject trade-off in biometric verification systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 1997.

[15] Li Gong. *Inside Java 2 platform security architecture, API design, and implementation*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[16] The Open Knowledge Initiative. O.k.i. architectural concepts, October 2008. `http://prdownloads.sourceforge.net/okiproject/OkiArchitecturalConcepts`

[17] A. Jain and A. Ross. Multibiometric systems. *Communication ACM*, 2004.

[18] A. K. Jain, A. Ross, and S. Prabhakar. An introduction to biometric recognition. *Circuits and Systems for Video Technology, IEEE Transactions*, 2004.

[19] Anil K. Jain and Arun Ross. Multimodal biometrics: An overview. In *Proceedings of the 12th European Signal Processing Conference*. ACM, 2004.

[20] Ed. K. Zeilenga. Lightweight directory access protocol (ldap):technical specification road map. RFC Editor, 2006.

[21] J Kohl, B Neuman, and Y. Theodore. The evolution of the kerberos authentication service. In *EurOpen Conference*. IEEE Computer Society Press, 1994.

[22] J. Kohl and C. Neuman. The kerberos network authentication service (v5). RFC Editor, 1993.

[23] T. Kwon, S. Park, and S. Shin. Multi-modal authentication for ubiquitous computing environments. In *HCI (6)*, pages 113–121, 2007.

[24] C Lai, L. Gong, L. Koved, A. Nadalin, and Schemers R. User authentication and authorization in the java(tm) platform. In *ACSAC '99: Proceedings of the 15th Annual Computer Security Applications Conference*.

[25] S Marcel and J Millan. Person authentication using brainwaves (eeg) and maximum a posteriori model adaptation. *IEEE Trans. Pattern Anal. Mach. Intell*, 2007.

[26] R. Mayrhofer. The candidate key protocol for generating secret shared keys from similar sensor data streams. In *ESAS*. Springer-Verlag, 2007.

[27] J. McCune, A. Perrig, and M. Reiter. Seeing-is-believing: Using camera phones for human-verifiable authentication. *International Journal of Security and Networks*, 2005.

[28] K. McDonald. *The Quick Python Book*. Manning Publications Co., Greenwich, CT, USA, 1999.

[29] W Meier. exist, open source native xml database - accessed july 2009, July 2009. `http://exist.sourceforge.net/`.

[30] SUN Microsystems. Glassfish community website, July 2009. `https://glassfish.dev.java.net/`.

[31] SUN Microsystems. Jersey community website, July 2009. `https://jersey.dev.java.net/`.

[32] SUN Microsystems. *SunOpenSSO Enterprise 8.0 - TechnicalOverview*. July 2009. `http://docs.sun.com/app/docs/doc/820-3740/`.

[33] OASIS. extensible access control markup language, December 2008.

[34] OASIS. extensible resource identifier, DECEMBER 2008. `http://www.oasis-open.org/committees/xri`.

[35] United States Government Department of Defense. *Trusted Computer System Evaluation Criteria (Orange Book)*. US DoD, 1985.

[36] Queensland University of Technology. Enterprise sign on engine webpage, December 2008. `http://www.esoeproject.org/accessed`.

[37] S. Perera, C. Herath, J. Ekanayake, E. Chinthaka, A. Ranabahu, D. Jayasinghe, S. Weerawarana, and G. Daniels. Axis2, middleware for next generation web services. *Web Services, IEEE International Conference*, 2006.

[38] John Podd, Julie Bunnell, and Ron Henderson. Cost-effective computer security: Cognitive and associative passwords. In *OZCHI '96: Proceedings of the 6th Australian Conference on Computer-Human Interaction (OZCHI '96)*. IEEE Computer Society, 1996.

[39] Source Forge project. Mobile one time passwords. `http://motp.sourceforge.net/`.

[40] ITU-T Recommendation. Osi x800 security architecture. ITU-T, 1991.

[41] D. Recordon and D. Reed. Openid 2.0: a platform for user-centric identity management. In *DIM '06: Proceedings of the second ACM workshop on Digital identity management*. ACM, 2006.

[42] RSA. Securid, October 2008.

[43] V Samar and C Lai. Making login services independent of authentication technologies. *3rd ACM Conference on Computer and Communications Security*, 1996.

[44] R Sandhu, E. Coyne, H Feinstein, and C Youman. Role-based access control models. *Computer*, 1996.

[45] N. Sastry, U. Shankar, and D Wagner. Secure verification of location claims. ACM, 2003.

[46] Jörgen Scheible. Mobile phone programming for multimedia. In *MULTIMEDIA '07: Proceedings of the 15th international conference on Multimedia*. ACM, 2007.

[47] R. Shirey. Internet security glossary, version 2. RFC Editor, 2007.

[48] SimpleAct. Quickmark - a multi device bar code reader, July 2009. `http://www.quickmark.com.tw`.

[49] William Stallings and Lawrie Brown. *Computer Security: Principles and Practice*. Prentice Hall Press, 2007.

[50] A. Tanenbaum and M. Van Steen. *Distributed Systems: Principles and Paradigms*. Prentice Hall, 2001.

[51] Bluecove Team. Bluecove project website, July 2009. `http://www.bluecove.org/`.

[52] Andrej Volchkov. Revisiting single sign-on: A pragmatic approach in a new context. *IT Professional*, 2001.

[53] Mark Weiser. The computer for the 21st century. *ACM SIGMOBILE Mobile Computing and Communications Review*, 1991.

[54] K. Ying Yu and L. Yiu, S.M.and Hui. Rfid forward secure authentication protocol: Flaw and solution. *Complex, Intelligent and Software Intensive Systems, International Conference*, 2009.

[55] N. Zhang, J. Chin, A. Rector, C. Goble, and Y. Li. Towards an authentication middleware to support ubiquitous web access. In *COMPSAC '04: Proceedings of the 28th Annual International Computer Software and Applications Conference - Workshops and Fast Abstracts*. IEEE Computer Society, 2004.

[56] L. Zhu and B. Tung. Public key cryptography for initial authentication in kerberos (pkinit). RFC Editor, 2006.

[57] M Zviran and W. J. Haga. A comparison of password techniques for multilevel authentication mechanisms. *The Computer Journal*, 1993.

A. CAM2ML XML Schema

A.1 Assertion.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
                    xmlns="http://CAM2ML.di.fct.unl.pt/Basic"
                    targetNamespace="http://CAM2ML.di.fct.unl.pt/Basic"
                    elementFormDefault="qualified"
>
 <xs:include schemaLocation="Principal.xsd" />
 <xs:include schemaLocation="Policy.xsd" />
  <xs:element name="assertion">
    <xs:complexType>
         <xs:sequence>
           <xs:element ref="issuer" maxOccurs="1" minOccurs="1"/>
           <xs:element name="application" type="xs:string" minOccurs="1" maxOccurs="1" />
           <xs:element name="levelOfAssurance" type="xs:string" minOccurs="1" maxOccurs="1" />
                <xs:choice>
                  <xs:element ref="policy_statement" />
                  <xs:element ref="authentication_statement"/>
                </xs:choice>
         </xs:sequence>
         <xs:attribute name="type" type="assertion_type" use="required"/>
         <xs:attribute name="id" type="assertion_id_type" use="required"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="authentication_statement">
   <xs:complexType>
        <xs:sequence>
         <xs:element ref="authentication_validity"  minOccurs="1" maxOccurs="1" />
        </xs:sequence>
   </xs:complexType>
  </xs:element>

  <xs:element name="policy_statement">
   <xs:complexType>
         <xs:sequence>
          <xs:element ref="policy" />
         </xs:sequence>
        </xs:complexType>
  </xs:element>

  <xs:simpleType name="assertion_type">
   <xs:restriction base="xs:string">
        <xs:enumeration value="Authentication"/>
        <xs:enumeration value="Policy"/>
```

```
    </xs:restriction>
   </xs:simpleType>

   <xs:simpleType name="assertion_id_type">
    <xs:restriction base="xs:string">
     <xs:pattern value="ASSER-\d"/>
    </xs:restriction>
   </xs:simpleType>
</xs:schema>
```

A.2 Request.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
                        xmlns="http://CAM2ML.di.fct.unl.pt/Basic"
                        targetNamespace="http://CAM2ML.di.fct.unl.pt/Basic"
                        elementFormDefault="qualified"
>
 <xs:include schemaLocation="Principal.xsd"/>
 <xs:include schemaLocation="Context.xsd"/>
 <xs:include schemaLocation="AuthenticationModes.xsd"/>

 <xs:element name="request">
  <xs:complexType>
   <xs:sequence>
        <xs:element ref="subject" minOccurs="1" maxOccurs="1"/>
    <xs:element name="application" minOccurs="1" maxOccurs="1"/>
        <xs:element name="levelOfAssurance" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:choice>
     <xs:element ref="policy_request"/>
     <xs:element ref="authentication_request"/>
    </xs:choice>
   </xs:sequence>
  </xs:complexType>
 </xs:element>

 <xs:element name="authentication_request">
  <xs:complexType>
   <xs:sequence>
        <xs:element ref ="context"/>
        <xs:element ref="authentication_data"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>

 <xs:element name="authentication_data">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="authentication_item" minOccurs="0" maxOccurs="unbounded"/>
```

```
   </xs:sequence>
  </xs:complexType>
 </xs:element>

 <xs:element name="authentication_item">
  <xs:complexType>
   <xs:simpleContent>
    <xs:extension base="xs:string">
     <xs:attribute name="mode" type="all_auth_modes"/>
    </xs:extension>
    </xs:simpleContent>
   </xs:complexType>
 </xs:element>

 <xs:element name="policy_request">
   <xs:complexType>
    <xs:sequence>
    <xs:element ref ="context"/>
    </xs:sequence>
     </xs:complexType>
    </xs:element>
</xs:schema>
```

A.3 AuthenticationModes.xsd

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
                   xmlns="http://CAM2ML.di.fct.unl.pt/Basic"
        targetNamespace="http://CAM2ML.di.fct.unl.pt/Basic"
           elementFormDefault="qualified">
>
    <xs:element name="authentication_modes" >
         <xs:complexType>
          <xs:sequence>
                <xs:element ref="mode" maxOccurs="unbounded" minOccurs="1" />
      </xs:sequence>
         </xs:complexType>
    </xs:element>

   <xs:element name="mode">
    <xs:complexType>
     <xs:attribute name="type" type="all_auth_modes"/>
    </xs:complexType>
   </xs:element>

   <xs:simpleType name="all_auth_modes">
    <xs:union memberTypes="basic_suh_values␣basic_sui_values␣basic_suk_values␣basic_sum_values" />
   </xs:simpleType>

   <xs:simpleType name="basic_suk_values">
```

```
    <xs:restriction base="xs:string">
     <xs:enumeration value="PASSWORD"/>
         <xs:enumeration value="PIN"/>
     <xs:enumeration value="PASSPHRASE"/>
        </xs:restriction>
   </xs:simpleType>

  <xs:simpleType name="basic_suh_values">
   <xs:restriction base="xs:string">
    <xs:enumeration value="HW_TOKEN"/>
        <xs:enumeration value="SW_TOKEN"/>
        <xs:enumeration value="BT_LOC"/>
        <xs:enumeration value="HW_TOKEN_WITH_SUK"/>
        <xs:enumeration value="HW_TOKEN_WITH_SUH"/>
   </xs:restriction>
  </xs:simpleType>


  <xs:simpleType name="basic_sum_values">
   <xs:restriction base="xs:string">
         <xs:enumeration value="MOVEMENTS"/>
         <xs:enumeration value="VISUAL_PROOF"/>
   </xs:restriction>
  </xs:simpleType>


  <xs:simpleType name="basic_sui_values">
   <xs:restriction base="xs:string">
        <xs:enumeration value="FINGERTIP"/>
   </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

A.4 Context.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns="http://CAM2ML.di.fct.unl.pt/Basic"
           targetNamespace="http://CAM2ML.di.fct.unl.pt/Basic"
           elementFormDefault="qualified"
>

 <xs:element name="context">
  <xs:complexType>
    <xs:sequence>
     <xs:element ref="time"minOccurs="0" />
     <xs:element ref="device"   minOccurs="0" />
     <xs:element ref="protocol" minOccurs="0" />
     <xs:element ref="channel"  minOccurs="0" />
```

```
    </xs:sequence>
 </xs:complexType>
</xs:element>

<xs:element name="time">
 <xs:complexType>
  <xs:sequence>
       <xs:choice>
           <xs:element ref="interval" minOccurs="0" maxOccurs="unbounded"/>
               <xs:element ref="value" minOccurs="0" maxOccurs="unbounded"/>
       </xs:choice>
  </xs:sequence>
 </xs:complexType>
</xs:element>

<xs:element name="device" >
 <xs:complexType>
  <xs:attribute name="type" type="basic_device"/>
 </xs:complexType>
</xs:element>

<xs:element name="channel" >
 <xs:complexType>
  <xs:attribute name="type" type="basic_channel"/>
 </xs:complexType>
</xs:element>

<xs:element name="protocol" >
 <xs:complexType>
  <xs:attribute name="type" type="basic_protocol"/>
 </xs:complexType>
</xs:element>

<xs:element name="interval">
 <xs:complexType>
       <xs:choice>
           <xs:attribute name="begin" type="xs:time"/>
               <xs:attribute name="end"   type="xs:time"/>
               <xs:attribute name="time" type="xs:time"/>
       </xs:choice>
 </xs:complexType>
</xs:element>

<xs:simpleType name="basic_device">
 <xs:restriction base="xs:string">
   <xs:enumeration value="Private␣Desktop␣Computer" />
       <xs:enumeration value="Public␣Desktop␣Computer" />
   <xs:enumeration value="Public␣Access␣Point" />
       <xs:enumeration value="Laptop␣Computer" />
   <xs:enumeration value="Cell␣Phone" />
```

```
    <xs:enumeration value="PDA" />
   </xs:restriction>
 </xs:simpleType>

 <xs:simpleType name="basic_protocol">
  <xs:restriction base="xs:string">
   <xs:enumeration value="HTTP" />
   <xs:enumeration value="HTTP_SSL" />
  </xs:restriction>
 </xs:simpleType>

 <xs:simpleType name="basic_channel">
  <xs:restriction base="xs:string">
   <xs:enumeration value="Ethernet" />
   <xs:enumeration value="WI-FI" />
   <xs:enumeration value="Bluetooth" />
   <xs:enumeration value="GPRS" />
   <xs:enumeration value="UMTS" />
  </xs:restriction>
 </xs:simpleType>
</schema>
```

A.5 Policy.xsd

```
<xs:schema
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns="http://CAM2ML.di.fct.unl.pt/Basic"
        targetNamespace="http://CAM2ML.di.fct.unl.pt/Basic"
        elementFormDefault="qualified">

        <xs:include schemaLocation="Context.xsd" />
        <xs:include schemaLocation="AuthenticationModes.xsd" />
        <xs:element name="policy">
         <xs:complexType>
         <xs:sequence>
      <xs:element ref="context" minOccurs="1" maxOccurs="unbounded" />
      <xs:element ref="authentication_modes" minOccurs="1" maxOccurs="1"/>
      <xs:element ref="authentication_validity"  minOccurs="1" maxOccurs="1" />
          <xs:element name="levelOfAssurance" type="xs:string" minOccurs="1" maxOccurs="1" />
         </xs:sequence>
        </xs:complexType>
   </xs:element>
   <xs:element name="authentication_validity">
        <xs:complexType>
         <xs:attribute name="duration" type="xs:time"/>
          </xs:complexType>
   </xs:element>
</xs:schema>
```

A.6 Principal.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
                    xmlns="http://CAM2ML.di.fct.unl.pt/Basic"
           targetNamespace="http://CAM2ML.di.fct.unl.pt/Basic"
            elementFormDefault="qualified">

 <xs:element name="cn"  type="xs:string" />

 <xs:element name="issuer">
  <xs:complexType>
        <xs:sequence>
         <xs:element ref="cn"  />
         <xs:element ref="signature" />
   </xs:sequence>
  </xs:complexType>
 </xs:element>

 <xs:element name="subject">
  <xs:complexType>
   <xs:sequence>
        <xs:element ref="cn"  maxOccurs="1" minOccurs="1"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>

 <xs:element name="signature">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="algorithm" type="xs:string" />
    <xs:element name="value" type="xs:string" />
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```