UNIVERSITÀ DEGLI STUDI DI PISA

Facoltà di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea in Informatica (Vecchio Ordinamento)
Anno Accademico 2004-2005

Tesi di Laurea:

Progettazione di un framework Open Source per l'Identity Management nella Pubblica Amministrazione

Candidato:
Giandomenico Napolitano

Relatori:

Prof. A. Corradini

Prof. T. Flagella

Controrelatore:

Prof. R. Barbuti

Abstract

In questi anni gli enti amministrativi, centrali e locali, del Paese stanno attraversando una fase di radicali trasformazioni nella direzione dell'e-Government, ovvero un modello di burocrazia fortemente automatizzata ed integrata tramite tecnologie di ICT. In questo contesto la tesi si pone come obiettivo quello di analizzare le esigenze della Pubblica Amministrazione e proporre soluzioni alle problematiche riguardanti la gestione delle identità digitali dei cittadini, in particolar modo per quanto riguarda l'autenticazione, l'autorizzazione, la gestione di profili utente e scenari avanzati di cooperazione in sicurezza tra le amministrazioni. Le soluzioni proposte riducono al minimo gli interventi sulle infrastrutture esistenti e si allineano all'attuale assetto politico-istituzionale improntato sul federalismo.

Indice

Introduzione **ix**

1 Nozioni di contesto **1**

1.1 L'Identity Management 2

1.1.1 Definizione di identità digitale 3

1.1.2 Tassonomia dell'Identity Management 3

1.1.3 L'Identity Management e l'e-Government 6

1.2 Il passaggio a tecnologie basate su XML e Web Services 7

1.2.1 XML 7

1.2.2 Il paradigma dei Web Services 8

1.2.3 Sicurezza nei Web Services 10

1.2.4 Le applicazioni web 12

1.3 Modelli per il controllo di accesso e l'autorizzazione 16

1.3.1 RFC 2904: Authorization Framework 17

1.3.2 ISO 10181-3: Access control framework 24

1.3.3 Mappa terminologica 28

1.3.4 Il modello RBAC 28

2 Analisi dei requisiti della Pubblica Amministrazione italiana in termini di Identity Management **37**

2.1 Le recenti iniziative del CNIPA 39

2.1.1 Il Sistema Pubblico di Connettività e Cooperazione . . 40

2.1.2 Valutazione del modello Open Source 50

2.2 Una selezione di progetti concreti e bandi 54

2.2.1 Il progetto ICAR 54
2.2.2 Regione Toscana: e.Toscana 56
2.2.3 Regione Campania: SPICCA 61
2.3 Sommario dei requisiti 68
2.3.1 Non-requisiti sui terminali per l'accesso ai servizi di e-Government 68
2.3.2 Paradigmi di controllo di accesso e profilazione avanzati 70
2.3.3 Distribuzione locale delle competenze e delle responsabilità 71
2.3.4 Integrazione con le infrastrutture esistenti 73
2.3.5 Sviluppo di nuove infrastrutture 74
2.3.6 Requisiti sul modello e le metodologie di sviluppo . . . 75

3 Standards e tecnologie di rilievo 77
3.1 Standards sul formato e lo scambio di credenziali 78
3.1.1 SAML 79
3.1.2 X.509 94
3.2 Soluzioni per il Single Sign On 96
3.2.1 Shibboleth 97
3.2.2 Liberty Alliance 111
3.3 Soluzioni per il controllo di accesso 115
3.3.1 PERMIS 115
3.3.2 Il controllo di accesso in Globus Toolkit 4 123

4 Proposta di framework per l'Identity Management nella Pubblica Amministrazione 131
4.1 Sistema federato di autenticazione 134
4.1.1 Modello di federazione 134
4.1.2 Gestione delle identità 138
4.2 Infrastrutture per l'attribuzione di ruoli e l'autorizzazione . . . 142
4.2.1 Modello di federazione 142
4.2.2 SAML Attribute Authority 143

4.2.3 SAML Policy Decision Point 145
4.2.4 Integrazione con il sistema federato di autenticazione . 153
4.3 Applicabilità del framework alle iniziative analizzate 154
4.3.1 Applicabilità del framework al progetto ICAR 154
4.3.2 Possibili integrazioni con SPC 154
4.3.3 Compatibilità con il bando di Regione Campania . . . 156
4.3.4 Compatibilità con il bando di Regione Toscana 157

5 Il prototipo realizzato 159
5.1 Finalità 160
5.1.1 Sviluppo di un'infrastruttura di autorizzazione per Shibboleth 160
5.1.2 Integrazione di applicazioni web in un SP Shibboleth . 160
5.2 Configurazione ed implementazione dei componenti 162
5.2.1 Shibboleth IdP 163
5.2.2 Shibboleth Java SP 164
5.2.3 SAML PDP 165
5.2.4 Profile Webapp 166
5.3 Possibili Sviluppi 169

Conclusioni 171

Bibliografia 175

Glossario degli acronimi e delle abbreviazioni 189

Codice sorgente del prototipo 197

Introduzione

L'obiettivo di questa tesi è quello di trovare soluzione ad alcune problematiche relative alla gestione delle identità digitali dei cittadini, nell'ambito della rivoluzione in atto nella Pubblica Amministrazione (PA) del Paese nella direzione dell'*e-Government*.

Per e-Government si intende l'introduzione massiccia di Information and Communication Technology (ICT) negli enti amministrativi allo scopo di migliorarne l'efficienza interna, l'integrazione con il restante tessuto amministrativo ed i rapporti con i fruitori dei propri servizi, siano essi cittadini, associazioni, imprese od altri enti del settore pubblico.

I vantaggi dell'adozione del modello e-Government nella (ri)organizzazione delle strutture amministrative dello Stato derivano in parte da quelli ottenibili tramite l'informatizzazione di una generica organizzazione, e quindi facenti riferimento all'area dell'e-Business. Per altri versi, tali vantaggi costituiscono caratteristiche e potenzialità specifiche del settore pubblico. Di seguito sono sintetizzati alcuni dei principali vantaggi di un tale modello:

Riduzione della componente discrezionale. Un effetto della traduzione di norme e processi amministrativi in *logica applicativa* (*business logic*) nelle applicazioni di e-Government è una maggiore impermeabilità degli stessi all'intervento umano. Dirigenti ed impiegati delle amministrazioni avranno quindi minore potere decisionale sul corso dei processi amministrativi. Ciò ha le seguenti implicazioni principali:

- miglioramento dell'efficienza;
- riduzione di fenomeni di favoritismo o corruzione;

- maggiori garanzie di privacy per i cittadini oggetto dei processi;
- più fedele e rapida propagazione dell'azione governativa nelle singole amministrazioni distribuite sul territorio.

Separazione delle funzioni di *front office* da quelle di *back office*. Molti degli enti amministrativi del Paese sono caratterizzati da una duplice natura: la gestione diretta dei rapporti con il cittadino (attività di sportello o *front office*) e quella dei processi amministrativi veri e propri (*back office*) innescati dal cittadino stesso o da altri enti pubblici o privati. Uno dei principi base delle iniziative di e-Government è quello di unificare le funzioni di *front office* in "sportelli unici" attraverso i quali le necessità del cittadino possano essere indirizzate agli enti opportuni, che si occuperanno di gestire le relative attività di *back office* secondo le proprie competenze. Gli sportelli unici hanno inoltre la funzione di realizzare servizi complessi componendo quelli offerti dalle singole amministrazioni.

Automatizzazione dei processi. La gestione automatizzata del ciclo di vita di una pratica e delle interazioni con le persone o gli enti in essa coinvolti permette di ridurne drasticamente i tempi di completamento – in alcuni casi anche di qualche ordine di grandezza – con ovvi vantaggi in termini di costi, efficienza e qualità del servizio.

Chiara suddivisione delle competenze. Una informatizzazione su larga scala della PA implica la definizione chiara degli oggetti applicativi (ad es. le pratiche, i cittadini, ecc.) e delle operazioni che i singoli enti amministrativi sono autorizzati a compiere su di essi. Eventuali conflitti di competenza tra più amministrazioni emergerebbero, quindi, già nella fase di progettazione ad alto livello delle applicazioni che prevedano la cooperazione delle stesse.

Abilitazione di nuovi servizi e scenari applicativi. La disponibilità di strumenti tecnologici efficienti e componibili come quelli previsti dall'e-

Government permetterà l'attuazione di nuove tipologie di servizi precedentemente impensabili.

Maggiore capillarità dell'azione amministrativa. Al giorno d'oggi buona parte[1] degli italiani dispone degli strumenti di accesso ai servizi di e-Government che diventano, a tutti gli effetti, degli 'sportelli personali'. È inoltre possibile allargare ulteriormente il bacino d'utenza rendendo i servizi di e-Government accessibili attraverso altre tipologie di terminali come i telefoni cellulari[2] (*m-Government*) e la televisione digitale (*t-Government*).

Il "Piano d'azione per l'e-Government"

In ottemperanza al programma europeo e-Europe [e-Europe-web], il Governo Italiano ha varato, a partire dal giugno 2000, il "Piano d'azione per l'e-Government" [Piano13], ovvero una serie di direttive ed interventi pratici volti a rivoluzionare l'utilizzo dell'ICT nella PA su scala nazionale.

Le linee guida di tale piano, ad opera dell'allora Ministro per la Funzione Pubblica Franco Bassanini [Bassanini1], erano molto ambiziose pur mantenendo scadenze a breve termine (1-2 anni). È stato ad esempio introdotto il concetto di *sportello unico*, ovvero la possibilità di accedere a tutti i servizi della PA recandosi presso gli sportelli di un qualsiasi ente amministrativo od utilizzando un terminale (browser) connesso alla rete Internet. Il cittadino che utilizzi un tale sportello non è tenuto a conoscere nei dettagli l'iter burocratico necessario ad ottenere un determinato servizio. Infatti, attraverso un'unica richiesta di servizio da parte del cittadino, lo sportello deve essere in grado di interpellare le amministrazioni di competenza che, a loro

[1] Nel 2003 si contavano in Italia 42 famiglie su 100 dotate di accesso ad Internet (fonte: [Eurostat]).

[2] Già nel 2003 si registrava in Italia la presenza di 96 abbonamenti di telefonia cellulare ogni 100 abitanti (fonte: [Eurostat]).

volta, provvedono a gestire la pratica, possibilmente scambiandosi servizi ed informazioni sul conto dell'utente[3].

Un altro interessante risultato del Piano di e-Government, connesso con il precedente, è stato il riconoscimento della centralità delle tecnologie di identificazione. Il modello dello sportello unico prevede, infatti, che l'autenticazione del cittadino richiedente sia effettuata in maniera omogenea ed interoperabile presso uno qualsiasi degli sportelli dello Stato, sia esso uno sportello 'reale' od un portale di servizi al cittadino. I dati relativi all'avvenuta autenticazione dovranno poi poter essere forniti a livello di back office alle amministrazioni che effettivamente erogheranno il servizio, in completa sicurezza e nel rispetto della privacy.

Il nodo probabilmente più delicato del Piano è stato trovare il giusto compromesso tra la necessità di uniformare e razionalizzare i processi amministrativi e quella di decentralizzare le competenze, in linea con l'evoluzione in senso federativo dello Stato, tuttora in atto. L'approccio scelto per realizzare un tale "federalismo efficiente" è stato quello di adottare modelli architetturali di impianto cooperativo anziché gerarchico.

Tra gli interventi fattivi previsti dal Ministro Bassanini è interessante citare:

- la realizzazione della "grande extranet della PA", ovvero l'infrastruttura di comunicazione e cooperazione che va ora sotto il nome di Sistema Pubblico di Connettività e cooperazione (SPC)[4];

- maggiore diffusione della Carta di Identità Elettronica (CIE), vista come strumento privilegiato per l'autenticazione sicura del cittadino[5];

[3]Un tale approccio implica l'eliminazione di gran parte dei certificati cartacei. La drastica riduzione del numero dei certificati emessi, dovuta soprattutto all'introduzione dell'autocertificazione, è stata utilizzata dal ministro Bassanini come indicatore della semplificazione in atto nei rapporti cittadino-PA.

[4]Si veda in proposito la sezione 2.1.1.

[5]In realtà, a 5 anni dal varo del Piano, la diffusione delle CIE (meno di 2 milioni di unità) non ha raggiunto ancora cifre tali da poter essere considerata lo strumento **unico** per l'autenticazione.

- l'istituzione di una serie di portali 'concentratori di servizi', come: italia.gov.it (servizi al cittadino), impresa.gov (servizi alle imprese), il "Portale per i servizi all'impiego", ecc.

- la realizzazione di un "Portale per i servizi di certificazione", ovvero uno strumento utilizzabile dalle amministrazioni di front office per verificare le autocertificazioni (o più in generale le credenziali) presentate dal cittadino per accedere ai servizi della PA;

- l'implementazione di specifici scenari applicativi che prevedano la cooperazione di più amministrazioni, come ad esempio quello per la circolarità anagrafica – l'Indice Nazionale delle Anagrafi (INA);

- l'emissione di apposite direttive per l'e-Procurement, ovvero la riduzione della discrezionalità in un settore delicato come quello dell'acquisizione di beni e servizi da parte della PA;

Il 13 maggio 2001 vi è stato il cambio di legislatura (dalla XIII alla XIV), con il contestuale avvicendarsi degli schieramenti al potere (dal centro-sinistra al centro-destra) e la suddivisione delle deleghe dell'ex Ministro Bassanini tra due nuovi Ministri, quello per la Funzione Pubblica[6] e quello per l'Innovazione e Tecnologie[7]. Nonostate questi importanti cambiamenti istituzionali, i principi e le direttive del "Piano d'azione per l'e-Government" sono rimasti sostanzialmente invariati, come testimonia l'avvenuto finanziamento del Piano all'inizio del 2002 [Finanz-e-Gov], ad ulteriore dimostrazione della trasversalità dell'e-Government rispetto alla scena politica.

È bene sottolineare, tuttavia, che il Piano di e-Government ha subito vari cambiamenti in corso d'opera, tra cui la revizione delle priorità di alcuni degli obiettivi. In particolare è stato posto l'accento sulla realizzazione delle infrastrutture di comunicazione – nella fattispecie SPC[8] – piuttosto che sull'implementazione di specifiche applicazioni.

[6] Ruolo inizialmente assunto da Franco Frattini e poi passato a Mario Baccini.
[7] Carica assunta da Lucio Stanca, fino ad allora alto dirigente in IBM
[8] cf. 2.1.1.

Il contributo della tesi

Come già accennato in apertura, questa tesi si concentra sugli aspetti di Identity Management (IM)[9] presenti nelle iniziative di e-Government. In particolare sono stati formalizzati (capitolo 2) i requisiti principali della PA italiana in termini di IM, a partire dall'analisi di un insieme di documenti di specifica, bandi di gara e progetti. In base a tali requisiti è stata svolta un'attività di ricerca (capitolo 3) di soluzioni tecnologiche adatte, tutte conformi al paradigma dell'Open Source, le quali sono state successivamente integrate (capitolo 4) all'interno di una architettura comprensiva. Per dimostrare la validità dell'architettura progettata, infine, è stato realizzato (capitolo 5) un prototipo contenente alcune delle funzionalità più interessanti dell'architettura.

Le tematiche principali affrontate nel lavoro di tesi sono state:

- Il Single Sign On (SSO) via web[10]: lo studio estensivo dei sistemi esistenti e della possibilità di estenderne le funzionalità, l'attitudine del modello federativo introdotto da alcuni di questi sistemi alle specificità dell'e-Government italiano.

- L'autorizzazione ed il controllo di accesso: separazione tra autenticazione ed autorizzazione nelle problematiche di sicurezza avanzate, i modelli di controllo di accesso distribuito (ovvero separazione tra controllo di accesso e decisione di autorizzazione).

- L'analisi di nuove potenzialità introdotte dall'uso della firma digitale come ad esempio l'implementazione elettronica della delega.

- Lo studio delle possibili integrazioni tra le funzioni di sicurezza/IM dei portali di front office e quelle presenti nel back office (SPC).

[9] La definizione di questa problematica è estensivamente trattata nella sezione 1.1.

[10] Esso consiste nella possibilità, per un utente, di accedere a servizi offerti da più siti web (su domini distinti) senza dover reiterare la fase di login. Per una definizione più esaustiva del SSO si veda la sezione 3.2.

- L'utilizzo delle informazioni utilizzate per l'autorizzazione anche a livello applicativo, ad esempio per realizzare profili utente.

Capitolo 1

Nozioni di contesto

Il presente capitolo raccoglie informazioni attinte da varie fonti (trends tecnologici, standards industriali, lavori accademici, ecc.) ed utili a comprendere le argomentazioni presenti nella restante parte della tesi.

Nella sezione 1.1 si definisce l'Identity Management, il 'contenitore tematico' di questa tesi, e se ne inquadrano le possibili varianti all'interno di una semplice tassonomia. La sezione 1.2 descrive uno dei maggiori trends tecnologici del momento: l'utilizzo di tecnologie derivate da quelle del World Wide Web anche nel campo delle applicazioni distribuite di tipo Business to Business (B2B), oltre che per lo sviluppo di siti web dinamici. La sezione 1.3, infine, riassume alcuni dei risultati teorici più interessanti (e più citati nelle fonti utilizzate per la tesi) per quanto riguarda il problema del controllo di accesso.

Laddove possibile, si è cercato di non entrare in dettagli tecnici, ma di privilegiare piuttosto una visione d'insieme.

1.1 L'Identity Management

> “ Identity Management (IM) is an integrated system of business processes, policies and technologies that enable organizations to facilitate and control their users' access to critical online applications and resources – while protecting confidential personal and business information from unauthorized users.[1] ”

L'IM è pertanto una vasta area tematica legata alla gestione dell'identità digitale degli utenti – nel caso dell'e-Business – e dei cittadini – nel caso dell'e-Government. Le principali problematiche afferenti all'IM sono:

- *user provisioning*, allocazione e revoca di privilegi, profili utente;
- delega di funzioni amministrative o di privilegi e profili utente;
- autenticazione;
- controllo di accesso ed autorizzazione;
- formato ed interoperabilità delle identità digitali;
- Single Sign On (SSO);
- servizi di *directory*;
- gestione della privacy;
- gestione del *trust*.

Inoltre, il termine IM viene spesso associato a sistemi informatici che integrano (anche parzialmente) al proprio interno le funzionalità appena elencate[2].

[1] Fonte: [Wikipedia], http://en.wikipedia.org/wiki/Identity_management.

[2] Ad esempio: Oracle Identity Management, Sun Java System Identity Manager, Novell Nsure Identity Manager, Entrust Secure Identity Management, Computer Associates eTrust Identity and Access Management Suite, ecc.

1.1.1 Definizione di identità digitale

Dei concetti di *identità digitale* (*digital identity*), *identità di rete* (*network identity*) o *identità elettronica* (*electronic identity*) esistono molteplici interpretazioni[3], spesso contrastanti. Nel corso di questa tesi si userà l'accezione di identità digitale come insieme delle informazioni in rete afferenti ad un unico soggetto, in base alla definizione data in [Liberty-web][4]:

> “ Network identity refers to the global set of attributes that are contained in an individual's various accounts with different service providers. These attributes include such information as name, phone numbers, social security numbers, addresses, credit records, and payment information. ”

1.1.2 Tassonomia dell'Identity Management

Secondo [TRIM], i sistemi (o i modelli) di IM possono essere classificati in base alla specifica 'filosofia' di gestione del trust:

IM isolato. Corrisponde al modello di IM attualmente più diffuso. Ogni servizio ha un proprio bacino d'utenza indipendente ed ad ogni utente viene assegnata una credenziale distinta per ogni servizio a cui fa accesso. Questo approccio semplifica l'IM per i Service Provider (SP), ma presenta seri problemi di usabilità per gli utenti all'aumentare dei servizi utilizzati.

Le istanze di trust coinvolte in questo modello sono piuttosto semplici:

T1 Il SP tutela la privacy dell'utente.

T2 Il SP ha implementato procedure di registrazione e meccanismi di autenticazione adeguati.

[3] Ad esempio: sociologiche, psicologiche, filosofiche, tecnologiche, giuridiche, ecc. Esistono inoltre diverse interpretazioni del termine in ambito prettamente tecnologico, corrispondenti ad altrettanti modelli o visioni dell'identità digitale.

[4] http://www.projectliberty.org/about/faq.php#05.

T3 Il client gestisce responsabilmente le credenziali ricevute dal SP.

IM federato. La *federazione dell'identità* (*identity federation*)[5] si può definire come l'insieme di tecnologie, standards ed accordi che permettono ad un insieme di SP di accettare come validi gli identificatori utente gestiti da un'altro insieme (non necessariamente distinto) di providers, detti Identity Provider (IdP). Una tale comunità di providers (SP ed IdP) viene tipicamente denominata *federazione* (*federation*)[6] o *circle of trust*[7].

La federazione dell'identità viene realizzata collegando[8] i diversi identificatori utilizzati dai providers della federazione e relativi ad uno stesso utente. Un tale approccio implementa implicitamente il Single Sign On (SSO), ovvero la possibilità per un utente di autenticarsi presso uno qualsiasi dei providers della federazione e, successivamente, di accedere ai servizi di tutti gli altri.

Questo approccio all'IM introduce nuove topologie di trust, ed in particolare la presenza di accordi di trust tra providers. Le istanze T1, T2 e T3 valgono anche in questo caso, ma in questo caso T1 e T2 si applicano ai providers che fungono da IdP. I clients ed i providers devono inoltre supportare le seguenti istanze:

T4 All'origine di una richiesta di servizio (risp. di identificazione) di un IdP (SP) nei confronti di un SP (IdP) vi deve sempre essere una esplicita richiesta dell'utente.

T5 La corrispondenza tra gli identificatori collegati in una federazione deve essere corretta, ovvero essi devono far tutti riferimento ad una stessa identità digitale.

[5] Gli standards e le tecnologie che utilizzano questo modello sono trattati estensivamente nel capitolo 3, sezioni 3.1.1 e 3.2.

[6] Termine utilizzato in SAML (cf. 3.1.1) e Shibboleth (cf. 3.2.1).

[7] Termine utilizzato in Liberty Alliance (cf. 3.2.2).

[8] In molti contesti si utilizza il verbo *federare* per indicare questa operazione.

T6 La circolarità tra i providers delle informazioni sull'utente deve essere regolata da opportune policies accettate da tutte le parti al momento della federazione dell'identità.

IM centralizzato. Questo modello di IM è costituito essenzialmente da un unico IdP che si occupa di identificare gli utenti per conto di una molteplicità di SP[9]. Le informazioni costituenti l'identità digitale di un utente possono anche in questo caso essere distribuite tra i providers, ma l'identificatore ad essa associato è unico e gestito dall'IdP. Come il precedente, anche questo modello permette il SSO.

Le istanze di trust applicabili all'IM centralizzato sono T1, T2, T3 e T6, di cui le prime due sono da applicarsi all'unico IdP.

IM personale. I problemi di usabilità del modello di IM isolato possono essere risolti tramite l'utilizzo di un dispositivo, hardware o software, in grado di immagazzinare in maniera sicura le molteplici credenziali possedute dall'utente. In [TRIM] tale dispositivo viene denominato Personal Authentication Device (PAD)[10].

Un tipico PAD prevede una procedura di abilitazione, tramite l'immissione di un PIN, di una master password o l'acquisizione di dati biometrici. Successivamente è possibile accedere, per un certo periodo di tempo, ai dati o alle funzionalità in esso immagazzinati. Alcuni PAD possono interoperare con i terminali di accesso ai servizi forniti dai SP. In quest'ultimo caso i PAD permettono una forma di SSO detta *virtuale*: una volta abilitato, il PAD abilita l'accesso ai servizi senza ulteriori interazioni di autenticazione da parte dell'utente.

[9] Alcuni dei sistemi che seguono questo modello: Kerberos [RFC1510], Microsoft .NET Passport [MSPassport] e Yale Central Authentication Service [Yale CAS-web].

[10] Esempi di PAD software: la funzionalità Password Manager del browser Mozilla (e derivati), il *Keychain* in Mac OS X ed analoghe funzionalità in altri sistemi operativi o applicativi ad-hoc. Esempi di PAD hardware: *smart cards*, *hardware tokens*, generatori di *one time passwords* come RSA SecurID, ecc.

L'unico vincolo di trust imposto dall'utilizzo di un PAD (oltre a T1, T2, T3 'ereditati' dal modello di IM isolato) è il seguente:

T7 Il PAD può essere utilizzato dalla sola persona associata all'identità digitale accessibile attraverso di esso (*tamper-resistance*).

1.1.3 L'Identity Management e l'e-Government

Se l'IM è fondamentale per l'e-Business, lo è allora a maggior ragione per l'e-Government. Lo Stato, attraverso le proprie istituzioni, è infatti il garante primario dell'identità dei cittadini, sia 'fisica' che digitale. È pertanto naturale ed auspicabile l'introduzione di infrastrutture di IM su larga scala (nazionale ed internazionale), che permettano ai cittadini l'accesso ai servizi di e-Government, in sicurezza e nel rispetto della privacy. È altresì desiderabile poter estendere la portata di tali infrastrutture al di là del confine di competenza dei governi, permettendone l'accesso a generici fornitori di servizi di e-Business (anche privati), ovviamente con le dovute garanzie e restrizioni.

È bene precisare che, oltre agli aspetti puramente tecnici, l'applicazione dell'IM all'e-Government necessita del supporto di opportuni interventi legislativi e normativi, che vanno oltre l'ambito di competenza di questa tesi. Sono infatti caratterizzate da tale duplice natura (tecnico/normativa) molte delle iniziative di IM nell'e-Government in atto, sia in ambito nazionale[11] che europeo[12] ed internazionale[13].

[11] Si pensi ad esempio alle iniziative del CNIPA [CNIPA-web] o del CISIS [CISIS-web], alcune delle quali sono descritte in maggior dettaglio nel capitolo 2.

[12] Vanno citati, in ambito europeo, i progetti: PRIME [PRIME-web], FIDIS [FIDIS-web][FIDIS-Models][FIDIS-IMS] e GUIDE [GUIDE-web].

[13] Di particolare rilievo in ambito internazionale è l'iniziativa "E-Authentication" del governo statunitense [E-Auth-web][E-Auth-tech].

1.2 Il passaggio a tecnologie basate su XML e Web Services

Un trend tecnologico degli ultimi anni, di forte impatto sul mondo dell'e-Business (e di conseguenza dell'e-Government), è l'adozione di XML come formato universale per i dati e del paradigma dei *Web Services* per lo sviluppo di applicazioni distribuite. Tale trend è in linea con l'evoluzione del Web da rete puramente informativa a rete di servizi, in atto da circa un decennio.

1.2.1 XML

eXtensible Markup Language (XML) [XML] è un linguaggio di *markup* generico che funge, allo stesso tempo, da linguaggio e metalinguaggio per descrivere documenti strutturati. Il meccanismo che permette di includere metadati contestualmente ai dati stessi è quello del markup, ovvero la possibilità di fare annotazioni semantiche o strutturali.

Per XML si è scelta una rappresentazione 'basata sul testo': un documento XML è una sequenza di **caratteri** anziché di **bit** come avviene per molti altri formati di rappresentazione di dati. La natura testuale di XML ne garantisce l'interoperabilità, permettendo la comunicazione e l'immagazzinamento di documenti XML utilizzando praticamente ogni tipo di supporto, ivi compresa la carta. Detta caratteristica agevola inoltre la lettura di documenti XML da parte di esseri umani dato che, in linea generale, non è necessario l'utilizzo di appositi strumenti software per decodificarne i contenuti.

Uno svantaggio della natura testuale di XML è invece la sua prolissità se confrontato con equivalenti formati di rappresentazione binari. Tuttavia, allo stato attuale dell'evoluzione tecnologica, ciò non è generalmente considerato uno handicap. Si può anzi affermare che la disponibilità di supporti di memorizzazione e di canali di trasmissione sempre più capienti, ad un costo sempre più ridotto, ha favorito la diffusione di XML in tempi recenti, nonostante esso si basi su concetti in discussione già da decadi all'interno della comunità scientifica.

Raramente per un documento XML viene utilizzata la forma generica standardizzata in [XML]. Più spesso si fa uso di *sottolinguaggi* specifici[14] derivati da XML tramite l'aggiunta di vincoli sintattici espressi da metalinguaggi come Document Type Definition (DTD) o XML Schema[15] [XML-Schema]. Un documento che rispetti lo standard [XML] viene detto *ben formato* (*well-formed*); se esso in aggiunta è conforme alle specifiche di un sottolinguaggio specifico si dice *valido* rispetto allo *schema* di quel sottolinguaggio, espresso secondo uno dei formalismi succitati.

Questa distribuzione dei requisiti sintattici su più livelli (testo – XML – schema) permette di (ri)utilizzare gli stessi parsers generici per i documenti basati su XML, ai quali verranno forniti in input, oltre ai documenti stessi, gli specifici schemi in base ai quali si vuole verificarne la validità.

1.2.2 Il paradigma dei Web Services

Un *Web Service* è un'applicazione software in grado di comunicare con altre applicazioni tramite scambi di documenti XML[16] attraverso protocolli Internet standard[17] detti *protocolli di trasporto*. L'interfaccia di un Web Service è tipicamente descritta tramite documenti Web Services Description Language (WSDL) [WSDL 1.1]. Come schematizzato in figura 1.1, una tale descrizione formale permette ad un client generico (non specializzato per l'interrogazione di uno specifico Web Service) di:

1. interrogare un apposito *registry* o *broker di servizi* specificando la tipologia di servizio a cui vuole accedere ed altre caratteristiche accessorie come la locazione (ente erogatore), vincoli di Quality of Service (QoS), requisiti di sicurezza, ecc.;

2. in risposta a tale interrogazione, ottenere un riferimento (una URI)

[14] Ad esempio: XHTML, RDF, SOAP, WSDL, ecc.

[15] Questo è a sua volta un sottolinguaggio di XML.

[16] Tipicamente, ma non necessariamente, tali documenti saranno incapsulati all'interno di *buste* (*envelopes*) SOAP [SOAP].

[17] Di norma si usa HTTP/HTTPS e più raramente SMTP.

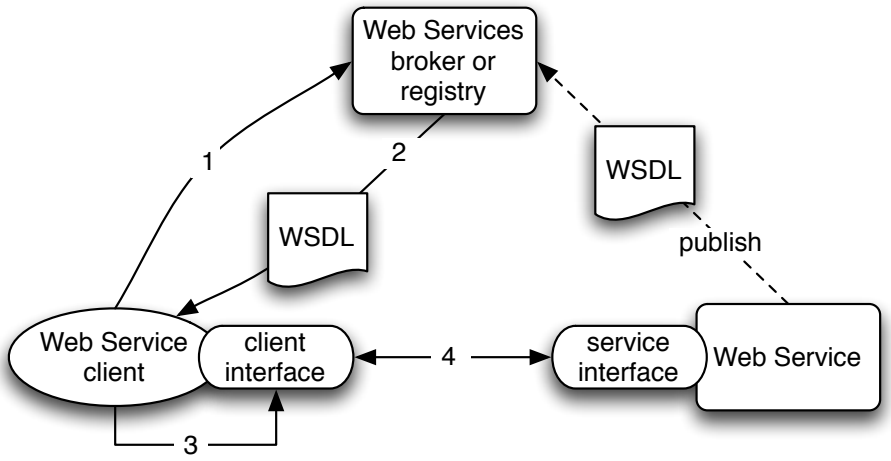

Figura 1.1: Schema di una tipica interazione client-server secondo il paradigma dei Web Services.

al Web Service del tipo richiesto e la descrizione formale della sua interfaccia (in WSDL o altri linguaggi analoghi);

3. adattare automaticamente la propria interfaccia, in modo da poter interrogare il particolare Web Service ottenuto al passo 2;

4. interrogare il Web Service.

Si noti che il modulo software che assume il ruolo di client può a sua volta essere un Web Service. L'utilizzo dei Web Services permette quindi di realizzare architetture distribuite basate su componenti *loosely joined*, ovvero facilmente componibili ed adattabili alle specifiche applicazioni. Tali architetture si dicono conformi al modello Service Oriented Architecture (SOA) [WSSOA].

Uno dei maggiori vantaggi del paradigma dei Web Services è la sua indipendenza dal linguaggio di programmazione dei componenti e più in generale dalla piattaforma software che li ospita. Un'altra caratteristica, molto gradita nel mondo e-Business, è la possibilità di utilizzare HTTP/HTTPS

come trasporto, permettendo in questo modo di attraversare molti firewalls aziendali[18].

1.2.3 Sicurezza nei Web Services

Nei tipici scenari di e-Business è richiesta l'implementazione di misure per garantire la sicurezza delle informazioni scambiate utilizzando tecniche crittografiche. Tali misure sono classificabili in due categorie[19]:

1. autenticità della fonte, integrità e non ripudiabilità;
2. confidenzialità.

Molti dei protocolli di trasporto utilizzati dai Web Services permettono di garantire entrambe le caratteristiche di sicurezza ai singoli scambi tra i clients ed i servizi. Tipicamente, tuttavia, le interazioni all'interno di una SOA sono più complesse e possono coinvolgere più di due componenti (architetture *multi-tier*). In tal caso, quindi, le proprietà di sicurezza dei messaggi (o di parti di essi) devono poter essere mantenute lungo tutto il corso dell'interazione complessa.

Si rende pertanto necessario affiancare ad un modello di sicurezza *hop-to-hop*, implementabile a livello di trasporto[20], anche un modello *end-to-end* che permetta di garantire integrità e confidenzialità di (parti di) messaggi, indipendentemente dal numero di moduli che ne realizzano il recapito.

1.2.3.1 Integrità e confidenzialità in documenti XML

Gli standards emessi dal World Wide Web Consortium (W3C) per la firma digitale (integrità) e la criptazione (confidenzialità) di parti di documenti XML sono, rispettivamente, XML Signature [XMLSig] e XML Encryption [XMLEnc]. I due standards permettono di applicare le rispettive funzioni

[18] Essi sono infatti tipicamente impostati per far passare il solo traffico HTTP su porta 80 e HTTPS su porta 443.

[19] cf. anche [Crypto]

[20] Ad esempio tramite SSL/TLS.

criptografiche selettivamente a parti (elementi) di un documento XML e, al contempo, includere (riferimenti a) le chiavi utili alla decodifica/validazione nel documento stesso.

Dato che i Web Services si basano sullo scambio di documenti XML, è possibile utilizzare XML Signature ed XML Encryption per implementare meccanismi di sicurezza avanzati, basati sulle proprietà di sicurezza del messaggio stesso e non del canale di comunicazione (il trasporto).

È in particolare possibile un modello di interazione complessa in cui un Web Service 'iniziale' crei un messaggio proteggendone l'integrità e/o la confidenzialità, globalmente o solo in alcune sezioni considerate 'critiche'. Successivamente tale messaggio potrà essere comunicato ad altri Web Services intermedi che potranno alterarne il contenuto, ad esempio arricchendolo con i risultati di un proprio processamento. Il Web Service 'terminale' dell'interazione potrà verificare che i nodi intermedi non abbiano alterato le sezioni critiche create dal nodo iniziale validandone la firma digitale (come specificato in XML Signature). Analogamente, l'utilizzo di XML Encryption garantisce che solo il Web Service terminale (ed eventualmente un sottoinsieme dei nodi intermedi) possa leggere le informazioni presenti nelle sezioni critiche del messaggio.

XML Signature ed XML Encryption, tuttavia, non trovano applicazione solo nel contesto dei Web Services, ma anche nella gestione di documenti XML in senso proprio. Seguendo il trend tecnologico che sta portando all'utilizzo estensivo di XML (e suoi derivati) come formato di rappresentazione dei dati, XML Signature ed XML Encryption sono probabilmente destinati a sostituire standards analoghi come PKCS#7 [RFC2315].

1.2.3.2 Web Services Security

Lo standard Web Services Security (WSS) [WSS] di Organization for the Advancement of Structured Information Standards (OASIS) estende ulteriormente le possibilità di sicurezza dei messaggi offerte da XML Signature ed XML Encryption. WSS specifica infatti:

- come includere generici *tokens*[21] di sicurezza all'interno di messaggi SOAP[22];
- come utilizzare tali tokens per firmare o criptare parti del messaggio utilizzando XML Signature ed XML Encryption.

I dettagli su come includere particolari tipi di tokens non sono specificati in [WSS], ma sono relegati a specifici *profili per WSS*, come ad esempio il "SAML Token Profile for WSS", descritto nella sezione 3.1.1.5.

1.2.4 Le applicazioni web

Un'*applicazione web* è un'applicazione client-server accessibile tramite un comune web browser, la cui logica applicativa risiede per la maggior parte nel lato server[23], implementato da un *application server*. Gli strumenti tecnologici che permettono di realizzare applicazioni web, come ad esempio la Common Gateway Interface (CGI), sono stati creati con l'intento iniziale di dotare i siti web, all'epoca puramente ipertestuali, di una qualche forma di interattività, ad esempio per implementare motori di ricerca o, più in generale, sottomissioni di dati da parte dell'utente.

Successivamente, data la diffusione capillare dei browsers web su tutte le maggiori piattaforme desktop, si è avviata la tendenza a realizzare un più ampio spettro di applicazioni client-server sotto forma di applicazioni web, ad un punto tale da arrivare a soppiantare applicativi ad-hoc precedentemente molto più diffusi[24].

Essendo basate su un protocollo (HTTP) basato sul ciclo richiesta/risposta, le applicazioni web hanno modalità di iterazione da esso influenzate:

[21] Ad esempio: certificati X.509 (cf. 3.1.2), asserzioni SAML (cf. 3.1.1), Kerberos tickets [RFC1510], ecc.

[22] I messaggi o *buste* (*envelopes*) SOAP [SOAP] sono il formato di messaggio più utilizzato in ambito Web Services. SOAP è un formato derivato da XML.

[23] Le applicazioni web seguono quindi il modello *thin client*.

[24] Si pensi ad esempio ai clients di posta elettronica.

- l'utente accede ad una pagina (*vista*) dell'applicazione web che può offrire modalità di interazione;
- l'utente fornisce il proprio input che viene inviato dal browser all'application server;
- l'application server presenta all'utente la vista successiva in funzione dell'input ricevuto;
- ...

Le modalità di interazione delle applicazioni web sono quindi più limitate rispetto a quelle possibili in una generica applicazione desktop. In particolare non è possibile implementare interazioni real-time[25], forme di input non testuale[26] o computazioni dal lato client[27]. Tali limiti possono essere parzialmente risolti attraverso l'utilizzo di tecnologie di *client-side scripting* come ad esempio JavaScript. Un altro approccio per superare tali limitazioni è quello di integrare applicazioni generiche all'interno di pagine web, utilizzando appositi plug-ins disponibili per i browsers più diffusi[28]. Tale approccio annulla però in parte i vantaggi del modello thin client portando le applicazioni web in una situazione simile a quella preesistente, ovvero caratterizzata da applicazioni client sviluppate ad-hoc per usufruire di un determinato servizio.

1.2.4.1 Integrazione con i Web Services

Le applicazioni web non sono classificabili come Web Services se non estendendo la definizione di questi ultimi. In particolare i dati scambiati tra il browser ed una applicazione web non sono in formato XML, ma HTML[29].

[25] Si pensi a: videogiochi, controllo a distanza, ecc.

[26] Ad esempio è estremamente difficile realizzare applicazioni di grafica.

[27] Un utilizzo di questa funzionalità potrebbe essere l'esecuzione di funzioni criptografiche per conto dell'utente che per ovvi motivi di sicurezza non possono essere delegate all'application server.

[28] Ad esempio: Sun Microsystems Java Applets [Applets] o Macromedia Flash [Flash].

[29] In realtà è in atto un processo di sostituzione di HTML con un sottolinguaggio di XML ugualmente espressivo: eXtensible HyperText Markup Language (XHTML).

Inoltre lo sviluppo di applicazioni web prevede la gestione di problematiche di presentazione che invece nei Web Services sono assenti, dato che questi ultimi non sono pensati per l'iterazione diretta con l'utente.

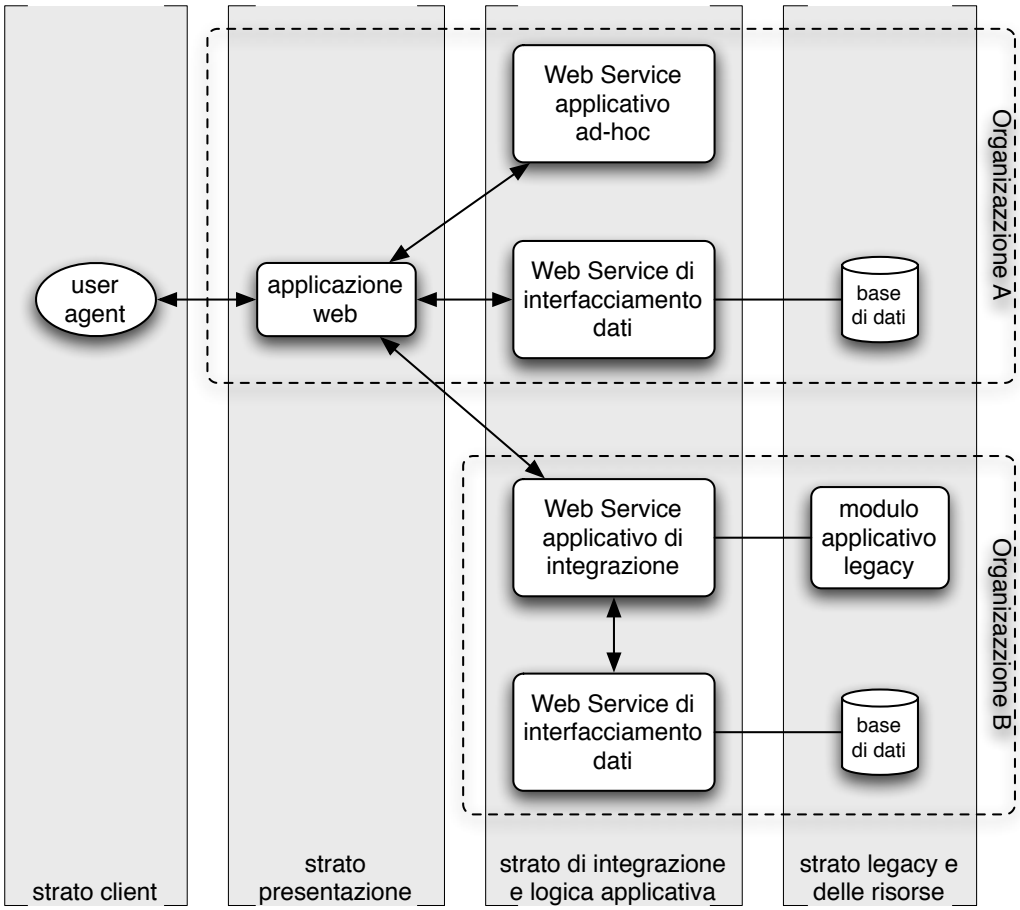

Figura 1.2: Esempio di architettura risultante dall'integrazione di Web Services ed applicazioni web.

Nelle applicazioni web di complessità medio/alta, tuttavia, i Web Services vengono tipicamente utilizzati nel backend a supporto dell'application server. Come illustrato dallo schema architetturale in figura 1.2, l'utilizzo dei Web Services nella realizzazione di applicazioni web permette di:

- incapsulare la logica applicativa, possibilmente distribuendola in vari componenti cooperanti;

- incapsulare basi di dati o componenti applicativi legacy[30], fornendo all'esterno un'interfaccia dinamica e facilmente componibile (come visto nella sezione 1.2.2);
- separare le funzioni di presentazione, gestite all'interno dell'application server, da quelle di logica applicativa, gestite dai Web Services.

[30]Da intendersi come: non conformi al paradigma dei Web Services.

1.3 Modelli per il controllo di accesso e l'autorizzazione

Questa sezione è una raccolta, non esaustiva né omogenea, di concetti relativi ad un tema centrale nell'IM: il controllo di accesso. Essa ha lo scopo di introdurre alcuni dei termini utilizzati nella restante parte della tesi ed in particolare nei capitoli 3 e 4.

Lo scopo del controllo di accesso è quello di combattere la minaccia dell'uso improprio di risorse critiche, come ad esempio:

- uso non autorizzato delle risorse;
- accesso, modifica o distruzione di informazioni sensibili;
- Denial of Service (DoS).

Esso si compone di due tipologie di attività: l'amministrazione della sicurezza, tipicamente svolta in un momento precedente all'accesso, e l'insieme di operazioni e scambi informativi innescati dal tentativo di accesso, che portano ad una prestazione di servizio nel caso in cui questo sia autorizzato o alla negazione dello stesso in caso contrario.

I meccanismi di base per garantire un controllo di accesso efficace sono principalmente tre:

Autenticazione: le parti in gioco devono essere identificate in maniera (ragionevolmente) certa;

Autorizzazione: affinché il tentativo di accesso abbia successo, è necessario che regole chiare attestino che il soggetto richiedente è autorizzato ad effettuare l'operazione richiesta su una data risorsa;

Auditing e Accounting: gli accessi a risorse sensibili devono poter essere indagati anche a posteriori (è pertanto necessario registrarli); deve inoltre essere possibile risalire ai responsabili degli accessi e, più in generale, a tutti i soggetti in essi coinvolti.

1.3.1 RFC 2904: Authorization Framework

Il documento [RFC2904], creato dall'AAAarch Research Group di Internet Engineering Task Force (IETF), fornisce un insieme di patterns architetturali per la realizzazione di servizi di Authentication, Authorization and Accounting (AAA) distribuiti.

Il modello di controllo di accesso previsto in [RFC2904] è quello in cui un **utente** desideri accedere ad una risorsa, detta **Service Equipment (SE)**, protetta dall'**AAA Server** del SP erogatore. Nell'effettuare il controllo di accesso, il SP può aver bisogno della partecipazione della **User Home Organization (UHO)**, responsabile per l'utente. Il diagramma in figura 1.3 mostra i principali attori in gioco nel modello ed i rispettivi accordi di trust e di servizio.

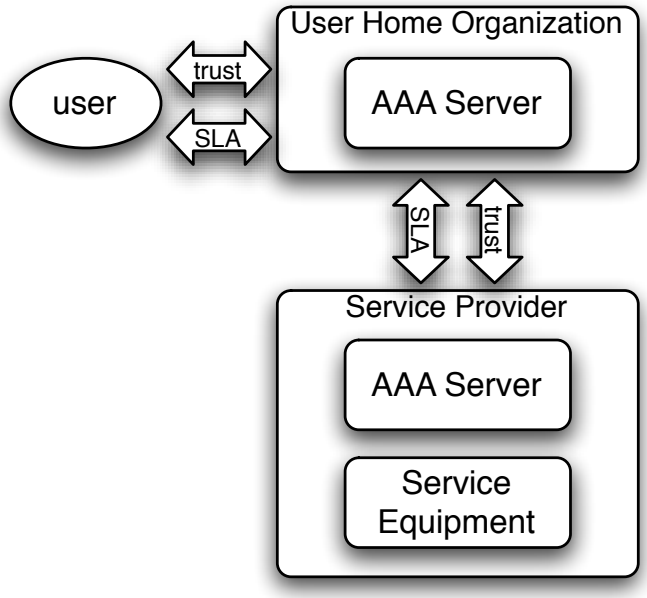

Figura 1.3: Macro-componenti ed accordi di servizio.

L'autorizzazione dell'utente si fonda su tali accordi di trust tra il dominio responsabile dell'erogazione del servizio (il SP) e quello responsabile per l'utente (la UHO). Quest'ultimo può partecipare attivamente, con funzioni

di mediatore, alle interazioni di autorizzazione. A seconda del numero di domini coinvolti, [RFC2904] individua tre casistiche distinte:

- Single Domain Case: la UHO non viene coinvolta nelle interazioni od è inglobata nel SP.
- Roaming: la UHO è distinta dal SP e partecipa all'autorizzazione.
- Distributed Services: è il caso più generale, in cui il servizio richiesto dall'utente è il risultato della composizione di servizi erogati da più SP.

1.3.1.1 Single Domain

Gli attori coinvolti in questo caso sono utente, l'AAA Server del SP ed il SE del SP. [RFC2904] considera tre possibili sequenze di interazioni, denominate *agent*, *pull* e *push*.

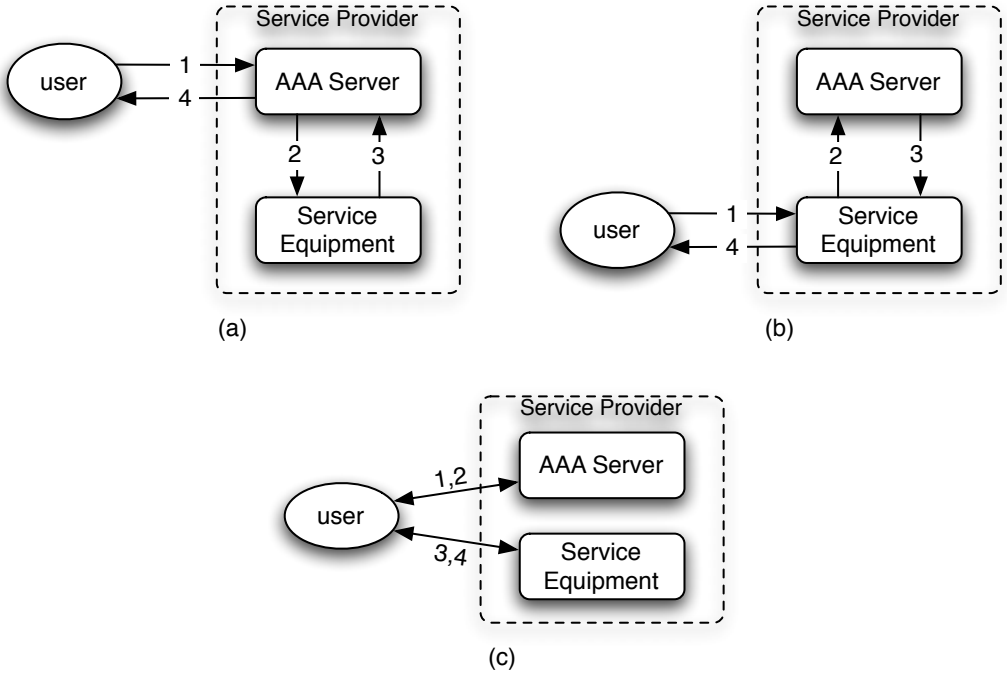

Figura 1.4: Sequenze 'base' di interazioni di autorizzazione secondo [RFC2904]: *agent* (a), *pull* (b) e *push* (c).

Agent Sequence. Nella sequenza *agent*, illustrata in figura 1.4.a, l'AAA Server funge da intermediario (agente) tra l'utente ed il SE. L'AAA Server riceve la richiesta di servizio dell'utente (1) e, dopo averlo autorizzato, la instrada verso il SE (2). Ricevuta la risposta dal SE (3), l'AAA Server la invia all'utente (4).

Pull Sequence. Nella sequenza *pull* (fig. 1.4.b) l'utente interagisce direttamente con il SE, come se non vi fosse autorizzazione. È il SE che, una volta ricevuta una richiesta dall'utente (1), provvede ad interrogare il servizio di AAA (2) per ottenerne l'autorizzazione. L'AAA Server effettua la decisione di accesso e ne comunica l'esito al SE (3). In caso di esito positivo, il SE può ora fornire il servizio richiesto all'utente (4).

Push Sequence. La sequenza *push* (fig. 1.4.c) prevede che sia l'utente (per l'esattezza lo *user agent*) a trasportare l'autorizzazione dal decisore (l'AAA Server) all'erogatore (il SE). Ciò avviene attraverso una prima interazione con l'AAA Server (1) attraverso la quale l'utente ottiene un *token*[31] (2) che gli garantisce l'accesso al servizio. In un istante successivo, ma non necessariamente correlato alle interazioni (1,2), l'utente può fare uso di tale token (anche più di una volta) per accedere ai servizi del SE (3,4).

1.3.1.2 Roaming

In molti scenari applicativi avanzati il controllo di accesso è un'operazione complessa che richiede la collaborazione di più domini di competenza. Il caso del *roaming* prevede la distribuzione delle competenze di autorizzazione tra la UHO ed il SP. La prima, infatti, ha una maggiore conoscenza dell'utente e può quindi basare la sua (parte di) decisione di accesso su informazioni sul suo conto non disponibili (ad es. per motivi di privacy) al SP. D'altro canto

[31] Da intendersi nell'accezione generica di documento asserente l'attitudine dell'utente a fruire del servizio. Per garantirne la non falsificabilità tale documento dovrà essere firmato (elettronicamente) dall'AAA Server.

il SP dispone di tutte le informazioni relative ai SE di propria responsabilità e le può utilizzare per effettuare la propria (parte di) autorizzazione.

Come descritto in figura 1.5, le sequenze di interazioni *agent*, *pull* e *push* relative al caso Single Domain sono generalizzabili al caso Roaming. Per semplificare la trattazione di tali 'sequenze generalizzate', l'AAA Server ed il SE del SP sono stati accorpati. Si noti tuttavia che nella pratica sarà uno dei due componenti del SP (od entrambi) a gestire le singole interazioni con la UHO e l'utente.

1.3.1.3 Distributed Services

In questo caso, il servizio richiesto dall'utente può essere il risultato della cooperazione di più SE locati presso SP distinti, ciascuno con una propria autorità in termini di autorizzazione. Per i Distributed Services si possono combinare gli stessi schemi visti nei casi precedenti, risultanti in molti sotto-casi possibili. Le relazioni 'contrattuali' tra i domini, nel caso di due providers, sono riportate in figura 1.6.

1.3.1.4 Policies per l'autorizzazione

Un altro aspetto importante delle problematiche di controllo di accesso è la rappresentazione delle regole che determinano le decisioni di accesso effettuate dagli AAA Servers dei vari providers. L'insieme di tali regole viene spesso denominato *policy*.

Nelle architetture di sicurezza ed IM complesse, come quelle trattate in questa tesi, avere delle policies separate dal resto della business logic è senz'altro una caratteristica desiderabile. Secondo [RFC2904], infatti, l'autorizzazione può essere vista come il risultato della valutazione delle policies di tutte le organizzazioni che hanno interesse nella decisione. Al fine di implementare una tale visione 'policy-centric' è necessario un framework che permetta di:

- recuperare le policies da opportune basi di dati;

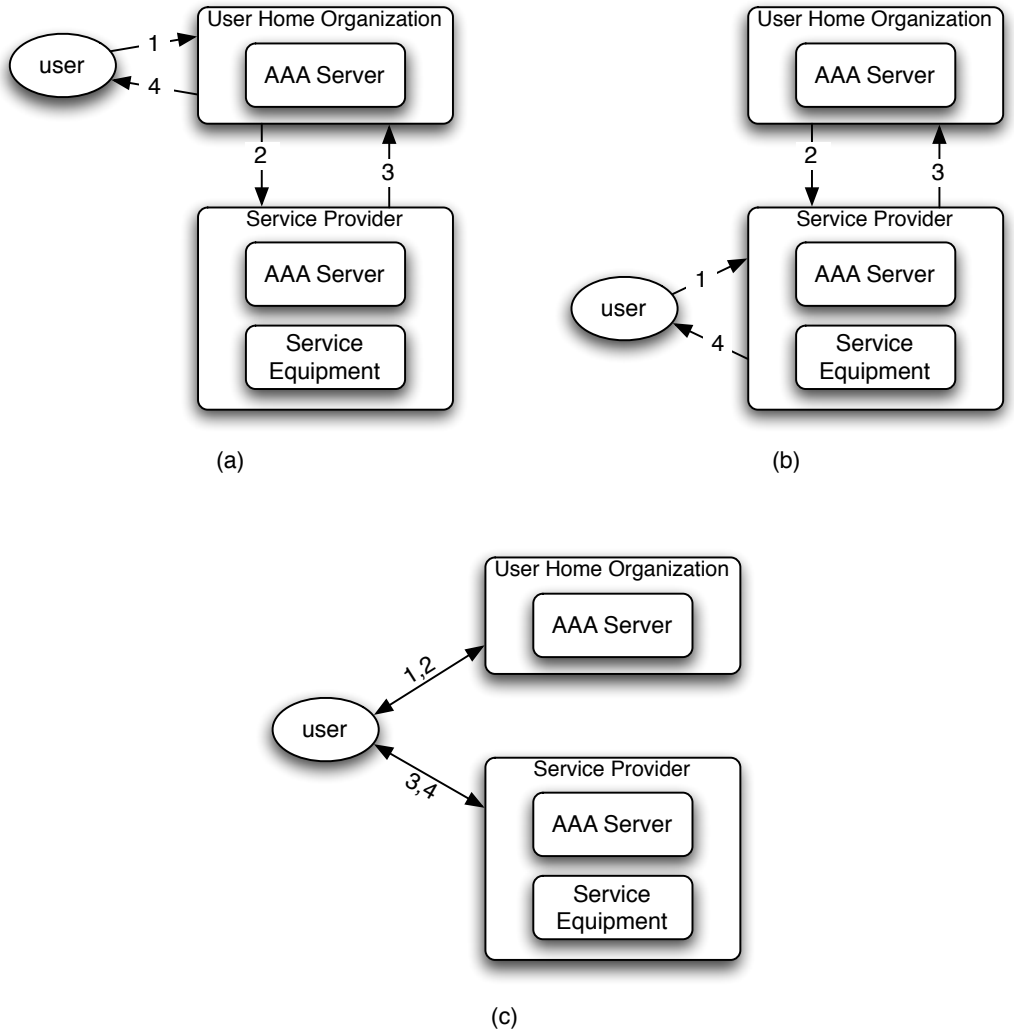

Figura 1.5: Sequenze di interazioni di autorizzazione generalizzate al caso Roaming: *agent* (a), *pull* (b) e *push* (c).

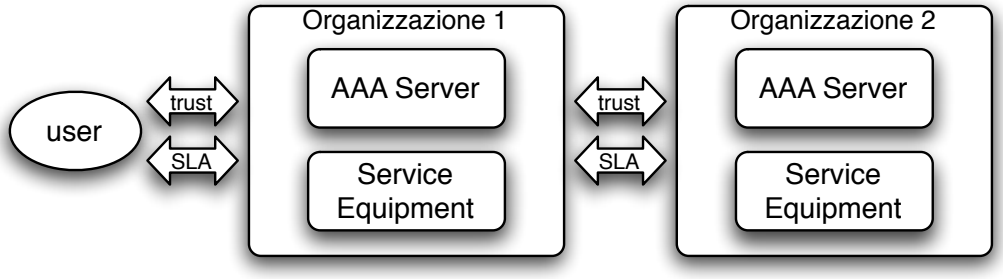

Figura 1.6: Macro-componenti ed accordi di servizio nel caso dei *Distributed Services*.

- valutare tali policies per ottenere una decisione di autorizzazione;
- applicare la decisione ottenuta alla particolare istanza di accesso;
- specificare come le policies possano essere distribuite tra più domini.

[RFC2904] non specifica un linguaggio particolare per definire le policies, in quanto è una competenza del Policy Working Group di IETF, si veda in proposito [RFC3060].

Policy Retrieval. Questa funzione è a carico dell'organizzazione che necessita della policy ed è svolta da un componente logico denominato Policy Retrieval Point (PRP). Esso si occupa di indicizzare le policies in base al richiedente (l'utente o la UHO), al servizio o ad altri tipi di chiave e di scegliere la policy pertintente alla specifica decisione di accesso.

Policy Evaluation. La funzione di autorizzazione vera e propria è svolta dal Policy Decision Point (PDP) è può essere effettuata in più domini distinti. La valutazione della policy necessita delle informazioni determinanti l'esito dell'accesso come lo stato della risorsa richiesta, l'identità dell'utente o condizioni ambientali (ora di accesso, locazione, route, ecc.).

Il PDP può servirsi di un altro componente, il Policy Information Point (PIP), per ottenere tali informazioni da opportune fonti, in maniera sincrona (pull) od asincrona (popolazione per eventi). In alternativa, l'organizzazione

che necessita della decisione può inviare la propria policy e farla valutare direttamente ai PDP delle organizzazioni che detengono le informazioni necessarie.

Policy Enforcement. L'attuazione della decisione emessa dal/dai PDP è tipicamente effettuata dall'organizzazione che protegge i SE, ovvero il SP, o dai SE stessi. Il componente logico che svolge tale funzione viene denominato Policy Enforcement Point (PEP).

Modello di distribuzione dei componenti. I vari componenti logici qui definiti (PRP, PDP, PIP, PEP) possono essere distribuiti in vari punti dell'architettura generale descritta in figura 1.3 nonché replicati in più istanze, eccezion fatta per il PEP. Tali componenti sono gli elementi costitutivi dell'AAA Server, ma possono essere integrati anche all'interno dello user agent e dei SE.

1.3.1.5 Altri argomenti trattati in [RFC2904]

[RFC2904] copre altri temi relativi al controllo di accesso che non sono tuttavia rilevanti per questa tesi o che sono trattati più estensivamente in altri documenti analizzati. In sintesi:

Utilizzo di Attribute Certificates (ACs) Il documento consiglia l'utilizzo di AC X.509 (successivamente standardizzati in [RFC3281]) per convogliare gli attributi in funzione dei quali la decisione di accesso viene valutata dai PDP. Si tratta in sostanza di documenti attestanti qualità o credenziali di un soggetto (ad es. l'utente) e firmati da un'autorità riconosciuta in analogia, quindi, ai Public Key Certificate (PKC) X.509, attestanti l'identità. Gli AC possono essere forniti (modalità push) dal richiedente al PDP o recuperati (modalità pull) dai PIP interrogando opportune basi di dati (tipicamente LDAP). Si veda in proposito anche la sezione 3.1.2.

Resource Management In molte applicazioni il risultato dell'autorizzazione è un servizio che si protrae per un certo lasso di tempo, detto *sessione*. Ogni AAA Server può mantenere una propria sessione e modificarne lo stato col procedere delle interazioni. Il componente logico addetto a tali mansioni viene denominato Resource Manager (RM). Tra le funzioni aggiuntive del RM è interessante citare la possibilità di comunicare con i RMs degli altri AAA Servers, ad esempio per notificare la terminazione della sessione od altre modifiche al suo stato.

Sicurezza end-to-end Nei casi in cui due AAA Servers comunichino tramite un terzo AAA Server intermediario, come ad esempio nel caso Distributed Services (sezione 1.3.1.3), è necessario assicurare caratteristiche di sicurezza end-to-end anziché solo hop-to-hop ai messaggi scambiati. Deve essere possibile inoltre per gli AAA Servers intermedi 'appendere' informazioni ai messaggi in transito senza invalidarne la sicurezza. La problematica è analoga a quella già vista nel contesto dei Web Services nella sezione 1.2.3.

1.3.2 ISO 10181-3: Access control framework

Lo standard "Security frameworks for open systems: Access control framework" [X.812] è un lavoro congiunto di ISO/IEC[32] ed ITU-T[33]. Esso fa parte di una suite[34] di standards per la sicurezza che coprono, oltre al controllo di accesso, problematiche come: l'autenticazione[35], la non ripudiabilità[36], l'integrità[37], la confidenzialità[38] e l'auditing[39].

Come per [RFC2904], in [X.812] si è scelto un approccio generico, non

[32] Il nome identificativo utilizzato da ISO/IEC è 10181-3.
[33] Da ITU-T esso viene denominato X.812.
[34] La serie 10181 per ISO/IEC, X.800-X.849 per ITU-T.
[35] ISO/IEC 10181-2 / ITU-T X.811.
[36] ISO/IEC 10181-4 / ITU-T X.813.
[37] ISO/IEC 10181-6 / ITU-T X.815.
[38] ISO/IEC 10181-5 / ITU-T X.814.
[39] ISO/IEC 10181-7 / ITU-T X.816.

legato a particolari piattaforme software o hardware. Lo standard presenta una serie di termini, concetti e patterns architetturali atti principalmente a costituire un vocabolario comune per gli esperti del settore.

Il componente principale per il controllo di accesso è quello che effettua la decisione di autorizzazione, la Access control Decision Function (ADF) secondo il linguaggio di [X.812]. Esso si interconnette con gli attori dell'accesso, l'*initiator* (il soggetto richiedente) ed il *target* (la risorsa) nel modo mostrato in figura 1.7, ovvero per mezzo del componente Access control Enforcement Function (AEF) che si occupa di eseguire (*enforce*) la decisione.

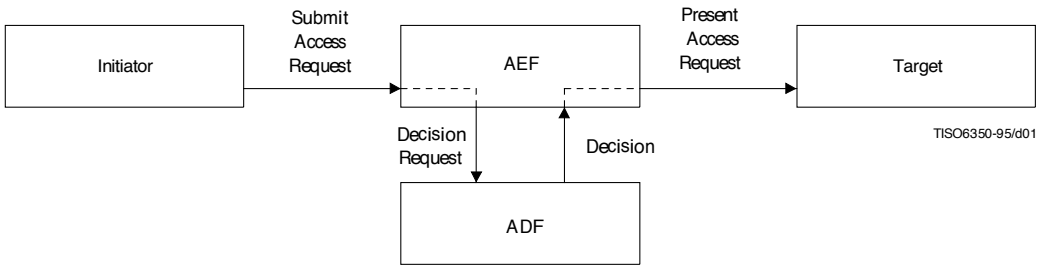

Figura 1.7: Componenti fondamentali per il controllo di accesso secondo [X.812]. Figura tratta da [X.812]

Le informazioni che, oltre alla policy, concorrono a determinare la decisione di accesso dell'ADF vengono chiamate Access control Decision Information (ADI). Come riportato in figura 1.8, [X.812] distingue tra le seguenti tipologie di ADI:

initiator ADI: informazioni circa il soggetto della richiesta come ad esempio identità, attributi, dominio di appartenenza, ecc.;

target ADI: informazioni sullo stato della risorsa, ad esempio il carico di lavoro, l'identità del soggetto responsabile, ecc.;

access request ADI: l'operazione e gli operandi relativi alla richiesta di accesso;

contextual ADI: qualsiasi informazione contestuale ritenuta influente sulla decisione, come l'istante in cui avviene l'accesso, la locazione o route dalla quale si cerca di accedere, ecc.;

retained ADI: l'ADF ha la possibilità di immagazzinare (*cache*) ADI ottenute in decisioni di accesso precedenti, utili ad esempio per autorizzare accessi ripetuti di uno stesso initiator o verso lo stesso target.

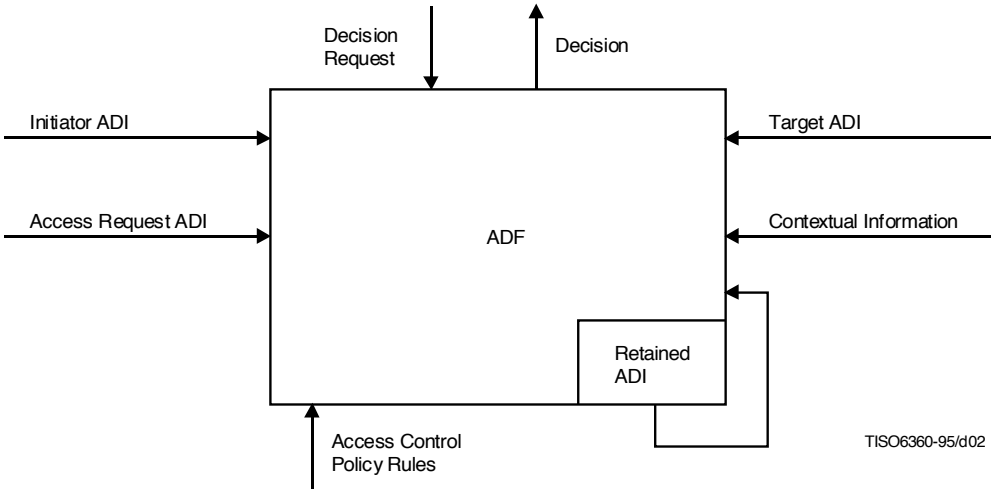

Figura 1.8: ADF: dettaglio delle informazioni affluenti nella decisione. Figura tratta da [X.812].

Lo standard distingue inoltre tra ADI e Access Control Information (ACI). Queste ultime possono esser viste come un insieme di attributi, nella forma <tipo, valore>, associati a vari "elementi" (initiator, target, access request, contesto). Le ADI vengono ottenute dalle ACI attraverso un'operazione detta *binding* che consiste sostanzialmente nella creazione di un legame (crittograficamente) forte tra le ACI e l'elemento a cui si riferiscono. Ad esempio la initiator ADI può essere creata facendo firmare ad un'autorità riconosciuta un documento contenente l'identificativo dell'initiator ed informazioni sul

suo conto (initiator ACI)[40]. Grazie a tale binding l'ADF ha la certezza che le informazioni utilizzate nella decisione sono affidabili.

[X.812] prevede che le ACI possano essere modificate, revocate o cedute da un componente (ad es. l'initiator) in favore di un altro (ad es. un broker) al fine di agire per suo conto (delega). Nel caso in cui una ACI venga revocata, è necessario che ciò venga fatto anche per tutte le ADI da essa derivate.

1.3.2.1 Aspetti di minore rilievo in [X.812]

Il resto dello standard definisce concetti e temi di minore interesse e poco citati in letteratura. Ad esempio:

Requisiti e classificazione delle policies. Pur non specificando un particolare linguaggio per esprimere le policies, lo standard ne individua i requisiti generali e ne classifica le tipologie (ACL, security labels, ecc.).

Componenti informativi e funzionali elementari. [X.812] classifica le varie categorie di ACI previste dal proprio modello: initiator, target, access request, operazione, operandi e contesto. La stessa operazione viene effettuata per le funzionalità di base dei componenti applicativi (*facilities*). Essi vengono inoltre suddivisi in funzioni amministrative (ad es. emissione di ACI) e funzioni operative (ad es. recupero delle ACI al momento di una decisione).

Classificazione dei meccanismi di controllo di accesso. Per [X.812], un meccanismo di controllo di accesso è determinato dalla scelta di una tipologia di policy (e quindi di uno *schema* di controllo di accesso) e dalla progettazione delle *facilities* di supporto per la gestione delle ACI. Le principali categorie di tali meccanismi sono:

Access Control List (ACL): le ACL sono delle ACI del target che contengono liste di coppie <initiator(s), operation(s)>;

[40] Ciò è esattamente ciò che accade nella pratica con i certificati digitali (cf. 3.1.2) o le asserzioni SAML (cf. 3.1.1).

Capability: è l'opposto della categoria ACL, le capabilities sono ACI dell'initiator nella forma di liste di coppie <target(s), operation(s)>;

Label based: il controllo di accesso è basato su etichette (ACI) associate agli initiators, ai targets ed alle richieste di accesso; le regole di accesso sono basate su relazioni tra le varie etichette in gioco;

Context based: l'autorizzazione in questo caso è basata solo su ACI contestuali; raramente questa categoria viene utilizzata in maniera esclusiva, più spesso funge da supporto per altri meccanismi di controllo di accesso.

Distribuzione dei componenti di controllo di accesso. Analogamente a [RFC2904], anche [X.812] analizza molte varianti dello schema in figura 1.7. In particolare viene studiato il caso in cui i componenti ADF e AEF siano distribuiti tra domini di sicurezza distinti.

1.3.3 Mappa terminologica

I documenti [X.812] e [RFC2904] utilizzano spesso termini differenti per denotare concetti analoghi. I termini utilizzati da questa tesi attingono indifferentemente dall'uno o dall'altro vocabolario in funzione della maggiore affinità con il concetto denotato. Inoltre, nell'esporre i lavori altrui (ad esempio gli standards e le tecnologie del cap. 3) si è scelto di utilizzare gli stessi termini dei testi originali. Allo scopo di ridurre le ambiguità o le incomprensioni, si riporta in tabella 1.1 le corrispondenze tra i termini dei due standards ed eventuali sinonimi utilizzati nel corso della tesi.

1.3.4 Il modello RBAC

Role-Based Access Control (RBAC), descritto per la prima volta in [RBAC] nel 1992, è un modello di controllo di accesso in cui i permessi non sono assegnati direttamente ai soggetti, ma a *ruoli* (o qualifiche). Ai soggetti potranno poi essere attribuiti uno o più ruoli. Questa separazione tra l'assegnazione

[RFC2904]	[X.812]	sinonimi utilizzati
utente	initiator	cittadino
SE	target	risorsa
UHO	initiator security domain	dominio fruitore
SP	target security domain	dominio erogatore
PDP	ADF	decisore, motore di decisione, motore di decisione basato su policy
PIP	acquire ACI facility	
PEP	AEF	
	ACI	attributi
AC	ADI	credenziali, attributi
RM		Session Manager
	access control	controllo di accesso
authorization	authorization	autorizzazione, decisione di accesso, decisione

Tabella 1.1: Mappa dei concetti espressi da [RFC2904] e [X.812].

di ruoli agli utenti e di permessi ai ruoli (che può quindi essere effettuata da autorità distinte) è la caratteristica principale di RBAC.

Il modello RBAC agevola significativamente la gestione dei permessi. Infatti l'associazione tra soggetti e permessi, usata dei modelli tradizionali, è tipicamente transitoria in molte applicazioni. La relazione tra ruoli e permessi tende invece a permanere più a lungo in quanto riflette più fedelmente la struttura di funzioni e responsabilità preesistente nelle organizzazioni. È pertanto possibile rimuovere la complessità della gestione dei permessi dalle mansioni degli amministratori addetti all'aggiunta, rimozione e (ri)qualificazione dei soggetti.

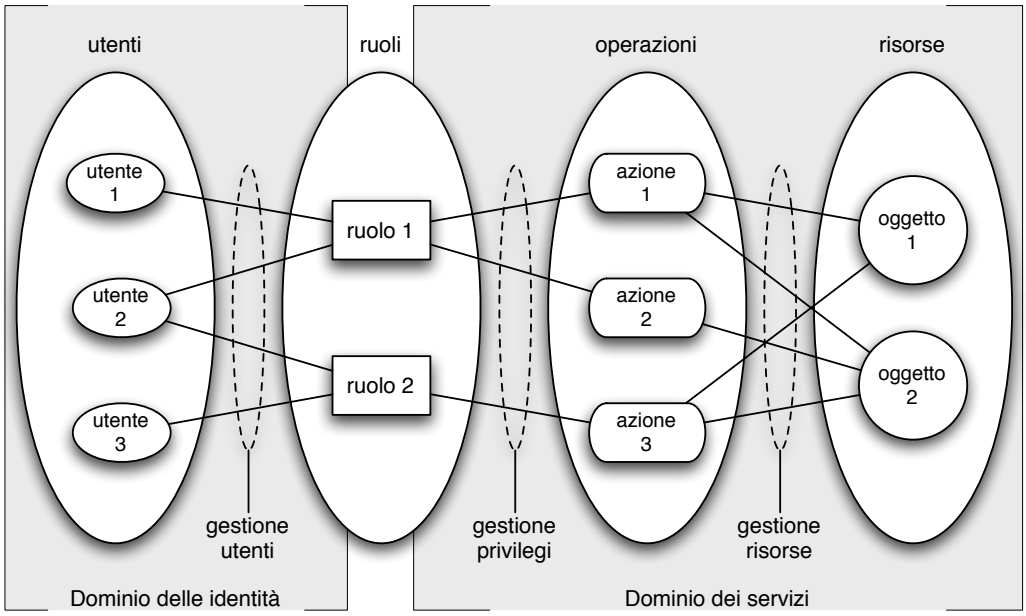

Figura 1.9: Modello insiemistico di RBAC.

Il concetto di ruolo è molto vicino a quello di *gruppo* di utenti utilizzato in molti altri modelli e sistemi reali. In effetti i ruoli possono essere visti come la collezione degli utenti che ne sono investiti. Una tale definizione non sarebbe tuttavia completa in quanto i ruoli caratterizzano anche l'insieme di permessi ad essi attribuiti. A tal proposito [RBAC-Sandhu] distingue i sistemi e i modelli RBAC 'veri' in base alla seguente proprietà:

> “ [...] it should be equally easy to determine role membership and role permissions [...][41] ”

[RBAC-Sandhu] estende il modello originale in [RBAC] definendo una famiglia di modelli RBAC:

RBAC$_0$ è sostanzialmente identico al modello in [RBAC];

RBAC$_1$ aggiunge il concetto di ereditarietà dei ruoli a RBAC$_0$;

RBAC$_2$ introduce l'uso di vincoli generici in RBAC$_0$;

RBAC$_3$ integra RBAC$_1$ e RBAC$_2$ in un unico modello.

1.3.4.1 Modello di base

RBAC$_0$ viene definito in termini insiemistici, come illustrato anche in figura 1.9. Siano U l'insieme dei soggetti, R quello dei ruoli e P quello dei permessi. Il modello evita di descrivere in maggiore dettaglio i permessi in quanto costituiscono una caratteristica specifica del dominio applicativo. Tuttavia essi possono essere visti, senza perdita di generalità, come insiemi di *azioni* su *risorse* (oggetti). Quindi: $P \subseteq 2^{A \times O}$, dove A denota l'insieme delle azioni ed O quello delle risorse.

L'assegnazione di ruoli ai soggetti e di permessi ai ruoli è modellata tramite due relazioni, rispettivamente: $UA \subseteq U \times R$ e $PA \subseteq P \times R$.

Il modello introduce inoltre il concetto di *sessione*[42], ovvero la possibilità per un soggetto di attivare un sottoinsieme dei propri ruoli, allo scopo di compiere una specifica attività e/o per un certo intervallo di tempo. A tale scopo viene introdotto un altro insieme, S (le sessioni), e due funzioni:

- $user : S \rightarrow U$ associa ogni sessione ad un singolo utente (che può avere più di una sessione attiva contemporaneamente);

[41] [RBAC-Sandhu] p. 4.

[42] Da non confondersi con la nozione di sessione applicativa, utilizzata ad esempio nelle applicazioni web.

- $roles : S \rightarrow 2^R$ associa ad ogni sessione l'insieme dei ruoli attivati dall'utente.

Le sessioni permettono di includere in RBAC$_0$ il principio del *privilegio minimo* (*least privilege*). Un utente dotato di più ruoli può scegliere, per una particolare sessione, il minimo sottoinsieme di essi che gli permetta di svolgere il proprio compito con successo.

RBAC$_0$ è quindi così formalizzato:

Definizione 1.1 *Sia:*
$RBAC_0 = \langle U, R, P, S, PA, UA, user, roles \rangle$
$PA \subseteq P \times R$
$UA \subseteq U \times R$
$user : S \rightarrow U$
$roles : S \rightarrow 2^R \mid \forall s_i \in S.\ roles(s_i) \subseteq \{r | (user(s_i), r) \in UA\}$ □

1.3.4.2 Ereditarietà

L'introduzione del concetto di *ereditarietà dei ruoli* (*role hierarchy*) permette di semplificare la gestione dei sistemi basati su RBAC, al tempo stesso modellando in maniera più fedele i contesti organizzativi dei tipici scenari applicativi. La grande maggioranza delle organizzazioni, infatti, è strutturata in base a relazioni gerarchiche di autorità e responsabilità.

Da un punto di vista matematico tale ereditarietà viene modellata da una relazione di semiordinamento sull'insieme dei ruoli: $RH \subseteq R \times R$, indicata anche con $\geq$. Si noti che un soggetto può attivare in una sessione sia ruoli direttamente in proprio possesso, sia ruoli ereditati. Si ha quindi la seguente ridefinizione di *roles*:

Definizione 1.2 *Sia:*
$roles : S \rightarrow 2^R$ *tale che:*
$\forall s_i \in S.\ roles(s_i) \subseteq \{r | \exists r' \in R.\ r' \geq r \Rightarrow (user(s_i), r') \in UA\}$ □

1.3.4.3 Vincoli

In aggiunta alle relazioni di ereditarietà, in molti scenari applicativi è necessario imporre vincoli aggiuntivi, come ad esempio:

Disgiunzione di ruoli. La possibilità di esprimere ruoli mutualmente esclusivi permette di realizzare il principio della *separazione delle responsabilità* (*separation of duties*). Ad esempio, nell'implementare un sistema di gestione delle gare d'appalto, è possibile imporre che i soggetti che assumano il ruolo di *partecipante* alla gara non possano contemporaneamente essere parte del processo di designazione del vincitore (ruolo di *designatore*).

Disgiunzione di permessi. Costituisce il vincolo duale del precedente, ovvero la possibilità di impedire che uno stesso permesso possa essere attribuito ad un solo elemento appartenente ad un insieme di ruoli mutualmente esclusivi.

Vincoli di cardinalità. Si può imporre ad esempio che un ruolo possa essere ricoperto da un numero limitato di soggetti, o viceversa che un soggetto possa assumere un numero limitato di ruoli.

Ruoli prerequisiti. Per assumere un determinato ruolo può essere necessario possederne già un'altro. Ad esempio per potersi iscrivere all'università viene di norma richiesto un diploma di scuola superiore.

Permessi prerequisiti. Dualmente al caso precedente, si può imporre che un permesso venga assegnato ad un ruolo solo se quest'ultimo è già in possesso di un permesso 'strumentale'. Si pensi ad esempio ai permessi in un *filesystem*: non ha senso permettere la lettura di un file se non si dà accesso alla directory che lo contiene.

Data la loro natura eterogenea, i vincoli vengono modellati in RBAC_2 sotto forma di generici predicati applicabili alle relazioni UA e PA ed alle funzioni $user$ e $roles$.

Si noti che l'ereditarietà dei ruoli definita in $RBAC_1$ potrebbe essere considerata un vincolo. Ciò ne implicherebbe la ridondanza, ovvero la sussunzione da parte di $RBAC_2$. Tuttavia in [RBAC-Sandhu] si sceglie di non aggiornare la precedente definizione di RH in quanto complicherebbe il modello.

1.3.4.4 Modello aggregato

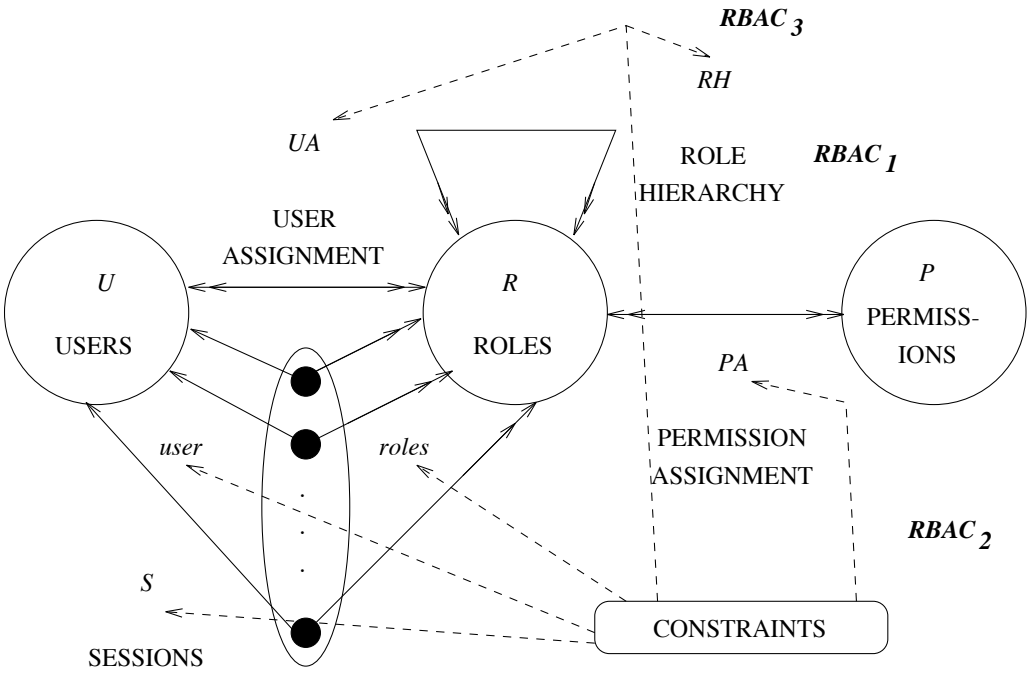

Figura 1.10: La famiglia dei modelli $RBAC_i$. Figura tratta da [RBAC-Sandhu].

$RBAC_3$, schematizzato in figura 1.10, unifica $RBAC_1$ (ereditarietà) e $RBAC_2$ (vincoli). Nel compiere tale operazione è necessario puntualizzare il fatto che i predicati di $RBAC_2$ devono poter essere applicati anche alla relazione di ereditarietà RH di $RBAC_1$. Bisogna inoltre considerare che, in presenza di ereditarietà, l'effetto dei vincoli può cambiare. Ad esempio, nell'imporre vincoli di cardinalità, è necessario chiarire se tali vincoli si ap-

plicano solo ai ruoli direttamente assunti da un soggetto od anche a quelli ereditati.

1.3.4.5 Modello amministrativo

Una delle caratteristiche più interessanti di RBAC è che può essere utilizzato per l'amministrazione di sé stesso. In realtà nei modelli RBAC_i si impone che i permessi non siano applicabili agli oggetti del modello stesso. Per realizzare l'amministrazione di RBAC è pertanto necessario introdurre una nuova famiglia di modelli, detta Administrative Role-Based Access Control (ARBAC), in tutto e per tutto speculare a RBAC_i. Ad esempio per ARBAC_0 si ha:

Definizione 1.3 *Sia:*

$ARBAC_0 = \langle U, AR, AP, S, APA, UA', user, roles' \rangle$

$APA \subseteq AP \times AR$

$UA' \subseteq U \times (R \cup AR)$

$user : S \to U$

$roles' : S \to 2^{R \cup AR} \mid \forall s_i \in S.\ roles'(s_i) \subseteq \{r | (user(s_i), r) \in UA'\}$ □

Si noti come i soggetti amministratori provengano dallo stesso insieme U dei soggetti di RBAC_0, mentre i ruoli ed i permessi amministrativi sono disgiunti dai corrispettivi in RBAC_0.

Un'aspetto interessante da considerare è per quali $i, j \in \{0, 1, 2, 3\}$ RBAC_i sia amministrabile da ARBAC_j. [RBAC-Sandhu] suppone che per la maggior parte degli scenari applicativi sia sufficiente il vincolo $i \geq j$, ovvero che sia possibile utilizzare modelli RBAC semplificati per l'amministrazione. L'utilizzo di ARBAC_1, ovvero delle gerarchie di ruoli amministrativi, è tuttavia fortemente consigliato in quanto permette di implementare la delega di funzioni amministrative, uno degli obiettivi dell'IM (cf. 1.1).

L'integrazione dei modelli ARBAC_i con i RBAC_i è schematizzata in figura 1.11.

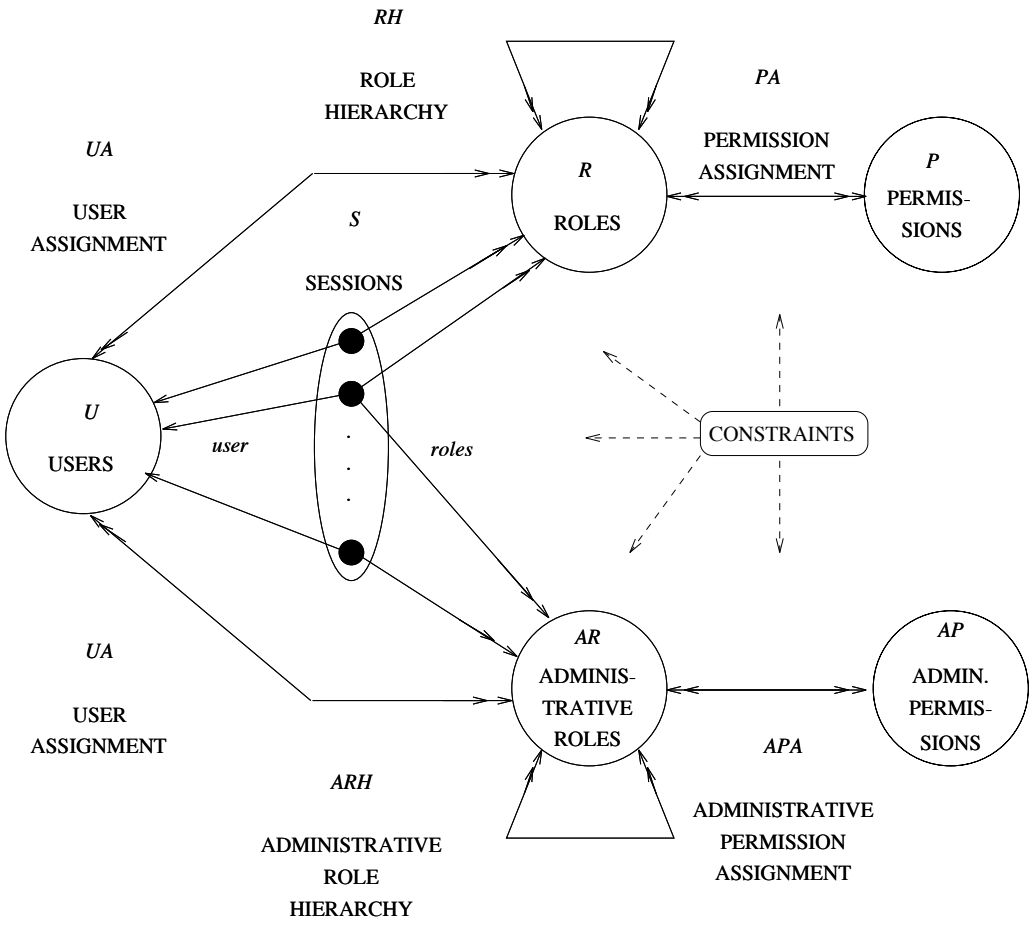

Figura 1.11: Il Modello amministrativo $ARBAC_3$. Figura tratta da [RBAC-Sandhu].

Capitolo 2

Analisi dei requisiti della Pubblica Amministrazione italiana in termini di Identity Management

Dopo aver delineato nel capitolo 1 i trends tecnologici in atto nel mondo dell'e-business, ci si appresta ora ad analizzare in maggior dettaglio quali siano le particolari esigenze della Pubblica Amministrazione (PA) italiana che l'architettura proposta (cap. 4) dovrà soddisfare utilizzando opportuni strumenti (cap. 3).

Al momento non esiste un modello architetturale preciso per l'IM promulgato da un autorità centrale, ma piuttosto una serie di iniziative indipendenti ad opera delle varie amministrazioni centrali o locali[1]. Questa situazione riflette da un lato l'attuale assetto politico/istituzionale improntato sul federalismo, dall'altro una effettiva novità delle problematiche introdotte dall'e-Government e la scelta di affrontarla attraverso una molteplicità di approcci distinti.

[1]Tipicamente amministrazioni regionali, province autonome o comuni di grandi dimensioni.

Di recente il Centro Nazionale per l'Informatica nella Pubblica Amministrazione (CNIPA)[2] si è occupato di promuovere modelli e standards per l'e-Government, soprattutto per quanto riguarda le comunicazioni informatiche (*cooperazione applicativa*) tra più amministrazioni. Tuttavia, molti aspetti di sicurezza ed IM non sono stati ancora sufficientemente trattati da questi standards come testimonia ad esempio il fatto che non esista ad oggi una soluzione per il SSO su scala nazionale tra tutti i portali delle amministrazioni.

Data la novità delle problematiche e la disomogeneità delle soluzioni studiate, l'analisi dei requisiti della PA viene qui effettuata esaminando due tipologie di iniziative:

- analisi diretta degli standards, laddove presenti;
- sintesi dei requisiti a partire da specifici progetti, bandi di gara, ecc.

Successivamente, nella sezione 2.3, tali requisiti verranno formalizzati e raggruppati in categorie.

Quella effettuata in questo capitolo vuole essere più un'indagine conoscitiva che una rassegna esaustiva. A tale scopo, le iniziative analizzate sono state scelte in modo che costituiscano un insieme rappresentativo dell'innovazione in atto nella PA in termini di IM.

[2] cf. 2.1.

2.1 Le recenti iniziative del CNIPA

“ Il Centro Nazionale per l'Informatica nella Pubblica Amministrazione (CNIPA) [CNIPA-web] opera presso la Presidenza del Consiglio per l'attuazione delle politiche del Ministro per l'Innovazione e le Tecnologie. Esso è stato istituito nel 2003[3] dalla fusione di due organismi preesistenti: l'Autorità per l'Informatica nella Pubblica Amministrazione (AIPA) ed il Centro Tecnico per la Rete Unitaria della Pubblica Amministrazione (RUPA). Il CNIPA ha l'obiettivo primario di dare supporto alla PA nell'utilizzo efficace dell'informatica per migliorare la qualità dei servizi e contenere i costi dell'azione amministrativa.

In sintesi il CNIPA:

- contribuisce alla definizione della politica del Governo e del Ministro per l'innovazione e le tecnologie e fornisce consulenza per la valutazione di progetti di legge nel settore informatico;
- coordina il processo di pianificazione e i principali interventi di sviluppo; detta norme e criteri per la progettazione, realizzazione, gestione dei sistemi informatici delle amministrazioni, della loro qualità e dei relativi aspetti organizzativi; definisce criteri e regole tecniche di sicurezza, interoperabilità, prestazione;
- controlla che gli obiettivi e i risultati dei progetti di innovazione della PA siano coerenti con la strategia del Governo; a tale scopo si affianca alle amministrazioni pubbliche nella fase di progettazione ed emette pareri di congruità tecnico-economica;
- cura l'attuazione di importanti progetti per l'innovazione tecnologica nella PA, la diffusione dell'e-Government e lo sviluppo

[3] In attuazione di quanto disposto dal decreto legislativo 30 giugno 2003, n. 196, “Codice in materia di protezione dei dati personali”, pubblicato sul supplemento ordinario n. 123 alla Gazzetta Ufficiale n. 174 del 29 luglio 2003.

delle grandi infrastrutture di rete del Paese per consentire agli uffici pubblici di comunicare tra loro e per portare i servizi della PA ai cittadini e alle imprese;

- cura la formazione dei dipendenti pubblici nel settore informatico, utilizzando le nuove tecnologie per favorire l'apprendimento continuo.

Dal punto di vista organizzativo il CNIPA è governato da un organo collegiale costituito dal Presidente e da quattro membri, scelti tra persone dotate di alta e riconosciuta competenza e professionalità, nominati dal Presidente del Consiglio dei Ministri.[4] ”

Dalla sua istituzione, il CNIPA ha emesso alcuni standards e best practices per l'uso dell'Information and Communication Technology (ICT) nella PA. Tra questi, è di interesse per la tesi il modello architetturale per la cooperazione applicativa tra le amministrazioni, descritto nella sezione 2.1.1. Non strettamente connessa con le problematiche di IM, viene inoltre riportata (sezione 2.1.2) la valutazione del modello di sviluppo Open Source in quanto costituisce un requistito incoraggiato dal CNIPA e ricorrente nelle committenze delle singole amministrazioni.

2.1.1 Il Sistema Pubblico di Connettività e Cooperazione

Sistema Pubblico di Connettività e cooperazione (SPC) è probabilmente il contributo di maggior rilievo apportato dal CNIPA alla “Società dell'Informazione” italiana, ed in particolare al sitema delle PA. Esso costituisce l'infrastruttura che permette la comunicazione tra le applicazioni informatiche dei soggetti e degli enti amministrativi, detta *cooperazione applicativa*.

Il processo di standardizzazione di SPC è avvenuto in due fasi principali:

[4] Adattato da: [CNIPA-web].

la stesura di una rete sicura di trasporto[5] tra le amministrazioni (Connettività) e la progettazione di un *messaging layer* che permetta protocolli complessi di comunicazione a livello applicativo (Cooperazione).

Per ciascuno dei due interventi nel 2003 è stato creato dal CNIPA un gruppo di lavoro specifico a cui hanno partecipato rappresentanti delle amministrazioni centrali e locali, del mondo accademico, dei maggiori fornitori di servizi ICT e le più importanti associazioni di categoria (150 e 120 persone rispettivamente).

Nel seguito si riportano i risultati della standardizzazione dell'architettura di cooperazione, pubblicati il 25 Novembre 2004 [CnipaArch][CnipaTech], in quanto di interesse per le problematiche e le tecnologie coinvolte.

2.1.1.1 Modello e contesto della cooperazione applicativa

Lo scenario preesistente ad SPC è quello di applicazioni sviluppate ad hoc per le singole amministrazioni. Nei casi in cui un atto amministrativo prevedesse la comunicazione di più enti, sono stati studiati caso per caso dei protocolli di cooperazione. In un periodo come quello attuale di forte spinta all'informatizzazione della burocrazia ci si è resi conto che l'approccio utilizzato precedentemente non era in grado di scalare adeguatamente con la domanda. Si è presentata pertanto una duplice necessità: da un lato innovare ed armonizzare le modalità di comunicazione tra amministrazioni, dall'altro preservare il più possibile il patrimonio applicativo preesistente.

Per soddisfare queste necessità il CNIPA ha scelto per SPC il modello di una SOA basata sui Web Services. Ogni organizzazione partecipante alla cooperazione espone (*eroga*) quindi il proprio parco applicativo sotto forma di *servizi* dotati di modalità di interfacciamento standardizzate. La stessa organizzazione può inoltre *fruire* dei servizi erogati da un'altro ente amministrativo. La situazione è illustrata in figura 2.1.

La fruizione di un servizio è regolata da un *accordo di servizio* stipulato

[5] Basata su connettività IP ed IPSec fornita da ISP *qualificati*, ovvero fornenti specifiche garanzie di affidabilità e qualità del servizio.

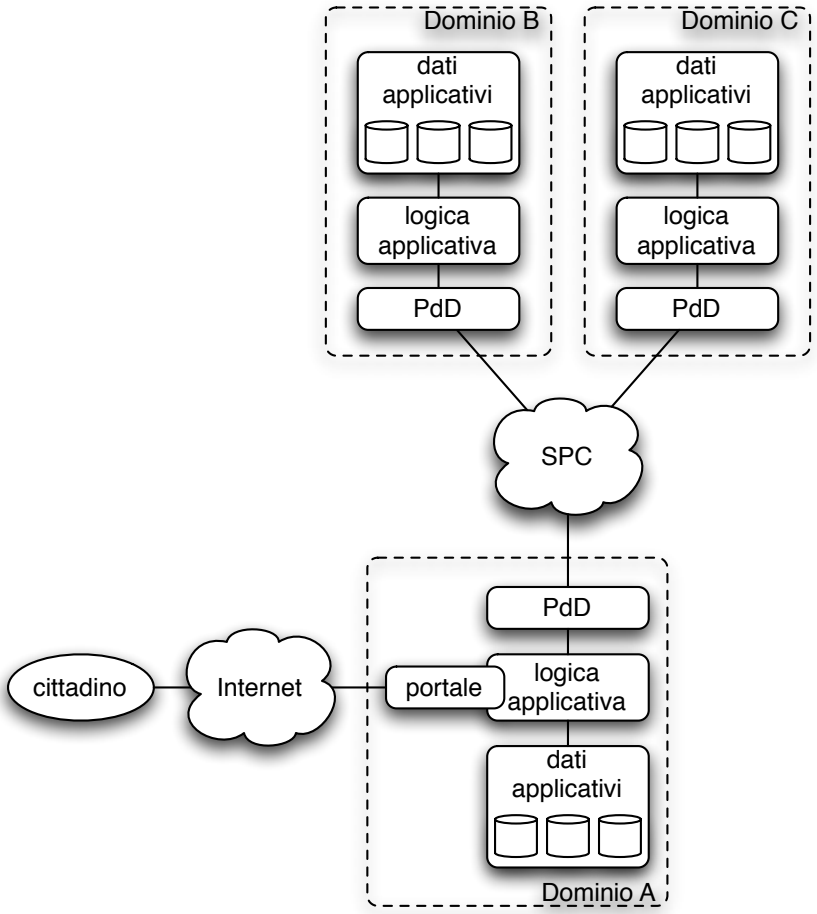

Figura 2.1: Modello architetturale di SPC.

tra l'ente fruitore e quello erogatore ed accessibile da un *registro dei servizi*[6]. L'accordo di servizio contiene anche le interfacce di scambio messaggi per l'interrogazione, i requisiti di sicurezza e qualità del servizio, la documentazione dello stesso, ecc.

Almeno la componente di interfacciamento dell'accordo deve essere scritta nel formato specificato da [WSDL 1.1] in modo che le applicazioni client presso gli enti fruitori possano adattarsi dinamicamente a variazioni nell'interfaccia del servizio.

[6] cf. 2.1.1.4.

2.1.1.2 La Porta di Dominio

In [CnipaArch][CnipaBusta] un *dominio* è definito come il confine di responsabilità di un ente o soggetto amministrativo e racchiude al suo interno tutte le applicazioni da esso gestite. Le comunicazioni da e verso un dominio devono attraversare la sua *Porta di Dominio (PdD)*. Si veda in proposito fig. 2.1.

La PdD è il componente infrastrutturale (eventualmente distribuito) di base di SPC ed è designata a svolgere i seguenti compiti:

- scambio a livello connessione (HTTP);
- sicurezza a livello connessione (SSL, TLS);
- gestione dei messaggi (cf. 2.1.1.3);
- tracciatura dei messaggi;
- smistamento (*routing*) dei messaggi;
- sicurezza a livello di messaggio (WSS);

Inoltre, a seconda delle applicazioni presenti all'interno del proprio dominio, la PdD può fornire funzionalità di integrazione delle stesse attraverso opportuni *adapters*. Infatti, le applicazioni *legacy* per le quali non fosse stato previsto, al momento della progettazione, l'utilizzo in un contesto di cooperazione SPC devono essere integrate nell'architettura secondo le modalità tipiche del paradigma dei Web Services, ovvero sviluppando dei *wrappers* che ne incapsulino la logica applicativa esponendo al contempo un'interfaccia interoperabile.

2.1.1.3 La Busta di e-Government

Lo standard CNIPA sulla *Busta di e-Government* [CnipaBusta] è stato il primo ad essere ratificato, il 21 Aprile 2004, tra quelli relativi a SPC. La Busta di e-Government specifica il formato dei messaggi scambiati tra le

PdD nelle interazioni di cooperazione applicativa e ne costituisce di fatto l'elemento informativo di base.

Struttura della Busta. Una Busta di e-Government è sostanzialmente una specializzazione di un messaggio SOAP[7] [SOAP] con l'aggiunta (opzionale) delle estensioni specificate in [SOAP-Attachment]. La struttura della Busta nel caso 'normale' e nel caso con attachments è illustrata in figura 2.2.

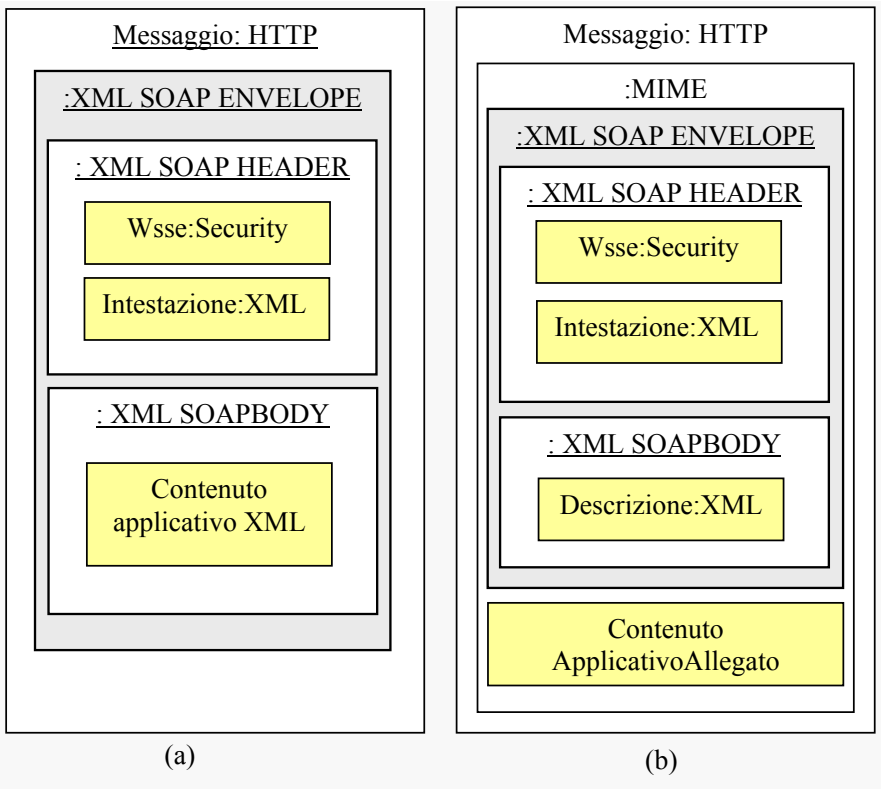

Figura 2.2: La struttura della Busta di e-Government, nei casi con (b) e senza (a) attachment. Figura tratta da [CnipaBusta].

In ambedue i casi la Busta si compone logicamente di una parte infrastrutturale o *header* (intestazione) e di una parte specifica dell'applicazione. Quest'ultima componente informativa è strutturata in due parti: il *body* (corpo) del messaggio ed eventuali allegati opzionali. Lo header ed il body devono

[7] Il trasporto raccomandato per i messaggi è HTTP o HTTPS.

essere in formato XML, mentre gli attachments possono anche essere binari purché codificati secondo lo standard MIME [RFC2045][RFC2046].

Composizione dello header. Ovviamente lo standard CNIPA non specifica quale debba essere il contenuto del body[8] o degli attachments ma solo il formato (di una parte) dello header. In particolare lo standard specifica un insieme di elementi XML dello header necessari ad implementare il *messaging layer* di SPC. Il CNIPA raccomanda inoltre l'utilizzo di uno o più elementi WSS [WSS] per gestire la sicurezza dei messaggi[9].

Elementi WSS. La possibilità di avere elementi WSS nello header di una Busta di e-Government è di particolare interesse per questa tesi in quanto lo standard in [WSS] definisce meccanismi generici per associare vari tipi di tokens di sicurezza ai messaggi, tra i quali è contemplata in [CnipaBusta][10] anche la presenza di asserzioni SAML (cf. 3.1.1.5).

La possibilità di usare tokens SAML in un messaggio di e-Government permette infatti di implementare scenari avanzati di IM come ad esempio l'utilizzo, da parte di un servizio cooperante, di credenziali 'derivate' da quelle del soggetto che ha innescato l'istanza di cooperazione[11]. Sono inoltre possibili modelli di autorizzazione dei messaggi basati non solo sull'identità del portatore, ma anche su sue qualifiche (attributi/autorizzazioni) certificate da una terza parte fidata.

Il capitolo 4 (sezione) esplora alcuni degli scenari applicativi possibili utilizzando tokens SAML in Buste di e-Government.

[8] Fa eccezione il caso in cui il messaggio venga utilizzato per trasferire documenti in attachments, nel qual caso è previsto che il body contenga anche il *manifesto* di tali documenti (titolo, formato o tipo, ecc.).

[9] cf. 1.2.3.

[10] L'ipotesi è stata tuttavia successivamente rinviata ad ulteriori analisi in [CnipaTech], pp. 66-67.

[11] Si pensi ad una delega di un cittadino o di un altro soggetto (un operatore, un altro servizio) in favore di un servizio per procedere in atti amministrativi elettronici per suo conto.

Elementi specifici di SPC. Si tratta della parte dello standard [CnipaBusta] che non costituisce una mera raccomandazione di altri standards, ma specifica gli elementi necessari al corretto funzionamento della cooperazione applicativa. Il listato 2.1 illustra un possibile header di Busta di e-Government; nel seguito ne sono descritti brevemente gli elementi principali.

Listing 2.1: Un esempio di header di Busta di e-Government, corrispondente alla richiesta di un servizio sincrono. Esempio tratto da [CnipaBusta].

```
<SOAP_ENV:Header
   xmlns="http://schemas.xmlsoap.org/soap/envelope/">
  <eGov_IT:Intestazione
     xmlns="http://www.cnipa.it/schemas/2003/eGovIT/Busta1_0/"
     SOAP_ENV:actor="http://www.cnipa.it/eGov_it/portadominio"
     SOAP_ENV:mustUnderstand="1">
    <eGov_IT:IntestazioneMessaggio>
      <eGov_IT:Mittente>
        <eGov_IT:IdentificativoParte
           tipo="CodicePA">
          ParteA
        </eGov_IT:IdentificativoParte>
      </eGov_IT:Mittente>
      <eGov_IT:Destinatario>
        <eGov_IT:IdentificativoParte
           tipo="CodicePA">
          ParteB
        </eGov_IT:IdentificativoParte>
      </eGov_IT:Destinatario>
      <eGov_IT:ProfiloCollaborazione>
        EGOV_IT_ServizioSincrono
      </eGov_IT:ProfiloCollaborazione>
      <eGov_IT:Servizio
         tipo="TEST">
        NomeServizio
      </eGov_IT:Servizio>
      <eGov_IT:Azione>
        NomeAzione
      </eGov_IT:Azione>
      <eGov_IT:Messaggio>
        <eGov_IT:Identificatore>
          ParteA_ANGPD_0000630_2003-06-05_15:58
        </eGov_IT:Identificatore>
        <eGov_IT:OraRegistrazione
           tempo="EGOV_IT_SPC">
          2003-06-05T17:58:10
        </eGov_IT:OraRegistrazione>
        <eGov_IT:Scadenza>
          2003-06-10T17:58:20
        </eGov_IT:Scadenza>
      </eGov_IT:Messaggio>
      <eGov_IT:ProfiloTrasmissione inoltro="EGOV_IT_PIUDIUNAVOLTA"/>
    </eGov_IT:IntestazioneMessaggio>
```

```
44      <eGov_IT:ListaTrasmissioni>
45        <eGov_IT:Trasmissione>
46          <eGov_IT:Origine>
47            <eGov_IT:IdentificativoParte
48               tipo="␣CodicePA"
49               indirizzoTelematico="http://137.5.3.11/pddo/servlet/s">
50              ParteA
51            </eGov_IT:IdentificativoParte>
52          </eGov_IT:Origine>
53          <eGov_IT:Destinazione>
54            <eGov_IT:IdentificativoParte
55               tipo="␣CodicePA"
56               indirizzoTelematico="http://158.0.1.21/pddo/servlet/s">
57              ParteB
58            </eGov_IT:IdentificativoParte>
59          </eGov_IT:Destinazione>
60          <eGov_IT:OraRegistrazione
61             tempo="EGOV_IT_SPC">
62            2003-06-05T17:58:10
63          </eGov_IT:OraRegistrazione>
64        </eGov_IT:Trasmissione>
65      </eGov_IT:ListaTrasmissioni>
66    </eGov_IT:Intestazione>
67  </SOAP_ENV:Header>
```

`IntestazioneMessaggio`. Contiene i seguenti sotto-elementi:

- l'indirizzo del `Mittente`;
- l'indirizzo del `Destinatario`;
- (opzionale) il `ProfiloCollaborazione`, ovvero la particolare modalità di scambio messaggi scelta (sincrona/asincrona, simmetrica/asimmetrica);
- (opzionale) l'identificatore della `Collaborazione` (ovvero gruppo di messaggi) alla quale appartiene il messaggio;
- (opzionale) il `Servizio` fruito/erogato;
- (opzionale) l'`Azione` richiesta;
- l'identificazione del `Messaggio` tramite un `Identificatore` unico, una marca temporale (`OraRegistrazione`), un (opzionale) messaggio ad

esso correlato[12] (`RiferimentoMessaggio`), la data di `Scadenza` del messaggio (opzionale);

- (opzionale) il `ProfiloTrasmissione`, ovvero se il messaggio può essere ritrasmesso e se l'avvenuta ricezione va riscontrata tramite l'invio di un messaggio di ritorno (*acknowledgement*).
- (opzionale) il numero di `Sequenza` all'interno dell'eventuale conversazione di appartenenza.

`ListaRiscontri`. Contiene l'acknowledgment, da parte del destinatario, di uno o più messaggi precedentemente ricevuti. L'elemento è opzionale.

`ListaTrasmissioni`. Nelle cooperazioni che richiedono l'implementazione della tracciabilità, questo elemento opzionale permette di registrare nel messaggio tutti i passaggi intermedi da esso attraversati, contestualmente all'istante in cui sono avvenuti.

`ListaEccezioni`. Opzionale. Contiene gli eventuali errori riscontrati nel processamento dell'elemento `IntestazioneMessaggio`.

2.1.1.4 Le altre infrastrutture

Oltre alla PdD, SPC necessita di un insieme minimo di componenti infrastrutturali centrali[13] che vengono complessivamente denominati Servizi Infrastrutturali di Cooperazione ed Accesso (SICA). I servizi SICA attualmente specificati in [CnipaArch] sono due:

- Servizio di registrazione e ricerca (Registro SICA),
- Servizio di Public Key Infrastructure (PKI SICA).

[12] Ad esempio nel caso in cui il messaggio sia un *acknowledgement* (riscontro) della ricezione di un altro messaggio.

[13] Anche per queste componenti è tuttavia prevista la possibilità di essere al loro interno distribuiti e/o replicati.

Il Registro dei servizi. La funzione principale del Registro dei servizi (o Registro SICA) è quella di permettere la pubblicazione ed il ricupero degli accordi di servizio al fine di agevolare lo sviluppo indipendente ed interoperabile dei componenti fruitori ed erogatori. [CnipaArch] specifica inoltre le seguenti funzionalità aggiuntive per il Registro:

- Indice dei domini che possono agire come fruitori od erogatori all'interno della comunità SPC.
- Elenco dei domini erogatori del SPC.
- Rubrica dei punti di accesso ai singoli servizi.

La PKI. Dato che tutti i soggetti cooperanti sono identificati per mezzo di un certificato X.509, è necessaria l'istituzione di una Public Key Infrastructure (PKI) composta da una gerarchia di Certification Authority (CA) al vertice delle quali vi è la CA del Comitato di gestione SPC[14]. Tale infrastruttura deve fornire le funzionalità di:

- emissione dei certificati (pubblicazione e rinnovo);
- sospensione e revoca degli stessi (utilizzando OCSP);
- marcatura temporale (TSA).

2.1.1.5 Paradigmi di cooperazione avanzati

Il paradigma di cooperazione nativamente supportato dai Web Services è quello dell'architettura di servizi (SOA), ovvero un modello in cui **un** fruitore accede ai servizi di **un** erogatore tramite uno scambio di messaggi. Tuttavia, alcuni scenari amministrativi necessitano di modelli di cooperazione più complessi che coinvolgano simultaneamente più di due attori. A tale scopo [CnipaArch] e [CnipaTech] raccomandano l'utilizzo di componenti infrastrutturali che permettano alle PdD, normalmente utilizzabili solo in modalità SOA, di cooperare secondo paradigmi avanzati.

[14] cf. 3.1.2.

Cooperazione per eventi. Nel modello di *cooperazione per eventi* gli attori si suddividono in due categorie: i *pubblicatori* che generano gli eventi ed i *sottoscrittori* che li consumano. Per implementare un tale modello usando i Web Services è necessaria l'istituzione di un servizio *gestore di eventi* che esponga due tipi di funzionalità ai domini cooperanti:

- La pubblicazione di un determinato *tipo* di evento da parte di un dominio.
- La sottoscrizione di un dominio per ricevere notifica di eventi di un certo tipo.

Orchestrazione. Il modello di *cooperazione per orchestrazione* è una generalizzazione di quello per eventi. Come quest'ultimo, il modello per orchestrazione prevede l'esistenza di un servizio centrale di coordinamento, ma le modalità di interazione sono più generiche e complesse. Lo standard raccomandato in [CnipaTech] per questo modello è [BPEL].

Coreografia. Il modello di *cooperazione per coreografia* cerca di affrontare scenari applicativi analoghi a quelli gestiti da quella per orchestrazione scegliendo tuttavia un approccio totalmente distribuito. In altre parole non è prevista la presenza di un componente centrale di coordinamento ma ogni attore ha conoscenza del suo ruolo all'interno della cooperazione. [CnipaTech] indica [WS-CDL] come standard di riferimento per questo modello.

2.1.2 Valutazione del modello Open Source

La recente diffusione del modello di sviluppo del software Open Source ha indotto il Ministero per l'Innovazione e le Tecnologie ad istituire, il 31 ottobre 2002, un'apposita "Commissione per il software a codice sorgente aperto nella Pubblica Amministrazione"[15] che ne esaminasse gli aspetti tecnici, economici ed organizzativi al fine di valutarne le possibilità di utilizzo nella PA. Sono

[15] Conosciuta anche come "Commissione Meo".

infatti evidenti i vantaggi che un tale modello può apportare al mondo della PA:

Risparmio economico: i prodotti Open Source sono tipicamente distribuiti gratuitamente; inoltre la loro natura 'aperta' ne incentiva la riusabilità in altri contesti.

Trasparenza: essendo il codice sorgente disponibile a chiunque lo voglia esaminare, è possibile per il cittadino conoscere i processi elettronici effettuati dalle amministrazioni nell'esecuzione delle pratiche amministrative, con particolare attenzione alla gestione della privacy.

Indipendenza dal fornitore: nel mondo del software commerciale spesso capita che l'azienda fornitrice di un determinato prodotto fallisca o non consideri più economicamente vantaggioso fornire supporto tecnico per l'intera durata del suo utilizzo (che può essere anche di svariati anni). Nel modello Open Source questo tipicamente non accade essendo le comunità degli utenti, degli sviluppatori e dei fornitori di supporto non disgiunte.

Modello di sviluppo collaborativo: più amministrazioni che condividano una stessa necessità applicativa possono unire gli sforzi e condividere il software sviluppato (o fatto sviluppare).

2.1.2.1 I risultati della "Commissione Meo"

La commissione, nel documento conclusivo [CommissioneMeo], analizza queste ed altre caratteristiche del modello di sviluppo Open Source e giunge ad alcune interessanti direttive che cercano di mettere sullo stesso piano prodotti commerciali ed Open Source:

- La scelta di soluzioni Open Source da parte di un'amministrazione deve essere effettuata in base al principio del Total Cost of Ownership (TCO) in totale parità rispetto a prodotti a sorgente chiuso.

- La PA deve essere libera di riusare (presso amministrazioni diverse da quelle appaltanti) i prodotti acquistati e ne deve avere la piena proprietà (non necessariamente esclusiva).

- Anche per i prodotti commerciali deve essere possibile, su richiesta della PA, l'ispezione del codice sorgente.

La commissione inoltre indica il modello Open Source come preferenziale per la realizzazione di progetti di ricerca ed innovazione tecnologica in modo da garantire una maggiore diffusione delle competenze acquisite ed il riuso delle soluzioni implementate. Questo è vero a maggior ragione nella realizzazione della cooperazione applicativa e di soluzioni di IM in quanto si tratta di progetti fondati sull'interoperabilità.

È opportuno citare infine il fatto che la commissione analizza l'impatto economico dell'adozione di soluzioni Open Source considerando la particolare struttura del tessuto produttivo italiano ed europeo in materia di software. In [CommissioneMeo] si riconosce infatti lo stato di profonda crisi del settore a partire dagli anni '80 ed il fatto che l'attuale struttura produttiva sia distribuita tra una miriade di piccole e medie imprese, in un settore che invece necessiterebbe di grandi investimenti per competere con i colossi statunitensi ed asiatici. In questo contesto l'Open Source potrebbe avere un ruolo strategico per promuovere l'innovazione nel Paese e di conseguenza per la rinascita dell'industria nazionale dell'ICT.

2.1.2.2 Il gruppo di lavoro del CNIPA.

A prosecuzione dei lavori della commissione, il CNIPA ha istituito, nel Febbraio 2004, il gruppo di lavoro "Codice sorgente aperto"[16]. Rispetto a quanto fatto dalla "Commissione Meo", l'approccio del gruppo di lavoro è stato più pratico che conoscitivo ed ha portato ad i seguenti risultati:

- la redazione di una metodologia di selezione per l'acquisto di soluzio-

[16] cf. [CnipaGdlOS].

ni ICT che permetta la comparazione sullo stesso piano di soluzioni commerciali ed Open Source;

- l'istituzione di un Centro di Competenza nazionale in materia di Open Source che permetta la diffusione di *best practices* e competenze alle istituzioni, l'incontro tra domanda ed offerta di prodotti e servizi, la condivisione delle esperienze delle singole amministrazioni.

2.2 Una selezione di progetti concreti e bandi

2.2.1 Il progetto ICAR

Il progetto triennale Interoperabilità e Cooperazione Applicativa tra le Regioni (ICAR) nasce nel 2004 su iniziativa di 17 regioni ed una provincia autonoma con il coordinamento del Comitato Permanente dei responsabili dei Sistemi Informatici delle regioni (CPSI)[17] del Centro Interregionale per il Sistema Informatico ed il sistema Statistico (CISIS)[18].

2.2.1.1 Gli interventi

Lo scopo del progetto è quello di promuovere l'uso di SPC per la cooperazione applicativa interregionale attraverso interventi concreti su due livelli:

Interventi infrastrutturali (INF). ICAR prevede l'implementazione da parte delle regioni di servizi infrastrutturali per la cooperazione applicativa in sé (INF-1), per la gestione di accordi di servizio (INF-2) e di un sistema federato di autenticazione (INF-3).

Casi di studio applicativi (AP). Allo scopo di verificare l'efficacia degli interventi infrastrutturali, il progetto prevede inoltre la stesura di sette casi di studio di specifici domini applicativi relativi alla cooperazione interregionale. In particolare:

AP-1 "Cooperazioni e Compensazioni Sanitarie Interregionali"

AP-2 "Cooperazione tra sistemi di Anagrafe"

[17] "Il Comitato Permanente dei Responsabili dei Sistemi Informatici delle Regioni, ha il compito di raccordare e costituire momento unificante per tutte le problematiche inerenti i Sistemi Informatici regionali con particolare riferimento all'innovazione tecnologica, alla standardizzazione dei sistemi ed alla cooperazione applicativa" (fonte: [CISIS-web]).

[18] Il CISIS "è una associazione tra le Regioni e le Province autonome costituita al fine di garantire un efficace coordinamento di strumenti informativi e di informazione statistica, nonché per assicurare il miglior raccordo tra le regioni, lo stato e gli enti locali" (fonte: [CISIS-web]).

AP-3 “Area Organizzativa Omogenea”

AP-4 “Lavoro e Servizi per l’Impiego”

AP-5 “Tassa automobilistica regionale”

AP-6 “Osservatorio Interregionale sulla rete distributiva dei carburanti”

AP-7 “Sistema Informativo Interregionale di Raccordo con Cinsedo”

L’intervento INF-3 riguarda alcune delle tematiche di questa tesi e pertanto viene qui descritto in dettaglio.

2.2.1.2 Intervento INF-3: “Realizzazione di un Sistema Federato interregionale di Autenticazione”

L’obiettivo di questo intervento è quello di realizzare un sistema federato di autenticazione ed attribuzione di ruoli che deleghi a componenti distribuiti detti Servizio di Identificazione e di Ruolo di Comunità (SIRC)[19] l’esecuzione di tali operazioni in funzione del dominio di appartenenza dell’utente. In base ad accordi di trust tra i domini partecipanti alla federazione, ogni dominio si impegna di riconoscere come valide le autenticazioni e le qualificazioni effettuate presso il SIRC che serve un altro dominio.

È interessante notare che la molteplicità SIRC-dominio non è uno-a-uno ma uno-a-molti. In questo modo è possibile implementare il concetto di *community network* ovvero più domini amministrativi che condividono lo stesso bacino d’utenza.

Affinché sia garantita l’interoperabilità tra i vari componenti dell’architettura federata (e quindi fortemente distribuita), ICAR individua la necessità di identificare univocamente gli utenti indipendentemente dal particolare meccanismo di autenticazione utilizzato dal dominio di appartenenza. Il progetto prescrive quindi l’utilizzo di un identificatore utente ‘federato’, ovvero valido presso ogni dominio. Lo stesso discorso si applica ai ruoli.

[19] Le modalità di funzionamento di questo componente sono analoghe a quelle di un Identity Provider nelle tecnologie di Single Sign On analizzate nella sezione 3.2.

È previsto che l'integrazione del sistema con i componenti di sicurezza preesistenti nei singoli domini avvenga sviluppando opportuni *wrappers*.

Infine, il progetto ICAR definisce come requisito anche una implementazione di riferimento del servizio, realizzata secondo il paradigma Open Source per garantirne il riuso, la condivisione dell'esperienza e la possibilità di adattarla alle esigenze dei singoli enti.

2.2.2 Regione Toscana: e.Toscana

Dal "Contesto di riferimento" in [BandoToscana]:

> "Con la Deliberazione del Consiglio Regionale n. 20 del 12.02.2003 è stato approvato il progetto e.Toscana nell'ambito del programma regionale degli investimenti strategici per gli anni 2003-2005.
>
> Tale progetto rappresenta l'impegno congiunto fra regione ed enti locali aderenti alla Rete Telematica Regionale Toscana (RTRT), per lo sviluppo di politiche di e-government sul territorio regionale ed in questa logica ha costituito la risposta toscana al piano nazionale di e-government che ha cofinanziato la realizzazione di progetti presentati sia dalla Regione che dagli enti sul territorio.
>
> Negli ultimi anni si è inoltre affermata a livello nazionale ed europeo una linea d'azione che vuole innovare la pubblica amministrazione ed il suo rapporto con tutte le componenti della società attraverso l'innovazione tecnologica applicata sia alle singole organizzazioni che per l'abbattimento delle barriere tecnologiche ed organizzative che limitano la circolarità delle informazioni e la condivisione delle banche dati e rendono difficile se non impossibile per gran parte della popolazione l'accesso telematico ai servizi. In questa ottica, con la deliberazione n. 867 del 08.09.2003, la Giunta Regionale individua come rilevante lo spaccato delle infrastrutture per la società dell'informazione e fra queste compaiono le "Infrastrutture per l'autenticazione e l'accesso ai servizi e alle informazioni - finalizzate alla

> diffusione di sistemi sicuri di riconoscimento telematico (certificati digitali) e alla creazione di modalità attraverso le quali a chi accede in rete sia possibile associare, nel rispetto della legge sulla privacy, i diritti di accesso e visibilità di classi di informazioni e servizi."
>
> Alla luce di questo si rende necessaria la costituzione di un sistema per l'accesso autenticato e sicuro ai servizi eToscana che si appoggi sulle infrastrutture già esistenti: il sistema userà infatti l'infrastruttura di RTRT per il trasporto, l'infrastruttura di PKI per la gestione dei certificati digitali e l'infrastruttura CART[20] per la Cooperazione Applicativa. ,,

Il sistema oggetto del bando della Regione Toscana si prefigge quindi di effettuare operazioni di autenticazione ed autorizzazione complesse per l'accesso a tutte le risorse (servizi, informazioni, ecc.) contenute all'interno del proprio dominio. Il contesto architetturale di riferimento è mostrato in figura 2.3.

2.2.2.1 Caratterizzazione del bacino d'utenza.

I soggetti da autorizzare si possono suddividere in **interni** ed **esterni** al dominio. I primi corrispondono a dipendenti (od operatori fidati) dell'ente titolare del dominio ed usufruiranno di servizi distinti o comunque esposti in maniera differente rispetto a quanto fatto per la seconda tipologia di utenza.

Gli utenti esterni ricadono in due ulteriori categorie: i soggetti amministrativi, che interrogano il dominio erogatore secondo le modalità della cooperazione applicativa[21] ed i cittadini che accedono ad esso tramite un terminale[22] che interroga un opportuno portale presso il dominio erogatore.

[20] Si tratta dell'istanza toscana di SPC. Tuttavia CART è stato progettato precedentemente a SPC e quindi non ne implementa tutte le funzionalità.

[21] Come specificato in SPC, cf. 2.1.1.

[22] Nel caso tipico si tratta di un browser web.

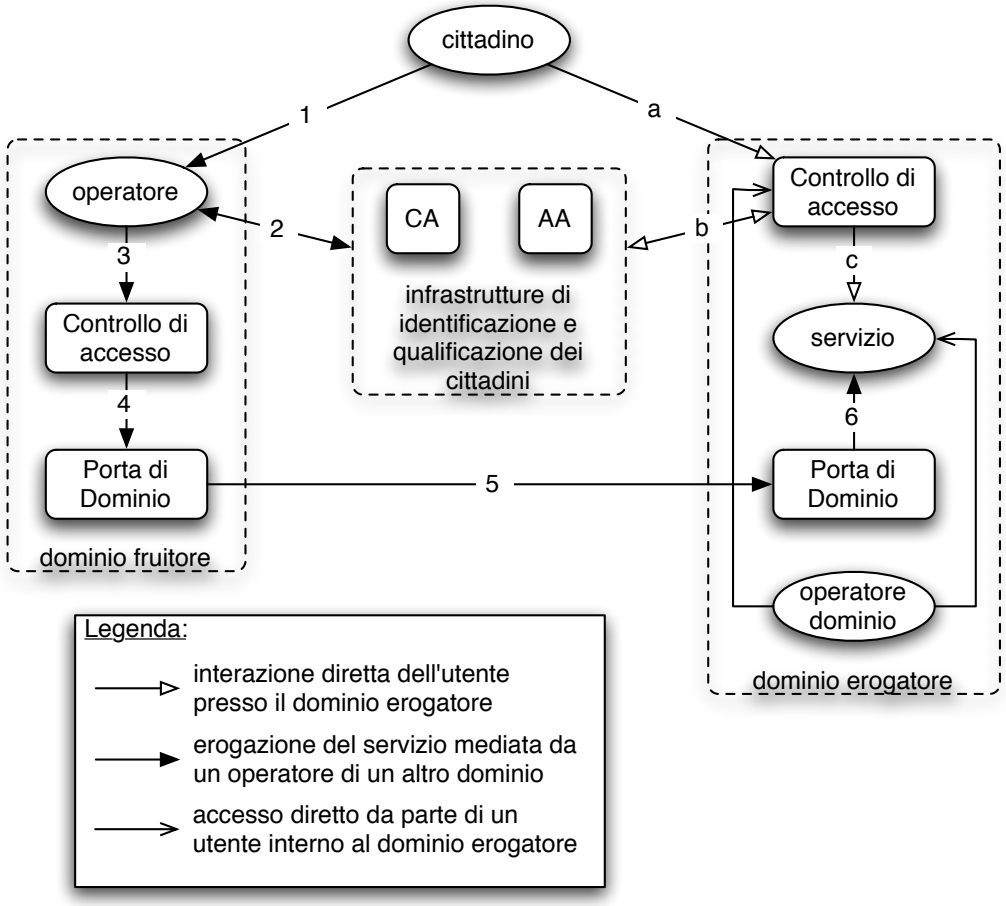

Figura 2.3: Modello architetturale del controllo di accesso in e.Toscana.

2.2.2.2 Funzionalità

Accesso di un cittadino. Nel caso di accesso di un cittadino, le funzionalità di alto livello del sistema descritto dal bando sono:

- Autenticare l'utente tramite un meccanismo non specificato[23]. L'identificativo da usare per riferire gli utenti è il Codice Fiscale.
- Autorizzare l'utente verificandone le credenziali, attribuendogli i ruoli[24]

[23]Il particolare meccanismo di autenticazione può essere pattuito tra i domini che condividono una stessa comunità di utenti.

[24]Si ha pertanto un controllo di accesso di tipo RBAC, si veda in proposito la sezione

e generando il profilo utente. Tale profilo sarà memorizzato in una sessione a lungo termine (tre mesi) con l'utente.

- Passare il controllo al servizio richiesto, inoltrando ad esso il profilo utente generato.

Accesso di un operatore di un dominio fidato. Nel caso di accesso da parte di soggetti amministrativi l'autenticazione e l'autorizzazione della controparte avviene invece secondo le modalità della cooperazione applicativa e può variare in funzione degli accordi di trust bilaterali o federativi tra i domini fruitore ed erogatore.

In particolare si assume che i cittadini a monte della richiesta amministrativa siano autenticati ed autorizzati dal dominio fruitore[25] e pertanto tali operazioni non vengono ripetute dal dominio erogatore. Quest'ultimo si limita a verificare che il dominio cliente ed il ruolo del soggetto richiedente (l'operatore) all'interno di esso corrispondano con quelli stipulati nell'accordo di trust.

Nel seguito ci si limita a descrivere le sole funzionalità relative all'accesso da parte di un cittadino.

2.2.2.3 Fase iniziale di Autocertificazione

Al momento del primo contatto, in caso di assenza di una sessione con l'utente, il sistema deve disporre di una modalità di autocertificazione. In questa fase l'utente, una volta autenticato, può dichiarare i ruoli di cui vuole disporre per l'accesso ai servizi protetti dal sistema oggetto di appalto. Quest'ultimo successivamente verificherà quanto dichiarato ed in caso di esito positivo fornirà accesso al servizio richiesto. In caso negativo, invece, l'utente verrà inserito in una lista nera che ne impedirà a priori futuri accessi al sistema.

1.3.4.

[25] Di fatto si tratta di un dominio *mediatore* o *aggregatore*.

2.2.2.4 Basi di dati utilizzate per l'autorizzazione

Archivio degli attributi (AA). Tale base di dati contiene le assegnazioni di ruoli agli utenti. Essa viene continuamente alimentata tramite cooperazione applicativa dagli enti detti *certificatori di ruolo*.

AA esterne. È prevista anche la possibilità di interrogare esplicitamente banche dati esterne per ottenere qualifiche degli utenti. Le comunicazioni con tali basi di dati possono avvenire al di fuori del paradigma della cooperazione applicativa[26].

Archivio dei permessi[27]. Contiene le associazioni tra i ruoli e la lista di servizi permessi per ciascuno di essi. I ruoli possono avere un'organizzazione gerarchica (requisito non obbligatorio).

Liste nere e liste bianche. Il sistema deve supportare l'accesso incondizionato in caso di appartenenza del soggetto ad una lista bianca (*white list*). Analogamente l'accesso deve essere negato a priori in caso di presenza dello stesso in una lista nera (*black list*). Il sistema deve fornire inoltre l'interfaccia di provisioning per gestire tali liste.

2.2.2.5 Il motore di profilazione SRTY

Il sistema oggetto di bando deve integrarsi con il motore di profilazione SRTY [SrtyToscana], sviluppato internamente da Regione Toscana. SRTY può essere visto come un componente che si antepone al servizio e ne filtra i contenuti in funzione del profilo[28] dell'utente che vi accede.

[26] È richiesto tuttavia che le basi di dati esterne comunichino secondo lo standard dei Web Services.

[27] Il nome utilizzato per questo componente in [BandoToscana] è in realtà "Repository dei ruoli", ma per uniformità di linguaggio è stato qui rinominato.

[28] Tale profilo sarà stato ottenuto dal sistema di autorizzazione.

Conforme al design pattern Model-View-Controller (MVC)[29], SRTY prevede che le viste possano dipendere, oltre che dalla vista precedente e dall'evento che le ha innescate, anche dal 'livello di visibilità' delle informazioni in esse contenute e delle interazioni[30] che presentano all'utente.

Tale livello di visibilità è elaborato da SRTY in funzione del profilo utente e viene implementato tramite l'uso di fogli di stile XSL. SRTY pertanto assume che l'output di ogni elaborazione del modello sia un documento XML che va successivamente filtrato, sia in funzione del profilo utente che della tipologia di terminale o il formato di output richiesto[31].

2.2.3 Regione Campania: SPICCA

Sistema Pubblico di Interoperabilità e Cooperazione applicativa CAmpana (SPICCA) [BandoCampania] è un progetto inizialmente presentato a marzo 2004 dalla Regione Campania per la cooperazione ed integrazione applicativa inter-istituzionale. Il progetto è all'incirca contemporaneo alla standardizzazione definitiva della Busta di e-Government[32] ad opera del CNIPA e ne sfrutta i risultati[33]. SPICCA utilizza inoltre molti concetti presenti anche in SPC[34] ma con terminologia differente dato che quest'ultimo era ancora in corso d'opera.

L'obiettivo di SPICCA e quello di dare a circa 250.000 utenti accesso a circa 10.000 servizi erogati dalle istituzioni centrali e locali della regione. Le disomogeneità tra i servizi in termini di meccanismi di accesso, formato dei dati, ecc. devono essere mediate da un componente centrale di aggregazione.

[29] cf. [POSA].

[30] È infatti inutile e spesso indesiderato presentare all'utente azioni che non è autorizzato a compiere.

[31] Si pensi ad esempio ad un utente che via browser richiede una 'versione stampabile' di un documento.

[32] cf. 2.1.1.3.

[33] All'epoca già disponibili sotto forma di bozza.

[34] cf. 2.1.1.

Quest'ultimo ha inoltre il compito di integrare tali servizi in nuovi servizi composti[35], laddove possibile.

Una decisa enfasi è stata usata da Regione Campania nel richiedere l'utilizzo del paradigma Open Source per le implementazioni di SPICCA, laddove possibile. In particolare è vincolante la richiesta di sviluppo Open Source dei componenti di gestione della Busta di e-Government e del proxy applicativo. In questo modo Regione Campania si assicura la possibilità di riusare ed estendere tali componenti man mano che le esigenze si evolvono nel tempo[36].

2.2.3.1 Integrazione ed Interoperabilità

Il modello SPICCA prevede due tipologie di nodo funzionale, corrispondente alla PdD del modello SPC: il Nodo di AGgregazione (NAG) ed il Nodo di DOMinio (NDOM).

Per il NAG sono previste le seguenti macro-funzionalità:

Presentazione di servizi. Dal punto di vista degli utenti, il nodo omogenizza ed integra i servizi forniti da diversi enti locali in modo che abbiano un formato standard e possano essere forniti su più ampia scala. il NAG inoltre combina più servizi locali in nuovi servizi complessi.

Broker di servizi. Dal punto di vista degli enti fruitori e/o erogatori, il nodo fornisce funzionalità di intermediazione tra domanda e offerta (*brokerage*), di integrazione e di bilanciamento del carico (dei nodi locali).

I NDOM, invece, sono sostanzialmente dei proxy applicativi che permettono, attraverso eventuali *adapters*, la partecipazione di SIL (anche legacy) alla cooperazione applicativa. Tali nodi possono erogare/fruire servizi direttamente e/o avvalersi delle facilities di integrazione dei NAG.

La figura 2.4 mostra l'architettura logica di SPICCA. Le comunicazioni tra NDOM e NAG avvengono sempre nel rispetto dello standard CNIPA sulla

[35] Si pensi ad esempio alle applicazioni statistiche o demografiche.

[36] cf. 2.1.2.

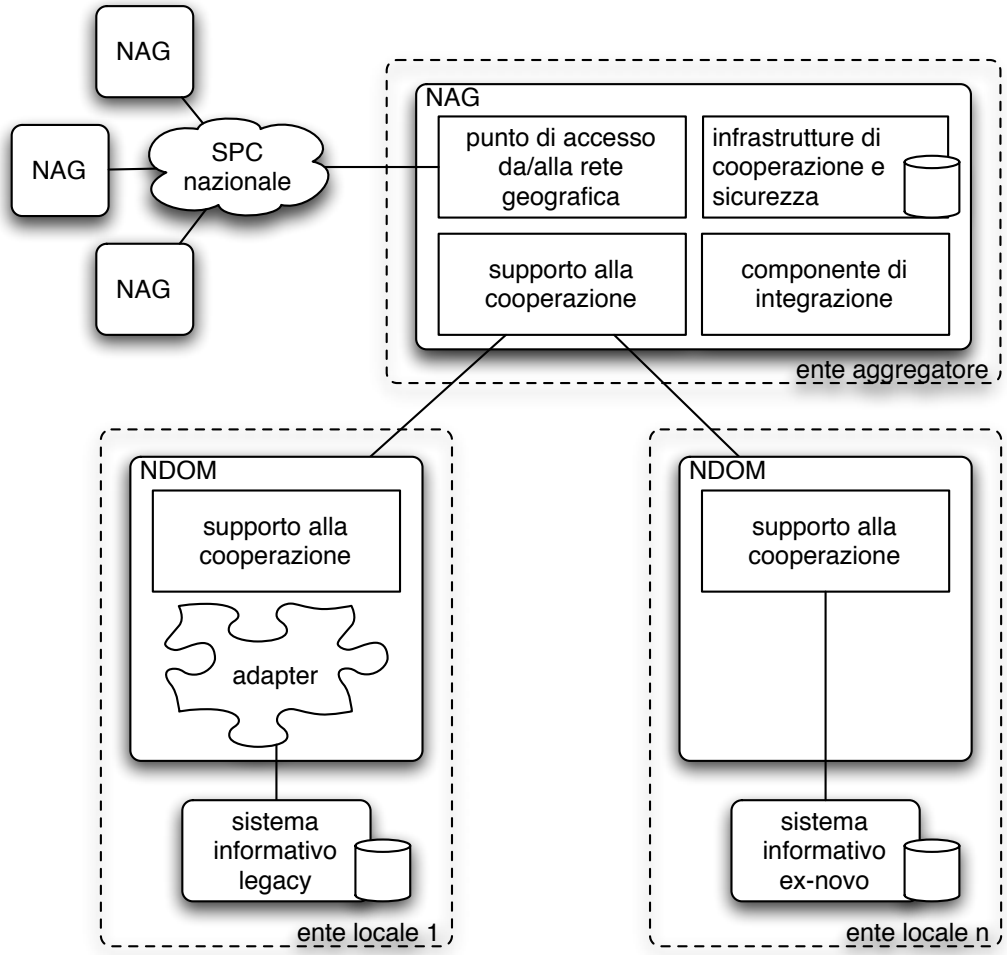

Figura 2.4: Modello architetturale di SPICCA.

Busta di e-Government[37] e quindi secondo il paradigma dei Web Services. Il NAG ha la possibilità di utilizzare le stesse modalità di comunicazione anche nei confronti di altri NAG. Esso inoltre espone un'interfaccia web accessibile direttamente dagli utenti finali attraverso un browser.

Le infrastrutture messe a disposizione dei NDOM dal NAG possono essere suddivise nelle seguenti categorie:

- integrazione ed interoperabilità (proxies applicativi, registry UDDI dei servizi, gestore eventi, ecc.),

[37] cf. 2.1.1.3.

- realizzazione di servizi aggregati,
- gestione della sicurezza,
- interfacciamento con le diverse tipologie di terminale (multicanalità),
- tracciabilità,
- monitoraggio della qualità dei servizi e del rispetto dei Service Level Agreement (SLA).

2.2.3.2 Gestione della sicurezza

I requisiti di SPICCA in merito di controllo di accesso, profilazione degli utenti e provisioning sono di particolare interesse per questa tesi e pertanto verranno trattati in maggiore dettaglio in questa sezione. Il modello propone infatti un'architettura di sicurezza complessa e distribuita tra NAG e NDOM che integra il controllo di accesso vero e proprio con altre problematiche di IM, come la presentazione di contenuti differenziati in funzione delle credenziali e scenari avanzati di provisioning.

Policies distribuite. SPICCA prevede la possibilità che l'accesso ad una determinata risorsa o servizio possa essere deciso in base a più policies distribuite. In particolare ogni NDOM può esercitare una propria policy in aggiunta alla policy centrale del NAG.

Modello di controllo di accesso. Il modello sceglie RBAC come paradigma di controllo di accesso. Esso permette la separazione tra l'assegnazione di permessi di accesso, operata dagli erogatori, dal provisioning di ruoli agli utenti, operata da una qualsiasi autorità che abbia conoscenza dell'utente e rapporti di trust con gli erogatori.

SPICCA non impone l'adozione di un modello RBAC gerarchico[38], pertanto non è previsto al momento che i ruoli abbiano relazioni di ereditarietà.

[38] cf. 1.3.4.

Tipologie di ACI supportate. Il modello SPICCA specifica l'utilizzo di una molteplicità di ruoli, privilegi ed informazioni contestuali (in generale ACI) in funzione dei quali autorizzare o meno l'accesso di un soggetto (utente o servizio fruitore):

- meccanismo di autenticazione: 'debole' (login/password, certificato su disco) o 'forte' (smart card, CIE, CNS, dispositivi biometrici);
- tipologia di terminale (canale): browser web, client applicativo, servizio, telefono cellulare, sportello elettronico;
- localizzazione: dominio o territorio di appartenenza, indirizzo IP, ecc.;
- attributi o ruoli assegnati al soggetto da un'autorità riconosciuta dal decisore;
- presenza del soggetto in una *white list* o *black list*;
- marcature temporali emesse da una Time Stamping Authority (TSA).

Archiviazione e ricupero dell'ACI. SPICCA non effettua una scelta tecnologica vincolante per quanto riguarda l'immagazzinamento e l'emissione delle ACI, ma propone un modello che prevede un insieme di autorità responsabili per le rispettive ACI. Fanno eccezione i certificati X.509 che dovranno essere pubblicati in un directory LDAP.

Le tipologie di autorità previste dal modello sono le seguenti:

CA certifica l'autenticità dei soggetti, sarà tipicamente conforme allo standard X.509;

Attribute Authority (AA) certifica le qualifiche, i ruoli, gli attributi del soggetto;

TSA certifica l'esistenza di un dato in (prima di) un certo istante;

Registration Authority certifica su delega di una CA o AA.

Tali autorità possono essere centrali o distribuite presso i NDOM ed, in quest'ultimo caso, avere una struttura gerarchica o basata su accordi di trust bilaterali o federativi.

Nel caso delle AA, SPICCA fornisce delle indicazioni su come gli attributi possano essere memorizzati all'interno di DB ad hoc, ovvero nel certificato (X.509) di identità, ovvero in un apposito AC. Si prevede inoltre sia la possibilità di avere AC asserenti un singolo ruolo, che AC 'aggregati' contenenti tutti i ruoli attribuibili ad un soggetto.

Modello delle interazioni per il controllo di accesso. Il diagramma in figura 2.5 illustra gli attori coinvolti in un'operazione di autorizzazione presso un NAG.

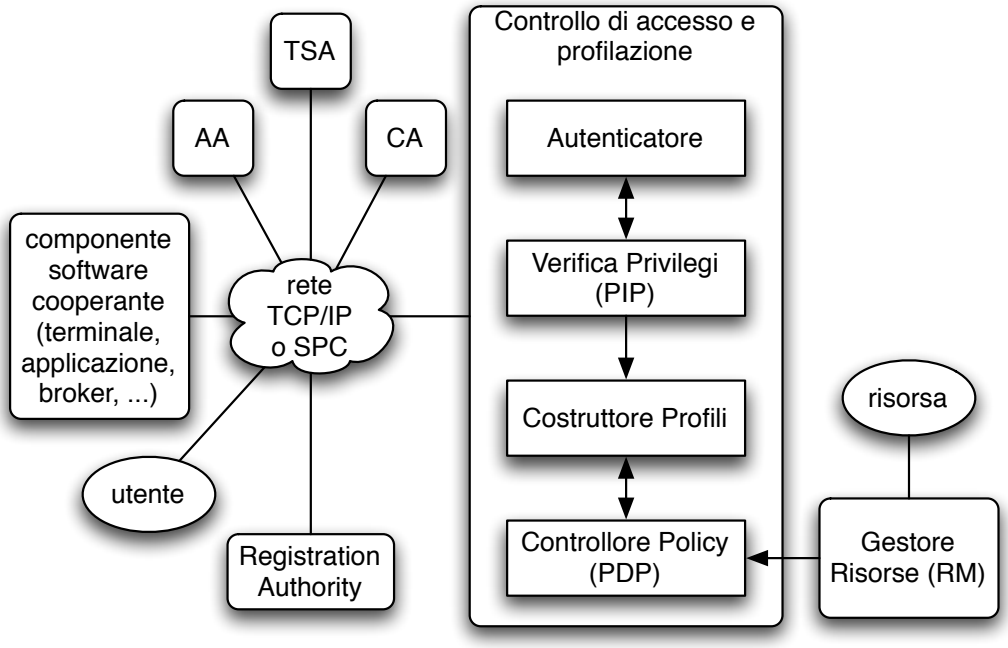

Figura 2.5: Modello concettuale del controllo di accesso in SPICCA.

In SPICCA, l'autorizzazione vera e propria e la generazione di contenuti personalizzati in funzione del profilo utente sono (logicamente) fuse in un'unica operazione, eseguita dal *Controllore di Policy*. Tale profilo sarà a sua

volta stato generato dal *Costruttore di Profili* a partire dalle credenziali dell'utente recuperate[39] e validate dai componenti *Verificatore di Privilegi* (nel caso di attributi) ed *Autenticatore* (nel caso di credenziali di autenticazione). Quest'ultimo infine ha anche il compito di eseguire il particolare meccanismo di autenticazione scelto per l'utente.

Il componente logico denominato in figura "componente software cooperante" rappresenta lo strato di software[40] che separa l'utente dal servizio esposto presso il NAG. In altre parole si tratta del componente che:

- utilizza certificati a chiave pubblica;
- utilizza certificati d'attributo;
- interagisce con l'Autenticatore per autenticarsi;
- interagisce con il Verificatore di privilegi per rivestire un ruolo;
- effettua richieste gestite dal resource manager e filtrate dal Controllore di Policy.

Ad ogni sessione, al soggetto è associata una Active Role List (ARL) che definisce i ruoli correntemente rivestiti. La ARL costituisce di fatto l'output del Costruttore di Profili. Per assumere un determinato ruolo il soggetto interagisce con il Verificatore di privilegi.

Mediazione dei meccanismi di sicurezza dei NDOM. Alla federazione di domini attuata da un NAG possono partecipare NDOM con meccanismi e policies di sicurezza disomogenei. Pertanto sul NAG ricade anche la responsabilità di mediare i meccanismi di sicurezza laddove necessario, nonché di verificare la compatibilità delle policies dei vari domini.

[39] È previsto che le credenziali di autenticazione e gli attributi possano sia essere fornite dal "componente software cooperante" (caso *push*) che ricuperate dal *Verificatore di Privilegi* interrogando opportune autorità (caso *pull*).

[40] Si pensi ad un browser web unito alla logica di creazione delle sessioni dal lato server, oppure ad un client applicativo, oppure al NDOM fruitore.

2.3 Sommario dei requisiti

In questa sezione vengono classificati e formalizzati i requisiti comuni (o comunque interessanti) ottenuti dall'analisi effettuata nella restante parte del capitolo (cf. sezioni 2.1, 2.2). L'architettura delineata nel capitolo 4 presenterà un insieme di soluzioni finalizzate al soddisfacimento, in toto o in parte, di questi requisiti.

2.3.1 Non-requisiti sui terminali per l'accesso ai servizi di e-Government

È buona pratica, nell'attuale momento tecnologico, quella di imporre la minor quantità di vincoli possibile sulle componenti hardware e software dei terminali utente. In uno scenario di servizi al cittadino i terminali sono infatti il componente architetturale di cui sono previste il maggior numero di istanze. Pertanto, anche la minima modifica per rendere i terminali maggiormente integrati con il resto dell'architettura ha un'impatto sui costi proporzionale al numero di utenti. L'approccio deve invece essere quello di adattare il resto dell'architettura ai terminali preesistenti, cercando inoltre di sfruttare le funzionalità specifiche delle singole tipologie di piattaforma client.

2.3.1.1 Indipendenza dal terminale

Le soluzioni proposte non dovrebbero imporre modifiche o aggiornamenti ai terminali client correntemente in uso per applicazioni analoghe[41]. La scelta della tipologia di terminale dovrebbe avvenire in base al principio del minimo costo e della massima diffusione sul mercato (prodotti *off the shelf*) in un regime di libera concorrenza. È un requisito implicito in tutti i progetti, i bandi e le standardizzazioni analizzati.

[41] Ad esempio browsers HTTP per applicazioni basate su web.

2.3.1.2 Multicanalità

Si deve cercare di sfruttare le potenzialità specifiche delle particolari tipologie di terminali disponibili presso l'utenza. Si deve cercare di adattare i formati di presentazione delle informazioni alle peculiarità del terminale ed alle preferenze dell'utente. Requisito presente in entrambi i bandi analizzati (cf. 2.2.2, 2.2.3).

2.3.1.3 Indipendenza da particolari applicativi ad hoc sui terminali

Lo sviluppo di applicazioni ad hoc da installare sul terminale è da evitare se possibile[42], in favore di applicativi di uso generico ed ampia diffusione[43].

2.3.1.4 Indipendenza da dispositivi di sicurezza ad hoc

Sebbene in questi anni il Governo stia promuovendo l'adozione di dispositivi hardware di autenticazione basati sulla tecnologia delle *smart cards*, la diffusione di tali dispositivi non ha ancora raggiunto un livello tale da poterli considerare strumentali per l'accesso ai servizi. Inoltre, anche in uno scenario di diffusione maggiore delle smart cards, potrebbe continuare ad essere desiderabile l'uso di altri meccanismi di autenticazione.

Pertanto, nelle soluzioni proposte si deve astrarre il più possibile dal particolare meccanismo software o hardware utilizzato per l'autenticazione. Requisito esplicito dei progetti e dei bandi della sezione 2.2.

[42] Nonostante esistano tecnologie come *Java Web Start* [JavaWebStart] che permettono di agevolare i compiti di distribuzione, installazione ed aggiornamento degli applicativi rendendoli trasparenti all'utente.

[43] Ad esempio un browser web od un client di posta elettronica.

2.3.2 Paradigmi di controllo di accesso e profilazione avanzati

Dalle analisi effettuate è emerso quanto possano essere complesse le decisioni di accesso relativamente alle problematiche di tipo amministrativo. Esse necessitano quindi di modelli di autorizzazione evoluti che da un lato permettano di esprimere la complessità di tali decisioni, dall'altro ne semplifichino la gestione da parte delle amministrazioni.

2.3.2.1 Necessità di autorizzazione basata su ruoli (RBAC)

Il requisito di un modello di controllo di accesso basato su qualifiche o ruoli attribuiti agli utenti (RBAC[44]) è espressamente formulato in tutti i progetti e i bandi della sezione 2.2.

2.3.2.2 Necessità di integrazione tra il controllo di accesso e la profilazione

Nei bandi analizzati (cf. 2.2.2, 2.2.3) è espresso il requisito di utilizzare la stessa infrastruttura di autenticazione ed attribuzione di ruoli anche per fornire all'utente contenuti personalizzati in funzione delle sue qualifiche ('diritti di visibilità') e/o preferenze.

2.3.2.3 Importanza di un linguaggio formale per esprimere le policies di sicurezza

È un requisito desiderabile di un qualsiasi motore di autorizzazione di un certo calibro che la logica di controllo di accesso (la policy) sia separata dal resto della *business logic* e sia rappresentata in un linguaggio formale specifico (distinto dal linguaggio di implementazione).

Una tale caratteristica permette la separazione dei compiti tra gli sviluppatori e gli addetti al provisioning ed abilita scenari di autorizzazione

[44] cf. 1.3.4.

complessi in cui la decisione di accesso può essere distribuita presso decisori esterni al dominio della risorsa richiesta.

Nel caso specifico della cooperazione applicativa (cf. 2.1.1) è auspicabile che tali policies vengano integrate all'interno degli accordi di servizio, in modo che le decisioni di accesso possano essere effettuate direttamente presso i domini fruitori, come tra l'altro è esplicitamente richiesto nel bando di Regione Toscana (requisito 2.3.3.1).

2.3.3 Distribuzione locale delle competenze e delle responsabilità

In tutti i documenti analizzati in questo capitolo emerge la necessità di una gestione della sicurezza distribuita tra i domini. I requisiti di questa classe sono collegati a molti di quelli della categoria 2.3.2.

2.3.3.1 Autenticazione, qualificazione ed autorizzazione presso il dominio di competenza od un'autorità delegata

Le operazioni di autenticazione, attribuzione di ruoli (qualificazione) ed autorizzazione devono avvenire presso il dominio responsabile per il soggetto o per la particolare qualifica[45] o autorizzazione[46]. Un dominio può delegare questo compito ad un'altro dominio che ne condivida il bacino d'utenza. Un dominio che necessiti di un'autenticazione/qualificazione/autorizzazione di competenza di un'altro dominio dovrebbe redirigere l'utente in maniera trasparente presso il dominio di competenza. Requisito espresso esplicitamente nel progetto ICAR (cf. 2.2.1), ma desiderabile anche negli altri casi.

[45] Si pensi ad esempio ad un albo professionale.

[46] Si pensi ad esempio ad un atto amministrativo che abiliti l'accesso a servizi od informazioni.

2.3.3.2 Necessità di standardizzare i tokens di sicurezza

Perché i domini possano accettare come valide le autenticazioni, le qualificazioni e le autorizzazioni emesse da altri domini, si rendono necessari dei formati di token[47] globalmente riconosciuti. Tali formati dovranno inoltre essere dotati dei meccanismi crittografici necessari a garantirne almeno la autenticità ed opzionalmente la confidenzialità. Requisito esplicito nei documenti di SPC (cf. 2.1.1) e nel progetto ICAR (cf. 2.2.1).

2.3.3.3 Un sistema federato di autenticazione, qualificazione ed autorizzazione

Il progetto ICAR (cf. 2.2.1) prevede espressamente la realizzazione di un sistema federato di autenticazione, qualificazione ed autorizzazione. Per 'federazione' si intende una comunità di domini che accettano le seguenti condizioni:

- considerare valide le autenticazioni, le qualificazioni e le autorizzazioni emesse dagli altri domini, purché all'interno delle rispettive competenze;
- effettuare autenticazioni, qualificazioni ed autorizzazioni in favore degli utenti di propria competenza, su richiesta di altri domini o degli utenti stessi.

2.3.3.4 Indipendenza dai meccanismi di sicurezza dei singoli domini

I singoli domini devono poter scegliere indipendentemente i meccanismi di autenticazione/qualificazione/autorizzazione considerati maggiormente adat-

[47] Il termine token a cui si fa qui riferimento può anche essere una credenziale effimera 'derivata' a partire da una a lungo termine, ad esempio un'asserzione SAML (cf. 3.1.1) emessa in seguito ad un'autenticazione avvenuta usando un certificato X.509. In effetti questa è l'unica accezione del termine token che permette la compatibilità di questo requisito con i requisiti 2.3.1.4, 2.3.3.4 e 2.3.3.5.

ti alle esigenze interne e di cooperazione. Requisito espresso nei documenti di SPC (cf. 2.1.1), del progetto ICAR (cf. 2.2.1) ed del bando di Regione Campania (cf. 2.2.3).

2.3.3.5 Mediazione dei meccanismi di sicurezza dei singoli domini

Qualora venga rispettato il requisito 2.3.3.4, si deve tuttavia poter rendere interoperabili i diversi meccanismi di sicurezza, attraverso l'utilizzo di opportuni componenti architetturali, siano essi centrali o sotto forma di adattatori applicati ai meccanismi dei singoli domini. È un requisito esplicito del bando di Regione Campania (cf. 2.2.3).

2.3.4 Integrazione con le infrastrutture esistenti

Molte delle architetture descritte nei documenti di questo capitolo prevedono l'integrazione con infrastrutture preesistenti.

2.3.4.1 Necessità di integrazione con le PKI preesistenti

Essendo lo standard di fatto per quanto riguarda l'autenticazione 'forte', tutti i documenti analizzati prevedono (in varia misura) il supporto di PKI X.509.

2.3.4.2 Necessità di integrazione con SPC

Sia il progetto ICAR (cf. 2.2.1) che il bando di Regione Toscana (cf. 2.2.2) richiedono esplicitamente l'integrazione delle soluzioni realizzate con le infrastrutture di SPC.

2.3.4.3 Necessità di integrazione con le basi di dati preesistenti

I dati applicativi sono per le amministrazioni una risorsa spesso più preziosa delle applicazioni stesse. Pertanto le soluzioni proposte dovranno ridurre al minimo l'impatto sugli stessi implementando opportuni adattatori laddove necessario. Requisito espresso in forma variamente esplicita in tutti i documenti analizzati.

2.3.5 Sviluppo di nuove infrastrutture

Dai documenti analizzati emerge l'esigenza di sviluppare specifiche infrastrutture a supporto delle nuove problematiche di IM gestite. A seconda delle specifiche esigenze, tali infrastrutture possono essere (logicamente) centralizzate o distribuite presso i domini di competenza.

2.3.5.1 Necessità di infrastrutture per l'attribuzione di ruoli

La soddisfazione del requisito 2.3.2.1 prevede l'esistenza di apposite infrastrutture per l'attribuzione certificata di ruoli ai soggetti della comunità. È un requisito espresso esplicitamente in tutti i bandi ed i progetti analizzati in questo capitolo (cf. 2.2).

2.3.5.2 Archivio dei permessi

Nel bando di Regione Toscana (cf. 2.2.2) viene esplicitamente richiesta l'implementazione di una base di dati che associ le azioni ammissibili ai ruoli, ovvero un *policy repository*. Una tale infrastruttura è implicitamente richiesta anche nel bando di Regione Campania (cf. 2.2.3).

2.3.5.3 Liste nere e liste bianche

L'appartenenza di un soggetto ad una di queste liste conduce tout court ad una decisione di autorizzazione (negativa per le prime, positiva per le seconde) senza bisogno di completare l'intero processo. Tali liste possono essere gestite in maniera cooperativa od indipendente dai vari domini. Requisito espresso esplicitamente nel bando di Regione Toscana (cf. 2.2.2), ma utile anche altrove.

2.3.5.4 Time Stamping Authority (TSA)

Nel bando di Regione Campania (cf. 2.2.3) viene richiesta la presenza di tali componenti. Le TSA sono fortemente raccomandate nelle PKI che supportano la revoca di certificati. Esse sono necessarie infatti per effettuare il *roll*

back nei casi in cui i certificati fossero stati utilizzati successivamente alla revoca senza che ne venisse contestualmente verificato lo stato di validità.

Le TSA possono tuttavia essere utilizzate anche per integrare funzionalità di marcatura temporale in applicativi che non le supportino nativamente.

2.3.6 Requisiti sul modello e le metodologie di sviluppo

Questa categoria raccoglie requisiti relativi a caratteristiche trasversali rispetto alle problematiche in esame ma che a seconda dei casi possono essere determinanti nella scelta della soluzione.

2.3.6.1 Preferibilità del modello Open Source, per nuove realizzazioni o per la scelta di componenti esistenti

Il modello di sviluppo Open Source è particolarmente appetibile per la realizzazione di progetti sperimentali, in particolar modo per quelli relativi a tecnologie di comunicazione (in senso lato). Pertanto l'adozione di tale modello per lo sviluppo di nuovi componenti è consigliabile. Una tale accezione di questo requisito compare nel progetto ICAR (cf. 2.2.1) e parzialmente nel bando di Regione Campania (cf. 2.2.3).

Per i motivi descritti in dettaglio nella sezione 2.1.2 è inoltre desiderabile la scelta di componenti Open Source preesistenti (pronti all'uso o da integrare) in fase di progettazione.

2.3.6.2 Riusabilità del software prodotto

Nel progetto ICAR (cf. 2.2.1), nel bando di Regione Campania(cf. 2.2.3) e nelle analisi del paradigma Open Source del CNIPA (cf. 2.1.2) si esprime esplicitamente il requisito della libera riusabilità del software commissionato dalla PA in altri domini applicativi o presso amministrazioni distinte da quelle appaltanti. Si tratta di una qualità desiderabile in una qualsiasi architettura

distribuita dotata di molte istanze dello stesso componente (è ad esempio il caso di SPC e SPICCA).

2.3.6.3 Predisposizione per lo sviluppo collaborativo

Nei progetti collaborativi come ICAR (cf. 2.2.1), vari enti contribuiscono alla stesura ed all'implementazione della stessa architettura. È pertanto desiderabile che le attività di sviluppo e la progettazione siano ben coordinate e che le competenze siano adeguatamente ripartite.

Capitolo 3

Standards e tecnologie di rilievo

In questo capitolo viene presentata una selezione degli standards e delle tecnologie pertinenti alle problematiche di Identity Management. Detta selezione è stata effettuata in base ai requisiti della Pubblica Amministrazione delineati nel capitolo 2.

Le soluzioni presentate verranno utilizzate in varia misura nel disegno architetturale del capitolo 4, alcune completamente[1], altre parzialmente[2]. Di alcune di esse, inoltre, si userà il solo modello architetturale[3], mentre altre[4] serviranno per un'analisi comparativa di altre soluzioni.

Gli standards e le tecnologie di questo capitolo sono suddivisi in tre categorie che riflettono altrettante problematiche fondamentali dell'area dell'Identity Management: il formato e lo scambio di credenziali, il Single Sign On ed il controllo di accesso, detto anche autorizzazione.

[1] SAML, cf. 3.1.1; Shibboleth, cf. 3.2.1.
[2] X.509, cf. 3.1.2; PERMIS, cf. 3.3.1.
[3] Globus Toolkit 4, cf. 3.3.2.
[4] Liberty Alliance, cf. 3.2.2.

3.1 Standards sul formato e lo scambio di credenziali

La *credenziale*, nell'accezione di "documento certificato comprovante l'identità o altre qualifiche di un soggetto"[5], è l'elemento informativo centrale nelle problematiche di Identity Management. Qualsiasi soluzione di Identity Management – per definizione – si trova a dover gestire delle identità digitali, ed ognuna di queste ultime non può essere ritenuta tale senza un insieme di credenziali che la supporti.

Pertanto è evidente la necessità di una buona standardizzazione e diffusione sul mercato delle tecnologie relative a questo aspetto, prima ancora della qualità intrinseca di esse. È per questo motivo che uno standard, per molti aspetti *legacy* e limitato come X.509[6], è tuttora il più utilizzato in ambito Internet e non, per l'autenticazione 'forte' di identità digitali.

Uno dei limiti maggiori di X.509 è la visione fortemente gerarchica dei soggetti (attivi e passivi) coinvolti nell'autenticazione e delle rispettive relazioni di fiducia *trust*, in contrapposizione a modelli decentrati, basati sul paradigma del *Web of Trust*[7], che si vanno affermando soprattutto nelle comunità Internet, ma anche nel mondo istituzionale.

Lo standard SAML[8] viene qui proposto come compromesso tra la visione completamente 'verticale' e quella 'orizzontale' del processo di certificazione, che va sotto il nome di approccio *federativo*[9]. Tale obiettivo viene raggiunto specificando un formato di credenziale, indipendente dal particolare metodo

[5]Nel seguito, tuttavia, verrà utilizzato il termine *credenziale* per denotare certificati di identità. Per i certificati di qualifica si utilizzeranno i termini di *attributo* nel caso di una qualità intrinseca del soggetto e di *autorizzazione* nel caso di permessi esplicitamente richiesti e consentiti.

[6]cf. 3.1.2.

[7]Si tratta di tecnologie che basano il trust sulla rete di contatti di un soggetto (la sua affidabilità è in funzione della quantità e della qualità dei soggetti che lo certificano) piuttosto che sulla certificazione o meno da parte di una autorità centrale.

[8]cf. 3.1.1.

[9]Si veda in proposito la tassonomia presentata nella sezione 1.1.2.

di autenticazione, che permetta ad un'autorità di *asserire* qualcosa su un soggetto. Il fatto che una seconda autorità riconosca o meno una tale asserzione è in funzione del rapporto di trust che ha con la prima e non è imposto dall'alto.

3.1.1 SAML

Security Assertion Markup Language (SAML) è uno standard basato su XML per la creazione e la comunicazione di tokens di sicurezza. Esso è stato prodotto, a partire dal 2001, dal Security Services Technical Commitee [SAML-web] di OASIS. Di recente è stata rilasciata la versione 2.0 dello standard, ma la maggior parte delle implementazioni – sia di SAML stesso sia di applicazioni che ne fanno uso – utilizzano ancora la versione 1.1 ratificata il 2 Settembre 2003. La più diffusa implementazione Open Source di SAML è OpenSAML [OpenSAML-web] che, allo stato attuale, copre la quasi totalità della versione 1.1 dello standard.

SAML ha come obiettivo principale la creazione di un framework di supporto per la problematica del SSO e, più in generale, dell'IM. Tuttavia, nel corso degli anni, sono emersi altri utilizzi dello standard come ad esempio la restrizione di deleghe in CAS (cf. 3.3.2.4) o la sicurezza di messaggi SOAP (cf. 3.1.1.5).

Lo standard si può suddividere in quattro parti essenziali: la specifica del formato delle credenziali, dette *asserzioni*, i protocolli (astratti) per comunicarle, i bindings e i *profili*[10].

[10]Traduzione letterale del termine inglese *profiles*, sta ad indicare il processo di inserzione (*embedding*) ed estrazione di asserzioni SAML in protocolli od oggetti di vario tipo (ad es. files o documenti web). Da non confondersi con il *profilo utente*.

3.1.1.1 Asserzioni

Nel documento [SAML-core] viene specificato il formato per tre diversi tipi di asserzione: di autenticazione, di attributi e di autorizzazione. Un esempio[11] di asserzione di autenticazione si può avere nel listato 3.1:

Listing 3.1: Un'asserzione di autenticazione

```
<Assertion
   xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
   AssertionID="_d17b16a4142d8bc0acab305788bda3d2"
   IssueInstant="2005-08-07T10:55:54.339Z"
   Issuer="https://idp.example.org/shibboleth"
   MajorVersion="1"
   MinorVersion="1">
  <Conditions
     NotBefore="2005-08-07T10:55:54.339Z"
     NotOnOrAfter="2005-08-07T11:00:54.339Z">
    <AudienceRestrictionCondition>
      <Audience>https://sp.example.org/shibboleth</Audience>
      <Audience>urn:mace:shibboleth:examples</Audience>
    </AudienceRestrictionCondition>
  </Conditions>
  <AuthenticationStatement
     AuthenticationInstant="2005-08-07T10:55:54.339Z"
     AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:unspecified">
    <Subject>
      <NameIdentifier
         Format="urn:mace:shibboleth:1.0:nameIdentifier"
         NameQualifier="https://idp.example.org/shibboleth">
        _d2da81fa476cb7a693e03d353ca601a5
      </NameIdentifier>
      <SubjectConfirmation>
        <ConfirmationMethod>
          urn:oasis:names:tc:SAML:1.0:cm:bearer
        </ConfirmationMethod>
      </SubjectConfirmation>
    </Subject>
    <SubjectLocality IPAddress="127.0.0.1"/>
  </AuthenticationStatement>
  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <!-- firma digitale omessa per ragioni di spazio -->
  </ds:Signature>
</Assertion>
```

L'elemento `Assertion` contiene l'istante di emissione (`IssueInstant`), l'identificativo dell'autorità emittente (`Issuer`) ed un identificatore unico utile

[11] Questo esempio, come gli altri di questo capitolo, è stato ottenuto dalle asserzioni effettivamente processate nelle fasi di test dal prototipo implementato.

per riferire l'asserzione (`AssertionID`). Annidato all'interno di `Assertion`, l'elemento `Conditions` impone le condizioni di validità dell'asserzione, tra le quali l'istante di entrata in vigore (`NotBefore`) e quello di scadenza (`NotOnOrAfter`). Per rendere l'asserzione una credenziale vera e propria tale elemento può essere firmato utilizzando lo standard XML Signature specificato dal W3C in [XMLSig].

L'elemento `Subject` fornisce l'identificazione del soggetto di cui si asserisce: un nome (`NameIdentifier`) assegnato dall'assertore (`NameQualifier`), di cui è specificato anche il formato (`Format`). L'elemento annidato `SubjectConfirmation` merita una menzione particolare in quanto specifica la procedura che il **ricevente** dell'asserzione può utilizzare per avere conferma del fatto che essa è effettivamente stata emessa su richiesta, diretta o indiretta, del soggetto indicato. Nel caso esemplificato il valore di tale elemento indica che il ricevente deve fidarsi del **portatore** (*bearer*) dell'asserzione: questo metodo è particolarmente utile nelle problematiche di SSO e, più in generale, nei casi in cui siano necessarie credenziali riusabili dall'utente stesso. Un altro utilizzo dell'elemento `SubjectConfirmation` è come contenitore[12] della chiave pubblica del soggetto. In questo caso il ricevente potrà innescare un'interazione di tipo *challenge/response* direttamente con il soggetto per avere conferma del suo legame con la chiave ricevuta nell'asserzione.

Il contenuto informativo specifico delle asserzioni è incapsulato nei cosiddetti *statements*. SAML specifica tre formati di statement predefiniti: `AuthenticationStatement`, `AttributeStatement` ed `AuthorizationDecisionStatement`, corrispondenti alle tre tipologie di asserzione citate pocanzi.

Nel caso degli statement di autenticazione, le informazioni di rilievo sono: l'istante in cui il soggetto si è autenticato (`AuthenticationInstant`, in generale antecedente ad `IssueInstant`), il particolare processo di autenticazione utilizzato (`AuthenticationMethod`) e l'indirizzo Internet del soggetto al momento dell'autenticazione (`SubjectLocality`).

[12] tramite l'elemento annidato `ConfirmationData`.

L'utilizzo previsto per le asserzioni di autenticazione è quello di certificare l'autenticazione del soggetto avvenuta tramite un'interazione tra esso e l'assertore. Tale autenticazione avviene in un'interazione distinta ed antecedente a quella che ha portato all'emissione dell'asserzione. Nel caso del SSO, ad esempio, ad un singolo sign-on dell'utente possono corrispondere più asserzioni di autenticazione. Ciascuna delle asserzioni viene emessa se già esiste una sessione dell'utente presso l'assertore e non innescando una nuova autenticazione ad ogni richiesta di asserzione. Un utilizzo di quest'ultimo tipo non sarebbe tuttavia incompatibile con lo standard e potrebbe costituire una forma di SSO per applicazioni.

Il listato 3.2 contiene un'asserzione di attributi:

Listing 3.2: Un'asserzione di attributi

```
<Assertion
   AssertionID="_b35595a50e6b329bcb7523927742a384"
   IssueInstant="2005-08-07T10:55:55.010Z"
   Issuer="https://idp.example.org/shibboleth"
   MajorVersion="1"
   MinorVersion="1">
  <Conditions
     NotBefore="2005-08-07T10:55:55.010Z"
     NotOnOrAfter="2005-08-07T11:25:55.010Z">
    <AudienceRestrictionCondition>
      <Audience>urn:mace:shibboleth:examples</Audience>
    </AudienceRestrictionCondition>
  </Conditions>
  <AttributeStatement>
    <Subject>
      <NameIdentifier
         Format="urn:mace:shibboleth:1.0:nameIdentifier"
         NameQualifier="https://idp.example.org/shibboleth">
        _d2da81fa476cb7a693e03d353ca601a5
      </NameIdentifier>
      <SubjectConfirmation>
        <ConfirmationMethod>
          urn:oasis:names:tc:SAML:1.0:cm:bearer
        </ConfirmationMethod>
      </SubjectConfirmation>
    </Subject>
    <Attribute
       AttributeName="urn:mace:dir:attribute-def:eduPersonAffiliation"
       AttributeNamespace="urn:mace:shibboleth:1.0:attributeNamespace:uri">
      <AttributeValue>staff</AttributeValue>
    </Attribute>
    <Attribute
       AttributeName="urn:mace:dir:attribute-def:eduPersonPrincipalName"
```

```
34          AttributeNamespace="urn:mace:shibboleth:1.0:attributeNamespace:uri">
35        <AttributeValue Scope="example.org">tomcat</AttributeValue>
36      </Attribute>
37    </AttributeStatement>
38    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
39      <!-- firma digitale omessa per ragioni di spazio -->
40    </ds:Signature>
41  </Assertion>
```

In questo caso il contenuto informativo specifico dell'`AttributeStatement` è costituito da uno o più elementi `Attribute` contenenti il nome dell'attributo (`AttributeName`), il relativo vocabolario[13] di appartenenza (`AttributeNamespace`) ed un insieme di uno o più valori incapsulati in elementi `AttributeValue`.

Le asserzioni di attributi possono rendersi necessarie nel caso in cui i consumatori di tali token necessitino di informazioni qualificanti il soggetto non ottenibili attraverso asserzioni di autenticazione. Esse si possono vedere come un accesso mediato ai *data backend* degli assertori. L'utilizzo tipico degli attributi è come ACI[14], ma possono essere usati anche per creare un profilo utente.

Per completare la trattazione possiamo vedere un'asserzione di autorizzazione esemplificata nel listato 3.3:

Listing 3.3: Un'asserzione di autorizzazione

```
1  <Assertion
2     xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
3     AssertionID="_2c3a58bb026ba9976cf437cdd357c73a"
4     IssueInstant="2005-08-07T10:56:01.610Z"
5     Issuer="https://idp.example.org/shibboleth"
6     MajorVersion="1"
7     MinorVersion="1">
8    <Conditions
9       NotBefore="2005-08-07T10:56:01.609Z"
```

[13] SAML assume che gli attributi possano avere valori scelti all'interno di un insieme (finito) prestabilito. Il modo in cui questi vocabolari possono essere definiti non è trattato dalla versione 1.1 dello standard. Per avere esempi su come è stato soddisfatto nella pratica questo requisito si veda 3.2.1 ed i file di configurazione di Shibboleth nel pacchetto del prototipo.

[14] cf. 1.3.2.

```
10      NotOnOrAfter="2005-08-07T11:56:01.609Z"/>
11    <AuthorizationDecisionStatement
12       Decision="Permit"
13       Resource="https://sp.example.org/profile-webapp/private">
14      <Subject>
15        <Nameidentifier
16           Format="urn:mace:shibboleth:1.0:nameIdentifier"
17           NameQualifier="https://idp.example.org/shibboleth">
18          _d2da81fa476cb7a693e03d353ca601a5
19        </NameIdentifier>
20        <SubjectConfirmation>
21          <ConfirmationMethod>
22            urn:oasis:names:tc:SAML:1.0:cm:bearer
23          </ConfirmationMethod>
24        </SubjectConfirmation>
25      </Subject>
26      <Action
27         Namespace="urn:oasis:names:tc:SAML:1.0:action:ghpp">
28        POST
29      </Action>
30      <Evidence>
31        <!-- evidenze omesse per ragioni di spazio -->
32      </Evidence>
33    </AuthorizationDecisionStatement>
34    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
35      <!-- firma digitale omessa per ragioni di spazio -->
36    </ds:Signature>
37  </Assertion>
```

L'assertore qui dichiara che il soggetto specificato in `Subject` ha richiesto una decisione sull'esecuzione dell'azione in `Action` sulla risorsa `Resource`, che la decisione ha avuto esito `Decision`[15] e che essa è stata effettuata in base alle prove in `Evidence`.

Le azioni possono essere più di una ed hanno ciascuna il proprio vocabolario (`Namespace`), esattamente come avveniva nell'asserzione 3.2 per gli attributi. Nel caso esemplificato l'azione "POST" appartiene al vocabolario "ghpp"[16] che raccoglie alcune delle azioni possibili del protocollo HTTP. Quindi, essendo la risorsa una URL, la decisione di fatto autorizza il soggetto ad accedere ad una 'pagina' web.

Le prove o evidenze allegate possono essere una o più asserzioni SAML, individualmente firmate se ritenuto necessario. Si può pensare di includere le asserzioni 3.1 e 3.2 all'interno dell'elemento `Evidence` in 3.3. L'inclusione

[15] Di solito ha valore "Permit" o "Deny".

[16] {GET, HEAD, PUT, POST}.

di evidenze nelle asserzioni di autorizzazione può essere omessa, ad esempio per motivi di privacy.

Le asserzioni di autorizzazione costituiscono l'output di un PDP, in un formato che permette al PDP stesso di certificare la decisione e di allegare parte delle informazioni utilizzate. Queste 'licenze' hanno il vantaggio di poter essere invocate da soggetti terzi rispetto ai PEP e possono essere riusate per più di un accesso.

3.1.1.2 Protocolli

In [SAML-core] si definisce un semplice protocollo richiesta/risposta per convogliare richieste di asserzione e le corrispondenti asserzioni. Tale protocollo, conforme al Document Exchange Model (DEM), astrae dal particolare livello di *trasporto* sottostante. Un esempio di richiesta di attributi è illustrato nel listato 3.4:

Listing 3.4: Una richiesta di attributi

```
<Request xmlns="urn:oasis:names:tc:SAML:1.0:protocol"
   xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
   xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   IssueInstant="2005-08-10T22:07:59.498Z"
   MajorVersion="1"
   MinorVersion="1"
   RequestID="_7f17e2145de5e7735b2071efd3f5576f">
  <AttributeQuery
     Resource="https://sp.example.org/profile-webapp/private">
    <Subject xmlns="urn:oasis:names:tc:SAML:1.0:assertion">
      <NameIdentifier
         Format="urn:mace:shibboleth:1.0:nameIdentifier"
         NameQualifier="https://idp.example.org/shibboleth">
        _20e29bfb0c0dbb06dd524c1dbccba727
      </NameIdentifier>
      <SubjectConfirmation>
        <ConfirmationMethod>
          urn:oasis:names:tc:SAML:1.0:cm:bearer
        </ConfirmationMethod>
      </SubjectConfirmation>
    </Subject>
    <AttributeDesignator xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
       AttributeName="urn:mace:dir:attribute-def:eduPersonPrincipalName"
       AttributeNamespace="urn:mace:shibboleth:1.0:attributeNamespace:uri"/>
  </AttributeQuery>
```

```
28 </Request>
```

L'elemento `Request` è il contenitore della richiesta e contiene informazioni utili per la tracciabilità (`IssueInstant`) e la correlazione con la corrispondente risposta (`RequestID`). Esso può essere firmato per supportare i casi di utilizzo in cui l'autenticazione del mittente non sia gestita dal protocollo sottostante oppure si utilizzino identità distinte per i vari livelli.

All'interno del contenitore `Request`, vi è la richiesta di asserzione vera e propria detta *query*. Il formato della query è ovviamente diverso per le varie tipologie di asserzione. Nel caso esemplificato si ha una query di attributi `AttributeQuery` contenente il soggetto di cui si chiedono informazioni (`Subject`), il nome (`AttributeName`) ed il vocabolario (`AttributeNamespace`) dell'unico attributo richiesto.

SAML prevede inoltre che gli elementi `Request` possano contenere riferimenti[17] ad asserzioni anziché query. In questo caso il contenuto delle risposte sarà la particolare asserzione richiesta così come immagazzinata nello stato del responder[18]. Non vengono emesse nuove asserzioni.

La corrispondente risposta è nel listato 3.5:

Listing 3.5: Una risposta ad una richiesta di attributi

```
1  <Response
2     xmlns="urn:oasis:names:tc:SAML:1.0:protocol"
3     xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
4     xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
5     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
6     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
7     InResponseTo="_7f17e2145de5e7735b2071efd3f5576f"
8     IssueInstant="2005-08-10T22:07:59.672Z"
9     MajorVersion="1"
10    MinorVersion="1"
11    ResponseID="_b7697c64c3f29b9cd6b42d11f2bba653">
12   <Status>
13     <StatusCode Value="samlp:Success"/>
14   </Status>
15   <Assertion
16      xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
17      AssertionID="_2620081591b8abde565003ff70c21295"
```

[17] Sotto forma di `AssertionID` o *artifact*.

[18] Si veda 3.2.1 per un esempio di responder di questo tipo: Artifact Resolution Service.

```
18      IssueInstant="2005-08-10T22:07:59.638Z"
19      Issuer="https://idp.example.org/shibboleth"
20      MajorVersion="1"
21      MinorVersion="1">
22     <Conditions
23        NotBefore="2005-08-10T22:07:59.638Z"
24        NotOnOrAfter="2005-08-10T22:37:59.638Z">
25       <AudienceRestrictionCondition>
26         <Audience>urn:mace:shibboleth:examples</Audience>
27       </AudienceRestrictionCondition>
28     </Conditions>
29     <AttributeStatement>
30       <Subject>
31         <NameIdentifier
32            Format="urn:mace:shibboleth:1.0:nameIdentifier"
33            NameQualifier="https://idp.example.org/shibboleth">
34           _20e29bfb0c0dbb06dd524c1dbccba727</NameIdentifier>
35         <SubjectConfirmation>
36           <ConfirmationMethod>
37             urn:oasis:names:tc:SAML:1.0:cm:bearer
38           </ConfirmationMethod>
39         </SubjectConfirmation>
40       </Subject>
41       <Attribute
42          AttributeName="urn:mace:dir:attribute-def:eduPersonPrincipalName"
43          AttributeNamespace="urn:mace:shibboleth:1.0:attributeNamespace:uri">
44         <AttributeValue>tomcat</AttributeValue>
45       </Attribute>
46     </AttributeStatement>
47   </Assertion>
48 </Response>
```

Si noti come l'attributo `InResponseTo` della risposta corrisponda all'attributo `RequestID` della richiesta. Un altro elemento interessante del contenitore `Response` è lo stato dell'operazione (`Status`) utile per comunicare al destinatario errori nella gestione della richiesta. Anche le risposte possono essere firmate secondo lo standard in [XMLSig].

Il contenuto informativo di una risposta SAML consiste sempre di una o più asserzioni, con il formato visto negli esempi 3.1, 3.2 e 3.3. È bene sottolineare che le asserzioni contenute in una risposta possono essere individualmente firmate[19]. Questo permette ad esse di avere un ciclo di vita più lungo di quello effimero delle risposte. L'utilizzo tipico delle asserzioni ricevute è infatti quello di essere immagazzinate ed utilizzate in un secondo momento come credenziali.

[19]Tuttavia, per semplicità di esposizione, in questo esempio le firme sono state omesse.

3.1.1.3 Bindings

In [SAML-bind] viene introdotto il termine di *protocol binding* per indicare l'inclusione di richieste e risposte SAML nei corrispondenti messaggi (o comunicazioni di altro tipo) del protocollo di trasporto. Lo standard specifica un solo binding, chiamato *SAML SOAP Binding*, che inserisce gli elementi `Request` e `Response` all'interno del `Body` di messaggi SOAP. Nei listati 3.6 e 3.7 viene esemplificato tale binding, nel caso in cui SOAP utilizzi a sua volta come trasporto HTTP. Per motivi di spazio i messaggi SAML veri e propri sono stati omessi, ma si può pensare di rimpiazzare i commenti con il contenuto dei listati 3.4 e 3.5 rispettivamente.

Listing 3.6: Il binding di una richiesta SAML

```
POST /shibboleth/AA/SOAP HTTP/1.1
Host: idp.example.org
Content-Type: text/xml
Content-Length: nnn
SOAPAction: http://www.oasis-open.org/committees/security

<?xml version="1.1" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <!-- inserire qui una richiesta SAML -->
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Listing 3.7: Il binding di una risposta SAML

```
HTTP/1.1 200 OK
Content-Type: text/xml
Content-Length: nnnn

<?xml version="1.1" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <!-- inserire qui una risposta SAML -->
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SAML delinea inoltre la metologia per specificare e pubblicare nuovi bindings. Questo permette di estendere lo standard per supportare altri tipi di trasporto.

3.1.1.4 Profili

In [SAML-bind] il concetto di *profilo* viene così definito:

> “ A profile describes how SAML assertions are embedded in or combined with other objects (for example, files of various types, or protocol data units of communication protocols) by an originating party, communicated from the originating site to a destination site, and subsequently processed at the destination. ”

In altre parole, un profilo è uno scenario di utilizzo di asserzioni SAML che prevede interazioni complesse non riconducibili ad una singola istanza del protocollo richiesta/risposta descritto nella sezione 3.1.1.3.

Come per i bindings, lo standard specifica la metodologia per creare e pubblicare nuovi profili. Oltre a questo, nel documento [SAML-bind], si descrivono due profili per il SSO da browser:

- il profilo browser/artifact
- il profilo browser/POST

Entrambi i profili descrivono lo scenario in cui un utente, utilizzando un browser web, si autentica presso un *sito sorgente* e da esso viene rediretto presso un *sito destinazione*, nel quale può accedere a risorse riservate senza autenticarsi una seconda volta. Il particolare meccanismo di redirezione, differente nei due profili, è il trasporto utilizzato dal sito sorgente per comunicare asserzioni sull'utente al sito destinazione.

Le modalità di autenticazione dell'utente presso il sito sorgente non sono specificate. Nella descrizione dei profili si assume che l'autenticazione sia già avvenuta in un istante precedente.

Il profilo browser/artifact si basa sul meccanismo di redirezione standard di HTTP, ovvero restituendo al browser uno *status code* di classe 300 e fornendo la nuova destinazione nello header `Location`. La URL di destinazione è l'unico trasporto utilizzabile da questo profilo per innescare la comunicazione tra i due siti. Pertanto nell'effettuare la redirezione il sito sorgente aggiunge uno o più parametri `SAMLart` alla *query string* contenenti dei riferimenti[20] ad asserzioni o *artifacts*. Il sito destinazione si occuperà in seguito di risolvere tali riferimenti interrogando direttamente il sito sorgente.

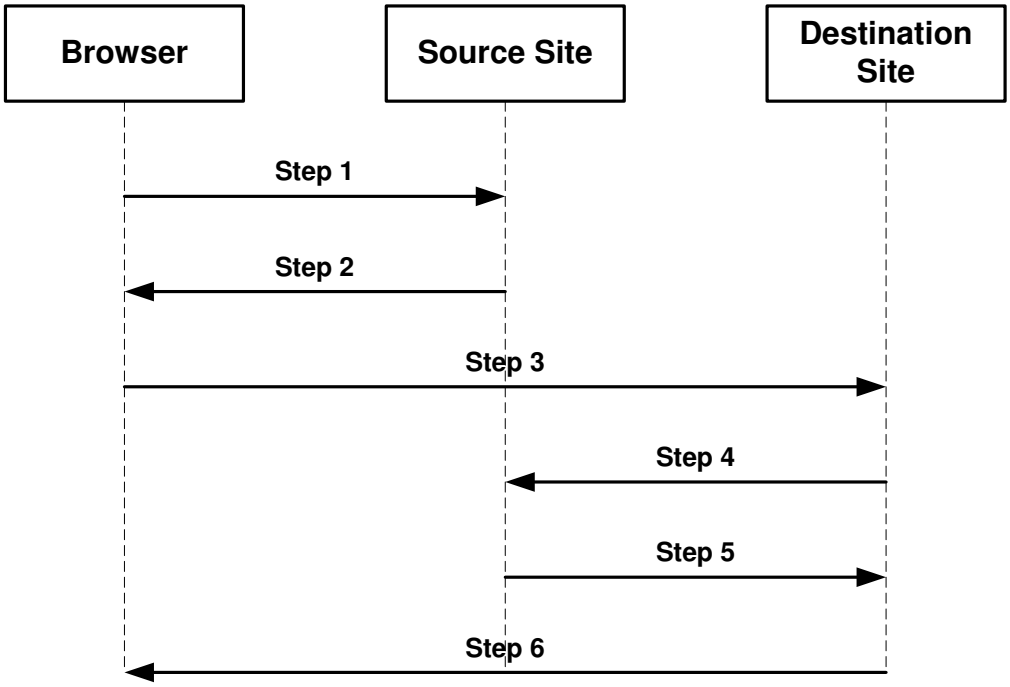

Figura 3.1: Diagramma temporale del profilo browser/artifact di SAML. Figura tratta da [SAML-bind].

Il dettaglio delle interazioni, come illustrato in figura 3.1, è come segue:

[20] A causa delle limitazioni sulla lunghezza delle URL non è possibile codificare un'intera asserzione SAML all'interno della query string. Si veda 3.2.2 per un altro approccio a questo problema.

1. L'utente accede all'*inter-site transfer service* presso il sito sorgente indicando il sito destinazione al quale desidera accedere.

2. Il sito sorgente redirige l'utente verso l'*artifact receiver service* presso il sito destinazione. La redirezione avrà codice di stato HTTP 302 e header `Location` contenente la URL del sito destinazione alla quale saranno stati aggiunti uno o più artifacts all'interno di parametri `SAMLart`. Ogni artifact è composto da un condensato della URI identificativa del sito origine e da un riferimento ad un'asserzione memorizzata nel sito sorgente.

3. Il browser segue la redirezione ed invoca, tramite una HTTP GET, l'artifact receiver service, convogliando nella URL di destinazione anche gli artifacts inseriti dal sito sorgente.

4. Utilizzando il "SAML SOAP Binding over HTTP(s)" il sito destinazione invia al sito sorgente una richiesta SAML contenente gli artifacts ricevuti.

5. Il sito sorgente risolve gli artifacts ricevuti interrogando il proprio database. Successivamente invia le risultanti asserzioni (o errori) nella risposta SAML.

6. Il sito destinazione dà accesso a o nega la risorsa richiesta dall'utente inviando una opportuna risposta HTTP al suo browser.

Il profilo browser/POST utilizza invece una pagina HTML contenente un form nascosto per implementare il meccanismo di redirezione. Questo metodo permette di convogliare una maggiore quantità di informazioni all'interno della richiesta dell'utente verso il sito destinazione, eliminando la necessità di una comunicazione esplicita di quest'ultimo con il sito sorgente. All'interno di tale form, infatti, viene codificata (in forma *base64*) un'intera risposta SAML contenente una o più asserzioni, di cui almeno una di autenticazione.

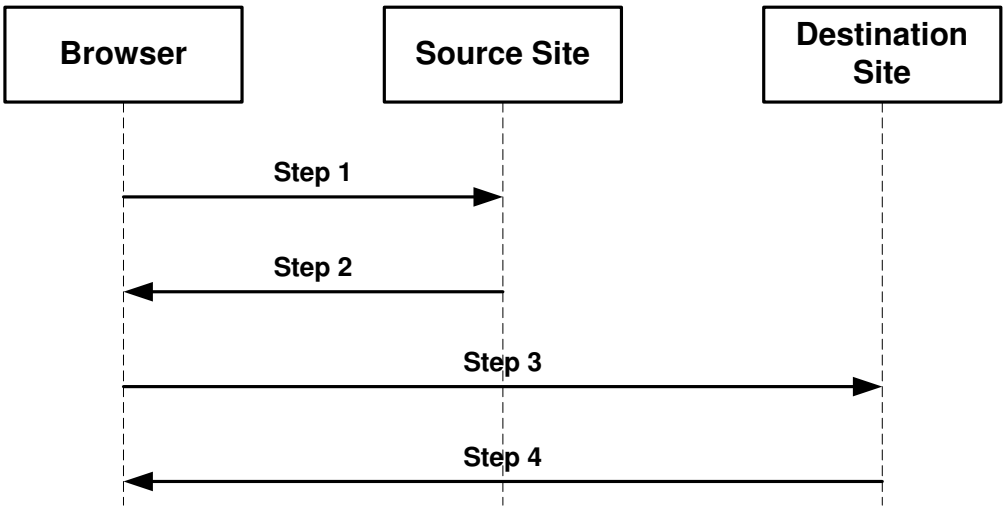

Figura 3.2: Diagramma temporale del profilo browser/POST di SAML. Figura tratta da [SAML-bind].

La sequenza di interazioni corrispondente al diagramma 3.2 è la seguente:

1. Come per l'altro profilo l'utente invoca una funzionalità del sito destinazione attraverso l'inter-site transfer service del sito sorgente.

2. Il sito sorgente invia al browser una pagina HTML contenente un form con la seguente struttura:

```
1 <FORM Method="Post" Action="https://sp.example.org/AssertionConsumer">
2   <INPUT TYPE="hidden" NAME="SAMLResponse" Value="B64(<Response>)">
3   <!-- ... -->
4   <INPUT TYPE="hidden" NAME="TARGET" Value="<Target>">
5 </FORM>
```

3. Il browser invia il form ricevuto dal sito sorgente al sito destinazione eseguendo un HTTP POST verso la URL dell'assertion consumer service. Le modalità nelle quali questa azione viene innescata non sono specificate, ma possono richiedere un'esplicita azione dell'utente[21].

[21] Ad esempio cliccando il bottone "Submit" del form. In alternativa si può evitare di coinvolgere l'utente utilizzando funzionalità di scripting client-side (es. JavaScript).

4. A questo punto il sito destinazione ha ricevuto le asserzioni necessarie alla propria decisione di accesso tramite il POST del browser. Quindi esso è in grado di accettare o rifiutare la richiesta dell'utente e di inviare al suo browser una risposta opportuna.

3.1.1.5 Web Services Security SAML Token Profile

Oltre a quelli descritti nei documenti ufficiali di SAML 1.1 [SAML-core][SAML-bind], diversi enti hanno pubblicato profili per altri scenari di utilizzo dello standard. Di particolare interesse è quello specificato in [WSS-saml] dal Web Services Security Technical Commitee di OASIS.

Il compito di questo profilo è quello di specificare come inserire, processare ed estrarre asserzioni SAML 1.1 all'interno di headers di messaggi SOAP conformi allo standard WSS[22].

Inclusione di asserzioni. L'inserimento di un'asserzione SAML in uno header WSS avviene in maniera del tutto simile alle altre tipologie di token supportate (Kerberos, X509, username/password, ecc.). Dato che è già in formato XML, l'asserzione viene inclusa senza ulteriori codifiche all'interno di un elemento `wsse:Security`, a sua volta contenuto nell'elemento `Header` della busta SOAP.

Riferimento ad asserzioni non incluse nel messaggio. Lo standard [WSS] prevede inoltre l'utilizzo di riferimenti a tokens 'esterni' che il ricevente dovrà recuperare nel processare il messaggio. Nel caso di tokens SAML, ciò si traduce nell'inclusione di riferimenti ad asserzioni[23] e delle informazioni di binding necessarie per recuperarle interrogando opportune *assertion authorities*.

[22] Lo standard WSS è un'estensione di SOAP che permette integrità e confidenzialità a livello di messaggio (si veda [WSS]).

[23] Come identificatore si utilizza il valore dell'attributo `AssertionID` dell'elemento `Assertion`, si veda anche 3.1.1.1.

Gestione dell'elemento `SubjectConfirmation`. Come indicato anche nella sezione 3.1.1.1, il produttore di un'asserzione SAML può indicare al consumatore uno o più metodi per avere conferma dell'autenticità del soggetto dell'asserzione. I metodi di conferma scelti in [WSS-saml] sono due: *holder-of-key* e *sender-vouches*.

Nel primo caso l'elemento `SubjectConfirmation` conterrà al suo interno una chiave (pubblica) che il ricevente potrà utilizzare per verificare l'associazione tra il soggetto dell'asserzione e chi l'ha inserita nel messaggio SOAP. La modalità tipica per effettuare questa associazione è quella di far utilizzare la corrispondente chiave (privata) a chi inserisce l'asserzione per firmare una parte del messaggio.

Il caso del *sender-vouches* è più semplice in quanto chi inserisce l'asserzione 'chiede' a chi riceve di fidarsi della sua associazione con il soggetto del messaggio. L'accettazione del messaggio sarà quindi in funzione dei vincoli di trust in atto tra le due entità.

3.1.2 X.509

X.509 [X.509] è uno standard di ITU-T che definisce una PKI. Nel corso degli anni, grazie anche agli adattamenti apportati da IETF in [RFC3280], è diventato lo standard di fatto per quanto riguarda l'autenticazione 'forte' in ambito Internet.

L'elemento informativo di base dello standard è il Public Key Certificate (PKC), ovvero il documento che associa il nome di un soggetto alla sua chiave pubblica. I certificati sono emessi e firmati da una CA su richiesta del soggetto ed hanno una validità temporale fissata.

Una PKI X.509 è tipicamente composta da più CA organizzate in una struttura ad albero, alla radice del quale si trova la *root* CA, detta anche *trust anchor*. Ciascuna CA è dotata di un certificato emesso e firmato da una CA di livello superiore[24] ad eccezion fatta per il trust anchor che utilizza un certificato auto-firmato. Le foglie dell'albero sono i soggetti per i quali è stato

[24] Per essere precisi dalla CA 'madre'.

emesso un certificato. Una struttura di questo tipo permette di avere garanzia dell'autenticità di un soggetto fidandosi esclusivamente dell'autenticità della root CA. Chi ha bisogno di questa garanzia può infatti risalire l'albero fino alla radice, verificando che ogni nodo intermedio abbia un certificato firmato dal nodo padre.

In caso di compromissione del soggetto associato ad un certificato[25], quest'ultimo può essere revocato pubblicandolo in una Certificate Revocation List (CRL) od utilizzando il protocollo Online Certificate Status Protocol (OCSP) definito in [RFC2560].

Nelle sue ultime incarnazioni, lo standard X.509 specifica un altro tipo di certificato: l'AC. Anziché la chiave pubblica di un soggetto, un AC contiene privilegi o credenziali utili per una autorizzazione ad opera di un'opportuna autorità. Gli AC sono emessi e firmati da una Attribute Authority (AA) e possono essere immagazzinati in un data base LDAP [LDAP].

[25] Può ad esempio accadere che la sua chiave privata venga scoperta.

3.2 Soluzioni per il Single Sign On

Il Single Sign On (SSO) è una particolare forma di autenticazione (ed autorizzazione) che permette l'accesso a più risorse protette a partire da un'unica, iniziale, interazione di autenticazione da parte del soggetto fruitore. Esistono varie tipologie di SSO in funzione dei contesti applicativi e dei soggetti coinvolti. Quella sulla quale si focalizza questa sezione viene denominata *Single Sign On su web*.

Il SSO su web si applica esclusivamente all'accesso autenticato a risorse web (statiche o dinamiche), tipicamente effettuato tramite un browser[26]. I soggetti che accedono ad una tale risorsa per la prima volta nell'arco di una stessa sessione, vengono dirottati presso un servizio di autenticazione, detto Identity Provider. Successivamente, se quest'ultima ha avuto successo, essi vengono rediretti nuovamente presso la risorsa iniziale, detta Service Provider, a cui verranno comunicati gli aggiornamenti sullo stato di autenticazione di tali soggetti.

L'approccio al SSO (ed all'Identity Management in generale) è di tipo federativo in tutte le soluzioni presentate. Non viene imposta dall'alto l'adozione di un particolare Identity Provider, ma vengono invece forniti gli strumenti per stringere legami di trust con una federazione[27] di tali providers senza vincoli gerarchici precostituiti.

I vantaggi del SSO su web in confronto alle soluzioni tradizionali (una credenziale distinta per ogni sito) sono evidenti:

Usabilità. Il soggetto fruitore deve effettuare meno interazioni per accedere agli stessi servizi.

Sicurezza. Dato che è un evento più raro, all'operazione di autenticazione può essere dedicata maggiore attenzione, eventualmente implementando tecniche 'forti' su base crittografica. Per il SSO di tipo federativo

[26] Esiste tuttavia la possibilità di estendere queste modalità di accesso al contesto dei Web Services.

[27] In alcuni contesti si usa anche il temine di *Circle of Trust*.

questo è vero a maggior ragione in quanto gli Identity Provider potrebbero essere gli unici ad avere le informazioni necessarie ad effettuare tale operazione sui propri soggetti.

Scalabilità. Un soggetto che si autentica presso un certo Identity Provider può accedere a risorse protette da un Service Provider che non ha bisogno di mantenere internamente informazioni sul suo conto: può ottenerle dal Service Provider. Tale fenomeno implica una maggiore scalabilità delle soluzioni SSO con il conseguente aumento dei soggetti servibili a parità d'investimento.

Incentivi alla federazione. Un Service Provider è motivato ad aderire ad una federazione dal fatto che può accedere ad un bacino d'utenza più ampio[28] (quello di tutti gli Identity Provider della federazione). D'altro canto un Identity Provider che entri in una federazione garantisce ai propri soggetti una migliore offerta di di servizi in termini di qualità, quantità e diversificazione.

3.2.1 Shibboleth

Shibboleth, [Shib-web][Shib-arch][Shib-proto], è un progetto inter-universitario del gruppo Middleware Architecture Committee for Education (MACE), appartenente al consorzio Internet2 [Internet2-web]. Le sue finalità sono la progettazione, la specifica e l'implementazione Open Source di sistemi per la condivisione inter-istituzionale di risorse web soggette a controllo di accesso.

Questo progetto nasce inizialmente per semplificare il problema dell'accesso a contenuti didattici riservati (o a pagamento) da più campus universitari, ciascuno con una differente infrastruttura di autenticazione. Shibboleth si può tuttavia applicare ad altri contesti – come quelli dell'e-Business e della PA – ed è probabilmente la più completa implementazione Open

[28] Questo è il caso, ad esempio, di molti fornitori di contenuti didattici a pagamento che stanno aderendo a federazioni Shibboleth.

Source di un sistema di SSO su web. Inoltre, diversamente da altri sistemi analoghi [Athens-web][Yale CAS-web], Shibboleth ha adottato fin dall'inizio SAML per implementare gran parte dei suoi protocolli, garantendosi l'interoperabilità con altri sistemi dell'area dell'IM.

3.2.1.1 Architettura generale

Le entità coinvolte in un protocollo di SSO usando Shibboleth sono:

Identity Provider (IdP): l'organizzazione in grado di autenticare l'utente e fornire informazioni aggiuntive o attributi sul suo conto.

Service Provider (SP): l'ente presso il quale è gestita la risorsa web a cui l'utente fa richiesta e che ha il compito di proteggerla attraverso una qualche forma di policy di accesso.

User Agent (UA): è l'applicazione[29] che, per conto dell'utente, innesca i protocolli di SSO richiedendo l'accesso ad una risorsa web protetta da un SP Shibboleth-interoperabile.

Where Are You From (WAYF): si tratta di un servizio gestito da una terza parte o dal SP stesso il cui compito è scoprire l'IdP di appartenenza dell'utente.

Non è prevista una gerarchia tra i providers: ogni organizzazione è responsabile dei propri utenti (nel caso degli IdP) e/o delle proprie risorse (nel caso dei SP). La rete di trust risultante è orizzontale ed in Shibboleth viene denominata *federazione*. Il SP che sceglie di aderire ad una federazione accetta implicitamente un legame di trust con tutti gli IdP che ne fanno parte. Simmetricamente, un IdP accetta di emettere asserzioni su richiesta di un qualsiasi[30] SP della federazione. Shibboleth non vieta tuttavia la possibilità di specificare rapporti di trust bilaterali tra singoli SP ed IdP. Quest'ultimo

[29] Tipicamente un browser web.

[30] Il contenuto informativo delle asserzioni può essere tuttavia differenziato in funzione dell'utente o del SP ricevente.

approccio ha una scalabilità ridotta ma può risultare utile per piccole configurazioni o per specificare rapporti differenziati tra providers all'interno di una federazione preesistente.

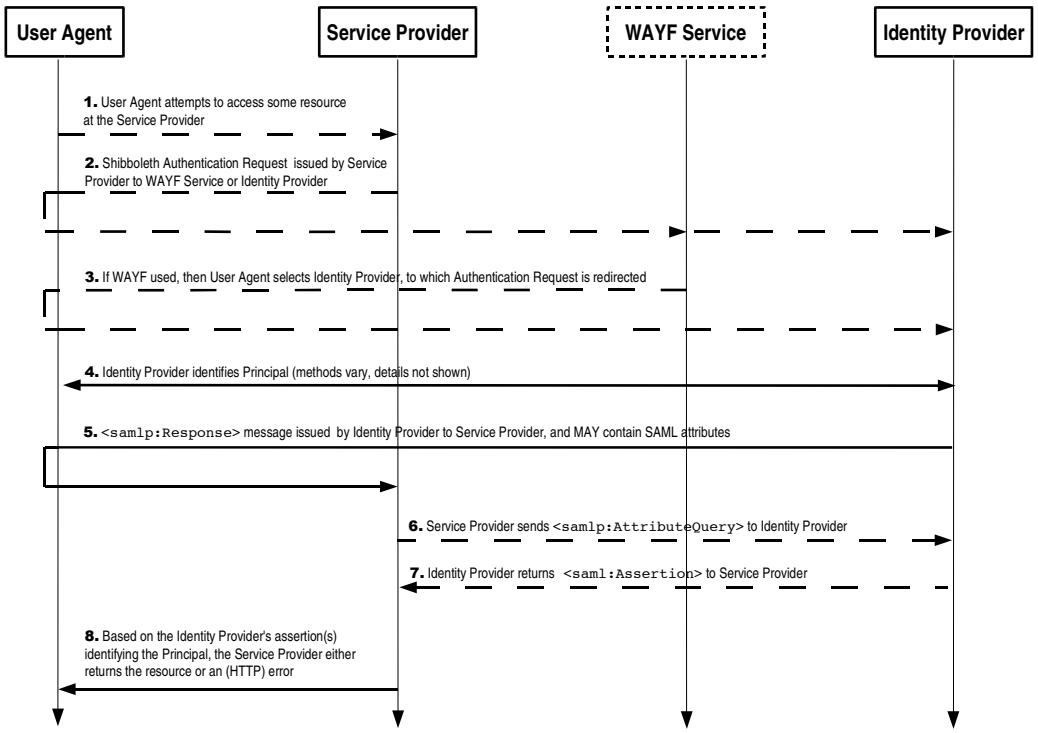

Figura 3.3: Diagramma temporale delle interazioni per il SSO e lo scambio di attributi usando Shibboleth. Figura tratta da [Shib-proto].

Come mostrato in figura 3.3, per un generico protocollo di SSO, le interazioni tra le entità sono le seguenti:

1. Lo UA, per conto dell'utente, richiede l'accesso ad una risorsa web presso il SP. Si assume che lo UA non abbia ancora una sessione attiva con il SP.

2. Il SP redirige lo UA verso il WAYF o direttamente presso l'IdP di appartenenza. Il contenuto della URL di destinazione costituisce una *authentication request*[31] e contiene informazioni sulla risorsa richiesta,

[31] Si veda in proposito la sezione 3.2.1.3.

un identificativo del SP e l'endpoint presso il quale il SP intende ricevere l'asserzione di autenticazione.

3. Se interpellato, il WAYF processa la authentication request del SP (ma trasportata dallo UA) ed interagisce con l'utente per conoscere l'IdP presso il quale intende autenticarsi. Tale informazione viene memorizzata in una sessione a lungo termine tra il WAYF e lo UA (ad es. un cookie). Successivamente il WAYF redirige lo UA verso l'IdP scelto dall'utente lasciando sostanzialmente invariata la query string della URL impostata dal SP al punto 2.

4. L'IdP identifica l'utente innescando un meccanismo di autenticazione o sfruttando una sessione ancora attiva. Shibboleth astrae dal particolare meccanismo di autenticazione utilizzato[32].

5. L'IdP invia un'asserzione di autenticazione al SP utilizzando il profilo SAML browser/POST o browser/artifact (cf. 3.1.1.4).

6. Se lo ritiene necessario, il SP invia una richiesta di attributi dell'utente all'IdP. In questo caso la comunicazione tra SP ed IdP non è trasportata dallo UA, ma avviene direttamente secondo le modalità indicate dal SAML SOAP Binding.

7. Se interrogato, IdP risponde alla richiesta di attributi del SP.

8. Sulla base delle informazioni ottenute sull'utente, il SP effettua la decisione per l'accesso dell'utente alla risorsa richiesta. In funzione dell'esito di tale decisione il SP invia una risposta HTTP opportuna allo UA.

[32]Nella configurazione del prototipo è stata utilizzata la funzionalità di autenticazione integrata nel servlet engine Apache Tomcat, [Tomcat-web].

3.2.1.2 Componenti

Per supportare i protocolli di SSO, i providers coinvolti espongono un certo numero di funzionalità distinte. Nella terminologia di Shibboleth[33], tali funzionalità vengono denominate *ruoli*. Un tipico IdP avrà un insieme di ruoli disgiunto da un tipico SP, ma non vanno esclusi i casi in cui uno stesso provider fornisca (alcune delle) funzionalità di IdP e SP contemporaneamente.

Nelle figure 3.4 e 3.5 sono illustrati i tipici componenti di un IdP e di un SP (rispettivamente), nonché le principali interconnessioni tra di essi.

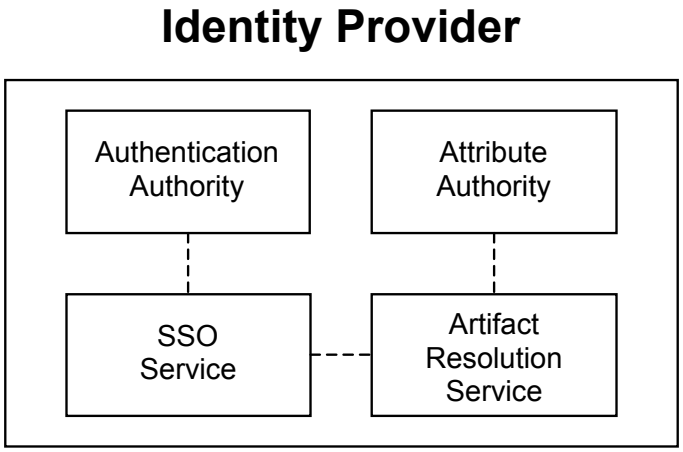

Figura 3.4: Componenti di un tipico IdP. Figura tratta da [Shib-arch].

Authentication Authority. Si tratta di un servizio che emette asserzioni SAML di autenticazione dopo aver ottenuto conferma dell'identità dell'utente dall'applicazione che esegue l'autenticazione vera e propria. Entra in azione al passo 4 del protocollo generico di SSO visto in 3.2.1.1.

Nell'uso comune di Shibboleth non è prevista l'invocazione diretta di questo servizio da parte del SP.

SSO Service. È un'applicazione web dell'IdP che riceve le richieste di autenticazione del SP ed innesca l'opportuno profilo SAML di SSO (passi 2/3,

[33] A sua volta derivata da quella di SAML 1.1 e 2.0.

4, 5). Questo componente costituisce il primo punto di contatto di un IdP se il protocollo di SSO ha inizio presso il SP[34].

Artifact Resolution Service. Utilizzato esclusivamente nei casi in cui si utilizza il profilo SAML browser/artifact, costituisce il web service che traduce gli artifacts in asserzioni. Viene innescato al passo 5 del protocollo in 3.2.1.1[35].

Attribute Authority (AA). Si tratta di un web service che fornisce asserzioni SAML di attributi utilizzando il SAML SOAP Binding. Gestisce i passi 6 e 7 del protocollo in 3.2.1.1 dal lato IdP.

L'accesso agli attributi da parte dei singoli providers richiedenti è a sua volta soggetto a controllo di accesso, regolato da una policy detta Attribute Release Policy (ARP). In tale policy confluiscono esplicite esigenze di privacy espresse dall'utente e direttive generiche dell'IdP, il tutto in funzione del particolare provider richiedente.

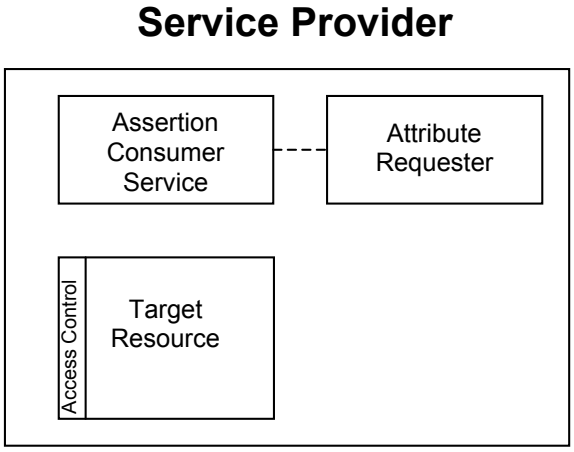

Figura 3.5: Componenti di un tipico SP. Figura tratta da [Shib-arch].

[34]Si tratta del caso più comune.

[35]Più in dettaglio, si tratta del servizio che agisce al passo 5 del profilo browser/artifact di SAML descritto nel paragrafo 3.1.1.4.

Assertion Consumer Service. È l'applicazione web alla quale vengono inviate le asserzioni di autenticazione da parte dell'IdP. Implementa il passo 5 del protocollo in 3.2.1.1 dal lato SP.

L'Assertion Consumer Service ha un comportamento differente a seconda del profilo SAML che si usa. Nel caso del browser/POST estrae direttamente le asserzioni dalla richiesta HTTP. Nel caso del browser/artifact, invece, deve risolvere gli artifacts[36] in asserzioni interrogando l'Artifact Resolution Service.

Attribute Requester. È un client web service che richiede attributi SAML alla AA come descritto nei passi 6 e 7 del protocollo generico di SSO. Simmetricamente al caso della AA, l'Attribute Requester è dotato di funzionalità di filtraggio degli attributi ricevuti. Essi infatti possono essere scartati in base ad una Attribute Acceptance Policy (AAP) o rinominati per una più agevole gestione da parte del provider richiedente.

Resource Manager (RM). Si tratta del componente che implementa i passi 1 e 8 del protocollo di SSO dal lato del SP. Il suo compito principale è quello di valutare la policy di accesso in funzione delle asserzioni ottenute. Esso può inoltre creare una sessione per la risorsa protetta, nel caso in cui questa sia dinamica (ovvero un'applicazione web). Sempre in quest'ultimo caso, il RM può delegare la decisione di accesso alla risorsa stessa, limitandosi ad innescare il protocollo di SSO ed a raccogliere le asserzioni.

3.2.1.3 Estensioni allo standard SAML

È evidente quanto l'interoperabilità sia la chiave di successo per sistemi di questo tipo. Una federazione Shibboleth può coinvolgere enti che hanno necessità di integrare (e quindi modificare) Shibboleth con la propria infrastruttura di ICT. Può inoltre rendersi necessario per un provider mantenersi interoperabile con altri sistemi di IM. Detto questo, non stupisce l'attenzio-

[36] Ricevuti sotto forma di parametri della query string.

ne riservata dal gruppo di lavoro MACE alla standardizzazione di interfacce e protocolli, sotto forma di estensioni al già diffuso e promettente standard SAML. Shibboleth può infatti essere visto come l'infrastruttura (minima) necessaria a rendere un'implementazione di SAML un sistema di SSO su web completo.

Authentication Request Profile. In [Shib-proto] si standardizza un nuovo profilo per SAML detto *Authentication Request Profile*. Il suo compito è quello di estendere i profili per il SSO di SAML – hanno tutti inizio presso l'IdP dell'utente – in modo che possano iniziare presso il SP. I passi 1-3 dell'esempio fornito nella sezione 3.2.1.1 seguono per l'appunto questo profilo.

In sostanza l'Authentication Request Profile specifica il formato della richiesta HTTP[37] del passo 2, l'introduzione del WAYF ed alcune regole di processamento da parte delle entità riceventi tale richiesta (IdP e WAYF).

Metadata Profile. Sia SAML che Shibboleth assumono l'esistenza di accordi, detti *metadati*, tra le entità partecipanti ai protocolli di SSO. Tali accordi riguardano i ruoli assunti, il formato dei nomi utente[38], quali bindings o profili siano supportati, gli endpoints forniti, i certificati e le chiavi posseduti, ecc.

I dettagli sulla forma ed il contenuto dei metadati sono tuttavia standardizzati solo in SAML 2.0. Pertanto, il gruppo di lavoro di Shibboleth ha creato un nuovo profilo per SAML 1.1, detto *SAML 1.x Metadata Profile* [Shib-SAML1Meta], che 'importa' alcune delle innovazioni presenti nella versione 2.0.

In pratica i metadati SAML 2.0 sono dei documenti XML che descrivono le peculiarità dei providers che li pubblicano. Tali documenti possono presentarsi anche in forma aggregata, ovvero descrivere tutti i providers all'interno di una federazione. Un esempio di metadati – relativo ad una semplice fede-

[37] Effettuata utilizzando il metodo GET.

[38] Ovvero il formato degli elementi SAML `NameIdentifier`, cf. 3.1.1.1.

razione composta di un solo IdP ed un solo SP – è riportato nei listati 3.8, 3.9 e 3.10, costituenti tre sezioni contigue dello stesso documento XML.

Listing 3.8: Un esempio di metadati, parte globale (federazione)

```
<!-- Descrittore di una federazione, puo' essere firmato -->
<EntitiesDescriptor
   xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
   xmlns:shibmd="urn:mace:shibboleth:metadata:1.0"
   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
   Name="urn:mace:shibboleth:examples"
   validUntil="2010-01-01T00:00:00Z">
```

Si noti la presenza di una data di scadenza del documento (`validUntil`) ed il nome assegnato alla federazione (`Name`). L'elemento `EntitiesDescriptor` può essere firmato per garantire l'integrità dei metadati.

Listing 3.9: Un esempio di metadati, parte relativa ad un IdP

```
  <!-- Descrittore di un Identity Provider -->
  <EntityDescriptor entityID="https://idp.example.org/shibboleth">

    <!-- Descrittore del componente di SSO del IdP -->
    <IDPSSODescriptor
       protocolSupportEnumeration="urn:oasis:names:tc:SAML:1.1:protocol [...]
           urn:mace:shibboleth:1.0">

      <!-- Il certificato corrispondente alla chiave privata utilizzata
           per firmare le asserzioni di autenticazione -->
      <KeyDescriptor use="signing">
        <ds:KeyInfo>
          <ds:X509Data>
            <ds:X509Certificate>
              <!-- Codifica base64 del certificato omessa -->
            </ds:X509Certificate>
          </ds:X509Data>
        </ds:KeyInfo>
      </KeyDescriptor>

      <!-- Descrittore del componente del IdP
           che risolve gli artifacts -->
      <ArtifactResolutionService
         index="1"
         Binding="urn:oasis:names:tc:SAML:1.0:bindings:SOAP-binding"
         Location="https://idp.example.org:8443/shibboleth-idp/Artifact"/[...]
             >

      <!-- formati per il NameIdentifier supportati -->
      <NameIDFormat>urn:mace:shibboleth:1.0:nameIdentifier</NameIDFormat>
```

```
40
41        <!-- Endpoint e binding da usare
42             per richiedere una autenticazione -->
43        <SingleSignOnService
44           Binding="urn:mace:shibboleth:1.0:profiles:AuthnRequest"
45           Location="https://idp.example.org/shibboleth-idp/SSO"/>
46
47      </IDPSSODescriptor>
48
49
50      <!-- Descrittore del componente Attribute Authority del IdP -->
51      <AttributeAuthorityDescriptor
52         protocolSupportEnumeration="urn:oasis:names:tc:SAML:1.1:protocol">
53
54        <!-- Il certificato corrispondente alla chiave privata utilizzata
55             per firmare le asserzioni di attributi -->
56        <KeyDescriptor use="signing">
57          <ds:KeyInfo>
58            <ds:X509Data>
59              <ds:X509Certificate>
60                <!-- Codifica base64 del certificato omessa -->
61              </ds:X509Certificate>
62            </ds:X509Data>
63          </ds:KeyInfo>
64        </KeyDescriptor>
65
66        <!-- Endpoint e binding del componente AA -->
67        <AttributeService
68           Binding="urn:oasis:names:tc:SAML:1.0:bindings:SOAP-binding"
69           Location="https://idp.example.org:8443/shibboleth-idp/AA"/>
70
71        <!-- Attributi emessi ed eventuali valori possibili -->
72        <Attribute
73           Name="urn:mace:dir:attribute-def:eduPersonAffiliation"
74           NameFormat="urn:mace:shibboleth:1.0:attributeNamespace:uri">
75          <AttributeValue>member</AttributeValue>
76          <AttributeValue>student</AttributeValue>
77          <AttributeValue>faculty</AttributeValue>
78          <AttributeValue>employee</AttributeValue>
79          <AttributeValue>staff</AttributeValue>
80        </Attribute>
81        <Attribute
82           Name="urn:mace:dir:attribute-def:eduPersonPrincipalName"
83           NameFormat="urn:mace:shibboleth:1.0:attributeNamespace:uri"/>
84
85        <!-- formati del NameIdentifier accettati
86             nelle richieste di attributi -->
87        <NameIDFormat>urn:mace:shibboleth:1.0:nameIdentifier</NameIDFormat>
88      </AttributeAuthorityDescriptor>
89
90      <!-- Informazioni generali
91           sulla organizzazione del IdP e contatti -->
92      <Organization>
93        <OrganizationName xml:lang="en">
94          Example Identity Provider
95        </OrganizationName>
96        <OrganizationDisplayName xml:lang="en">
97          Identities R Us
98        </OrganizationDisplayName>
```

```
99        <OrganizationURL xml:lang="en">
100         http://idp.example.org/
101       </OrganizationURL>
102     </Organization>
103     <ContactPerson contactType="technical">
104       <SurName>Technical Support</SurName>
105       <EmailAddress>support@idp.example.org</EmailAddress>
106     </ContactPerson>
107
108   </EntityDescriptor>
```

In questa porzione del documento sono presenti informazioni generali sull'IdP (`Organization`), un identificatore unico (`entityID`) e le chiavi (`KeyDescriptor`) utilizzate per firmare i vari tipi di asserzione. È inoltre riportato dove (`Location`) e come (`Binding`) accedere ai diversi servizi offerti (`SingleSignOnService`, `ArtifactResolutionService`, `AttributeService`) e quale formato di handle utilizzare nelle richieste (`NameIDFormat`). Infine, per quanto riguarda la configurazione della AA, sono presenti gli attributi rilasciati (`Attribute`) e l'insieme dei possibili valori (`AttributeValue`).

Listing 3.10: Un esempio di metadati, parte relativa ad un SP

```
113   <EntityDescriptor entityID="https://sp.example.org/shibboleth">
114
115     <!-- Componente di SSO del SP -->
116     <SPSSODescriptor
117        protocolSupportEnumeration="urn:oasis:names:tc:SAML:1.1:protocol">
118
119       <!-- Il certificato utilizzato dal SP
120            per autenticarsi con IdP -->
121       <KeyDescriptor use="signing">
122         <ds:KeyInfo>
123           <ds:X509Data>
124             <ds:X509Certificate>
125               <!-- Codifica base64 del certificato omessa -->
126             </ds:X509Certificate>
127           </ds:X509Data>
128         </ds:KeyInfo>
129       </KeyDescriptor>
130
131       <!-- formati per il NameIdentifier supportati -->
132       <NameIDFormat>urn:mace:shibboleth:1.0:nameIdentifier</NameIDFormat>
133
134       <!-- Gli endpoints e i bindings che IdP puo usare
135          per inviare asserzioni di autenticazione al SP -->
136       <AssertionConsumerService
137          index="1" isDefault="true"
138          Binding="urn:oasis:names:tc:SAML:1.0:profiles:browser-post"
```

```
139         Location="https://sp.example.org/shibboleth-sp/Shibboleth.sso/[...]
                SAML/POST"/>
140       <AssertionConsumerService
141         index="2"
142         Binding="urn:oasis:names:tc:SAML:1.0:profiles:artifact-01"
143         Location="https://sp.example.org/shibboleth-sp/Shibboleth.sso/[...]
                SAML/Artifact"/>
144       <AssertionConsumerService
145         index="3"
146         Binding="urn:oasis:names:tc:SAML:1.0:profiles:browser-post"
147         Location="https://sp.example.org:9443/shibboleth-sp/Shibboleth.[...]
                sso/SAML/POST"/>
148       <AssertionConsumerService
149         index="4"
150         Binding="urn:oasis:names:tc:SAML:1.0:profiles:artifact-01"
151         Location="https://sp.example.org:9443/shibboleth-sp/Shibboleth.[...]
                sso/SAML/Artifact"/>
152
153       <!-- Gli attributi di cui il SP necessita (e gli eventuali valori)
154            per effettuare la propria decisione di accesso -->
155       <AttributeConsumingService isDefault="true" index="1">
156         <ServiceName xml:lang="en">Service Provider</ServiceName>
157         <RequestedAttribute
158            Name="urn:mace:dir:attribute-def:eduPersonAffiliation"
159            NameFormat="urn:mace:shibboleth:1.0:attributeNamespace:uri">
160           <AttributeValue>member</AttributeValue>
161           <AttributeValue>student</AttributeValue>
162           <AttributeValue>faculty</AttributeValue>
163           <AttributeValue>employee</AttributeValue>
164           <AttributeValue>staff</AttributeValue>
165         </RequestedAttribute>
166       </AttributeConsumingService>
167     </SPSSODescriptor>
168
169     <!-- Informazioni generali
170          sulla organizzazione del SP e contatti -->
171     <Organization>
172       <OrganizationName xml:lang="en">
173         Example Service Provider
174       </OrganizationName>
175       <OrganizationDisplayName xml:lang="en">
176         Services R Us
177       </OrganizationDisplayName>
178       <OrganizationURL xml:lang="en">
179         http://sp.example.org/
180       </OrganizationURL>
181     </Organization>
182     <ContactPerson contactType="technical">
183       <SurName>Technical Support</SurName>
184       <EmailAddress>support@sp.example.org</EmailAddress>
185     </ContactPerson>
186
187   </EntityDescriptor>
188
189 </EntitiesDescriptor>
```

Nella parte relativa al SP le informazioni di rilievo sono il certificato

utilizzato per autenticarsi (`KeyDescriptor`), il formato di handle richiesto (`NameIDFormat`), uno o più servizi di ricezione di asserzioni di SSO (`AssertionConsumerService`) ciascuno con il suo endpoint (`Location`) ed il binding da utilizzare (`Binding`). Sono inoltre presenti gli attributi ritenuti indispensabili per le proprie decisioni di accesso (`RequestedAttribute`) con i relativi valori (`AttributeValue`). Come per il caso dell'IdP sono presenti informazioni generali e di contatto sull'organizzazione del SP.

È bene sottolineare che i metadati sono distinti dai files di configurazione di IdP e SP. Essi pubblicano solo alcune delle informazioni presenti nelle configurazioni dei singoli providers.

Formati per il `NameIdentifier`. Una caratteristica decisamente interessante di Shibboleth è il supporto per nomi utente opachi[39] ed effimeri[40] detti *handle*. Al momento di emettere l'asserzione di SSO (punto 5 del protocollo in sez. 3.2.1.1) l'IdP converte il nome utente utilizzato internamente[41] in un handle tale che sia difficile risalire alla sua effettiva identità. Esso potrà essere utilizzato successivamente dal SP ricevente per richiedere ulteriori asserzioni per conto dell'utente.

Questo accorgimento permette di garantire ottimi livelli di privacy dell'utente. Il SP infatti, è in grado di effettuare la propria decisione di accesso ignorando completamente l'identità dell'utente richiedente e basandosi solo sulle asserzioni ottenute dall'IdP[42].

Il particolare mapping utilizzato per associare handles a nomi utente è estensibile e configurabile per-peer. In altre parole un IdP può utilizzare differenti formati di handle a seconda del livello di trust che ha con il SP per il quale sta emettendo asserzioni.

[39] Dai quali non è possibile risalire al nome utente originale.

[40] Utilizzabili per un numero di interazioni o per un intervallo di tempo limitati.

[41] Il formato della rappresentazione interna del nome utente dipende dal particolare backend di autenticazione utilizzato. Nel caso di LDAP sarà ad esempio il CN.

[42] Purché l'insieme di tali asserzioni a sua volta non identifichi univocamente l'utente.

Shibboleth prevede i seguenti formati predefiniti per il `NameIdentifier` nelle asserzioni e nelle corrispondenti query (se presenti):

`SharedMemoryShibHandle` è sostanzialmente un numero casuale generato al momento dell'emissione di un'asserzione. Tale valore viene mantenuto in associazione con il nome utente utilizzato internamente per un certo periodo di tempo (`handleTTL`). I listati 3.1, 3.2, 3.3, 3.4, 3.5 utilizzano uno handle di questo tipo.

`CryptoHandleGenerator` come il caso precedente, ma lo handle viene criptato utilizzando una chiave simmetrica. Permette di garantire la confidenzialità dello handle tra l'IdP assertore e il SP consumatore, senza che eventuali SP intermediari possano intercettarlo ed utilizzarlo per ottenere nuove asserzioni per conto dell'utente.

`Principal` in questo caso lo handle è 'in chiaro', ovvero corrisponde al nome utente utilizzato internamente. Può essere utile per emettere asserzioni verso SP maggiormente fidati o che basano la propria decisione di accesso anche sull'identità dell'utente.

3.2.1.4 Implementazione

Tutte le implementazioni di Shibboleth si compongono di due pacchetti separati: il software da installare presso gli IdP e quello per i SP. L'implementazione 'ufficiale', ad opera del gruppo di lavoro MACE, fornisce un IdP scritto come insieme di applicazioni web J2EE che realizzano i singoli ruoli (AA, SSO Service, ecc.). Il SP è invece scritto in C/C++ ed è costituito da un modulo per il server HTTP[43] che implementa il RM e da un *demone* che gestisce i protocolli di SSO.

Di recente l'università di Yale [Yale Shib-web] ha reimplementato con tecnologia J2EE il software per i SP. Il ruolo del RM viene gestito da un *servlet filter*[44], mentre gli altri servizi sono implementati da servlets e pagine JSP

[43] Attualmente i servers supportati sono Apache [Apache-web] e IIS [IIS-web].

[44] cf. [Servlet-API].

esattamente come nell'IdP. Il lavoro eseguito a Yale è stato sucessivamente integrato nella release ufficiale di Shibboleth a partire dalla versione 1.3.

Dal punto di vista dello sviluppatore delle applicazioni web protette, la differenza sostanziale tra le due implementazioni del SP è nel formato dei dati di sessione e nel meccanismo di passaggio degli stessi dal SP all'applicazione. Il SP Java può passare dati strutturati[45] utilizzando la Servlet API, mentre la versione C/C++ converte gli attributi in opportuni headers HTTP[46].

Pertanto quest'ultima versione del SP si adatta meglio a proteggere documenti web statici, mentre quella in Java a risorse dinamiche. L'implementazione in Java SP può inoltre essere integrata all'interno di applicazioni web complesse che vogliano avere visibilità dell'intero protocollo di SSO.

3.2.2 Liberty Alliance

Liberty Alliance [Liberty-web] è un consorzio di più di 150 organizzazioni fondato da Sun Microsystems nel 2001 per sviluppare e promuovere standards aperti nell'area dell'Identity Management federato. Lo scopo principale del consorzio è di fornire ai providers gli strumenti per rendere interoperabili le credenziali ed i profili dei propri utenti, cercando al contempo di rispettarne la privacy e mantenere i propri standards di sicurezza.

Gli standards di Liberty Alliance sono al momento organizzati in tre ambiti principali o *fasi*:

I. Identity Federation Framework (ID-FF), un sistema federato per il SSO su web;

II. Web Services Framework (ID-WSF), infrastrutture per l'IM in ambito Web Services;

[45] Si può ad esempio passare le asserzioni ottenute nella fase di SSO all'applicazione web protetta per ulteriori processamenti. Questo approccio è stato seguito nel prototipo realizzato, si veda in proposito il capitolo 5.

[46] I nomi di tali headers si possono impostare nella Attribute Acceptance Policy, cf. 3.2.1.2.

III. Services Interface Specifications (ID-SIS), la standardizzazione di interfacce per servizi applicativi riguardanti l'IM.

Sebbene esistano implementazioni sia di ID-FF che di ID-WSF, l'accettazione di quest'ultimo è limitata, data la novità delle tecnologie Web Services ed il gran numero di standards concorrenti[47]. Ai fini di questa tesi è pertanto interessante analizzare esclusivamente ID-FF, focalizzando l'attenzione sulle differenze con Shibboleth[48].

3.2.2.1 ID-FF

Come Shibboleth, anche ID-FF [LibertyProtSchema][LibertyBindProf] si basa su SAML 1.1[49]. Le interrelazioni tra i tre standards sono tuttavia più articolate. La nuova versione 2.0 di SAML utilizza infatti concetti e tecnologie inizialmente standardizzate in ID-FF ai quali anche Shibboleth di recente si sta adeguando. Inoltre, alcune delle organizzazioni e delle persone che hanno lavorato a Shibboleth hanno anche partecipato alla stesura degli standards di Liberty Alliance.

Un aspetto distintivo di ID-FF è l'approccio maggiormente decentrato alla problematica dell'IM. Mentre nel dominio applicativo di Shibboleth è ben chiara la ripartizione dei providers tra IdP (le università) e SP (i fornitori di contenuti didattici), ID-FF punta al più eterogeneo mercato dell'*e-business*. Lo standard suppone infatti di partire dall'attuale situazione di fatto in cui gli utenti hanno diversi accounts gestiti separatamente da altrettanti providers, senza voler imporre l'adozione di un unico IdP per ciascun gruppo di utenti.

L'unione degli attributi e delle credenziali associati a tutti i propri accounts costituisce per ID-FF l'*identità digitale* dell'utente. Lo standard definisce un meccanismo per collegare due accounts presso providers distinti detto *identity federation*. Ciascuno dei due providers continuerà a gestire la pro-

[47] Tra cui quelli della 'famiglia' WS-*: WS-Interoperability, WS-Federation, WS-Security, WS-Policy, WS-Trust.

[48] cf. 3.2.1

[49] cf. 3.1.1.

pria parte dell'identità digitale, ma avrà a disposizione un nome comune per riferire univocamente l'utente nelle comunicazioni[50] con l'altro provider.

Di conseguenza, nella visione di Liberty Alliance, i providers si trovano tipicamente a dover gestire le funzionalità di IdP e SP simultaneamente, sarà l'utente a scegliere presso quale provider autenticarsi e di quali attributi autorizzare la condivisione[51]. È inoltre prevista in ID-FF la possibilità, per un IdP, di delegare l'autenticazione di un utente ad un altro IdP attraverso l'IdP proxying[52].

Un'altra feature interessante di ID-FF è il *single logout*. L'utente ha la possibilità di terminare globalmente la propria sessione distribuita di SSO invocando tale funzionalità presso uno qualsiasi dei providers della federazione. Il provider ricevente provvederà ad inoltrare la richiesta di logout a tutti gli altri che, a loro volta, invalideranno le proprie sessioni locali. Questa feature offre indubbi vantaggi di sicurezza nei casi in cui lo UA sia un terminale di pubblico accesso.

Meritano inoltre di essere citate le seguenti caratteristiche:

URL-encoded messages la possibilità di codificare messaggi ID-FF all'interno di una query string HTTP;

Federation Termination un protocollo che permette all'utente di richiedere lo scioglimento di un legame di identity federation tra un IdP ed un SP.

Liberty-Enabled Client and Proxy (LECP) la predisposizione dello standard all'utilizzo di terminali e proxies applicativi 'intelligenti', ovvero dotati della logica necessaria per processare ed emettere messaggi ID-FF.

[50] Si tratta tipicamente di asserzioni di autenticazione o di attributi.

[51] Tuttavia ID-FF non standardizza nulla di simile alla ARP di Shibboleth, cf. 3.2.1.2.

[52] Ad esempio nel caso in cui esso ritenga che l'utente si sia già autenticato presso il provider destinatario della delega.

Le implementazioni complete[53] dell'ultima versione (la 1.2) di ID-FF sono al momento cinque:

- OpenView Select Federation di Hewlett-Packard;
- Novell Identity Provider di Novell, Inc.;
- i-dLive di Nippon Telegraph and Telephone corporation (NTT);
- Sun Java System Access Manager di Sun Microsystems;
- IdentityBridge di Trustgenix.

Sono inoltre disponibili due implementazioni Open Source di ID-FF 1.2. Si tratta di implementazioni parziali, ma conformi ai requisiti di base[54] di Liberty Alliance:

- SourceID Liberty di Ping Identity Corporation;
- Lasso di Entr'ouvert.

[53] Si tratta delle implementazioni che hanno passato i tests di conformità eseguiti dal consorzio Liberty Alliance, fonte: [LibertyImpl-web].

[54] Si veda [LibertyReq] per ulteriori dettagli.

3.3 Soluzioni per il controllo di accesso

Nelle applicazioni dai requisiti di sicurezza elevati come quelle in cui sono in gioco attributi confidenziali di un'identità digitale (sanità, banche, ecc.), il problema del *controllo di accesso* (o *autorizzazione*) è distinto da quello dell'*autenticazione*. La decisione sull'accesso di un soggetto autenticato ad una risorsa protetta viene valutata in funzione degli attributi[55] del soggetto stesso e della *policy* dell'organizzazione responsabile per la risorsa. Tali decisioni possono coinvolgere altre organizzazioni e le loro rispettive policies.

Altri contesti applicativi di e-Government, come quello della firma digitale a valore legale e dell'emissione di deleghe in forma elettronica, necessitano di infrastrutture più complesse, ma sempre incentrate sul problema dell'autorizzazione.

Pertanto, vengono qui presentati un motore di decisione basato su policy[56] (o Policy Decision Point), ed un'infrastruttura avanzata di autorizzazione[57] sviluppata nell'ambito del *Grid computing*, ma che vale la pena di essere studiata come modello per implementare scenari avanzati di e-Government.

3.3.1 PERMIS

PrivilEge and Role Management Infrastructure Standards validation (PERMIS), [Permis-web][PermisSEC][PermisACM], specifica ed implementa una API Java per un PDP in base al framework di autorizzazione ISO 10181-3 / ITU-T X.812 [X.812]. Il progetto è opera dell'Information Systems Security Research Group (ISSRG) presso l'università di Salford[58] in Gran Bretagna e di recente è stato rilasciato con licenza Open Source[59].

PERMIS nasce per effettuare decisioni di autorizzazione complesse in ambito di e-Government. Su finanziamento europeo, infatti, è stato testato su

[55] Ivi comprese eventuali autorizzazioni ottenute in precedenza.
[56] PERMIS, cf. 3.3.1.
[57] cf. 3.3.2.
[58] Il gruppo di ricerca si è successivamente spostato all'università di Kent.
[59] cf. [OpenPermis-web].

progetti pilota in tre paesi membri per risolvere problematiche amministrative:

- A Bologna, per permettere la gestione distribuita del piano regolatore da parte degli architetti autorizzati.
- A Barcellona, per gestire le contravvenzioni di auto prese a nolo.
- A Salford, per garantire la correttezza degli appalti comunali.

Le decisioni di accesso operate da PERMIS sono valutate in funzione di una policy – di cui viene specificato il formato – e dell'insieme di AC X.509 afferenti al soggetto coinvolto. Tali AC possono essere forniti da chi richiede la decisione (modello *push*) od estratti da un'opportuna base di dati (modello *pull*).

Gli standards su cui si basa PERMIS sono X.509 per i certificati e LDAP [LDAP] per l'accesso a gli AC ed alle policies. Tuttavia la API è provvista degli opportuni punti di estensione per supportare altre tipologie di AA[60].

3.3.1.1 Architettura di una PMI

In [PermisSEC] si introduce il termine di Policy Management Infrastructure (PMI), in analogia con PKI[61], per denotare un'infrastruttura di autorizzazione basata su X.509. La corrispondenza tra i concetti principali di una PMI e quelli di una PKI è illustrata in tabella 3.1.

Mentre un PKC crea un legame (crittograficamente) forte tra una chiave pubblica ed il nome di un soggetto, un AC permette di associare a quest'ultimo una forma più generale di informazione detta attributo. Le entità che firmano gli AC vengono denominate AA ed anch'esse possono essere organizzate in una gerarchia il cui vertice viene denominato Source of Authority (SoA).

[60] Questo è stato l'approccio seguito nel disegno dell'architettura del cap. 4 e nella realizzazione del prototipo (cap. 5) per permettere l'utilizzo di attributi SAML nella decisione.

[61] cf. 3.1.2.

concetto	entità PKI	entità PMI
certificato	PKC	AC
emissario del certificato	CA	AA
utente del certificato	soggetto	portatore (holder)
legame (binding) del certificato	soggetto ↔ chiave pubblica	soggetto ↔ attributi
revoca	CRL	ACRL
terza parte fidata	root CA	SoA
autorità subordinata	CA subordinata	AA

Tabella 3.1: Comparazione tra una PKI ed una PMI.

Anche per gli AC esistono liste di revoca denominate Attribute Certificate Revocation List (ACRL).

In figura 3.6 è illustrata l'architettura della PMI di PERMIS. L'utilizzo di tale PMI avviene in due momenti (logicamente) distinti, detti *privilege allocation* e *privilege verification*.

Privilege Allocation. Nella prima fase la SoA crea, firma e pubblica la policy di accesso nella base di dati. Inoltre la SoA, e/o le AA da essa delegate, assegnano privilegi ai soggetti sotto forma di AC, pubblicando anch'essi nella base di dati (modello pull) o rendendoli in altro modo disponibili ai soggetti (modello push). I soggetti verranno inoltre dotati di credenziali di autenticazione dalla CA nei modi specifici della particolare tecnologia utilizzata[62].

Privilege Verification. La seconda fase viene innescata quando un soggetto (Client) invoca l'accesso ad una risorsa (Target) protetta da un gateway applicativo (PEP), eventualmente includendo le credenziali di autenticazione e gli attributi (questi ultimi solo nel caso push).

[62]Nel caso in cui essa sia X.509 si tratta di PKC forniti al soggetto e/o pubblicati nella base di dati.

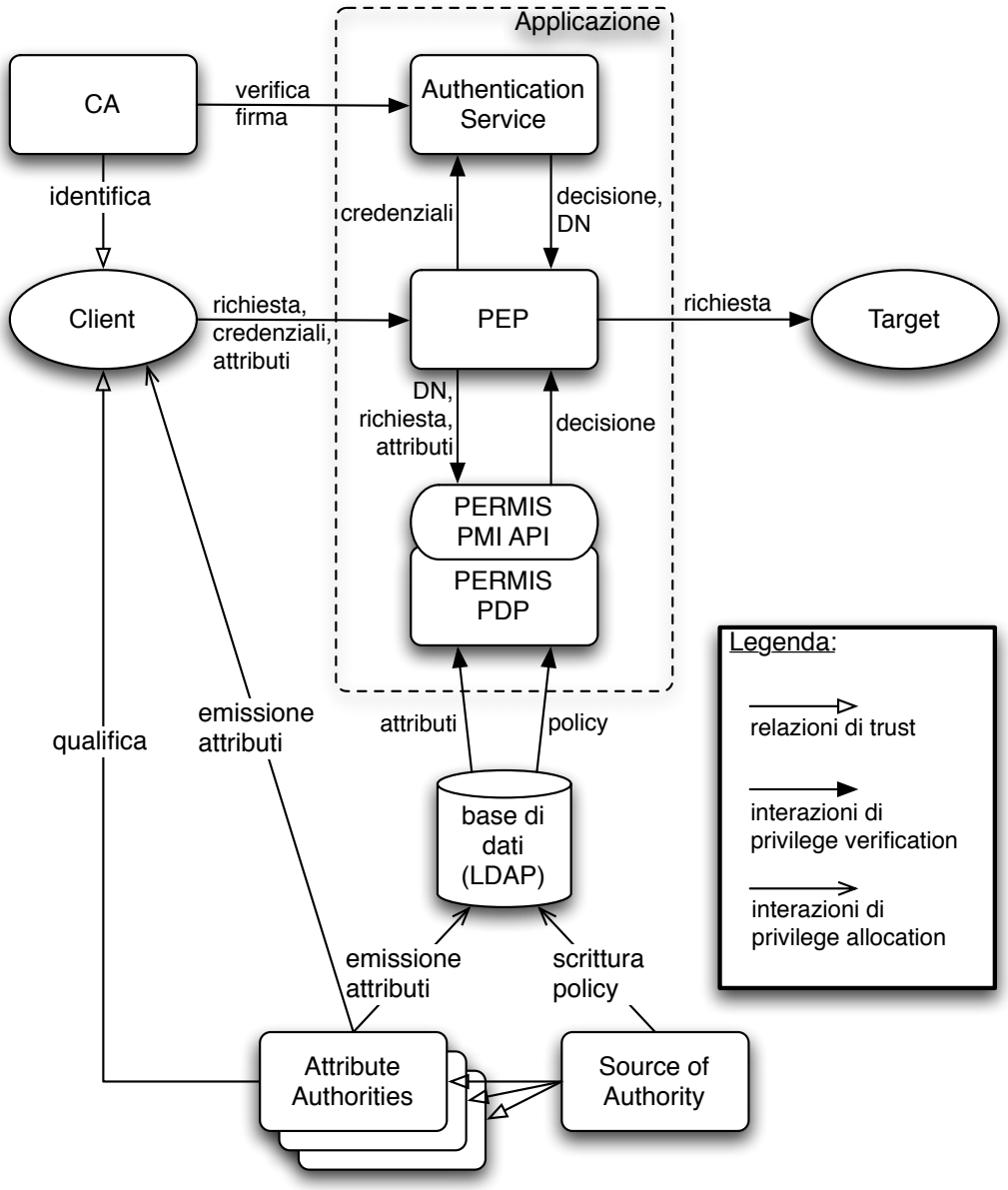

Figura 3.6: Architettura della PMI di PERMIS.

Ricevuta la richiesta, il PEP provvede ad autenticare il Client validandone le credenziali. Questa operazione può essere delegata ad un opportuno componente applicativo (Authentication Service) che agirà secondo le modalità previste dalla particolare tecnologia di autenticazione scelta, fornendo in output il nome del soggetto (DN) od un'eventuale messaggio di errore[63]. Avendo così ottenuto il DN, il PEP può invocare la funzione di decisione della API di PERMIS, eventualmente fornendo (caso push) gli attributi ricevuti contestualmente alla richiesta.

Il PDP, una volta invocato, verifica se la policy permette l'**azione** richiesta dal **soggetto** sulla **risorsa** Target, dati gli **attributi** (AC) di cui è dotato. Questi attributi possono essere quelli ricevuti nella richiesta di decisione (caso push) ovvero estratti dalla base di dati (caso pull)[64].

In caso di risposta positiva il PEP inoltrerà la richiesta al Target, altrimenti presenterà un messaggio di errore al Client.

3.3.1.2 Paradigma di controllo di accesso

PERMIS supporta il controllo di accesso di tipo RBAC gerarchico[65]. Gli attributi presenti negli AC sono pertanto interpretati come *ruoli* e possono avere relazioni di ereditarietà[66]. PERMIS permette quindi un *provisioning* a due livelli: assegnazione dei ruoli agli utenti e dei privilegi ai ruoli. Questi ultimi possono inoltre essere delegati.

3.3.1.3 Struttura della policy

In PERMIS la policy è costituita da un unico documento XML che viene utilizzato per istanziare (l'oggetto Java che implementa) il motore di decisio-

[63] Ad esempio in caso di fallita autenticazione.

[64] È possibile anche il caso 'ibrido' che utilizza entrambe le fonti di attributi.

[65] cf. 1.3.4.

[66] Si veda 3.3.1.3 per i dettagli.

ne[67]. Tale documento è suddiviso in alcune sotto-policies che regolamentano ciascuna un aspetto separato del controllo di accesso:

`SubjectPolicy` definisce i domini dei soggetti cui possono essere assegnati ruoli (o attributi in generale) ed in favore dei quali possono essere emesse decisioni di autorizzazione. Ogni dominio consiste nell'unione e/o differenza[68] di sottoalberi LDAP e può essere visto come 'gruppo' di utenti dotato di un proprio nome identificativo (`ID`).

`SOAPolicy` contiene i DN LDAP del creatore della policy – la SoA – e delle AA delegate dalla SoA ad emettere AC.

`RoleHierarchyPolicy` descrive eventuali rapporti di ereditarietà, anche multipla, tra i ruoli. È pertanto possibile definire sia molteplici ruoli superiori che ereditano i privilegi di un ruolo subordinato comune[69], sia il viceversa, ovvero singoli ruoli superiori che ereditano i privilegi di una molteplicità di ruoli subordinati[70].

`RoleAssignmentPolicy` stabilisce quali ruoli possono essere assegnati a quali (gruppi di) soggetti da quale SoA/AA. Possono essere aggiunti vincoli sul numero di deleghe possibili[71] e sulla validità temporale degli AC emessi[72].

`TargetPolicy` questo elemento definisce i domini delle risorse alle quali può essere o meno autorizzato l'accesso. La definizione dei domini avvie-

[67] Non è quindi possibile combinare policies emesse da più autorità all'interno della stessa decisione.

[68] in termini insiemistici.

[69] Si pensi ad un'Amministratore ed un Project Leader che ereditano i privilegi di un Impiegato.

[70] Si pensi ad un Manager che eredita i privilegi di tutti i suoi sottoposti.

[71] Impostare a zero per impedire la delega.

[72] I vincoli della policy hanno priorità sui quelli presenti negli AC.

ne come nella `SubjectPolicy` con l'ulteriore possibilità di specificare risorse sotto forma di URL[73].

`ActionPolicy` specifica la lista delle azioni possibili, compresi gli argomenti di ciascuna, se presenti.

`TargetAccessPolicy` costituisce la logica della decisione di accesso e si compone di un insieme di clausole. Ogni clausola autorizza una insieme di **ruoli** ad effettuare un insieme di **azioni** su di un'insieme di **risorse**. Le clausole possono essere ulteriormente ristrette tramite l'aggiunta di un'espressione booleana che opera su variabili ambientali[74] e/o sugli argomenti delle azioni.

Il listato 3.11 esemplifica i concetti appena esposti.

Listing 3.11: Un semplice esempio di policy di PERMIS.

```
<?xml version="1.0" encoding="UTF-8"?>
<X.509_PMI_RBAC_Policy OID="1.2.3.4.5">
  <SubjectPolicy>
    <SubjectDomainSpec ID="dominioItalia">
      <Include LDAPDN="c=it"/>
    </SubjectDomainSpec>
  </SubjectPolicy>
  <RoleHierarchyPolicy>
    <RoleSpec OID="1.2.826.0.1.3344810.1.1.14" Type="permisRole">
      <SupRole Value="administrator">
        <SubRole Value="user"/>
      </SupRole>
      <SupRole Value="user"/>
    </RoleSpec>
  </RoleHierarchyPolicy>
  <SOAPolicy>
    <SOASpec
       ID="LucaBianchi"
       LDAPDN="cn=Luca␣Bianchi,o=cnipa,ou=direzione"/>
    <SOASpec
       ID="MarioRossi"
       LDAPDN="cn=Mario␣Rossi,o=anagrafe,ou=provisioningCentre"/>
  </SOAPolicy>
  <RoleAssignmentPolicy>
    <RoleAssignment ID="amministrazione">
```

[73] In PERMIS l'identificatore di una risorsa può avere due formati, un DN LDAP o una URL.

[74] Ovvero contextual ACI (cf. 1.3.2) come ad es. la data, il giorno della settimana, ecc.

```
      <SubjectDomain ID="dominioItalia"/>
      <RoleList>
        <Role Type="permisRole" Value="administrator"/>
      </RoleList>
      <Delegate Depth="0"/>
      <SOA ID="LucaBianchi"/>
      <Validity/>
    </RoleAssignment>
    <RoleAssignment ID="creazioneUtenti">
      <SubjectDomain ID="dominioItalia"/>
      <RoleList>
        <Role Type="permisRole" Value="user"/>
      </RoleList>
      <Delegate Depth="0"/>
      <SOA ID="MarioRossi"/>
      <Validity/>
    </RoleAssignment>
  </RoleAssignmentPolicy>
  <TargetPolicy>
    <TargetDomainSpec ID="resource">
      <Include URL="https://example.org"/>
    </TargetDomainSpec>
  </TargetPolicy>
  <ActionPolicy>
    <Action Name="read"/>
    <Action Name="write"/>
  </ActionPolicy>
  <TargetAccessPolicy>
    <TargetAccess ID="simplePrivileges">
      <RoleList>
        <Role Type="permisRole" Value="user"/>
      </RoleList>
      <TargetList>
        <Target Actions="read">
          <TargetDomain ID="resource"/>
        </Target>
      </TargetList>
    </TargetAccess>
    <TargetAccess ID="advancedPrivileges">
      <RoleList>
        <Role Type="permisRole" Value="administrator"/>
      </RoleList>
      <TargetList>
        <Target Actions="write">
          <TargetDomain ID="resource"/>
        </Target>
      </TargetList>
    </TargetAccess>
  </TargetAccessPolicy>
</X.509_PMI_RBAC_Policy>
```

3.3.1.4 Interfaccia Amministrativa

Per agevolare il provisioning, PERMIS fornisce anche delle GUI[75] per gestire la creazione della policy (Privilege Allocator) e l'attribuzione di ruoli agli utenti (Attribute Certificate Manager). Le figure 3.7 e 3.8 ne mostrano alcune schermate catturate durante la creazione della policy nel listato 3.11.

3.3.2 Il controllo di accesso in Globus Toolkit 4

Globus Toolkit (GT) [Globus-web] è una raccolta di *middleware* Open Source per sviluppare applicazioni ed infrastrutture distribuite per il *Grid computing*[76], giunto recentemente alla sua quarta major release. Nonostante il dominio applicativo[77] sia molto diverso da quello della PA e dell'e-business in generale, le problematiche di sicurezza legate al Grid computing sono ugualmente interessanti. Infatti, la natura estremamente dinamica ed inter-istituzionale delle attività di ricerca, unita al valore (monetario e non) delle risorse condivise, hanno fatto sì che il controllo di accesso fosse, sin dall'inizio, un problema prioritario da risolvere utilizzando tecnologie allo stato dell'arte e, dove queste non esistessero, sviluppandone di nuove.

Le ultime versioni di GT (3.0 e successive) sono, in tutto e per tutto, dei frameworks di cooperazione applicativa basati sui Web Services, ivi compreso l'application server[78] ed una serie di servizi infrastrutturali che coprono problematiche di gestione dei job, archiviazione e trasferimento dei dati, monitoraggio, discovery, orchestrazione, sicurezza, ecc.

[75] sviluppate su piattaforma Java.

[76] cf. [Grid].

[77] Lo scopo originale del Grid è la condivisione e l'aggregazione inter-istituzionale di risorse computazionali per la ricerca scientifica.

[78] Si tratta di una versione customizzata di Apache Axis [Axis].

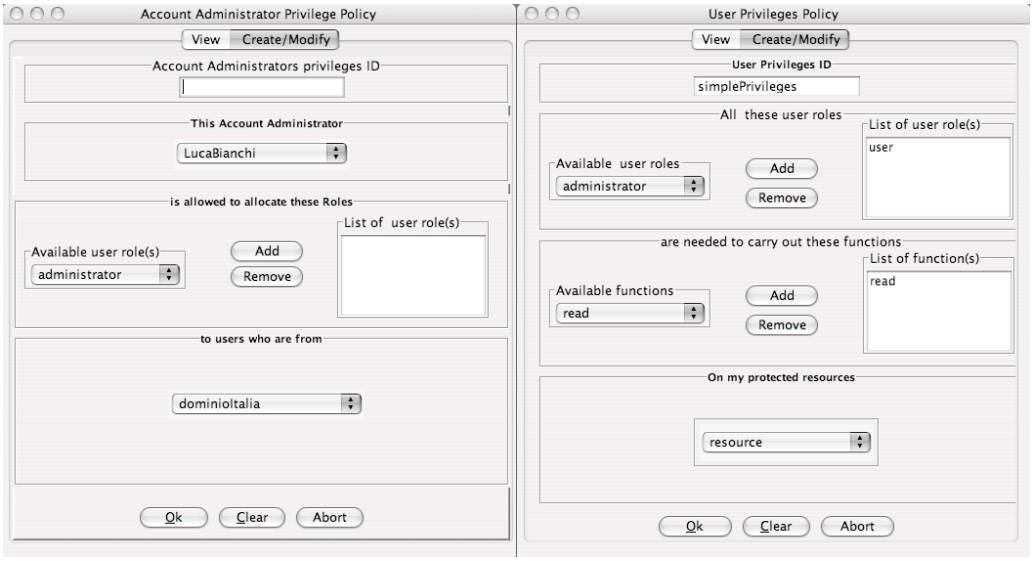

Figura 3.7: Alcune schermate dell'interfaccia per la creazione della policy di PERMIS.

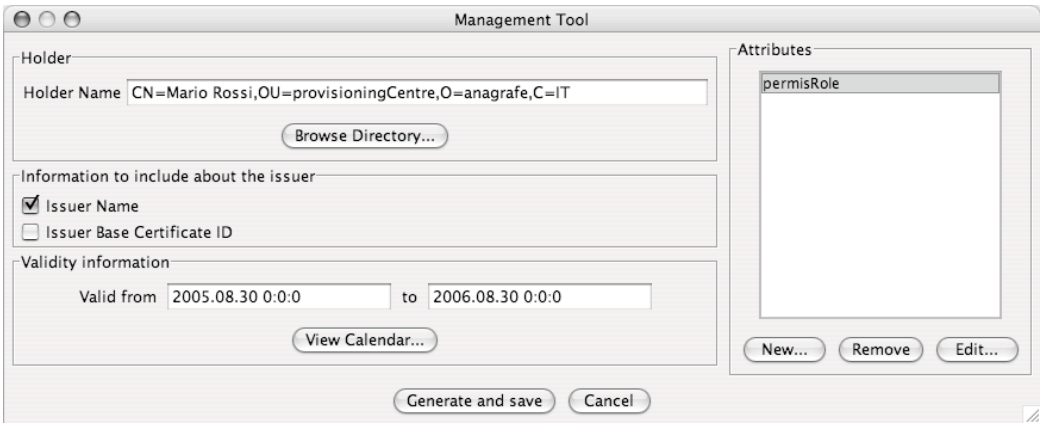

Figura 3.8: Interfaccia amministrativa di PERMIS per la creazione di AC.

3.3.2.1 Grid Security Infrastructure

L'infrastruttura di sicurezza di GT, detta Grid Security Infrastructure (GSI)[79], utilizza PKC X.509[80] per identificare entità persistenti come le applicazioni e gli utenti. GSI supporta inoltre il meccanismo dei *proxy certificates*, come specificato in [GridProxyCert], per generare tokens dinamici a partire dai PKC. Tali tokens potranno successivamente essere utilizzati come deleghe di privilegi a terze entità[81] o per implementare il SSO. Le deleghe possono (e dovrebbero) essere ristrette imponendo intervalli di validità più ristretti delle credenziali originali ed eventualmente includendo un'asserzione di autorizzazione SAML[82] al loro interno che specifichi le sole azioni per le quali possono essere utilizzate.

Basate su tali token, confidenzialità, non ripudiabilità ed autenticità dei messaggi possono essere implementate a livello di trasporto (usando TLS) e/o a livello SOAP (usando WSS[83] o WS-SecureConversation[84]). In GT 4.0 è stata aggiunta la possibilità di usare per questi scopi anche nome utente e password come credenziali anziché X.509.

3.3.2.2 Authorization Framework

Una caratteristica di GSI di sicuro interesse per questa tesi è la particolare infrastruttura che l'application server mette a disposizione – a partire dalla

[79] cf. [GSI].
[80] cf. 3.1.2.
[81] umane o applicative.
[82] Di fatto questa possibilità rende ridondante l'utilizzo dei proxy certificate per l'emissione di deleghe. Le asserzioni SAML contengono già tutti gli elementi informativi e crittografici necessari ad implementare una delega, senza la necessità di essere incapsulate in un certificato. L'adozione di questa scelta da parte GT è giustificata soprattutto da motivi compatibilità con le precedenti versioni che non facevano uso di SAML. L'utilizzo dei proxy certificate tuttavia impedisce una piena compatibilità con le specifiche WS-Iteroperability.
[83] cf. [WSS].
[84] cf. [WS-SC]

versione 4.0 di GT – per il controllo di accesso ai singoli Web Services (o più propriamente Grid Services).

Come mostrato in figura 3.9, immediatamente dopo l'autenticazione del soggetto richiedente, il controllo viene passato ad una catena di moduli di autorizzazione. Ciascuno di essi può essere un PDP vero e proprio oppure un PIP che si occupa di recuperare attributi che potranno essere utilizzati dagli stadi successivi. La decisione negativa di un PDP provoca il fallimento dell'intera catena ed il rigetto della richiesta del Client.

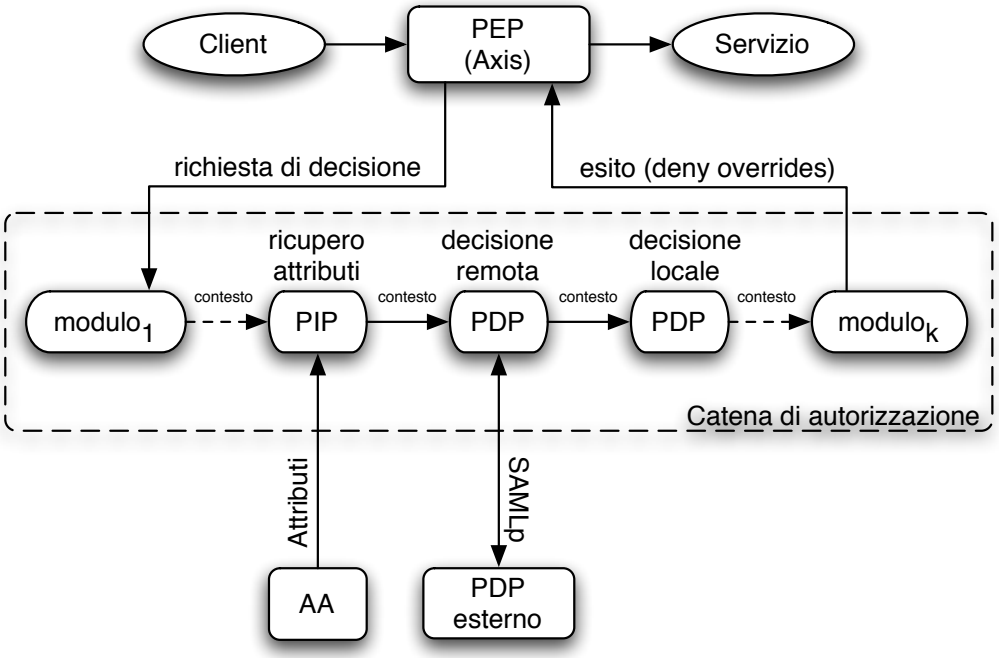

Figura 3.9: La catena di moduli di autorizzazione (lato server) in GT 4.

La conformazione della catena è liberamente configurabile a tempo di deployment ed è possibile inserire lo stesso PDP o PIP in più istanze fornendo configurazioni diverse per ciascuna di esse. Oltre alle interfacce Java per sviluppare i propri moduli di autorizzazione, il framework di GT 4 fornisce allo sviluppatore anche un certo numero di moduli predefiniti tra i quali va senz'altro citato `SAMLAuthorizationCallout`. Esso permette di delegare la

decisione di accesso ad un PDP esterno interrogandolo tramite una richiesta di asserzione di autorizzazione SAML[85].

3.3.2.3 Delegation Service

Delegation Service è l'infrastruttura che permette ad un soggetto di emettere e rinnovare le deleghe verso i servizi ospitati dall'application server. Tali credenziali vengono immagazzinate[86] per poter essere successivamente richieste dai singoli servizi (tramite un'apposita API Java) che le useranno per impersonare il soggetto nel rispetto delle restrizioni presenti nel proxy certificate emesso. I servizi possono inoltre registrarsi con il Delegation Service per ottenere la notifica di ogni rinnovo di dette credenziali.

Ad esempio, per l'emissione di una delega, si hanno le seguenti interazioni[87], come illustrato in figura 3.10:

1. Il client ottiene il PKC del Delegation Service.
2. Il client crea il proxy certificate contenente la delega al Delegation Service.
3. Il client invia il proxy certificate al Delegation Service ed ottiene un riferimento (EPR[88]) alla Delegated Credential Resource creata.
4. Il client invoca il servizio delegato, fornendogli il puntatore (EPR) alla credenziale.
5. Il servizio delegato può accedere alla delega utilizzando un'opportuna API.

[85] cf. 3.1.1.

[86] Per ogni nuova delega viene creata una WS-Resource [WSRF].

[87] Sequenza semplificata per motivi di sintesi. Si veda http://www.globus.org/toolkit/docs/4.0/security/delegation/developer-index.html per i dettagli.

[88] End Point Reference.

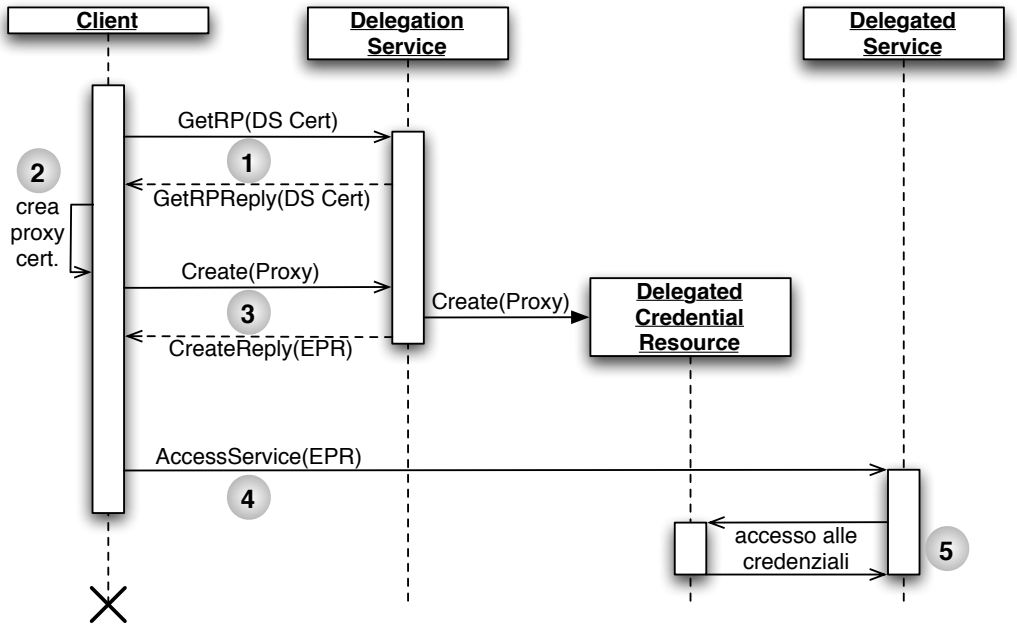

Figura 3.10: Diagramma di sequenza per la creazione di una delega attraverso Delegation Service.

3.3.2.4 Community Authorization Service

Community Authorization Service (CAS) [CAS] è un PDP che emette asserzioni di autorizzazione SAML su richiesta dei clients che, a loro volta, le incapsuleranno in proxy certificates. CAS non è un motore di decisione generico come PERMIS[89], ma ha l'obiettivo specifico di implementare il concetto di Virtual Organization (VO).

In [VO], una VO è definita come una collezione dinamica di risorse ed utenti, unificata da un obiettivo comune (es. un progetto di ricerca) e possibilmente distribuita su più domini amministrativi. Pertanto il compito principale di un CAS è quello di gestire utenti, risorse e policies per conto di una VO. In altre parole, per ogni VO sarà istanziato un CAS a cui i possessori delle risorse allocate alla VO potranno delegare (parte de) l'autorizzazione degli accessi da parte di utenti della VO.

[89] cf. 3.3.1, 3.3.2.5.

Il linguaggio di policy di CAS, implementato tramite un piccolo DB SQL, permette di esprimere l'appartenenza di utenti a gruppi e di assegnare a questi ultimi un insieme di azioni autorizzate su risorse. Esso pertanto è ben lontano dalla ricchezza del paradigma RBAC, ma è più che sufficiente per le esigenze di molti gruppi di ricerca.

3.3.2.5 PERMIS Authorization Service

ISSRG ha contribuito al progetto Globus fornendo il proprio PDP PERMIS[90] sotto forma di Grid Service. Si tratta sostanzialmente di un'authority SAML[91] che dispensa asserzioni di autorizzazione create in funzione delle decisioni di accesso operate da PERMIS.

Utilizzando SAML sia per la comunicazione[92] che per la creazione[93] di asserzioni certificate, PERMIS Authorization Service può (ed è pensato per) essere invocato come decisore remoto da parte delle catene di autorizzazione di altri deployments di GT, secondo le modalità descritte nella sezione 3.3.2.2. Ciò non esclude che possa essere utilizzato anche localmente o da un qualsiasi client SAML.

[90] cf. 3.3.1.
[91] cf. 3.1.1.
[92] cf. 3.1.1.2, 3.1.1.3.
[93] cf. 3.1.1.1.

Capitolo 4

Proposta di framework per l'Identity Management nella Pubblica Amministrazione

In questo capitolo viene definito in dettaglio il framework che la tesi propone a soluzione dei requisiti della PA espressi nel capitolo 2. Esso pone le sue fondamenta sulle tecnologie di SSO scelte nel capitolo 3[1] e considerate lo stato dell'arte per implementare soluzioni di sicurezza distribuite ed interoperabili su larga scala.

Per garantire il pieno soddifacimento dei requisiti, su queste 'fondamenta' (sezione 4.1) sono state posate specifiche infrastrutture distribuite per l'attribuzione di ruoli (sezione 4.2.2) ed il controllo di accesso basato su policy (sezione 4.2.3). A quest'ultima è stata dedicata una particolare attenzione in quanto si tratta di un'infrastruttura raramente implementata in prodotti e contesti analoghi, ma che soddisfa i requisiti ed abilita nuovi scenari applicativi al contempo integrandosi con il resto del framework.

[1] Si tratta in particolare di Shibboleth, cf. 3.2.1.

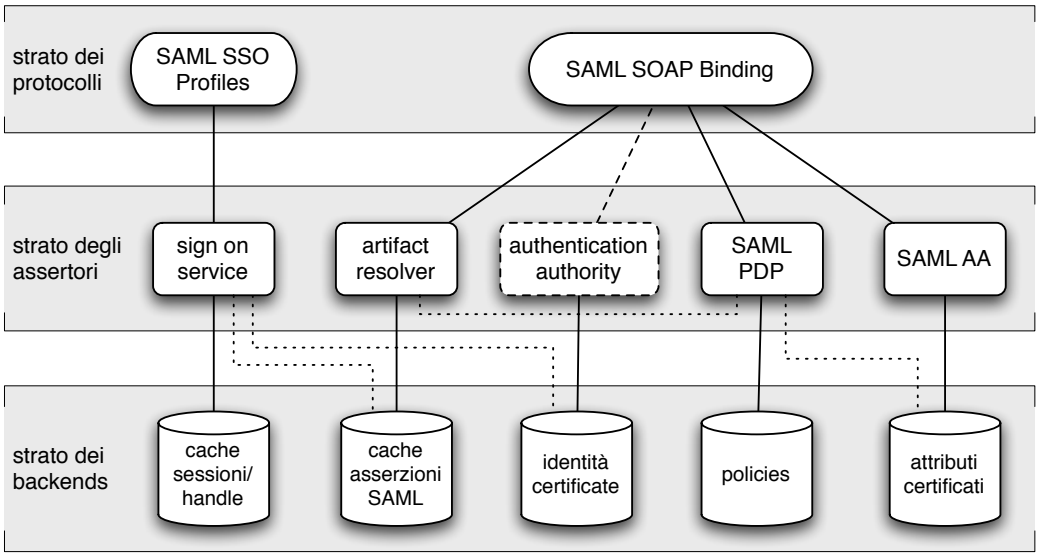

Figura 4.1: Diagramma a strati delle varie tipologie di authority previste dallo standard SAML.

Centralità di SAML. Come mostrato anche in figura 4.1, il 'cemento' che unisce i componenti dell'architettura è SAML[2], una tecnologia che ha recentemente ottenuto grande successo in molteplici ambiti di utilizzo. All'interno del framework qui progettato, essa viene utilizzata sia per i protocolli di comunicazione che per rendere interoperabile il contenuto delle credenziali e delle autorizzazioni.

Oltre che come tecnologia, SAML è stato utilizzato come modello di IM. I concetti di asserzione ed autorità assertrice presenti in [SAML-core][3] costituiscono infatti un elemento di intermediazione tra i particolari meccanismi di sicurezza (ed in generale di IM) in atto presso i singoli domini cooperanti. Di conseguenza, il problema di riconoscere come valide credenziali od autorizzazioni (in generale ACI) emesse da altri domini si riduce ad un problema di trust tra i domini stessi[4], a sua volta agevolato dalla presenza di apposite

[2] cf. 3.1.1

[3] Si veda in particolare il modello descritto nelle pp. 8-9.

[4] L'approccio assertivo è qui visto in contrapposizione ad una visione più ontologica tipica delle credenziali emesse da una struttura fortemente gerarchica come ad esempio

soluzioni[5].

Da SAML è stata infine tratta parte dell'architettura del componente di autorizzazione distribuito (il PDP), in particolar modo per quanto riguarda i flussi di asserzioni che concorrono all'emissione della decisione, come è documentato più in dettaglio nella sezione 4.2.3.

una PKI X.509. In altre parole la validità di un'asserzione non è assoluta, ma subordinata ad un rapporto di fiducia di chi la consuma nei confronti di chi la produce.

[5]Ad esempio i metadati SAML 2.0 implementati in Shibboleth, cf. 3.2.1.3.

4.1 Sistema federato di autenticazione

La tecnologia scelta per soddisfare il requisito 2.3.3.3 è Shibboleth[6], in quanto:

- non impone la presenza di particolari applicativi sulle piattaforme client oltre ad un browser web che supporti HTTP/HTTPS, redirect e POST (requisiti 2.3.1.1 e 2.3.1.3);
- permette una distribuzione il più possibile periferica delle competenze e delle responsabilità in termini di autenticazione (tutti i requisiti della classe 2.3.3) ed inoltre non necessita di dispositivi fisici di sicurezza per i terminali (requisito 2.3.1.4);
- è predisposto ad utilizzare una PKI X.509 per l'autenticazione dei domini (requisito 2.3.4.1) e comunicazioni SOAP dirette tra i domini nei protocolli in cui non è necessario un collegamento con l'utente (requisito 2.3.4.2).

Shibboleth inoltre è dotato di una AA (requisito 2.3.5.1) e sopperisce autonomamente alle funzionalità di marcatura temporale (non richiede 2.3.5.4). Il progetto infine è distribuito con licenza Open Source (requisiti 2.3.6.1 e 2.3.6.2) ed è opera di una comunità di sviluppatori (principalmente accademici) che ne migliora ed arricchisce costantemente le funzionalità (requisito 2.3.6.3).

4.1.1 Modello di federazione

Da un punto di vista prettamente legale e normativo, gli enti responsabili dell'identificazione del cittadino sono in primo luogo le anagrafi comunali. Volendo quindi implementare un sistema federato di autenticazione su web usando Shibboleth, la soluzione ideale sarebbe quella di istituire un IdP per

[6] cf. 3.2.1

ogni comune, che a sua volta potrebbe sfruttare per l'autenticazione le infrastrutture per la gestione della Carta di Identità Elettronica (CIE) e l'Indice Nazionale delle Anagrafi (INA), laddove presenti.

Tuttavia non è affatto scontato che i comuni siano tutti disposti a fornire un servizio di IdP per i propri cittadini. Si pensi ai comuni di piccole dimensioni che potrebbero non essere ancora stati coinvolti dagli ammodernamenti previsti dal piano di e-Government, o non avere i fondi o le infrastrutture per gestire tali servizi.

È necessario pertanto un modello di federazione che permetta all'autenticazione di essere effettuata su vari livelli amministrativi: a livello di comune, 'area' (insieme di comuni), provincia o centralmente a livello di regione.

Al fine di semplificare l'esposizione, inoltre, conviene astrarre dalla particolare natura istituzionale degli enti amministrativi classificandoli in due categorie logiche:

Dominio delle Identità (DI) corrisponde al dominio[7] amministrativo responsabile dell'autenticazione di un insieme di cittadini. Più DI possono delegare un'unica infrastruttura IdP per l'autenticazione dei propri cittadini. Un DI *produce* identità digitali.

Dominio dei Servizi (DS) è il dominio amministrativo responsabile per un insieme di servizi al cittadino, che accetti le autenticazioni effettuate dagli IdP. Si suppone che ogni DS sia dotato di un'infrastruttura di SP[8]. Un DS *consuma* identità digitali.

La descrizione dell'architettura prosegue mostrando diversi pattern architetturali che esprimono altrettante topologie di trust tra DI, IdP, DS e SP.

[7] Ai sensi della terminologia SPC (cf. 2.1.1).

[8] Un SP risulta infatti meno oneroso di un IdP, sia in termini di risorse che di responsabilità. Inoltre, se un dominio ha la possibilità di gestire dei servizi, è probabile che sia in grado di affiancare a questi un SP.

4.1.1.1 Autenticazione completamente distribuita

Come già accennato, il modello di federazione Shibboleth ideale è quello in cui ogni SP protegge le risorse di un DS ed ogni IdP autentica gli utenti di un DI. L'IdP è quindi responsabile del minimo numero di utenti (istituzionalmente) possible. La struttura di trust di una tale federazione è mostrata in figura 4.2.

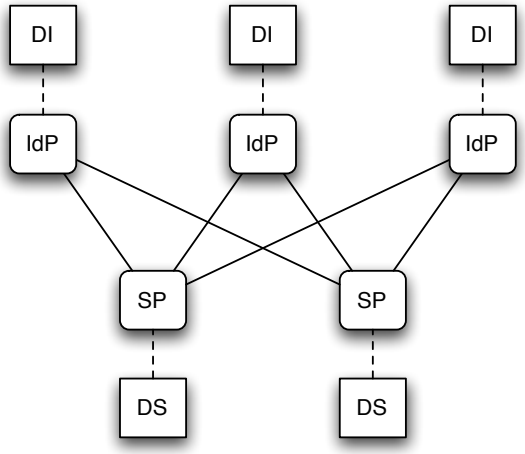

Figura 4.2: Topologia del trust tra i domini in caso di autenticazione completamente distribuita. Le linee tratteggiate indicano le deleghe per l'autenticazione e la fornitura di servizio, quelle continue i legami di trust della federazione Shibboleth implementata.

Uno svantaggio di questa soluzione, oltre all'imposizione di adottare un IdP per tutti i DI, è quello di portare a strutture di trust più complesse.

4.1.1.2 Autenticazione completamente centralizzata

Il modello completamente centralizzato prevede l'esistenza di un'unico IdP fidato per tutta la federazione, come mostrato in figura 4.3.

Lo svantaggio di questa struttura di trust è che il provisioning degli utenti viene effettuato centralmente, e ciò può non essere possibile per grandi bacini

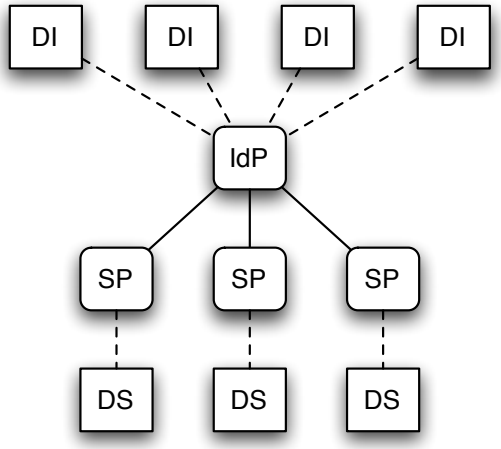

Figura 4.3: Topologia di trust tra i domini in caso di autenticazione completamente centralizzata.

di utenza[9]. La soluzione, tuttavia, permette comunque la separazione tra la fornitura dei servizi e la gestione delle utenze ed ha il minimo impatto infrastrutturale.

4.1.1.3 Modello ibrido

Il modello completamente distribuito e quello completamente centralizzato appena descritti costituiscono due casi estremi all'interno dei quali è possibile trovare una soluzione di compromesso. È necessario infatti da un lato limitare l'impatto infrastrutturale della soluzione, dall'altro cercare di evitare colli di bottiglia, sia da un punto di vista delle prestazioni che della complessità di gestione.

A tal fine è bene consentire la presenza di più IdP e contemporaneamente anche la possibilità per i DI più piccoli di aggregarsi in *comunità di cittadini*[10] servite da un'unico IdP. La soluzione è illustrata in figura 4.4.

[9] Si pensi ai 250.000 utenti previsti nel bando di Regione Campania.

[10] Analogamente a quanto fatto anche dal progetto ICAR (cf. 2.2.1.2).

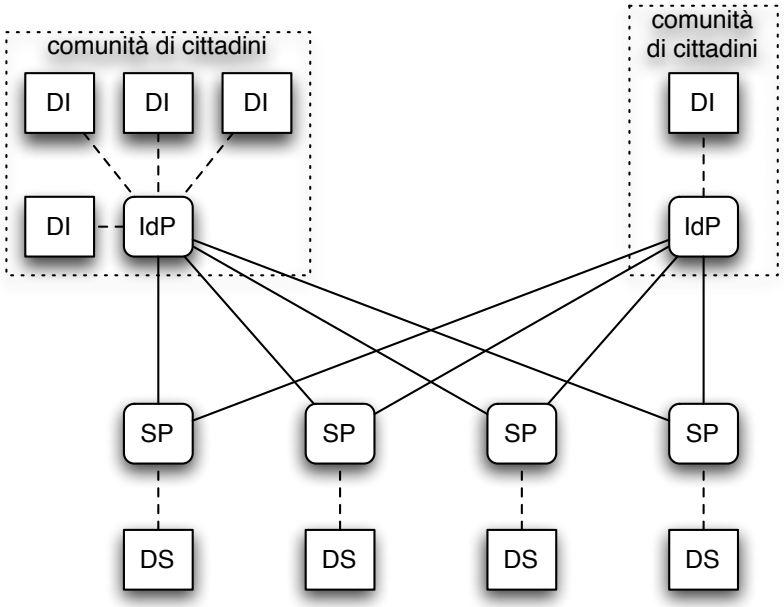

Figura 4.4: Modello ibrido di trust tra i domini. Ad ogni IdP corrisponde una comunità di cittadini che lo utilizzano per autenticarsi.

4.1.2 Gestione delle identità

Fermo restando che la tecnologia scelta permette ai singoli IdP di mantenere la propria rappresentazione interna delle identità digitali, è necessario standardizzare il formato dell'identificatore utente 'federato' `NameIdentifier`[11], utilizzato nelle comunicazioni tra IdP e SP e nelle asserzioni. All'interno dei singoli IdP è inoltre importante puntualizzare quale sia la corrispondenza (*mapping*) tra il nome utilizzato internamente e quello federato, con particolare riferimento ad eventuali tecniche di offuscamento e pseudonimizzazione[12] a tutela della privacy.

Essendo indispensabile per l'interoperabilità, il formato del `NameIdentifier` utilizzato dovrà essere pubblicato nei metadati[13] della federazione.

[11] cf. 3.1.1.1, 3.2.1.3.
[12] Tramite l'uso di handles opachi ed effimeri (cf. 3.2.1.3).
[13] cf. 3.2.1.3.

4.1.2.1 Formato per il nome utente federato

Nella scelta del formato del `NameIdentifier`, tenendo conto delle infrastrutture preesistenti e delle soluzioni suggerite dalla PA stessa nei documenti analizzati nel capitolo 2, si può restringere il campo a tre possiblità:

- utilizzare i certificati X.509 (laddove presenti),
- riferire gli utenti tramite il Codice Fiscale
- non effettuare alcuna conversione ed utilizzare l'identificatore locale dei singoli IdP.

`NameIdentifier` derivato dal certificato X.509. Nel caso in cui si usino certificati X.509 per l'autenticazione dei cittadini, è possibile usare il campo Subject del certificato (od un sottoinsieme delle informazioni in esso contenute) per identificarli univocamente nella federazione. Si può quindi configurare Shibboleth in modo che non trasformi il `NameIdentifier` utilizzato localmente (name mapping di tipo `Principal`, cf. 3.2.1.3)[14].

`NameIdentifier` pari al Codice Fiscale. Il bando di Regione Toscana (cf. 2.2.2) prevede espressamente l'utilizzo del Codice Fiscale per identificare univocamente i cittadini. Per implementare questo requisito vi sono due possibilità:

- Imporre agli IdP l'utilizzo del Codice Fiscale come nome utente anche internamente. Si tratta della soluzione più semplice, ma è in contraddizione con il requisito 2.3.3.4 (che tuttavia non è stato espresso da Regione Toscana).
- Implementare un componente di name mapping all'interno di Shibboleth che si occupi di convertire il nome utente locale nel suo Codice

[14] In questo caso SAML prevede inoltre in [SAML-core] un formato specifico per l'elemento `NameIdentifier`, caratterizzato dalla presenza dell'attributo `Format` con valore `urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName`.

Fiscale, eventualmente interrogando opportuni servizi. È una soluzione più complessa e costosa, ma rispetta il requisito 2.3.3.4 ed implementa inoltre il requisito 2.3.3.5.

Nome utente locale ai singoli IdP. Si tratta di fatto di una generalizzazione del primo caso (certificati X.509) e ne utilizza la stessa configurazione per Shibboleth (name mapping di tipo `Principal`) ma senza imporre l'utilizzo di un particolare meccanismo di autenticazione. Si noti che l'unicità dell'elemento `NameIdentifier` è garantita dal fatto che Shibboleth utilizza sempre l'attributo `NameQualifier` (cf. 3.1.1.1) con valore pari all'identificatore unico dell'IdP autenticante. In altre parole Shibboleth risolve a monte il problema dell'unicità facendo sì che il nome utente federato sia la coppia <nome locale, IdP>.

4.1.2.2 Pseudonimizzazione degli utenti

Sebbene non sia nei requisiti, è interessante considerare la possibilità di utilizzare `NameIdentifier` opachi ed effimeri come descritto nella sezione 3.2.1.3. Una tale funzionalità potrebbe essere usata per garantire il servizio di SSO anche a SP ai quali non si volesse rilasciare l'identità effettiva degli utenti.

Si aprirebbe così la strada a scenari in cui la PA si occuperebbe dell'autenticazione e dei servizi essenziali, delegando la fornitura di servizi addizionali ad enti privati, senza per questo compromettere la privacy dei propri cittadini.

Per abilitare tale funzionalità, è necessario configurare Shibboleth affinché utilizzi un name mapping di tipo `SharedMemoryShibHandle` o `CryptoHandleGenerator`.

4.1.2.3 Federazioni multiple

È da notare infine il fatto che un IdP può gestire federazioni multiple[15] e pertanto configurazioni che usano un formato di `NameIdentifier` piuttosto che

[15] A partire da Shibboleth 1.3.

un'altro possono esistere simultaneamente per la stessa comunità di cittadini.

Quindi, si può pensare di realizzare federazioni multi-livello, con caratteristiche distinte in termini di privacy e sicurezza a seconda del servizio al quale l'utente desidera accedere.

4.2 Infrastrutture per l'attribuzione di ruoli e l'autorizzazione

Per i servizi di attribuzione di ruolo (AA) ed autorizzazione (PDP) si è scelto di utilizzare modalità di interfacciamento simili, basate sul concetto di *SAML responder*.

Come mostrato anche in figura 4.1, un SAML responder è un web service che emette asserzioni SAML se opportunamente interrogato utilizzando protocollo SAML SOAP Binding[16]. La scelta di un protocollo basato su SOAP permette di convogliare agevolmente le comunicazioni da e verso un SAML responder attraverso le infrastrutture di scambio messaggi di SPC[17], soddisfacendo in tal modo il requisito 2.3.4.2.

4.2.1 Modello di federazione

Diversamente da quanto avviene per il caso dell'autenticazione, la distribuzione degli enti qualificatori/autorizzatori può avvenire anche in base alla funzione (albo degli avvocati, camera di commercio, questura, ecc.) oltre che alla collocazione territoriale.

4.2.1.1 Modello completamente distribuito

La collocazione naturale per le AA ed i PDP è presso gli enti qualificatori ed autorizzatori. Tuttavia, come nel caso dell'autenticazione federata non è detto che tali enti abbiano la possibilità di fornire servizi e pertanto deve essere prevista la delegabilità di una tale funzione ad un ente aggregatore.

[16] cf. 3.1.1.3.

[17] Si tratta sostanzialmente di incapsulare richieste e risposte SAML all'interno di buste di e-Government (cf. 3.1.1.2, 2.1.1.3).

4.2.1.2 Modello completamente centralizzato

Un modello abbastanza diffuso[18] in PA è quello di gestire questo tipo di servizi centralmente (tipicamente a livello regionale) e permettere agli enti qualificatori/autorizzatori di popolare in maniera asincrona[19] i backends di tali servizi, come mostrato in figura 4.5.

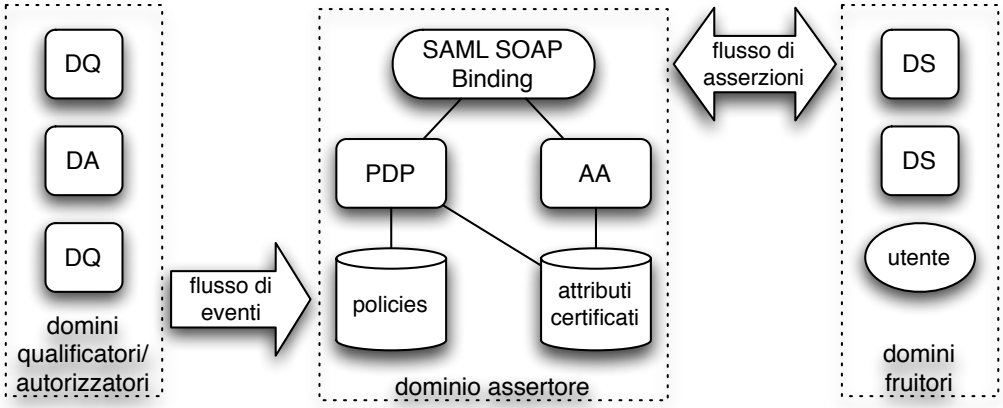

Figura 4.5: Modello centralizzato per l'attribuzione di ruoli e l'autorizzazione con aggiornamento per eventi.

4.2.1.3 Modello ibrido

Laddove possibile, la soluzione di compromesso che permetta alle asserzioni di essere emesse sia centralmente che localmente è preferibile. Tuttavia una tale soluzione dovrebbe comunque prevedere la presenza di PDP e AA centralizzati presso ogni IdP.

4.2.2 SAML Attribute Authority

La tecnologia scelta per implementare la AA è ancora una volta Shibboleth[20]. La AA di Shibboleth infatti può essere utilizzata indipendentemente dal resto

[18] Il bando di Regione Toscana fa ad esempio riferimento a questo modello (cf. 2.2.2).
[19] Tipicamente attraverso pubblicazioni di eventi (cf. 2.1.1.5).
[20] cf. 3.2.1.

del pacchetto di SSO.

Questa soluzione permette di soddisfare, oltre a quelli già riportati nella sezione 4.1, i seguenti requisiti:

2.3.5.1 La AA di Shibboleth implementa (ovviamente) una infrastruttura per l'attribuzione di ruoli.

2.3.4.3 La AA di Shibboleth è dotata degli opportuni adattatori per interfacciarsi con le tipologie più diffuse di base di dati.

L'utilizzo di una AA Shibboleth presenta inoltre il grosso vantaggio di poter essere utilizzata contestualmente alle interazioni di SSO ('interamente' ad un IdP) per fornire qualifiche dell'utente oltre che mere asserzioni di autenticazione.

La AA di Shibboleth può pubblicare le informazioni sulla propria configurazione (formato del `NameIdentifier` e degli attributi, endpoints, chiave pubblica, ecc.) all'interno dei metadati[21] di federazione.

È bene notare che la AA qui descritta non vuole porsi in alternativa alle soluzioni preesistenti nella PA per l'attribuzione di ruoli, ma piuttosto migliorarne l'interoperabilità. Queste ultime infatti sono tipicamente dei DB LDAP contenenti AC X.509 (e quindi in un formato binario piuttosto ostico da gestire) che la AA proposta può aiutare ad 'esternalizzare' in un formato moderno, portabile ed autodocumentante come XML.

4.2.2.1 Identificazione dei soggetti

La AA di Shibboleth supporta le stesse tipologie di `NameIdentifier` del sottosistema di SSO per riferire gli utenti (cf. 4.1.2). Nel caso in cui la AA sia usata all'interno di un IdP, essa potrà accettare anche formati di `NameIdentifier` distinti da quelli usati per il SSO.

L'utilizzo di handles opachi[22] per riferire gli utenti ha senso solo nel caso in cui la AA partecipi ad interazioni di SSO.

[21] cf. 3.2.1.3.
[22] cf. 3.2.1.3.

4.2.2.2 Identificazione degli attributi

La AA di Shibboleth può pubblicare nei metadati[23] il proprio *vocabolario* di attributi, ovvero l'insieme degli attributi supportati (nome e tipo) ed opzionalmente per ciascuno di essi l'enumerazione dei possibili valori.

4.2.2.3 Integrazione con i data backends

La AA di Shibboleth è fornita degli adattatori – chiamati *attribute resolvers* – per basi di dati LDAP e SQL, quest'ultima via API JDBC[24]. Sono forniti inoltre le API e degli esempi in codice sorgente per implementare nuovi attribute resolvers[25].

4.2.2.4 Gestione della privacy

Come accennato nella sezione 3.2.1.2, l'emissione (di asserzioni) di attributi può essere regolata da una ARP. Tramite le ARP è possibile specificare quali valori di quali attributi rilasciare a quali domini richiedenti.

L'attuale implementazione di Shibboleth supporta due ARP per ogni utente: quella espressa centralmente dalla AA e quella specificata dall'utente stesso. In quest'ultimo caso non sono ancora state sviluppate apposite interfacce di gestione, ma sarebbero facilmente realizzabili ex-novo.

Al fine di garantire la privacy e la trasparenza verso i cittadini si raccomanda l'utilizzo di ARP utente e di opportune interfacce che ne permettano la configurazione, contestualmente agli altri aspetti del profilo.

4.2.3 SAML Policy Decision Point

Per alcune problematiche avanzate di e-Government può essere necessario effettuare delle decisioni di accesso che verranno poi utilizzate da un dominio

[23] cf. 3.2.1.3.

[24] cf. [JDBC].

[25] Questa caratteristica è stata utilizzata nell'implementazione del prototipo (cf. capitolo 5) allo scopo di permettere il testing delle funzionalità del sistema realizzato indipendentemente dalla particolare tipologia di backend.

(od un componente distribuito) distinto dal decisore. La separazione tra le funzioni di decisione e di attuazione della stessa è un concetto che ha un'utilità tecnica, ma anche un corrispettivo nel mondo burocratico[26].

Viene qui mostrata l'archiettura di un possibile PDP, basata su componenti Open Source[27], che utilizza SAML per i protocolli ed il formato della decisione emessa.

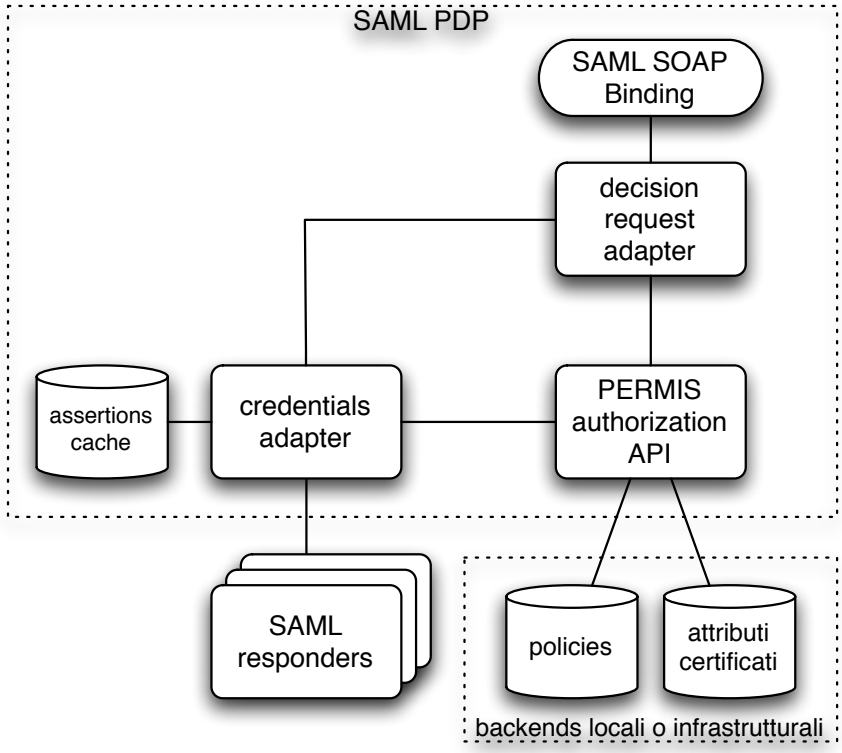

Figura 4.6: Architettura interna del SAML PDP proposto.

L'architettura del PDP, mostrata in figura 4.6, è fortemente ispirata al PERMIS Authorization Service[28] sviluppato all'interno di GT[29]. Si è scelto tuttavia di non riutilizzare il codice del GT in quanto si basa su delle esten-

[26] Si pensi ai "nulla osta", alle "autorizzazioni a procedere", alle deleghe, alle procure, ecc.

[27] In questo modo vengono soddisfatti i requisiti 2.3.6.1 e 2.3.6.2.

[28] cf. 3.3.2.5.

[29] cf. 3.3.2.

sioni allo standard SAML. PERMIS Authorization Service utilizza inoltre una versione customizzata (e peraltro piuttosto datata) di OpenSAML 0.8.

Coerentemente con il modello architetturale delineato in [SAML-core], lo scopo del SAML PDP è quello di emettere asserzioni di autorizzazione SAML che potranno essere successivamente utilizzate come credenziali per accedere a servizi o per ottenere ulteriori asserzioni. Tale componente dovrà soddisfare tutti i requisiti della categoria 2.3.2.

SAML[30] prevede che una decisione di accesso si basi su quattro elementi informativi:

- il **soggetto** che richiede l'autorizzazione;
- la **risorsa** di cui viene richiesto l'accesso;
- l'**azione** che il soggetto intende compiere sulla risorsa;
- le **credenziali** (*evidence*) che rendono il soggetto idoneo ad ottenere l'autorizzazione.

4.2.3.1 Identificazione dei soggetti

Per l'identificazione dei soggetti valgono le stesse considerazioni effettuate nelle sezioni 4.1.2 e 4.2.2.1.

Va tuttavia precisato che Shibboleth non prevede la presenza di PDP nei propri protocolli e quindi nel caso in cui un PDP venga usato all'interno di un IdP esso dovrà essere opportunamente integrato. Una tale integrazione dovrà comprendere un meccanismo[31] che permetta al PDP di risolvere i nomi utente federati in soggetti utilizzabili per la decisione, accedendo eventualmente alle funzionalità di name mapping di Shibboleth.

Quest'ultima operazione è tuttavia necessaria solo per le decisioni basate sull'identità dell'utente. È pensabile anche un modello di PDP che

[30] cf. 3.1.1.1.

[31] Per un esempio si veda il componente "handle resolver" realizzato nell'implementazione del PDP del prototipo e descritto nel capitolo 5.

esprima decisioni 'anonime' basate sulle sole asserzioni presentate dall'utente nella richiesta, od ottenute dall'IdP che lo ha autenticato (utilizzando come 'chiave' lo handle). Un tale modello non potrebbe tuttavia implementare internamente il controllo di accesso basato su ACL.

4.2.3.2 Identificazione delle risorse e delle azioni

SAML prevede che, nelle asserzioni di autorizzazione[32] come anche nelle corrispondenti richieste[33], le risorse vengano identificate tramite URI. Le azioni potranno invece essere stringhe qualsiasi a condizione che appartengano al vocabolario (*namespace*) prescelto per la particolare risorsa.

Ad esempio, nel caso in cui le risorse siano applicazioni web, queste potranno essere identificate dalla propria URL. Le azioni saranno tipicamente quelle previste dal protocollo HTTP[34] oppure specifiche per la particolare applicazione[35].

4.2.3.3 Gestione delle credenziali in `Evidence`

Sia le richieste di autorizzazione che le corrispondenti asserzioni prevedono l'esistenza di un elemento `Evidence` contenente un insieme di asserzioni[36]. Nel primo caso (la richiesta di autorizzazione) le asserzioni in `Evidence` sono da interpretarsi come le credenziali **presentate** dal client per ottenere l'asserzione di autorizzazione dal PDP. Nel secondo caso (l'autorizzazione) le asserzioni in `Evidence` sono quelle **utilizzate** nel processo di decisione[37].

Quindi le 'evidenze' nell'asserzione di autorizzazione emessa dal PDP sono l'unione di quelle fornite dal client (*push*) e delle informazioni ottenute da opportuni SAML responders o da basi di dati locali durante il processo di autorizzazione (*pull*). Le ACI ottenute da queste ultime tuttavia possono

[32] cf. 3.1.1.1.
[33] cf. 3.1.1.2.
[34] Namespace: `urn:oasis:names:tc:SAML:1.0:action:ghpp`.
[35] In questo caso il namespace dovrà essere definito e concordato preventivamente.
[36] cf. 3.1.1.2, 3.1.1.1.
[37] cf. [SAML-core]

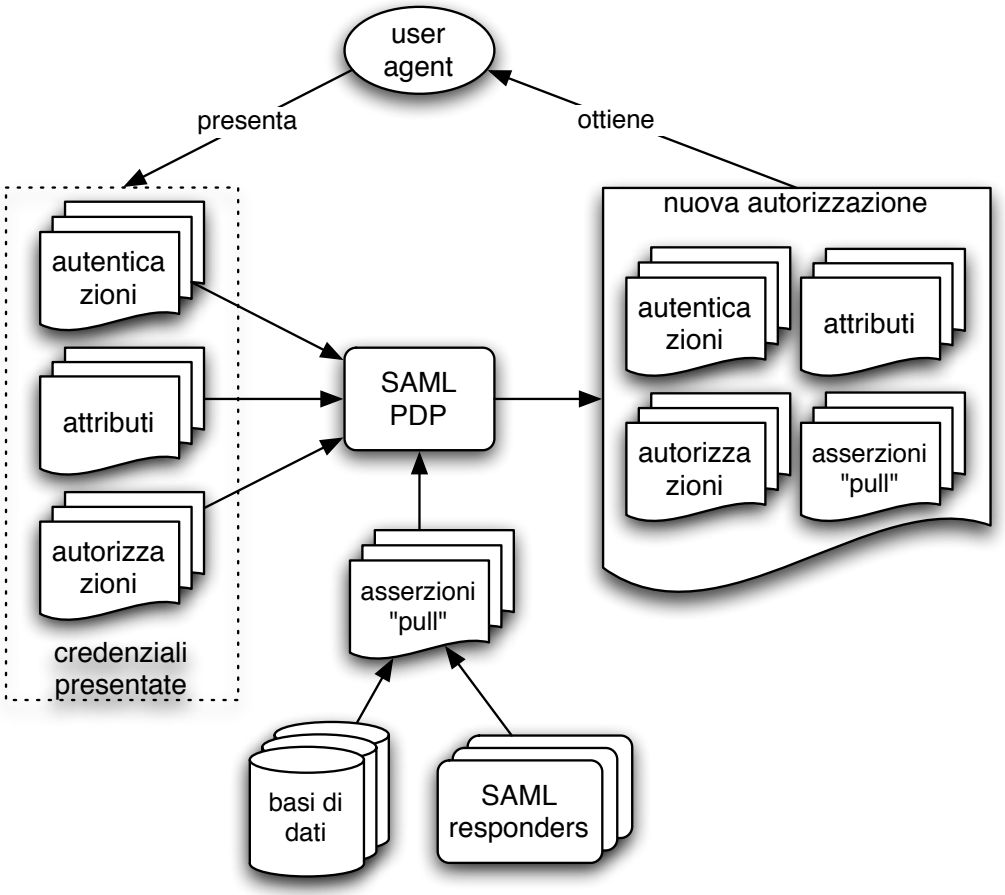

Figura 4.7: Flusso delle asserzioni nell'emissione di una autorizzazione.

non essere sotto forma di asserzione ed in tal caso dovrebbero essere generate a partire da esse nuove asserzioni da includere nell'autorizzazione. Il flusso informativo risultante è mostrato in figura 4.7.

Questa soluzione presenta tuttavia due inconvenienti:

- la generazione di asserzioni 'locali' a partire dalle informazioni ottenute dalle basi di dati può essere complessa da implementare;
- alcune delle ACI potrebbero non essere rilasciabili ai fruitori dell'autorizzazione, ovvero il client, l'utente o i servizi a cui verrà presentata.

Pertanto la scelta di quali asserzioni includere nell'elemento `Evidence`

di un'asserzione di autorizzazione è lasciata a discrezione dell'implementatore del PDP mentre resta l'obbligo di validare e considerare nella decisione tutte le credenziali presenti nell'elemento omonimo della richiesta di autorizzazione.

4.2.3.4 Integrazione con il motore di decisione

SAML specifica solo come la decisione di un PDP debba essere richiesta ed ottenuta, non come venga effettuata. il SAML PDP necessita quindi di un motore di decisione che implementi i requisiti della categoria 2.3.2.

La tecnologia scelta per tale componente è PERMIS[38]. PERMIS supporta il paradigma di autorizzazione RBAC (requisito 2.3.2.1) e basa le proprie decisioni su una policy espressa in linguaggio formale (requisito 2.3.2.3).

Il soddisfacimento del requisito 2.3.2.2 (integrazione tra il controllo di accesso e la profilazione) è ottenuto condividendo, laddove possibile, le stesse basi di dati della AA od interrogandola direttamente tramite protocollo SAML. Si veda in proposito la figura 4.6.

4.2.3.5 Integrazione con i backends

PERMIS basa le sue decisioni, oltre che sulle policies e sulle credenziali fornite dall'utente, su una o più fonti di AC X.509 accessibili via LDAP, HTTP o filesystem. Questa caratteristica consente a PERMIS di allacciarsi direttamente ad eventuali infrastrutture di attribuzione di ruolo preesistenti.

Qualora sia invece previsto (anche) l'utilizzo di asserzioni SAML per effettuare le decisioni, come nel caso delle credenziali presentate dall'utente (push) o di AA SAML, è necessario implementare degli opportuni componenti di integrazione, denominati rispettivamente "decision request adapter" e "credentials adapter" nel diagramma in figura 4.6. Tali componenti provvederanno a convertire le informazioni contenute nelle asserzioni SAML in un formato compatibile con la API di PERMIS dopo averne ovviamente validato la sintassi e la firma digitale.

[38] cf. 3.3.1.

4.2.3.6 Sequenza delle operazioni

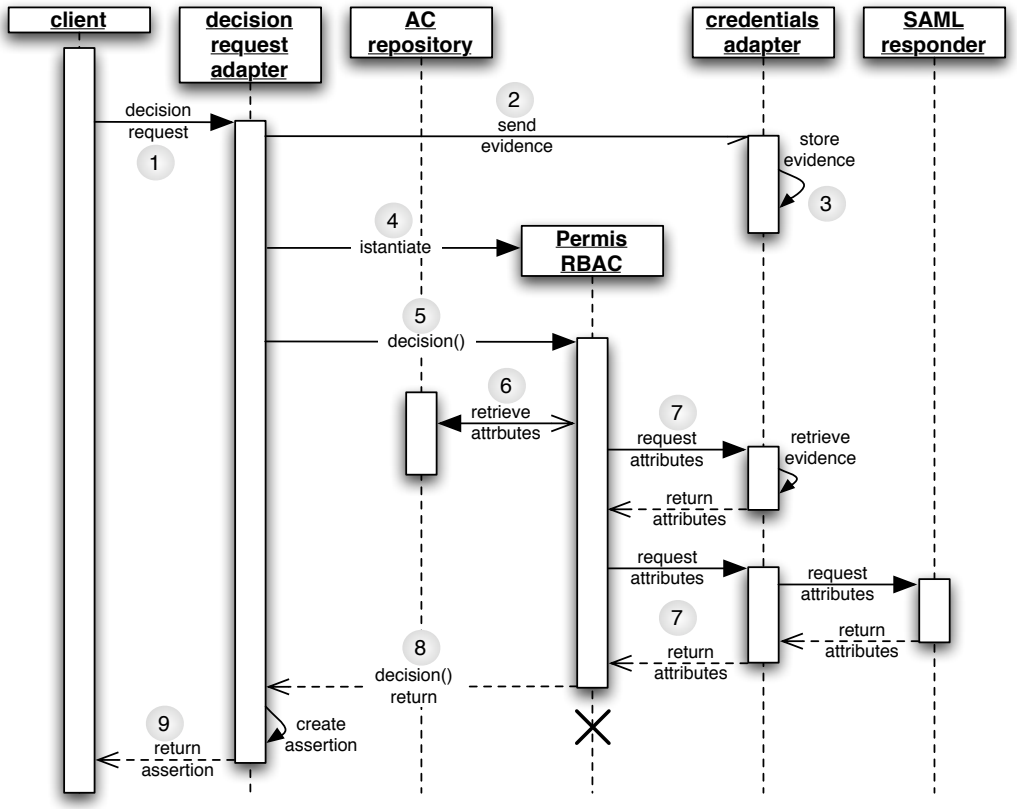

Figura 4.8: Diagramma di sequenza delle interazioni relative all'emissione di un'autorizzazione.

Il dettaglio delle interazioni tra i componenti del SAML PDP nell'emissione di un'asserzione di autorizzazione è mostrato in figura 4.8[39]:

1. Il client interroga il PDP utilizzando il SAML SOAP Binding. La richiesta di autorizzazione contiene il **soggetto** che richiede l'accesso, la **risorsa** a cui vuole accedere e l'**azione** che su di essa intende compiere. È inoltre possibile, opzionalmente, fornire un insieme di **credenziali** utili per la decisione sotto forma di asserzioni SAML.

[39]La numerazione utilizzata non indica l'effettivo ordine di esecuzione, alcune interazioni possono infatti avvenire concorrentemente.

2. Il decision request adapter valida la richiesta di autorizzazione ed invia al credentials adapter le credenziali ricevute in push.

3. Il decision request adapter provvede ad istanziare un oggetto di classe `PermisRBAC` (che effettuerà la decisione vera e propria) fornendo al costruttore la policy reputata pertinente alla decisione richiesta[40] ed il riferimento al credentials adapter.

4. Il credentials adapter, che implementa l'interfaccia `AttributeRepository` della API di PERMIS, valida le credenziali ottenute in push ed immagazzina le informazioni in esse contenute sotto forma di coppie <nome, valore>[41]. A tal fine il credentials adapter può avvalersi di un'opportuna base di dati detta "assertions cache" od immagazzinare tali informazioni direttamente in memoria.

5. Il decision request adapter invoca il metodo `decision()` della classe `PermisRBAC`, con parametri: <soggetto, risorsa, azione>.

6. Nell'effettuare la decisione, `PermisRBAC` può interrogare l'archivio degli AC X.509.

7. `PermisRBAC` può inoltre richiedere attributi derivati dalle asserzioni presentate o da interrogazioni di SAML responders. In questo caso le richieste saranno mediate dal componente credentials adapter.

8. `PermisRBAC` restituisce infine al decision request adapter la decisione presa.

9. Il decision request adapter provvede a generare un'asserzione di autorizzazione SAML contenente l'esito della decisione ed opzionalmente le asserzioni utilizzate nel processo.

[40] Questa operazione è necessaria per ovviare all'impossibilità di avere policies distribuite in PERMIS.

[41] La rappresentazione interna delle assezioni può non corrispondere fedelmente al contenuto delle stesse, in particolar modo per le asserzioni di autenticazione o di autorizzazione che, diversamente da quelle di attributi, non sono già in forma di coppie <nome, valore>.

4.2.4 Integrazione con il sistema federato di autenticazione

Sia il SAML PDP che la SAML AA possono essere utilizzati contestualmente al protocollo di SSO descritto nella sezione 3.2.1.1. Ciò permette di supportare scenari in cui il recupero di qualifiche ed autorizzazioni sia automaticamente innescato (ad opera del SP) a partire da una richiesta di servizio da parte del cittadino.

L'integrazione della AA nelle interazioni di SSO è gia prevista ed implementata in Shibboleth, come dimostrano i passi 6 e 7 del protocollo della sezione 3.2.1.1. La presenza di un PDP non è invece prevista al momento dagli sviluppatori di Shibboleth e pertanto l'integrazione di tale componente deve essere appositamente progettata ed implementata.

La soluzione più naturale a questo problema è quella di generalizzare il protocollo di SSO sostituendo i passi 6 e 7 con l'interrogazione di una molteplicità di autorità SAML (AA e PDP) effettuata dall'IdP per conto del SP o direttamente da quest'ultimo.

Questa soluzione apre tuttavia una problematica nuova, ovvero come sapere quali sono le autorità da interpellare data la particolare istanza applicativa. Si tratta chiaramente di un problema di cooperazione applicativa specifico dei singoli scenari di utilizzo la cui soluzione generale va al di là degli scopi di questa tesi.

Tuttavia, a scopo di esemplificazione e validazione dell'archiettura, nel prototipo realizzato (e descritto nel capitolo 5) viene mostrata una possibile integrazione dei tre componenti (SSO, AA e PDP) nel caso più semplice in cui siano tutti centralizzati all'interno dell'IdP.

4.3 Applicabilità del framework alle iniziative analizzate

Viene qui brevemente descritto come l'architettura generale delineata nelle sezioni 4.1 e 4.2 possa essere adattata alle particolarità dei singoli progetti, bandi o standards analizzati nel capitolo 2.

4.3.1 Applicabilità del framework al progetto ICAR

Il sistema di autenticazione federato descritto nella sezione 4.1, unito all'infrastruttura di attribuzione di ruoli specificata nella sezione 4.2.2 costituiscono una soluzione completa per l'intervento INF-3[42] del progetto ICAR. Inoltre, grazie alla mediazione dei meccanismi di sicurezza dei singoli domini operata da SAML, non è necessario alterare lo spazio degli identificatori utente e degli attributi perché risultino interoperabili in un contesto federativo.

4.3.2 Possibili integrazioni con SPC

Come già notato in fase di analisi (sezione 2.1.1.3) ed approfondito nel capitolo 3 (sezione 3.1.1.5), vi è la possibilità di inserire asserzioni SAML all'interno di buste di e-Government, con la funzione di tokens di sicurezza. Tale caratteristica introduce modalità avanzate di autorizzazione[43] a livello SPC (*back office*), così come è già possibile fare a livello di portale (*front office*) grazie al framework delineato in questo capitolo.

La figura 4.9 mostra un possibile uso di asserzioni SAML nell'autorizzazione di comunicazioni SPC. Un dominio ricevente può decidere se accettare una busta non più solo in base all'identità del mittente (ed all'appartenenza dello stesso ad un particolare 'gruppo'[44] di cooperazione), ma anche in base

[42] cf. 2.2.1.2.

[43] Peraltro accennate anche in [CnipaTech].

[44] Il termine esatto utilizzato nei documenti normativi di SPC [CnipaBusta][CnipaArch][CnipaTech] è *dominio di cooperazione* e denota un gruppo di domini partecipanti in una particolare istanza di cooperazione applicativa.

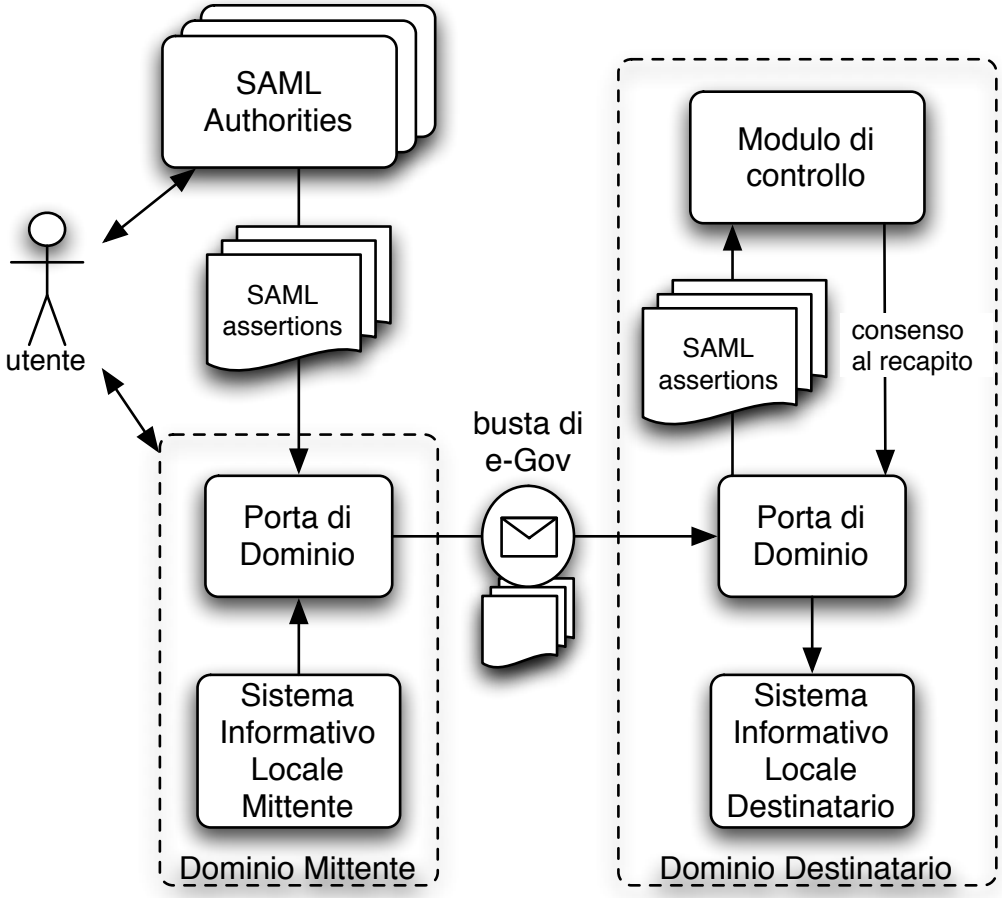

Figura 4.9: Utilizzo degli headers WSS con token SAML nelle decisioni di accesso su buste di e-Government.

ad asserzioni SAML[45] in suo possesso. Tali asserzioni possono essere ottenute da opportune autorità SAML che possono appartenere, a seconda dei casi, al dominio mittente, a quello destinatario o a domini terzi rispetto a quelli comunicanti.

È inoltre possibile prevedere che l'emissione di tali asserzioni venga innescata da opportune interazioni dell'utente con le autorità SAML ed eventualmente con il dominio mittente in maniera analoga a quanto avviene per il SSO su web. In questo caso, le asserzioni conterranno informazioni di

[45] Ad esempio attribuenti ruoli od autorizzazioni al mittente.

autenticazione, qualifica ed autorizzazione dell'utente stesso e potranno ad esempio essere utilizzate dal modulo di controllo per verificare se il dominio mittente ha un'esplicita delega[46] a procedere a suo nome.

Da un punto di vista più concettuale, utilizzando asserzioni SAML nello header WSS delle buste di e-Government, il dominio ricevente può fidarsi di ciò che riceve in funzione della sua fiducia nei confronti delle autorità SAML e non necessita quindi di esplicite relazioni di trust con i mittenti. Questa possibilità, unita all'utilizzo dei paradigmi di cooperazione avanzati descritti nella sezione 2.1.1.5, permette di realizzare scenari di cooperazione in sicurezza in cui i domini possono dinamicamente assumere il ruolo di funtore o di autorità a seconda delle specifiche istanze applicative.

4.3.3 Compatibilità con il bando di Regione Campania

La gestione della sicurezza per la funzione di "presentazione di servizi" del NAG[47] è già prevista dall'architettura delineata in questo capitolo. In particolare la presentazione di contenuti differenziati in funzione delle credenziali[48] e l'aggregazione dei contenuti stessi[49] possono essere gestite nelle modalità esemplificate nel capitolo 5.

Per quanto riguarda invece la funzione di "broker di servizi", si ha a che fare con uno schema architetturale in tutto e per tutto analogo a quello di SPC. Si possono pertanto applicare le stesse considerazioni effettuate nella sezione 4.3.2 e riguardanti l'utilizzo di asserzioni SAML come tokens di buste di e-Government.

[46] Si potrebbe estendere ulteriormente questa funzionalità utilizzando un approccio simile al Delegation Service di GT, descritto nella sezione 3.3.2.3.

[47] cf. 2.2.3.1.

[48] cf. 2.2.3.2.

[49] cf. 2.2.3.1.

4.3.4 Compatibilità con il bando di Regione Toscana

Le particolarità del progetto e.Toscana[50], rispetto al modello generale descritto in questo capitolo, sono principalmente tre:

- utilizzo di un'infrastruttura di scambio messaggi inter-dominio (CART) non completamente compatibile con SPC;
- necessità di integrazione con il motore di profilazione SRTY;
- supporto per l'autocertificazione.

La prima caratteristica implica l'impossibilità di utilizzare tokens SAML nei messaggi inter-dominio a meno che non si estenda CART. Per quanto riguarda invece l'interazione di SRTY con il resto dell'architettura, lo si potrebbe collegare con le stesse fonti di ACI disponibili al/ai PDP (approccio *common backend*). Si potrebbe in alternativa fornire un componente di intermediazione in grado di derivare, a partire da un insieme di asserzioni SAML, un profilo utente compatibile con il formato di SRTY.

Le funzionalità di portale delle PdD, come specificate nelle sezioni 2.2.2.2 e 2.2.2.4, possono invece essere implementate secondo il modello generale, senza particolari indicazioni. Il supporto all'autocertificazione, richiesto nella sezione 2.2.2.3, può essere implementato come una richiesta di provisioning che dovrà essere successivamente autorizzata ed eseguita da un operatore.

[50] cf. 2.2.2

Capitolo 5

Il prototipo realizzato

Viene qui documentato il lavoro di sviluppo svolto con l'intento di verificare nella pratica alcune delle argomentazioni di questa tesi. Esso segue le raccomandazioni enunciate nel capitolo 4 ed utilizza le tecnologie selezionate nel capitolo 3 al fine di valutare la fattibilità dei requisiti chiave del capitolo 2. Il software prodotto non ha pertanto la pretesa di costituire un prodotto completo e definitivo, ma ha piuttosto la natura di prototipo.

Nel seguito verranno descritti gli obiettivi (sezione 5.1) principali che ci si è posti nella realizzazione di tale prototipo e le interconnessioni (sezione 5.2) tra i componenti (ri)utilizzati od implementati ex-novo. Sarà infine delineato un possibile percorso evolutivo per il software sviluppato (sezione 5.3).

5.1 Finalità

5.1.1 Sviluppo di un'infrastruttura di autorizzazione per Shibboleth

Come già evidenziato nella sezione 4.2.3, Shibboleth non prevede nella propria architettura la presenza di un PDP, nonostante il componente sia specificato nello standard SAML. Pertanto l'obiettivo primario del prototipo realizzato è quello di integrare questa funzionalità nell'attuale implementazione di Shibboleth.

La realizzazione di questo componente segue le specifiche espresse nella sezione 4.2.3 e si integra con il resto dell'architettura nella maniera descritta nella sezione 4.2.4. Il PDP è stato inoltre pensato per supportare tutti i modelli di federazione descritti nella sezione 4.2.1. L'implementazione realizzata comprende infine un meccanismo per risolvere gli handle opachi in nomi utente utilizzabili per la decisione di accesso come suggerito nella sezione 4.2.3.1.

5.1.2 Integrazione di applicazioni web in un SP Shibboleth

Il secondo obiettivo di questo prototipo è quello di mostrare come un'applicazione web possa essere integrata con il SP di Shibboleth in modo da partecipare attivamente ai protocolli di SSO, attribuzione di ruoli ed autorizzazione, interagendo in quest'ultimo caso con il PDP implementato.

Il comportamento di default di Shibboleth nei confronti di una risorsa protetta, statica o dinamica che sia, è infatti quello di opacizzare l'intero processo di SSO e qualifica, rendendo così possibile la protezione di risorse non progettate specificamente per il deployment all'interno di un SP.

Per l'applicazione web qui documentata, si vuole invece pieno accesso alle asserzioni emesse dall'IdP (e da eventuali SAML responders esterni) in modo da permetterne l'uso anche a livello applicativo. Questa possibilità abilita

l'utilizzo delle asserzioni come profilo utente, oltre che come credenziali di accesso, allo scopo di personalizzare le funzioni ed i contenuti presentati.

Per ottenere tali funzionalità, sono state anche apportate modifiche al codice sorgente del SP di Shibboleth, come verrà illustrato nella sezione 5.2.2.

5.2 Configurazione ed implementazione dei componenti

L'insieme dei componenti utilizzati od implementati ex novo per realizzare il prototipo in esame è illustrato in figura 5.1.

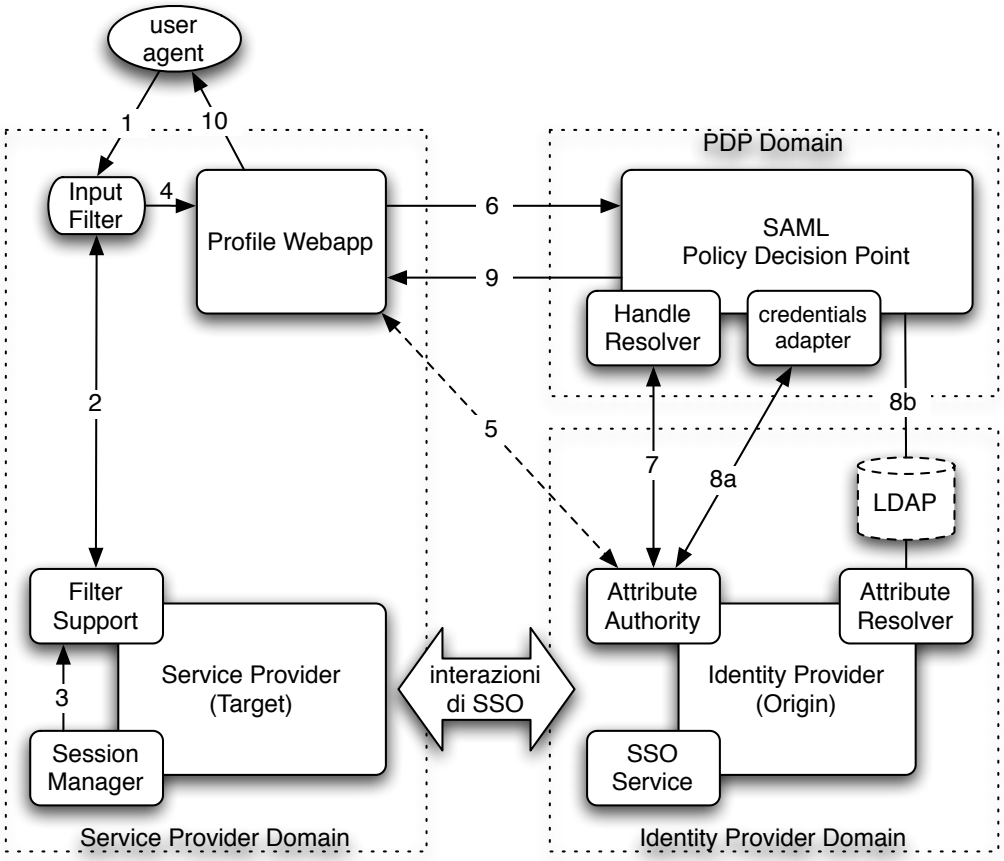

Figura 5.1: Architettura del prototipo implementato.

La federazione utilizzata come ambiente di sviluppo è minimale, ovvero comprende un solo IdP ed un solo SP. L'applicazione web realizzata, "Profile Webapp", è installata nello stesso dominio del SP e ne sfrutta le funzionalità di protezione.

In aggiunta ai due macro-componenti standard di Shibboleth vi è il SAML PDP che può o meno risiedere nello stesso dominio dell'IdP. Nel caso di

quest'ultima eventualità si può avere una federazione di domini con privilegi differenziati:

- il dominio dell'IdP, che ha accesso all'effettiva identità digitale dei cittadini;
- il dominio del SP, a cui tali identità possono essere anche totalmente mascherate; il SP effettuerà in questo caso le proprie decisioni di accesso in base alle asserzioni ottenute da opportune autorità SAML, anche esterne all'IdP;
- il dominio del PDP che, se ne necessita per le proprie decisioni, può accedere ad una porzione maggiore (o comunque in generale distinta) dell'identità digitale dei cittadini accessibile al SP.

5.2.1 Shibboleth IdP

L'IdP utilizzato è quello dell'implementazione ufficiale di Shibboleth descritta nella sezione 3.2.1.4 e costituito da un insieme di applicazioni web J2EE. Per l'IdP non sono state utilizzate particolari configurazioni ad eccezion fatta per la scelta di un attribute resolver[1] 'virtuale' per la AA.

Tale attribute resolver, implementato dalla classe `SampleConnector`[2], permette alla AA di emettere un attributo speciale contenente l'identificatore utente utilizzato localmente dall'IdP. Questa funzionalità viene utilizzata dal PDP per implementare la risoluzione di handles opachi in nomi utente. La ARP è stata di conseguenza configurata per garantire l'accesso a tale attributo al PDP ma non al SP.

L'utilizzo della classe `SampleConnector` è stato inoltre utile in fase di test, in quanto permette di non utilizzare delle basi di dati vere e proprie per ottenere gli attributi, ma di generarli programmaticamente.

[1] cf. 4.2.2.3.

[2] Appartenente al package `edu.internet2.middleware.shibboleth.aa.attrresolv.provider`.

Per il deployment dell'IdP è stato usato il servlet/Java Server Pages (JSP) engine Apache Tomcat [Tomcat-web]. L'IdP è stato inoltre configurato per utilizzare le funzionalità di autenticazione di Tomcat, più che sufficienti per l'ambiente di sviluppo.

5.2.2 Shibboleth Java SP

Per il SP è stata scelta l'implementazione in Java, recentemente contribuita al progetto Shibboleth dall'università di Yale, anziché la versione originale in C/C++ per le motivazioni già argomentate nella sezione 3.2.1.4. Pertanto si è scelto di utilizzare Tomcat come contenitore anche per il SP.

Come nel caso dell'IdP, la configurazione non è dissimile a quella di default. Tuttavia è stato modificato il servlet filter che intercetta le richieste degli utenti (passo 1) ed innesca il meccanismo di SSO se questi non hanno una sessione aperta con il SP (passi 2,3).

Tale filtro è suddiviso in due componenti cooperanti:

Input Filter È un'implementazione dell'interfaccia Java `javax.servlet.Filter` della Servlet API[3]. Operante nel contesto applicativo del servlet container Tomcat, intercetta le richieste HTTP prive di sessione ed innesca il protocollo di SSO, creando successivamente una nuova sessione applicativa. Tomcat supporta l'utilizzo di filtri differenziati per ogni singola applicazione web gestita, abilitando in questo modo la possibilità di avere SP multipli (o distintamente configurati) nello stesso dominio.

Filter Support Opera nel contesto applicativo del SP e può accedere ai dati di sessione[4] presenti nel Session Manager che può successivamente comunicare all'Input Filter mediante l'utilizzo di un'interfaccia Java[5] fornita da quest'ultimo.

[3] cf. [Servlet-API].

[4] Contenenti tra le altre cose tutte le asserzioni SAML ottenute dall'IdP.

[5] Si tratta dell'interfaccia `FilterSupport` del package `edu.internet2.middleware.shibboleth.resource`.

Nella versione originale il Filter Support inseriva nella nuova sessione creata per l'applicazione protetta le sole coppie <nome (stringa), valore (stringa)> degli attributi ottenuti dalla AA. Esso è stato pertanto modificato perchè inserisse nella sessione le asserzioni ottenute dall'IdP nella loro interezza. Questa funzionalità si è resa necessaria soprattutto per permettere all'applicazione di ottenere lo handle opaco in modo da poterlo utilizzare successivamente nell'interrogazione di opportuni SAML responders (come il PDP).

La possibilità di accedere agli handle opachi anziché ai nomi utente 'in chiaro' permette un modello di applicazione web totalmente ignara dell'identità digitale degli utenti in linea con quanto detto nella sezione 4.1.2.2. Ad una tale applicazione interesserà esclusivamente il **profilo** dell'utente servito, visto come insieme di autenticazioni (pseudonimizzate), qualifiche ed autorizzazioni emesse da opportune autorità SAML. Il fatto di poter accedere direttamente alle asserzioni ottenute permette inoltre all'applicazione di applicare proprie politiche di trust nei confronti delle suddette autorità, distinte dalle politiche (di federazione) del SP.

5.2.3 SAML PDP

Come già detto in precedenza, il PDP è stato implementato seguendo le direttive espresse nella sezione 4.2.3, con l'aggiunta del componente "Handle Resolver" atto ad ottenere l'identificatore utente utilizzato internamente dall'IdP. Tale identificatore potrà essere utilizzato per accedere direttamente al repository di attributi della AA (passo 8b) o per interrogare ulteriori autorità SAML (nelle stesse modalità del passo 8a).

La richiesta di autorizzazione (passo 6), può essere infatti a nome di un utente pseudonomizzato per mezzo di uno handle. In questo caso, lo Handle Resolver si occupa di interrogare la AA dell'IdP per ottenere l'attributo speciale (`eduPersonPrincipalName`) contenente l'identificatore utilizzato internamente (passo 7).

5.2.4 Profile Webapp

Come si può vedere in figura 5.1, la Profile Webapp è l'attore principale dell'architettura. Una volta ricevuta la richiesta dell'utente – che l'Input Filter avrà provveduto a contestualizzare (passo 4) con le asserzioni ottenute dall'IdP tramite i protocolli standard di SSO[6] – la Profile Webapp si occupa di interrogare opportuni SAML responders per arricchire il profilo in proprio possesso.

La Profile Webapp può richiedere attributi alla AA dell'IdP (passo 5). Questa possibilità potrebbe essere utile per realizzare diversi livelli di 'conoscenza dell'utente' tra il SP e l'applicazione. Il SP potrebbe fare del tutto a meno degli attributi ed occuparsi solo dell'autenticazione. L'applicazione potrebbe così anche risiedere in un altro dominio, a patto che si reimplementino i punti 2 e 3 che attualmente si basano su chiamate di API (all'interno della stessa JVM) e non su uno scambio di messaggi.

La richiesta di asserzioni può avvenire anche verso SAML responders esterni al dominio dell'IdP, tipicamente AA o PDP; quest'ultima eventualità e illustrata nei passi 6 e 9. È interessante puntualizzare che nel passo 6 l'applicazione allega nella richiesta di autorizzazione[7] tutte le asserzioni in proprio possesso, ivi compresa quella di autenticazione contenente l'identificatore utente (eventualmente pseudonimizzato).

5.2.4.1 Architettura interna

Come mostrato in figura 5.2, la composizione interna della Profile Webapp si ispira al design pattern MVC [POSA] e nella fattispecie al "Model 2" descritto in [Model2].

La richiesta HTTP dello user agent viene intercettata dall'Input Filter che provvede a creare una sessione per l'applicazione nel caso in cui non ve ne fosse già una attiva. Avvenute le interazioni di SSO, laddove necessarie[8],

[6] Descritti più dettagliatamente nella sezione 3.2.1.1.

[7] Utilizzando l'elemento `Evidence` come indicato nelle sezioni 3.1.1.2 e 4.2.3.3.

[8] Le sessioni dell'utente con l'applicazione sono disaccoppiate da quelle con il SP. Può

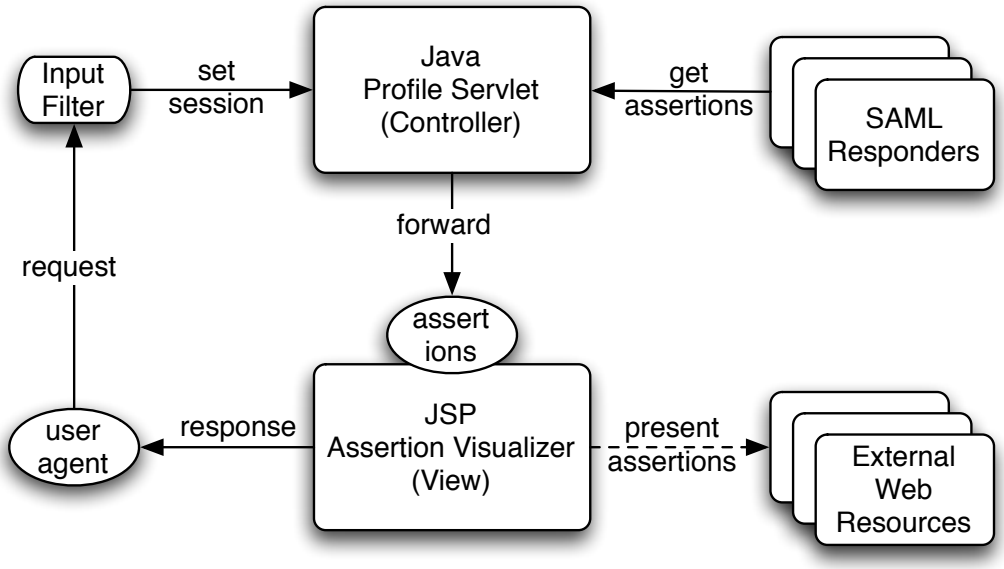

Figura 5.2: Struttura interna della Profile Webapp.

la richiesta verso la Profile Webapp conterrà le asserzioni emesse dall'IdP e sarà passata al modulo controller ("Profile Servlet").

Il Profile Servlet raccoglie le informazioni necessarie per creare il profilo utente interrogando un insieme di SAML responders. Nella particolare federazione utilizzata come ambiente di sviluppo l'unica interrogazione che fornisce nuove informazioni (rispetto alle asserzioni già ricevute nella sessione) è quella verso il PDP. Il servlet provvede ad arricchire la sessione con le asserzioni ottenute e passa il controllo all' "Assertion Visualizer".

Quest'ultimo componente provvede a presentare una vista all'utente in funzione delle asserzioni trovate nei dati di sessione. L'attuale implementazione dell'Assertion Visualizer ha scopo unicamente dimostrativo e si limita a visualizzare gli elementi informativi delle asserzioni ottenute dal Profile Servlet. È tuttavia ipotizzabile una situazione in cui le viste generate presentino dei contenuti veri e propri, differenziati in funzione delle asserzioni ottenute,

accadere che la richiesta verso l'applicazione avvenga in un istante in cui la sessione dell'utente presso il SP (creata precedentemente) sia ancora attiva, nel qual caso il filtro si limiterà a copiare le asserzioni nella nuova sessione applicativa.

implementando di fatto la profilazione[9]. Questi contenuti potrebbero inoltre essere richiesti a risorse web esterne al dominio della Profile Webapp utilizzando eventualmente le asserzioni stesse come credenziali di accesso.

[9]Così come esplicitamente richiesta nei bandi di Regione Toscana e Regione Campania. Una tale archiettura soddisfa inoltre il requisito 2.3.2.2.

5.3 Possibili Sviluppi

L'attuale logica applicativa della Profile Webapp potrebbe essere trasferita all'interno dell'Input Filter. In questo modo le applicazioni web protette potrebbero esternalizzare anche le funzionalità di profilazione, oltre che quelle di autenticazione ed autorizzazione.

Un tale filtro costituirebbe di fatto un *agent* che, per conto dell'applicazione protetta e dell'utente, si occuperebbe di ottenere[10] le asserzioni necessarie per generare il profilo utente. Esso inoltre continuerebbe a controllare gli accessi (come già fa) in funzione delle stesse asserzioni.

Essendo in Tomcat la configurazione dei filtri indipendente per ciascuna applicazione, ognuna di queste potrebbe specificare nel proprio descrittore gli attributi e le autorizzazioni ritenuti rilevanti per costruire il profilo utente. Sempre nella stessa configurazione, le applicazioni potrebbero specificare le policies per l'accesso alle proprie 'aree protette', ovvero un insieme di espressioni booleane sugli elementi informativi presenti nelle asserzioni ritenute indispensabili per l'accesso.

[10] Direttamente, interrogando SAML responders, od indirettamente, ricuperando le asserzioni ottenute dal SP nelle interazioni con l'IdP.

Conclusioni

Risultati raggiunti

In questa tesi sono stati analizzati alcuni dei più importanti requisiti (capitolo 2) della PA italiana in termini di Identity Management ai quali si è cercato di dare soluzione utilizzando le tecnologie e gli standards attualmente presenti sul mercato (capitolo 3). Tale soluzione è costituita da un disegno architetturale generale (capitolo 4), che permette di integrare le varie tecnologie scelte, e dall'implementazione (capitolo 5) di alcune delle funzionalità più interessanti dell'architettura stessa.

Oltre al soddisfacimento dei requisiti, l'adozione di tali tecnologie, standards e patterns architetturali concorre a creare nuove potenzialità applicative, ritenute utili ed interessanti ma finora poco o per nulla esplorate dalla PA.

Sviluppi futuri

È senz'altro ravvisabile un percorso evolutivo nella direzione intrapresa da questa tesi. Da un lato è possibile perfezionare il disegno architetturale del capitolo 4 e completarne l'implementazione, dall'altro è interessante esplorare nuove possibilità, sia in termini di scelte tecnologiche, sia per quanto riguarda ulteriori modelli architetturali.

Motori di decisione alternativi

PERMIS è il motore di decisione che è stato scelto[11] ed inserito nel framework[12] in quanto costituisce una soluzione pronta all'uso che soddisfa gli attuali requisiti della PA in termini di autorizzazione. Tuttavia, nella fase di ricerca a monte di questa tesi, sono stati analizzati (e scartati) molti strumenti analoghi.

Tra di essi è senz'altro da citare eXtensible Access Control Markup Language (XACML), il linguaggio di policy (e relativo modello di PDP) standardizzato dall'omonimo Technical Committee di OASIS in [XACML][13]. XACML è dotato delle stesse funzionalità di PERMIS (controllo di accesso RBAC e basato su policy, possibilità di decisioni distribuite, ecc.), ma ad esse affianca:

- un protocollo per l'invocazione e l'emissione delle decisioni (in questo si sovrappone a SAML);
- la possibilità di valutare più policies nell'esecuzione della stessa decisione;
- appositi punti di contatto (da ambo le parti) con lo standard SAML;
- la possibilità di interrogare opportune autorità XACML per ottenere la policy applicabile/applicata ad una particolare decisione;
- estensibilità e vastità del linguaggio di policy, in particolare per quanto riguarda le funzioni booleane e gli algoritmi di combinazione delle regole e delle policies (es. Deny-overrides, Permit-ovverides, Only-one-applicable, ecc.).

Il prezzo da pagare per avere queste caratteristiche è la relativa instabilità del linguaggio e l'assenza di implementazioni Open Source aggiornate alla versione 2.0 dello standard. Ciò detto, l'applicazione di XACML a problematiche di e-Government merita senz'altro di essere approfondita[14].

[11] nella sezione 3.3.1.
[12] nella sezione 4.2.3.
[13] Si veda anche [XACML-web].
[14] Dato anche che la versione 1.0 dello standard è già parte della suite Electronic Business

Futuro delle tecnologie basate su SAML

Come accennato nella sezione 3.2.2.1, la versione 2.0 dello standard SAML, ratificata il 15 Marzo 2005, include molte delle innovazioni apportate da Liberty Alliance a SAML 1.1[15]. SAML 2.0 include anche caratteristiche proprie di Shibboleth[16], e quest'ultimo a sua volta implementerà a breve SAML 2.0.

È pertanto ipotizzabile, col passare del tempo, una sempre maggiore integrazione di queste tre iniziative, che potrebbe a detta di molti sfociare in una convergenza.

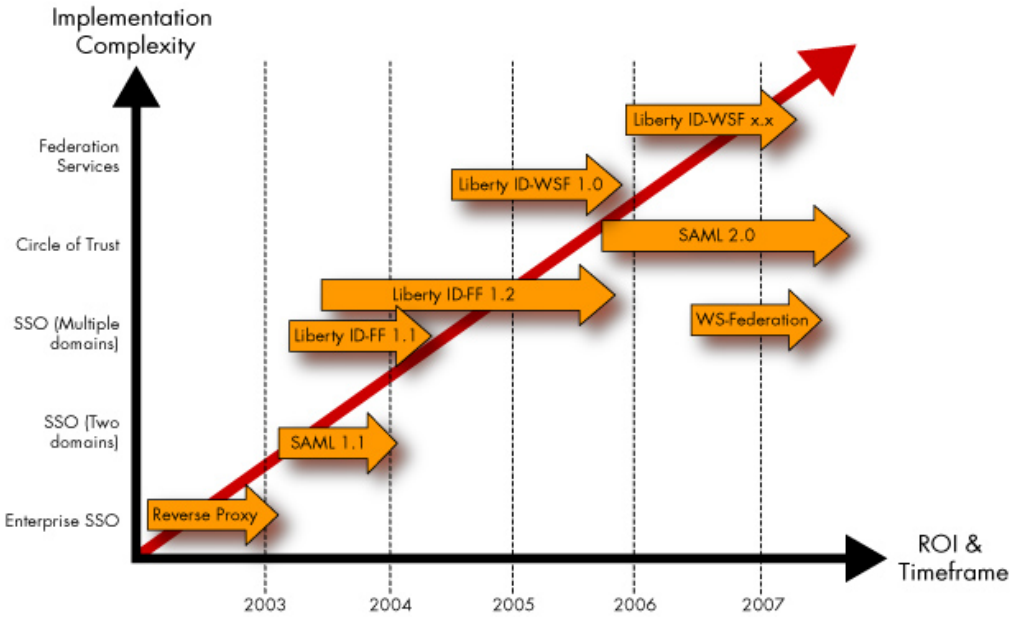

Figura 5.3: Intervalli temporali di implementazione, osservati e stimati, dei principali standards di IM federato. Figura tratta da [Rouault].

using eXtensible Markup Language (ebXML) a sua volta scelta dal CNIPA come tecnologia di registry in [CnipaTech].

[15] Single logout, `NameIdentifier` mapping, identity federation, metadati, LECP, ecc.

[16] Handles opachi ed effimeri.

Problematiche avanzate di e-Government

> “ Ogni qualvolta si sviluppa una tecnologia innovativa, questa viene utilizzata riproponendo gli schemi ed i modelli precedenti, poi pian piano quest'ultimi vengono modificati per sfruttare meglio le possibilità offerte dalle nuove tecniche. È ciò che è successo alle prime automobili che avevano ancora forme e caratteristiche delle carrozze; è ciò che oggi succede agli strumenti elettronici, ancora utilizzati con modalità e schemi tipici della carta.[17] ”

Sebbene ad un livello infrastrutturale la PA italiana si stia dotando delle tecnologie allo stato dell'arte, vi è ancora molto lavoro da fare per sfruttarne appieno le potenzialità a livello applicativo[18], man mano che si passa dai processi cartacei a quelli elettronici.

Per questo motivo questa tesi non si è soffermata molto sulle potenzialità più evolute delle tecnologie di sicurezza ed IM utilizzate. In particolare si suggerisce di indagare più approfonditamente[19], in futuri lavori, sulle interrelazioni possibili tra le infrastrutture progettate e gli headers WSS della busta di e-Government.

[17] Fonte: *Firma elettronica e carrozze* di Gianfranco Pontevolpe, Taccuino tecnico del CNIPA, [CNIPA-web].

[18] Si vedano ad esempio le critiche mosse in [Bassanini2] dall'ex Ministro per la Funzione Pubblica Franco Bassanini, iniziatore del “Piano d'azione per l'e-Government”, alle scelte intraprese in questo senso dai suoi successori.

[19] di quanto fatto nella sezione 4.3.2

Bibliografia

[Apache-web] Apache HTTP Server Project, http://httpd.apache.org/.

[Applets] Sun Microsystems Java Applets, http://java.sun.com/applets/.

[Athens-web] Eduserv Athens, http://www.athensams.net/.

[Axis] Apache Axis SOAP Server, http://ws.apache.org/axis/.

[BPEL] OASIS Web Services Business Process Execution Language Technical Committee. http://www.oasis-open.org/committees/documents.php?wg_abbrev=wsbpel.

[BandoCampania] Regione Campania, Assessorato all'Università e Ricerca Scientifica, Innovazione Tecnologica e Nuova Economia, Sistemi Informativi e Statistica, Musei e Biblioteche. *Appalto-concorso per la realizzazione del Sistema Regionale per la Cooperazione Applicativa in Sicurezza*. Decreto Dirigenziale n. 302 del 21 aprile 2005, allegato "E", pubblicato nel Bollettino Ufficiale della Regione Campania Numero 25 del 9 maggio 2005. http://www.sito.regione.campania.it/burc/pdf05/burc25or_05/decdir302_05/decdir302_05allE.pdf

[BandoToscana] Regione Toscana, Direzione Generale Organizzazione e Sistema Informativo, Area di Coordinamento Ingegneria dei Sistemi Informativi e della Comunicazione, Settore I.I.T.R. *Capitolato Speciale di Appalto per l'attivazione di un'infrastruttura per l'autenticazione e l'accesso sicuro ai servizi di e.Toscana*. Allegato "1", Capitolato Tecnico. Aprile 2005. http://www.rete.toscana.it/gar/arpa/cap_tecnico.pdf.

[Bassanini1] Bassanini F. *Cinque anni di riforma della Amministrazione Pubblica italiana 1996-2001*. http://www.bassanini.it/docs/5anni/index.html.

[Bassanini2] Bassanini F., ed. *L'ammodernamento del sistema amministrativo italiano, Note per un programma di governo*. Luglio 2005. http://www.bassanini.it/docs/dossier_ammodernamento.pdf.

[CAS] Pearlman, L. et al. *The Community Authorization Service: Status and Future*. CHEP03. March 2003. http://www.globus.org/alliance/publications/papers/CAS_update_CHEP_03-final.pdf.

[CISIS-web] CISIS - Centro Interregionale per il Sistema Informatico e il Sistema Statistico, homepage: http://www.cisis.it/.

[CNIPA-web] CNIPA: Centro Nazionale per Informatica nella Pubblica Amministrazione, http://www.cnipa.gov.it/.

[CnipaArch] Gruppo di lavoro "Servizi per l'interoperabilità, la cooperazione applicativa e l'accesso del SPC". *Sistema pubblico di cooperazione: Architettura*. Versione 1.0, 25 Novembre

2004. http://www.cnipa.gov.it/site/_files/SPCoop-Architettura_v1.0_20041125_.pdf.

[CnipaBusta] Gruppo di lavoro "Servizi per l'interoperabilità, la cooperazione applicativa e l'accesso del SPC". *Specifiche della Busta di e-Government*. 21 Aprile 2004. http://www.cnipa.gov.it/site/_files/4.SPC_Busta%20e-Gov%20v.1_0-21-04-2004.pdf.

[CnipaGdlOS] CNIPA, Gruppo di Lavoro "Codice sorgente aperto". *Rapporto conclusivo*. Versione 1.0. http://www.cnipa.gov.it/site/_files/Rapporto%20conclusivo_OSS.pdf.

[CnipaTech] Gruppo di lavoro "Servizi per l'interoperabilità, la cooperazione applicativa e l'accesso del SPC". *Sistema pubblico di cooperazione: Standard e Tecnologie*. Versione 1.0, 25 Novembre 2004. http://www.cnipa.gov.it/site/_files/SPCoop-Standard_v1.0_20041125_1.pdf.

[CommissioneMeo] Ministro per l'Innovazione e le Tecnologie, *Indagine conoscitiva sul software a codice sorgente aperto nella Pubblica Amministrazione*. Rapporto della Commissione, Maggio 2003. http://www.cnipa.gov.it/site/_files/indagine_commissione_os.pdf.

[Crypto] Menezes, A. et al. *Handbook of Applied Cryptography*. CRC Press, 1996. http://www.cacr.math.uwaterloo.ca/hac/.

[E-Auth-tech] US President e-Government Initiative, US General Services Administration. *Technical Approach for the Authentication Service Component*. E-Authentication Initiative. Version 1.0.0, June 2004.

http://www.cio.gov/eauthentication/documents/TechApproach.pdf.

[E-Auth-web] US E-Authentication Initiative home page, http://www.cio.gov/eauthentication/.

[Eurostat] Lumio, M. *Statistics in focus: Telecommunications in Europe*. Eurostat, August 2005. http://epp.eurostat.cec.eu.int/cache/ITY_OFFPUB/KS-NP-05-008/EN/KS-NP-05-008-EN.PDF.

[FIDIS-IMS] Bauer M. et al., eds. *D3.1: Structured Overview on Prototypes and Concepts of Identity Management Systems*. FIDIS consortium - EC Contract No. 507512. Version 1.1, September 2005. http://www.fidis.net/fileadmin/fidis/deliverables/fidis-wp3-del3.1.overview_on_IMS.final.pdf.

[FIDIS-Models] Nabeth, T. ed. *D2.3: Models*. FIDIS consortium - EC Contract No. 507512. Version 1.0, May 2005. http://www.fidis.net/fileadmin/fidis/deliverables/fidis-wp2-del2.3.models.pdf.

[FIDIS-web] [FIDIS]: Home, http://www.fidis.net/.

[Finanz-e-Gov] Consiglio dei Ministri, *Utilizzazione di quota dei proventi derivanti dalle licenze UMTS per il piano e-government*. D.P.C.M. 14 febbraio 2002. Pubblicato nella Gazz. Uff. 21 marzo 2002, n. 68. http://www.cnipa.gov.it/site/_contentfiles/01378200/1378212_DPCM%2014%20febbraio%202002.pdf.

[Flash] Macromedia Flash, http://www.macromedia.com/software/flash/.

[GSI] Welch, V. ed. *Globus Toolkit Version 4 Grid Security Infrastructure: A Standards Perspective*. Version 3. July 2005. http://www.globus.org/toolkit/docs/4.0/security/GT4-GSI-Overview.pdf.

[GUIDE-web] GUIDE Project, http://www.guide-project.org/.

[Globus-web] The Globus Alliance, http://www.globus.org/.

[Grid] Foster, I. Kesselman, C. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1999. ISBN 1-558-60475-8

[GridProxyCert] Welch, V. *X.509 proxy certificates for dynamic delegation*. 3rd Annual PKI R&D Workshop, April 2004. http://www.globus.org/alliance/publications/papers/pki04-welch-proxy-cert-final.pdf.

[IIS-web] Microsoft Internet Information Services, http://www.microsoft.com/iis.

[Internet2-web] Internet2 - Home, http://www.internet2.edu/.

[JDBC] Sun Microsystems' JDBC Technology, http://java.sun.com/products/jdbc/.

[JavaWebStart] Sun Microsystems, Java Web Start Technology, http://java.sun.com/products/javawebstart/.

[LDAP] Howes, T.A. *The Lightweight Directory Access Protocol: X.500 Lite*, CITI Technical Report 95-8, July 1995. http://www.openldap.org/pub/umich/ldap.pdf.

[LibertyBindProf] Cantor, S. Kemp, J. Champagne, D. eds. *Liberty ID-FF Bindings and Profiles Specification*, Version 1.2-errata-v2.0, Liberty Alliance Project (12 September 2004). http://www.projectliberty.org/specs/

draft-liberty-idff-bindings-profiles-1.2-errata-v2.0.pdf.

[LibertyImpl-web] Liberty Alliance Project - Conformant Products, http://www.projectliberty.org/activities/conformant_products.php.

[LibertyProtSchema] Cantor, Scott, Kemp, John, eds. *Liberty ID-FF Protocols and Schema Specification*, Version 1.2-errata-v3.0, Liberty Alliance Project (12 September 2004). http://www.projectliberty.org/specs/draft-liberty-idff-protocols-schema-1.2-errata-v3.0.pdf

[LibertyReq] Eric Tiffany ed., *Liberty ID-FF 1.2 Static Conformance Requirements*, Liberty Alliance Project. http://www.projectliberty.org/specs/liberty-idff-1.2-scr-v1.0.pdf.

[Liberty-web] Liberty Alliance / Project Liberty, http://www.projectliberty.org.

[MSPassport] Microsoft Corporation *Microsoft .NET Passport Review Guide*. January 2004. http://download.microsoft.com/download/a/f/4/af49b391-086e-4aa2-a84b-ef6d916b2f08/passport_reviewguide.doc.

[Model2] Understanding JavaServer Pages Model 2 architecture, http://www.javaworld.com/javaworld/jw-12-1999/jw-12-ssj-jspmvc.html.

[OpenPermis-web] Open PERMIS, http://sec.isi.salford.ac.uk/openPermis/.

[OpenSAML-web] OpenSAML - an Open Source Security Assertion Language implementation, http://www.opensaml.org/.

[OpenSPCoop-web] OpenSPCoop Home Page, http://www.openspcoop.org/.

[POSA] Buschmann, F. et al. *Pattern-Oriented Software Architecture*. John Wiley and Sons, 1996. ISBN 0-471-95869-7.

[PRIME-web] Portal for the PRIME Project - PRIME, http://www.prime-project.eu.org/.

[PermisACM] Chadwick, D.W., Otenko, A. *The PERMIS X.509 Role Based Privilege Management Infrastructure*, SACMAT'02, June 3-4, 2002. http://sec.isi.salford.ac.uk/download/SACMATfinal.pdf.

[PermisSEC] Chadwick, D.W., Otenko, A. *RBAC Policies in XML for X.509 Based Privilege Management*, SEC, 2002. http://sec.isi.salford.ac.uk/download/Sec2002Final.pdf.

[Permis-web] Privilege and Role Management Infrastructure Standards Validation, http://sec.isi.salford.ac.uk/permis/.

[Piano13] *e-Government: Il piano di azione del Governo*. Rapporto del 23 giugno 2000. http://www.innovazione.gov.it/ita/soc_info/politiche_governo/palchigi_rapp_neweconomy_sint.shtml.

[RBAC] Ferraiolo, D. Kuhn, R. *Role-Based Access Controls*. 15th NIST-NCSC National Computer Securi-

ty Conference, 1992. http://csrc.nist.gov/rbac/Role_Based_Access_Control-1992.html.

[RBAC-Sandhu] Sandhu, R. S. et al. *Role-Based Access Control Models*. IEEE Computer 29(2): 38-47, IEEE Press, 1996. http://csrc.nist.gov/rbac/sandhu96.pdf.

[RBAC-web] National Institute of Standards and Technology – Role Based Access Control, http://csrc.nist.gov/rbac/.

[RFC1510] Kohl, J. et al. *The Kerberos Network Authentication Service*. IETF RFC 1510, September 1993. http://www.ietf.org/rfc/rfc1510.txt.

[RFC2045] Freed, N. et al. *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*. IETF RFC 2045, November 1996. http://www.ietf.org/rfc/rfc2045.txt.

[RFC2046] Freed, N. et al. *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*. IETF RFC 2046, November 1996. http://www.ietf.org/rfc/rfc2046.txt.

[RFC2315] Kaliski, B. *PKCS #7: Cryptographic Message Syntax*. IETF RFC 2315, March 1998. http://www.ietf.org/rfc/rfc2315.txt.

[RFC2560] M. Myers et al., *X.509 Internet Public Key Infrastructure: Online Certificate Status Protocol - OCSP*. IETF RFC 2560, June 1999. http://www.ietf.org/rfc/rfc2560.txt.

[RFC2904] Vollbrecht, J. et al. *AAA Authorization Framework*. IETF RFC 2904, August 2000. http://www.ietf.org/rfc/rfc2904.txt.

[RFC3060] Moore, B. et al. *Policy Core Information Model – Version 1 Specification*. IETF RFC 3060, February 2001. http://www.ietf.org/rfc/rfc3060.txt.

[RFC3280] R. Housley et al., *Internet X.509 Public Key Infrastructure: Certificate and Certificate Revocation List (CRL) Profile*. IETF RFC 3280, April 2002. http://www.ietf.org/rfc/rfc3280.txt.

[RFC3281] Farrell, S. et al. *An Internet Attribute Certificate Profile for Authorization*. IETF RFC 3281, April 2002. http://www.ietf.org/rfc/rfc3281.txt.

[Rouault] Rouault, J. *Making sense of the federation protocol landscape*. Hewlett-Packard, July 2005. http://devresource.hp.com/drc/resources/fed_land/federation_landscapeHP.pdf.

[SAML-bind] E. Maler et al., *Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-bindings-profiles-1.1. http://www.oasis-open.org/committees/download.php/3405/oasis-sstc-saml-bindings-1.1.pdf.

[SAML-core] E. Maler et al., *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-core-1.1. http://www.oasis-open.org/committees/download.php/3406/oasis-sstc-saml-core-1.1.pdf.

[SAML-web] OASIS Security Services (SAML) Technical Committee, http://www.oasis-open.org/committees/security/.

[SOAP-Attachment] Nielsen, H. F. et al. eds. *SOAP 1.2 Attachment Feature*. W3C Working Group Note, 8 June 2004. http://www.w3.org/TR/soap12-af/.

[SOAP] Gudgin, M. et al. eds. *SOAP Version 1.2 Part 1: Messaging Framework*. W3C Recommendation, 24 June 2003. http://www.w3.org/TR/soap12-part1/.

[Servlet-API] Servlet API, http://java.sun.com/products/servlet/docs.html.

[Shib-SAML1Meta] G. Whitehead and S. Cantor, *Metadata Profile for the OASIS Security Assertion Markup Language (SAML) V1.x*. OASIS SSTC Committee Draft 01, March 2005. Document ID sstc-saml1x-metadata-cd-01. http://www.oasis-open.org/committees/download.php/13254/sstc-saml1x-metadata-cd-01.pdf.

[Shib-arch] S. Cantor et al., *Shibboleth Architecture: Technical Overview*. Internet2-MACE Working Draft 02, June 2005. Document ID draft-mace-shibboleth-tech-overview-02. http://shibboleth.internet2.edu/docs/draft-mace-shibboleth-tech-overview-02.pdf.

[Shib-proto] S. Cantor et al., *Shibboleth Architecture: Protocols and Profiles*. Internet2-MACE Working Draft 10, August 2005. Document ID draft-mace-shibboleth-arch-protocols-10. http://shibboleth.internet2.edu/docs/draft-mace-shibboleth-arch-protocols-10.pdf.

[Shib-web] Shibboleth Project - Internet2 Middleware, http://shibboleth.internet2.edu/.

[SrtyToscana] Regione Toscana. *SRTY*. http://www.rete.toscana.it/gar/arpa/srty.pdf

[SunXACML-web] Sun's XACML Implementation, http://sunxacml.sourceforge.net/.

[TRIM] Josang, A. et al. *Trust Requirements in Identity Management*. Third Australasian Information Security Workshop (AISW2005). http://crpit.com/confpapers/CRPITV44Josang.pdf.

[Tomcat-web] Apache Tomcat, http://jakarta.apache.org/tomcat/.

[VO] Foster, I. et al. *The Anatomy of the Grid: Enabling Scalable Virtual Organizations*. International Journal of Supercomputer Applications. 2001. http://www.globus.org/alliance/publications/papers/anatomy.pdf.

[WS-CDL] *Web Services Choreography Description Language*, W3C Working Draft, 17 December 2004. http://www.w3.org/TR/ws-cdl-10/.

[WSDL 1.1] *Web Services Description Language (WSDL) 1.1*. W3C Note, 15 March 2001. http://www.w3.org/TR/wsdl.

[WSRF] Czajkowski, K. et al. *The WS-Resource Framework*. Version 1.0. May 2004. http://www.globus.org/wsrf/specs/ws-wsrf.pdf.

[WS-SC] Anderson, S. et al. *Web Services Secure Conversation Language (WS-SecureConversation)*. Specification. http://www-128.ibm.com/developerworks/library/specification/ws-secon/.

[WSSOA] Web Services and Service-Oriented Architectures, http://www.service-architecture.com/.

[WSS] P. Hallam-Baker et al., *Web Services Security: SOAP Message Security 1.0*. OASIS, August 2003. Document ID {WSS: SOAP Message Security }-{1.0}. http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf.

[WSS-saml] P. Hallam-Baker et al., *Web Services Security: SAML Token Profile*. OASIS, December 2004. Document ID oasis-wss-saml-token-profile-1.0. http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0.pdf.

[Wikipedia] Wikipedia, http://www.wikipedia.org/.

[Windley] Windley, P. *Digital Identity*. O'Reilly, August 2005. ISBN: 0-596-00878-3.

[X.509] ITU-T Recommendation X.509 (1997 E): *Information Technology - Open Systems Interconnection - The Directory: Authentication Framework*, June 1997.

[X.812] ITU-T Rec X.812 (1995) | ISO/IEC 10181-3:1996, *Information technology - Open Systems Interconnection - Security frameworks for open systems: Access control framework*.

[XACML] Moses, T. ed. *eXtensible Access Control Markup Language (XACML) Version 2.0*. OASIS Standard, 1 Feb 2005. Document id: oasis-access_control-xacml-2.0-core-spec-os. http:

//docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf.

[XACML-web] OASIS eXtensible Access Control Markup Language (XACML) TC, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml.

[XML] Bray, T. et al. eds. *Extensible Markup Language (XML) 1.1*. W3C Recommendation, February 2004. http://www.w3.org/TR/xml11/.

[XMLEnc] Eastlake D. et al. eds. *XML Encryption Syntax and Processing*. W3C Recommendation, December 2002. http://www.w3.org/TR/xmlenc-core/.

[XML-Schema] Fallside, D. C. et al. eds. *XML Schema Part 0: Primer Second Edition*. W3C Recommendation. October 2004. http://www.w3.org/TR/xmlschema-0/.

[XMLSig] Eastlake D. et al. eds. *XML-Signature Syntax and Processing*. W3C Recommendation, February 2002. http://www.w3.org/TR/xmldsig-core/.

[Yale CAS-web] Yale Central Authentication Service, http://www.yale.edu/tp/auth/

[Yale Shib-web] Yale Shibboleth Service Provider Java implementation, http://tp.its.yale.edu/shib/tiki-index.php.

[e-Europe-web] European Commission - Information Society - eEurope 2005, http://europa.eu.int/information_society/eeurope/2005/index_en.htm.

Glossario degli acronimi e delle abbreviazioni

AAA Authentication, Authorization and Accounting

AAAarch RG Authorization, Authentication and Accounting architecture Research Group, http://www.aaaarch.org/

AA Attribute Authority

AAP Attribute Acceptance Policy

AC Attribute Certificate

ACI Access Control Information

ACL Access Control List

ACRL Attribute Certificate Revocation List

ADF Access control Decision Function

ADI Access control Decision Information

AEF Access control Enforcement Function

AIPA Autorità per l'Informatica nella Pubblica Amministrazione

API Application Programming Interface

ARBAC Administrative Role-Based Access Control

ARL Active Role List

ARP Attribute Release Policy

B2B Business to Business

CA Certification Authority

CART Cooperazione Applicativa Regione Toscana

CAS Community Authorization Service

CGI Common Gateway Interface

CIE Carta di Identità Elettronica

CISIS Centro Interregionale per il Sistema Informatico ed il sistema Statistico

CN Common Name

CNIPA Centro Nazionale per l'Informatica nella Pubblica Amministrazione, http://www.cnipa.gov.it/

CNS Carta Nazionale dei Servizi

CPSI Comitato Permanente dei responsabili dei Sistemi Informatici delle regioni

CRL Certificate Revocation List

CSO Chief Security Officer

DAC Discretionary Access Control

DB Data Base

DEM Document Exchange Model

DI Dominio delle Identità

DN Distinguished Name

DOM Document Object Model

DS Dominio dei Servizi

DTD Document Type Definition

DoS Denial of Service

EC European Community

FIDIS Future of IDentity in the Information Society, http://www.fidis.net/

GSI Grid Security Infrastructure

GT Globus Toolkit

GUIDE Government User IDentity for Europe

GUI Graphical User Interface

HTML HyperText Markup Language

HTTP HyperText Transfer Protocol

HTTPS Secure HyperText Transfer Protocol

ICAR Interoperabilità e Cooperazione Applicativa tra le Regioni

ICT Information and Communication Technology

ID-FF Identity Federation Framework

ID-SIS Services Interface Specifications

ID-WSF Web Services Framework

IEC International Electrotechnical Commission, http://www.iec.ch/

IEEE Institute of Electrical and Electronics Engineers, inc., http://www.ieee.org/

IETF Internet Engineering Task Force, http://www.ietf.org/

IIS Internet Information Services

IM Identity Management

INA Indice Nazionale delle Anagrafi

IP Internet Protocol

ISO International Organization for Standardization, http://www.iso.org/

ISP Internet Service Provider

ISSRG Information Systems Security Research Group

ITU-T International Telecommunication Union, Telecommunication standardization sector, http://www.itu.int/ITU-T/

IdP Identity Provider

J2EE Java 2 Platform Enterprise Edition, http://java.sun.com/j2ee/

JDBC Java DataBase Connectivity

JSP Java Server Pages

JVM Java Virtual Machine

LDAP Ligthweight Directory Access Protocol

LECP Liberty-Enabled Client and Proxy

MACE Middleware Architecture Committee for Education, http://middleware.internet2.edu/MACE/

MAC Mandatory Access Control

MIME Multipurpose Internet Mail Extensions

MVC Model-View-Controller

NAG Nodo di AGgregazione

NDOM Nodo di DOMinio

NIST National Institute of Standards and Technology, http://www.nist.gov/

NTT Nippon Telegraph and Telephone corporation

OASIS Organization for the Advancement of Structured Information Standards, http://www.oasis-open.org/

OCSP Online Certificate Status Protocol

OSI Open Systems Interconnection

PAD Personal Authentication Device

PA Pubblica Amministrazione

PDP Policy Decision Point

PEC Posta Elettronica Certificata

PEP Policy Enforcement Point

PERMIS PrivilEge and Role Management Infrastructure Standards validation

PIN Personal Identification Number

PIP Policy Information Point

PKC Public Key Certificate

PKCS Public Key Cryptography Standards

PKI Public Key Infrastructure

PMI Policy Management Infrastructure

PRIME PRivacy and Identity Management for Europe, http://www.prime-project.eu.org/

PRP Policy Retrieval Point

PdD Porta di Dominio

QoS Quality of Service

RBAC Role-Based Access Control

RDF Resource Description Framework

RFC Request For Comments

RM Resource Manager

RTRT Rete Telematica Regionale Toscana

RUPA Rete Unitaria della Pubblica Amministrazione

SAIA Sistema di Accesso e di Interscambio Anagrafico

SAML Security Assertion Markup Language

SE Service Equipment

SICA Servizi Infrastrutturali di Cooperazione ed Accesso

SIL Sistema Informativo Locale

SIRC Servizio di Identificazione e di Ruolo di Comunità

SLA Service Level Agreement

SMTP Simple Mail Transfer Protocol

SOA Service Oriented Architecture

SPC Sistema Pubblico di Connettività e cooperazione

SPICCA Sistema Pubblico di Interoperabilità e Cooperazione applicativa CAmpana

SP Service Provider

SQL Structured Query Language

SSL Secure Sockets Layer

SSO Single Sign On

SoA Source of Authority

TCO Total Cost of Ownership

TLS Transport Layer Security

TSA Time Stamping Authority

UA User Agent

UDDI Universal Description, Discovery and Integration

UHO User Home Organization

URI Uniform Resource Identifier

URL Uniform Resource Locator

VO Virtual Organization

W3C World Wide Web Consortium, http://www.w3.org/

WAYF Where Are You From

WSDL Web Services Description Language

WSS Web Services Security

WWW World Wide Web

XACML eXtensible Access Control Markup Language

XHTML Extensible HyperText Markup Language

XHTML eXtensible HyperText Markup Language

XML eXtensible Markup Language, http://www.w3.org/XML/

XSL eXtensible Stylesheet Language, http://www.w3.org/Style/XSL/

ebXML Electronic Business using eXtensible Markup Language

Codice sorgente del prototipo

Listing 5.1: Codice Java del Profile Servlet, il modulo *controller* della Profile Webapp.

```
1  import java.io.IOException;
2  import javax.servlet.http.*;
3  import javax.servlet.*;
4
5  import java.security.*;
6  import java.security.cert.Certificate;
7  import java.security.cert.CertificateException;
8  import java.security.cert.X509Certificate;
9  import java.util.*;
10 import java.io.*;
11
12 import org.apache.xml.security.signature.XMLSignature;
13 import org.opensaml.*;
14 import org.opensaml.provider.SOAPHTTPBindingProvider;
15 import org.w3c.dom.Element;
16 import org.apache.log4j.*;
17
18 /**
19  * <p>Implementazione di un servlet che utilizza le facilities di [...]
        Shibboleth per il SSO.
20  * Oltre a ricevere gli attributi utente in "push" questo Servlet [...]
        interroga esplicitamente
21  * AA per ottenere altri attributi in "pull", con i quali costruira' un [...]
        profilo utente
22  * in base al quale verra' fornito contenuto customizzato
23  *
24  * <p>Lo stesso meccanismo puo' essere usato per interrogare altri tipi [...]
        di SAML Responder
25  * come ad esempio una Authorization Authority
26  */
27 @SuppressWarnings("serial")
28  public class ProfileServlet extends HttpServlet
29  {
30
31
32   private Logger logger = null;
33   private PrintWriter writer = null;
34
```

```
35
36   /**
37    * Recupera un attributo tra quelli "iniettati" nella sessione dal [...]
          Filter
38    * @param localSession, la sessione locale del servlet
39    * @param attributeName, il nome dell'attributo da ricuperare
40    * @return l'attributo ottenuto dalla sessione del Service Provider [...]
          Shibboleth
41    */
42   private Object
43  getShibAttribute(HttpSession localSession,
44           String attributeName)
45   {
46    if(localSession == null) {
47     log("sessione␣(locale)␣assente.");
48     return null;
49    }
50
51    String shibAttributesKey = getInitParameter("shibAttributesKey");
52
53    return ((Map) localSession.getAttribute(shibAttributesKey)).get(...]
          attributeName);
54   }
55
56
57
58   /**
59    * Crea un SAMLSubject a partire da un nameIdentifier (handle)
60    * @param handle, il "nome" del soggetto
61    * @param handleFormat, il formato dello handle
62    * @return il blocco \<Subject\> sotto forma di istanza di classe [...]
          OpenSAML
63    * @exception ServletException in caso di errore SAML
64    */
65   private SAMLSubject
66  createSubject(String handle,
67         String handleFormat)
68    throws ServletException
69   {
70    SAMLSubject subject = null;
71
72    try {
73
74     SAMLNameIdentifier nameId =
75      new SAMLNameIdentifier(handle, //il valore del blocco <[...]
            NameIdentifier>
76                             getInitParameter("defaultNameQualifier"), //[...]
                                 il nameQualifier
77                             handleFormat); //il formato
78
79     TreeSet<String> confirmationMethods = new TreeSet<String>();
80     confirmationMethods.add(SAMLSubject.CONF_BEARER);
81     subject = new SAMLSubject(nameId,
82                               confirmationMethods,
83                               null, //confirmation data
84                               null); //ds:KeyInfo
85
86    } catch(SAMLException e) {
```

```
87      throw new ServletException("errore SAML nella creazione del <Subject[...]
            >: " + e);
88     }
89     return subject;
90    }
91
92
93    /**
94     * Crea una richiesta di attributi (vuota = tutti gli attributi)
95     * conforme allo standard SAML 1.1 a partire da un SAMLSubject
96     * @param subject, il soggetto della richiesta
97     * @param resource, la risorsa per la cui decisione di accesso sono [...]
           necessari gli attributi, puo' essere null
98     * @return la richiesta sotto forma di istanza di classe OpenSAML
99     * @exception ServletException in caso di errore SAML
100    */
101   private SAMLRequest
102  createAttributeRequest(SAMLSubject subject,
103              String resource)
104    throws ServletException
105   {
106    SAMLRequest request = null;
107
108    try {
109
110     SAMLAttributeQuery attributeQuery =
111      new SAMLAttributeQuery(subject,
112       resource, //la risorsa
113       null); //gli attributi richiesti, vuoto per averli tutti
114
115     request = new SAMLRequest(attributeQuery);
116
117    } catch(SAMLException e) {
118     throw new ServletException("errore SAML nella creazione della [...]
           richiesta di attributi: " + e);
119    }
120    return request;
121   }
122
123
124   /**
125    * Crea una <samlp:request> contenente una query di autorizzazione
126    * Si e' scelto di chiedere una sola azione alla volta (col namespace [...]
           di default)
127    * Inoltre nel campo evidence vengono aggiunte l'asserzione di [...]
           autenticazione
128    * (ottenuta al momento del sign-on) e quella di attributi
129    * @param subject, il soggetto per conto di cui si chiede l'[...]
           autorizzazione
130    * @param resource, la risorsa a cui si chiede l'accesso
131    * @param action, l'azione nel namespace GHPP
132    * @param authentication,  (opzionale) l'asserzione di autenticazione
133    * @param attributes, (opzionale) l'asserzione di attributi
134    * @return la <samlp:request>
135    * @throws ServletException
136    */
137   private SAMLRequest
138  createAuthorizationRequest(SAMLSubject subject,
139              String resource,
```

```
140                 String action,
141                 SAMLAssertion authentication,
142                 SAMLAssertion attributes)
143     throws ServletException
144   {
145     try {
146      SAMLAction sAction =
147     new SAMLAction(SAMLAction.SAML_ACTION_NAMESPACE_GHPP, action);
148
149      SAMLAuthorizationDecisionQuery query =
150       new SAMLAuthorizationDecisionQuery(subject, resource, null, null);
151
152      query.addAction(sAction);
153
154      if(authentication != null)
155       query.addEvidence(authentication);
156
157      if(attributes != null)
158       query.addEvidence(attributes);
159
160      return new SAMLRequest(query);
161     } catch(SAMLException e) {
162      throw new ServletException("errore␣SAML␣nella␣creazione␣della␣" +
163                                 "richiesta␣di␣autorizzazione" + e);
164     }
165   }
166
167
168   /**
169    * Interroga un SAML responder per ottenere asserzioni (in verita' "[...]
           risposte")
170    * Si ipotizza che il responder sia nel dominio dell'IdP e che il [...]
           Servlet (noi) sia
171    * nel dominio del SP
172    * @param sReq, la SAML request sotto forma di oggetto OpenSAML
173    * @param responderURL, la URL del SAML responder
174    * @return la risposta SAML o null in caso di errore
175    */
176   private SAMLResponse
177  sendRequest(SAMLRequest sReq,
178        String responderURL)
179    throws ServletException
180   {
181    //inizializzazione del singleton SAMLConfig
182    SAMLConfig config = SAMLConfig.instance();
183
184    //keystore con la chiave privata del client (il servlet)
185    config.setProperty("org.opensaml.ssl.keystore",
186            getInitParameter("localKeyStore"));
187    config.setProperty("org.opensaml.ssl.keystore-type",
188            getInitParameter("localKeyStoreType"));
189    config.setProperty("org.opensaml.ssl.keystore-pwd",
190            getInitParameter("localKeyStorePass"));
191    config.setProperty("org.opensaml.ssl.key-pwd",
192            getInitParameter("localKeyPass"));
193
194    //keystore con il certificato del server (il responder)
195    config.setProperty("org.opensaml.ssl.truststore",
196            getInitParameter("remoteKeyStore"));
```

```
197     config.setProperty("org.opensaml.ssl.truststore-type",
198             getInitParameter("remoteKeyStoreType"));
199     config.setProperty("org.opensaml.ssl.truststore-pwd",
200             getInitParameter("remoteKeyStorePass"));
201
202     //creazione del binding SAML-over-SOAP-over-HTTPS
203     //un altro modo per fare la stessa cosa e':
204     //SAMLBinding binding = SAMLBindingFactory.getInstance(SAMLBinding.[...]
            SOAP);
205     SOAPHTTPBindingProvider binding = null;
206     try {
207      binding =
208       new SOAPHTTPBindingProvider("urn:oasis:names:tc:SAML:1.0:bindings:[...]
              SOAP-binding",
209                   null);
210     } catch(SAMLException e) {
211      throw new ServletException("errore nella creazione del binding SAML-[...]
            over-SOAP-over-HTTPS " +
212                                 "per l'invocazione del SAML responder: " +[...]
                                      e);
213     }
214
215     SAMLResponse sResp = null;
216     try {
217      sResp = binding.send(responderURL, sReq);
218     } catch(SAMLException e) {
219      throw new ServletException("errore nell'invio della SAML request al [...]
            SAML responder: " + e);
220     }
221
222     return sResp;
223   }
224
225
226   /**
227    * Effettua la XML Digital Signature di un qualsiasi blocco SAML [...]
          firmabile
228    * Utilizza la chiave privata del mittente (il Servlet)
229    * Include il certificato del mittente (il Servlet)
230    * @param toSign, il blocco da firmare
231    * @return lo stesso blocco dopo la firma
232    * @throws ServletException in caso di eccezioni catturate
233    */
234   private void
235  signUtil(SAMLSignedObject toSign)
236    throws ServletException
237   {
238    //recupero delle chiavi e dei certificati dai Java Keystores
239    //forniti nella configurazione della webapp
240
241    String localKeyStorePath = getInitParameter("localKeyStore");
242    String localKeyStoreType = getInitParameter("localKeyStoreType");
243    String localKeyStorePass = getInitParameter("localKeyStorePass");
244    String localKeyPass = getInitParameter("localKeyPass");
245    String localKeyStoreUser = getInitParameter("localKeyStoreUser");
246
247    if(!localKeyStoreType.equals("JKS"))
248     throw new ServletException("formato del keystore non supportato");
249    try {
```

```
250
251     KeyStore localKeyStore = KeyStore.getInstance(localKeyStoreType);
252
253     localKeyStore.load(new FileInputStream(localKeyStorePath),
254                        localKeyStorePass.toCharArray());
255
256     PrivateKey localPrivateKey =
257      (PrivateKey) localKeyStore.getKey(localKeyStoreUser,
258                     localKeyPass.toCharArray());
259     if (localPrivateKey == null)
260      throw new ServletException("nessuna chiave privata trovata nel Java [...]
           KeyStore");
261
262     Certificate[] certificates =
263    localKeyStore.getCertificateChain(localKeyStoreUser);
264     if (certificates == null)
265      throw new ServletException("nessun certificato trovato nel Java [...]
           KeyStore");
266
267     X509Certificate[] x509Certs = new X509Certificate[certificates.length[...]
          ];
268     for (int i = 0; i < certificates.length; i++) {
269      if (certificates[i] instanceof X509Certificate) {
270       x509Certs[i] = (X509Certificate) certificates[i];
271      } else {
272       throw new
273     ServletException("solo i certificati di tipo X509 sono supportati");
274      }
275     }
276
277
278     //firma vera e propria (usando OpenSAML)
279
280     toSign.sign(XMLSignature.ALGO_ID_SIGNATURE_RSA_SHA1,
281         localPrivateKey,
282         Arrays.asList(x509Certs));
283
284    } catch(KeyStoreException kse) {
285     throw new
286    ServletException("errore nell'accesso al Java KeyStore: "
287            + kse);
288    } catch(NoSuchAlgorithmException nsae) {
289     throw new
290    ServletException("impossibile trovare l'appropriato provider JCE: "
291            + nsae);
292    } catch(CertificateException ce) {
293     throw new
294    ServletException("impossibile caricare il certificato dal Java [...]
          KeyStore: "
295            + ce);
296    } catch(IOException ioe) {
297     throw new
298    ServletException("errore in lettura del Java KeyStore: "
299            + ioe);
300    } catch(UnrecoverableKeyException uke) {
301     throw new
302    ServletException("impossibile leggere la chiave privata dal Java [...]
          KeyStore: "
303            + uke);
```

```
304     } catch(SAMLException se) {
305      throw new
306     ServletException("errore␣nell'esecuzione␣della␣firma:␣"
307             + se);
308     }
309
310   }
311
312
313   /**
314    * Metodo principale
315    */
316   protected void
317   doGet(HttpServletRequest httpReq,
318     HttpServletResponse httpResp)
319    throws ServletException, IOException
320   {
321    //la sessione dovrebbe gia' essere stata creata dal Filter
322    HttpSession localSession = httpReq.getSession(false);
323
324    //httpResp.setContentType("text/plain");
325    //PrintWriter out = httpResp.getWriter();
326
327    //recupera gli attributi di sessione
328    String handle = (String) getShibAttribute(localSession, "handle");
329    String handleFormat =
330   (String) getShibAttribute(localSession, "handle-format");
331    SAMLAssertion authN = null;
332    SAMLAssertion attrib = null;
333    try {
334     authN = new SAMLAssertion((Element)
335                 getShibAttribute(localSession,
336                         "authentication-assertion"));
337     attrib  = new SAMLAssertion((Element)
338                   getShibAttribute(localSession,
339                           "attribute-assertion"));
340    } catch(SAMLException e) {
341     throw new
342    ServletException("impossibile␣estrarre␣dalla␣sessione␣" +
343            "le␣asserzioni␣di␣autenticazione␣ed␣attributi:␣" + e);
344    }
345
346    log("handle␣=␣" + handle + "␣␣handle-format␣=␣" + handleFormat);
347    log("authN:\n" + authN);
348    log("attrib:\n" + attrib);
349
350    SAMLRequest sReq = null;
351    try {
352     sReq = createAuthorizationRequest((SAMLSubject) ((...]
         SAMLSubjectStatement) authN.getStatements().next()).getSubject().[...]
         clone(),
353                                       "https://sp.example.org/profile-[...]
                                            webapp/private",
354                                       "POST",
355                                       authN,
356                                       attrib);
357    } catch(CloneNotSupportedException e) {
358     throw new
359    ServletException("errore␣nella␣clonazione␣di␣un␣oggetto␣SAML:␣"
```

```
360             + e);
361       }
362       log("request:\n" + sReq);
363
364       //signUtil(sReq);
365       log("<samlp:request>␣=\n" + sReq);
366       log("isSigned:␣" + sReq.isSigned());
367
368       SAMLResponse sResp = sendRequest(sReq,
369                       getInitParameter("PDPurl"));
370       log("<samlp:response>␣=\n" + sResp);
371
372       SAMLAssertion authZ =
373      (SAMLAssertion) sResp.getAssertions().next();
374       log("sResp.isSigned():␣" + sResp.isSigned());
375       log("authZ.isSigned():␣" + authZ.isSigned());
376       try {
377        authZ.verify();
378       } catch(SAMLException e) {
379        log("authZ.verify():␣error:␣" + e);
380        throw new
381       ServletException("firma␣non␣valida␣nell'asserzione␣di␣autorizzazione:␣[...]
             "
382             +e);
383       } finally {
384        log("authZ.verify():␣ok");
385       }
386
387       try {
388        X509Certificate issuerCert =
389       (X509Certificate) authZ.getX509Certificates().next();
390        log("authZ␣issuer␣Cert␣subject:␣"
391        + issuerCert.getSubjectDN().getName());
392        log("authZ␣issuer␣Cert␣root␣CA:␣"
393        + issuerCert.getIssuerDN().getName());
394       } catch(SAMLException e) {
395        throw new
396       ServletException("impossibile␣estrarre␣il␣certificato␣X509␣" +
397             "dall'asserzione␣di␣autorizzazione" + e);
398       }
399
400       //passiamo l'autorizzazione alla pagina JSP
401       try {
402        localSession.setAttribute("authorization-assertion",
403                    ((SAMLAssertion)
404                     sResp.getAssertions().next()).toDOM());
405       } catch(SAMLException e) {
406        throw new ServletException("impossibile␣salvare␣l'autorizzazione␣" +
407                                   "negli␣attributi␣di␣sessione:␣" + e);
408       }
409
410       getServletContext().getRequestDispatcher("/display.jsp").forward(...]
             httpReq, httpResp);
411
412       //localSession.invalidate();
413      }
414
415
416      @Override
```

```
417   public void
418  init(ServletConfig config)
419   throws ServletException {
420
421   super.init(config);
422
423    try {
424     writer =
425    new PrintWriter(new BufferedWriter(new FileWriter(getInitParameter("[...]
           logfile"), true)), true);
426    } catch(IOException e) {}
427   }
428
429
430   @Override
431   public void
432  log(String message)
433   {
434    writer.println("[" + new Date() + "] " + message);
435   }
436
437  }
```

Listing 5.2: Codice JSP dell'Assertion Visualizer, il modulo *view* della Profile Webapp.

```
1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//IT">
2  <html>
3   <head>
4    <meta http-equiv="Content-Type"
5       content="text/html; charset=iso-8859-1">
6    <title>Profilo Utente</title>
7   </head>
8   <body>
9    <%@ page
10     language="java"
11     import="org.opensaml.*, org.w3c.dom.*, java.util.*" %>
12   <%--
13      Non e' stato usato un bean perche' in OpenSAML il
14      costruttore di default si puo' usare solo per
15      creare *nuove* asserzioni, ovvero non firmate e
16      quindi inservibili come token
17   --%>
18   <%  SAMLAssertion authZ =
19     new SAMLAssertion((Element) session.getAttribute("authorization-[...]
           assertion")); %>
20   <p> Issuer =
21    <%= authZ.getIssuer() %>
22   <p> Validity = from
23    <%= authZ.getNotBefore() %>
24    to
25    <%= authZ.getNotOnOrAfter() %>
26   <p> Signed =
27    <%= authZ.isSigned()  %>
```

```
28    <p>
29    <p> Statement:
30     <%  SAMLAuthorizationDecisionStatement statement =
31       (SAMLAuthorizationDecisionStatement) authZ.getStatements().next(); %[...]
             >
32     <%  SAMLSubject subject =
33       statement.getSubject(); %>
34     <%  SAMLNameIdentifier nameId =
35       subject.getNameIdentifier(); %>
36    <p> Subject =
37     <%= nameId.getName() %>
38     @
39     <%= nameId.getNameQualifier() %>
40    <p> SubjectConfirmation =
41     <%= subject.getConfirmationMethods().next() %>
42    <p>
43    <p> Resource =
44     <%= statement.getResource() %>
45    <p>
46     <%  Iterator actions = statement.getActions(); %>
47     <% SAMLAction i = null; %>
48    <p> Actions:
49     <%  while(actions.hasNext()) { %>
50     <%  i = (SAMLAction) actions.next(); %>
51    <p> Name =
52     <%= i.getData() %>
53     , Namespace =
54     <%= i.getNamespace() %>
55     <%  } %>
56    <p>
57    <p> Permission:
58     <%= statement.getDecision() %>
59    <p>
60     <%  Iterator evidence = statement.getEvidence(); %>
61    <p> Evidence: present =
62     <%= evidence.hasNext() %>
63    <p>
64     <%  SAMLAssertion assertion = null; %>
65     <%  int j = 1; %>
66     <%  while(evidence.hasNext()) {%>
67    <p> Assertion n.
68     <%= j %>:
69     <%  assertion = (SAMLAssertion) evidence.next(); %>
70     signed =
71     <%= assertion.isSigned() %>
72     ,
73     <% boolean valid ;
74       try { assertion.verify();
75       } catch(SAMLException e)
76       {valid = false;}
77       finally
78       {valid = true;} %>
79    valid =
80     <%= valid %>
81     <%  j++; } %>
82     <%  session.invalidate(); %>
83   </body>
84  </html>
```

Listing 5.3: File di configurazione di Tomcat per la Profile Webapp. Indica come si possa integrare un'applicazione web all'interno di Shibboleth.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE web-app
  PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
  "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>

 <display-name>Applicazioni Web SAML</display-name>
  <description>
  Queste applicazioni web mostrano l''utilizzo dello standard SAML per
  dialogare con Shibboleth e con altri SAML Responder generici, come
  ad esempio servizi di autorizzazione
  </description>

 <filter>
    <filter-name>ShibFilter</filter-name>
    <filter-class>edu.internet2.middleware.shibboleth.resource.[...]
        AuthenticationFilter</filter-class>
    <init-param>
      <param-name>shireURL</param-name>
      <param-value>https://sp.example.org:9443/shibboleth-sp/Shibboleth.[...]
          sso/SAML/POST</param-value>
    </init-param>
  <init-param>
      <param-name>wayfURL</param-name>
      <param-value>https://idp.example.org:443/shibboleth-idp/SSO</param-[...]
          value>
    </init-param>
     <init-param>
      <param-name>providerId</param-name>
      <param-value>https://sp.example.org/shibboleth</param-value>
    </init-param>
    <init-param>
      <param-name>requireId</param-name>
      <param-value>.+/ProfileServlet</param-value>
    </init-param>
   </filter>

 <filter-mapping>
  <filter-name>ShibFilter</filter-name>
  <url-pattern>/ProfileServlet</url-pattern>
 </filter-mapping>

 <servlet>
    <servlet-name>TestServlet</servlet-name>
    <display-name>Test Servlet</display-name>
    <servlet-class>TestServlet</servlet-class>
  </servlet>

 <servlet>
  <servlet-name>ProfileServlet</servlet-name>
  <display-name>Profile Servlet</display-name>
  <description>
   Implementazione di un''applicazione che ricupera il proprio profilo
```

```
53      da asserzioni di attributi ed autorizzazioni ottenuti da un SAML
54      responder.
55    </description>
56    <servlet-class>ProfileServlet</servlet-class>
57    <init-param>
58      <param-name>shibAttributesKey</param-name>
59      <param-value>edu.internet2.authentication.PrincipalAttributes</param-[...]
            value>
60    </init-param>
61    <init-param>
62      <param-name>defaultNameQualifier</param-name>
63      <param-value>https://idp.example.org/shibboleth</param-value>
64    </init-param>
65    <init-param>
66      <param-name>localKeyStore</param-name>
67      <param-value>../profile-webapp/keys/sp-example.jks</param-value>
68    </init-param>
69    <init-param>
70      <param-name>localKeyStoreType</param-name>
71      <param-value>JKS</param-value>
72    </init-param>
73    <init-param>
74      <param-name>localKeyStoreUser</param-name>
75      <param-value>tomcat</param-value>
76    </init-param>
77    <init-param>
78      <param-name>localKeyStorePass</param-name>
79      <param-value>exampleorg</param-value>
80    </init-param>
81    <init-param>
82      <param-name>localKeyPass</param-name>
83      <param-value>exampleorg</param-value>
84    </init-param>
85    <!--
86      ATTENZIONE al momento supponiamo che tutti i SAML responders
87      abbiano lo stesso server cert
88    -->
89    <init-param>
90      <param-name>remoteKeyStore</param-name>
91      <param-value>../profile-webapp/keys/idp-example.jks</param-value>
92    </init-param>
93    <init-param>
94      <param-name>remoteKeyStoreType</param-name>
95      <param-value>JKS</param-value>
96    </init-param>
97    <init-param>
98      <param-name>remoteKeyStoreUser</param-name>
99      <param-value>tomcat</param-value>
100   </init-param>
101   <init-param>
102     <param-name>remoteKeyStorePass</param-name>
103     <param-value>exampleorg</param-value>
104   </init-param>
105   <init-param>
106     <param-name>AAurl</param-name>
107     <param-value>https://idp.example.org:8443/shibboleth-idp/AA</param-[...]
            value>
108   </init-param>
109   <init-param>
```

```
110     <param-name>PDPurl</param-name>
111     <param-value>https://idp.example.org:8443/PDP/PDP</param-value>
112    </init-param>
113    <init-param>
114     <param-name>logfile</param-name>
115     <param-value>../profile-webapp/log</param-value>
116    </init-param>
117   </servlet>
118
119    <servlet-mapping>
120      <servlet-name>TestServlet</servlet-name>
121      <url-pattern>/TestServlet</url-pattern>
122    </servlet-mapping>
123
124   <servlet-mapping>
125      <servlet-name>ProfileServlet</servlet-name>
126      <url-pattern>/ProfileServlet</url-pattern>
127    </servlet-mapping>
128
129  </web-app>
```

Listing 5.4: Codice Java del SAML PDP.

```
1   package org.openspcoop.sec;
2
3   import java.io.*;
4   import java.util.*;
5
6   import javax.servlet.*;
7   import javax.servlet.http.*;
8   import org.opensaml.*;
9   import org.w3c.dom.*;
10
11
12  /**
13   * Implementa un SAML responder che dispensa decisioni di autorizzazione.
14   * Implementa sia l'interfaccia HttpServlet per le comunicazioni su [...]
         protocollo SAML,
15   * sia l'interfaccia AuthorizationService per agire da web service in [...]
         stile RPC
16   * */
17  @SuppressWarnings("serial")
18   public class PDPServlet extends HttpServlet
19   {
20
21
22    private SAMLBinding binding = null;
23    private PrintWriter writer = null;
24    private SignUtil signUtil = null;
25
26
27    /**
28     * Estrae la <samlp:request> (in formato SAML 1.1) dalla richiesta HTTP[...]
            in POST
29     * @param httpReq, la richiesta HTTP
```

```
30      * @return la richiesta SAML
31      * @throws ServletException in caso di errore SAML
32      */
33     private SAMLRequest
34     getSAMLRequest(HttpServletRequest httpReq)
35      throws SAMLException
36     {
37      SAMLRequest sReq = binding.receive(httpReq, 1);
38
39      return sReq;
40     }
41
42
43     /**
44      * Estrae la richiesta di autorizzazione dalla <samlp:request>
45      * Controlla che esista una tale richiesta
46      * @param sReq la <samlp:request> da processare
47      * @return la richiesta di autorizzazione
48      * @throws ServletException in caso che la <samlp:request> non contenga
49      * una richiesta di autorizzazione
50      */
51     private SAMLAuthorizationDecisionQuery
52     getAuthorizationQuery(SAMLRequest sReq)
53      throws SAMLException
54     {
55      SAMLQuery query = sReq.getQuery();
56      if(!(query instanceof SAMLAuthorizationDecisionQuery)) {
57       throw new SAMLException("la <samlp:request> non contiene una [...]
            richiesta di autorizzazione");
58      }
59
60      return (SAMLAuthorizationDecisionQuery) query;
61     }
62
63
64     /**
65      * Imposta la chiave pubblica e privata nel singleton SAMLConfig
66      * va invocata appena prima di processare richieste e risposte tramite [...]
           la
67      * classe SOAPHTTPBindingProvider
68      */
69     private void
70     initSAMLSingleton()
71     {
72      //inizializzazione del singleton SAMLConfig
73      SAMLConfig config = SAMLConfig.instance();
74
75      //keystore con la chiave privata
76      config.setProperty("org.opensaml.ssl.keystore",
77                getInitParameter("keyStore"));
78      config.setProperty("org.opensaml.ssl.keystore-type",
79                getInitParameter("keyStoreType"));
80      config.setProperty("org.opensaml.ssl.keystore-pwd",
81                getInitParameter("keyStorePass"));
82      config.setProperty("org.opensaml.ssl.key-pwd",
83                getInitParameter("keyPass"));
84
85      //keystore con il certificato
86      config.setProperty("org.opensaml.ssl.truststore",
```

```
87              getInitParameter("trustStore"));
88     config.setProperty("org.opensaml.ssl.truststore-type",
89              getInitParameter("trustStoreType"));
90     config.setProperty("org.opensaml.ssl.truststore-pwd",
91              getInitParameter("trustStorePass"));
92   }
93
94
95   /**
96    * Invia una <samlp:response> al richiedente
97    * @param httpResp, il contesto in cui incapsulare la risposta
98    * @param sResp, la <samlp:response> in formato SAML 1.1
99    */
100  private void
101  sendSAMLResponse(HttpServletResponse httpResp,
102           SAMLResponse sResp)
103  {
104   try {
105    binding.respond(httpResp, sResp, null);
106   } catch(SAMLException e) {
107    log("errore SAML nell'invio della <samlp:response>: " + e);
108   }
109  }
110
111
112  /**
113   * Invia una <samlp:response> contenente un blocco \<status\> con [...]
         dentro l'eccezione
114   * e riscontrata in una delle fasi del processo di gestione della [...]
         richiesta
115   * @param httpResp, il contesto in cui incapsulare la risposta
116   * @param e, l'eccezione SAML da riportare
117   * @param requestId, la richiesta a cui e' correlata questa risposta
118   */
119  private void
120  sendSAMLException(HttpServletResponse httpResp,
121           SAMLException e,
122           String requestId)
123  {
124   try {
125    binding.respond(httpResp,
126             new SAMLResponse(requestId,
127                      null,
128                      null,
129                      e),
130             null);
131   } catch(SAMLException ne) {
132    log("errore SAML nell'invio della <samlp:response> (contenente un [...]
         errore): "
133       + ne);
134   }
135  }
136
137
138  /**
139   * Crea una <samlp:response> a partire da uno statement di [...]
         autorizzazione
140   * @param authZ, lo statement da incapsulare
141   * @param requestId, la richiesta a cui associare questa risposta
```

```
142     * @return la <samlp:response> contenente lo statement (a sua volta in [...]
            una asserzione)
143     * @throws ServletException in caso di errore SAML
144     */
145    private SAMLResponse
146    createSAMLResponse(SAMLAuthorizationDecisionStatement statement,
147              String requestId)
148     throws SAMLException
149    {
150     //la data di emissione
151     Date notBefore = new Date();
152     //la data di scadenza (= data di emissione + Time To Live)
153     Date notOnOrAfter =
154      new Date(notBefore.getTime() +
155           1000*Long.parseLong(getInitParameter("assertions-ttl")));
156
157     Vector<SAMLAuthorizationDecisionStatement> statements =
158      new Vector<SAMLAuthorizationDecisionStatement>();
159     statements.add(statement);
160
161     SAMLAssertion assertion =
162      new SAMLAssertion(getInitParameter("default-issuer"),
163               notBefore,
164               notOnOrAfter,
165               null, //conditions
166               null, //advice
167               statements);
168
169     log("firma");
170     try {
171      signUtil.sign(assertion);
172     } catch (Exception e) {
173      log("errore firma:" + e);
174      throw new SAMLException("impossibile firmare la risposta di [...]
             autorizzazione" + e);
175     }
176
177     Vector<SAMLAssertion> assertions =
178      new Vector<SAMLAssertion>();
179     assertions.add(assertion);
180
181     SAMLResponse resp =
182      new SAMLResponse(requestId, //in response to
183               null, //recipient
184               assertions, //collezione di asserzioni
185               null); //eccezioni da riportare
186
187     return resp;
188    }
189
190
191    /**
192     * Genera l'asserzione corrispondente alla query, in funzione che le [...]
            singole
193     * azioni siano individualmente state autorizzate dal motore di [...]
            decisione
194     * @param query, la richiesta da autorizzare
195     * @return l'asserzione risultante, contenente le asserzioni [...]
            autorizzate,
```

```
196     * o tutte quelle chieste in caso le prime siano state tutte rifiutate
197     * @throws ServletException in caso di errore SAML
198     */
199    private SAMLAuthorizationDecisionStatement
200    processAuthorizationQuery(SAMLAuthorizationDecisionQuery query)
201     throws SAMLException
202    {
203     String subjectName = null;
204     subjectName =
205      resolveSubject(query.getSubject(),
206              query.getResource());
207     if(subjectName == null) {
208      subjectName =
209       new String(query.getSubject().getNameIdentifier().getName().toString[...]
            ());
210     }
211     log("handle " +
212       query.getSubject().getNameIdentifier().getName() +
213       " is " +
214       subjectName);
215
216     String resourceURI = query.getResource();
217     SAMLAuthorizationDecisionStatement statement = null;
218
219     //fornisce al motore di decisione con le asserzioni in push (<evidence[...]
          >)
220     for(Iterator<SAMLAssertion> i = query.getEvidence();
221       i.hasNext();
222       pushCred(i.next()));
223
224     Iterator<SAMLAction> requestedActionsIterator =
225      query.getActions();
226     Vector<SAMLAction> requestedActions =
227      new Vector<SAMLAction>();
228     Vector<SAMLAction> permittedActions =
229      new Vector<SAMLAction>();
230     try {
231      while(requestedActionsIterator.hasNext()) {
232       SAMLAction action =
233        requestedActionsIterator.next();
234       requestedActions.add((SAMLAction)
235                  action.clone());
236       if(isPermitted(subjectName,
237               resourceURI,
238               action.getData())) {
239        permittedActions.add((SAMLAction)
240                   action.clone());
241       }
242      }
243
244      statement =
245       new SAMLAuthorizationDecisionStatement(
246                           (SAMLSubject) query.getSubject().clone(),
247                           resourceURI,
248                           permittedActions.isEmpty()?SAMLDecision.DENY:[...]
                               SAMLDecision.PERMIT,
249                           permittedActions.isEmpty()?requestedActions:[...]
                               permittedActions,
250                           null);//evidence
```

```
251
252     //copia le asserzioni in push nel campo evidence dello statement
253     Iterator i = query.getEvidence();
254     Object entry = null;
255     while(i.hasNext()) {
256      try {
257       entry = i.next();
258       if(entry instanceof SAMLAssertion) {
259        statement.addEvidence(secureClone((SAMLAssertion) entry));
260       } else if(entry instanceof String) {
261        statement.addEvidence(new String((String) entry));
262       } else log("ricevuta una evidence non valida");
263      } catch(SAMLException e) {
264       log("asserzione in <evidence> non corretta: "+e);
265       log("class:" + entry.getClass().getName());
266      }
267     }
268
269    } catch(CloneNotSupportedException c) {
270     throw new SAMLException("impossibile clonare l'oggetto: " + c);
271    }
272
273    return statement;
274   }
275
276
277   /**
278    * Converte un handle in un nameId in chiaro, in forma di stringa
279    * Il nome in chiaro viene ottenuto interrogando la AA e corrisponde
280    * al valore dell'attributo urn:mace:dir:attribute-def:[...]
           eduPersonPrincipalName
281    * @param opaqueSubject, il soggetto da de-anonimizzare
282    * @param resource, la risorsa per il cui accesso per cui si chiedono [...]
           gli attributi
283    * @return nameId in chiaro
284    * @throws SAMLException in caso di errore
285    */
286   private String
287   resolveSubject(SAMLSubject opaqueSubject,
288           String resource)
289    throws SAMLException
290   {
291    SAMLSubject subjectToSend = null; //va copiato
292    try {
293     subjectToSend =
294      (SAMLSubject) opaqueSubject.clone();
295    } catch(CloneNotSupportedException e) {
296     throw new
297      SAMLException("errore nella clone() di un subject: "
298             + e);
299    }
300
301    SAMLAttributeQuery attributeQuery =
302     new SAMLAttributeQuery(subjectToSend,
303                 new String(resource),
304                 null); //attribute designators
305
306    SAMLAttributeDesignator reqAttribute =
307     new SAMLAttributeDesignator(getInitParameter("specialAttribute"), //[...]
```

```
        name
308                 getInitParameter("specialAttributeNameSpace")); //[...]
                        namespace
309
310   attributeQuery.addDesignator(reqAttribute);
311
312   SAMLRequest request =
313    new SAMLRequest(attributeQuery);
314   log("AArequest=\n" + request);
315
316   SAMLResponse response = null;
317   try {
318    response =
319     (SAMLResponse) binding.send(getInitParameter("AAurl"),
320                   request).clone();
321   } catch(Exception e) {
322    throw new
323     SAMLException("errore nella comunicazione con AA: "
324           +e);
325   }
326   log("AAresponse=\n" + response);
327
328   String principalName = null;
329   Iterator assertions =
330    response.getAssertions();
331   if(assertions.hasNext()) {//ci aspettiamo una sola asserzione
332    Iterator statements =
333     ((SAMLAssertion) assertions.next()).getStatements();
334    if(statements.hasNext()) {//ci aspettiamo un solo AttributeStatement
335     SAMLStatement attributeStatement =
336      (SAMLStatement) statements.next();
337     if(attributeStatement instanceof SAMLAttributeStatement) {
338      Iterator attributes =
339       ((SAMLAttributeStatement) attributeStatement).getAttributes();
340      while(attributes.hasNext()) {
341       SAMLAttribute attribute =
342        (SAMLAttribute) attributes.next();
343       if(attribute.getValues().hasNext()) {//ci aspettiamo un solo [...]
             valore
344        if(attribute.getName().equals(getInitParameter("specialAttribute"[...]
             ))) {
345         principalName =
346          attribute.getValues().next().toString();
347        }
348       }
349      }
350     } else throw new
351       SAMLException("statement non valido nella risposta dalla AA");
352    } else throw new
353      SAMLException("nessuno statement nella risposta dalla AA");
354   } else throw new
355     SAMLException("nessuna asserzione nella risposta dalla AA");
356
357   return principalName;
358  }
359
360
361  private void initDecisionEngine()
362  {
```

```
363   }
364
365
366   private void pushCred(SAMLAssertion assertion)
367   {
368   }
369
370
371   private boolean isPermitted(String subject,
372                String resource,
373                String action)
374   {
375    return true;
376   }
377
378
379   /**
380    * Copia un'asserzione SAML preservandone la firma
381    * @param original, l'asserzione da copiare
382    * @return l'asserzione copiata
383    */
384   private SAMLAssertion secureClone(SAMLAssertion original)
385   {
386    try {
387     Element e =
388      (Element) original.toDOM();
389     Element ne =
390      (Element) e.cloneNode(true);
391     SAMLAssertion copy =
392      new SAMLAssertion(ne);
393     return copy;
394    } catch(SAMLException err) {
395     log("secureClone(): " + err);
396     return null;
397    }
398   }
399
400
401   @Override
402   protected void
403   doPost(HttpServletRequest httpReq,
404       HttpServletResponse httpResp)
405    throws ServletException, IOException
406   {
407    String requestId = null;
408
409    log("inizio invocazione");
410
411    //decisore senza stato, lifetime == invocazione
412    initDecisionEngine();
413
414    SAMLResponse sResp =null;
415    try {
416     SAMLRequest sReq =
417      getSAMLRequest(httpReq);
418     requestId = sReq.getId();
419     SAMLAuthorizationDecisionQuery aQuery =
420      getAuthorizationQuery(sReq);
421     SAMLAuthorizationDecisionStatement statement =
```

```
422       processAuthorizationQuery(aQuery);
423      sResp =
424       createSAMLResponse(statement,
425                 requestId);
426     } catch(SAMLException e) {
427      log("errore nel processamento della richiesta: " +
428        e +
429        "\n",
430        e);
431      sendSAMLException(httpResp, e, requestId);
432     } finally {
433      log("invio");
434      sendSAMLResponse(httpResp, sResp);
435     }
436
437     log("fine invocazione");
438
439    }
440
441
442    @Override
443    public void log(String message)
444    {
445     writer.println("[" + new Date() + "] " + message);
446    }
447
448
449    public void log(String message,
450            Throwable e)
451    {
452     writer.println("[" + new Date() + "] " + message);
453     e.printStackTrace(writer);
454    }
455
456
457    @Override
458    public void init(ServletConfig config)
459     throws ServletException
460    {
461     super.init(config);
462
463     signUtil =
464      new SignUtil(getInitParameter("keyStore"),
465             getInitParameter("keyStoreType"),
466             getInitParameter("keyStorePass"),
467             getInitParameter("keyPass"),
468             getInitParameter("keyStoreUser"));
469
470     try {
471      writer = new PrintWriter(new BufferedWriter(new FileWriter(...
          getInitParameter("logfile"), true)), true);
472     } catch(IOException e) {}
473     log("inizializzazione");
474
475     //creazione del binding SAML-over-SOAP-over-HTTPS
476     initSAMLSingleton();
477     try {
478      binding = SAMLBindingFactory.getInstance(SAMLBinding.SOAP);
479     } catch(NoSuchProviderException e) {
```

```
480     throw new
481      ServletException("errore nella creazione del binding " +
482               e);
483    }
484
485   }
486
487
488   @Override
489   public void destroy()
490   {
491    log("rimozione");
492    super.destroy();
493   }
494
495  }
```

Listing 5.5: File di configurazione di Tomcat per il PDP SAML.

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2
3  <!DOCTYPE web-app PUBLIC "-//Sun␣Microsystems,␣Inc.//DTD␣Web␣Application␣[...]
       2.3//EN" "http://java.sun.com/dtd/web-app_2_3.dtd">
4
5  <web-app>
6
7   <display-name>SAML PDP</display-name>
8    <description>
9    Implementazione di un SAML responder che processa richieste di [...]
         autorizzazione
10   </description>
11
12  <servlet>
13   <servlet-name>PDPServlet</servlet-name>
14   <display-name>PDP Servlet</display-name>
15   <description>
16    Implementazione di un SAML responder che processa richieste di [...]
         autorizzazione
17   </description>
18   <servlet-class>PDPServlet</servlet-class>
19
20   <init-param>
21    <param-name>defaultNameQualifier</param-name>
22    <param-value>https://idp.example.org/shibboleth</param-value>
23   </init-param>
24   <!--
25      ATTENZIONE al momento supponiamo di avere gli stessi server cert
26      dell''IdP
27    -->
28   <init-param>
29    <param-name>keyStore</param-name>
30    <param-value>../PDP/keys/idp-example.jks</param-value>
31   </init-param>
32   <init-param>
33    <param-name>keyStoreType</param-name>
```

```
34      <param-value>JKS</param-value>
35    </init-param>
36    <init-param>
37      <param-name>keyStoreUser</param-name>
38      <param-value>tomcat</param-value>
39    </init-param>
40    <init-param>
41      <param-name>keyStorePass</param-name>
42      <param-value>exampleorg</param-value>
43    </init-param>
44    <init-param>
45      <param-name>keyPass</param-name>
46      <param-value>exampleorg</param-value>
47    </init-param>
48    <!--
49       ATTENZIONE al momento supponiamo che tutti i SAML requesters
50       abbiano lo stesso server cert del Target (SP)
51      -->
52    <init-param>
53      <param-name>trustStore</param-name>
54      <param-value>
55        ../PDP/keys/idp-example.jks
56      </param-value>
57    </init-param>
58    <init-param>
59      <param-name>trustStoreType</param-name>
60      <param-value>JKS</param-value>
61    </init-param>
62    <init-param>
63      <param-name>trustStoreUser</param-name>
64      <param-value>tomcat</param-value>
65    </init-param>
66    <init-param>
67      <param-name>trustStorePass</param-name>
68      <param-value>exampleorg</param-value>
69    </init-param>
70    <init-param>
71      <param-name>trustPass</param-name>
72      <param-value>exampleorg</param-value>
73    </init-param>
74    <init-param>
75      <param-name>AAurl</param-name>
76      <param-value>
77        https://idp.example.org:8443/shibboleth-idp/AA
78      </param-value>
79    </init-param>
80    <init-param>
81      <param-name>default-issuer</param-name>
82      <param-value>
83        https://idp.example.org/shibboleth
84      </param-value>
85    </init-param>
86    <init-param>
87      <param-name>assertions-ttl</param-name>
88      <param-value>3600</param-value>
89    </init-param>
90    <init-param>
91      <param-name>logfile</param-name>
92      <param-value>../PDP/log</param-value>
```

```
93    </init-param>
94    <init-param>
95     <param-name>specialAttribute</param-name>
96     <param-value>
97      urn:mace:dir:attribute-def:eduPersonPrincipalName
98     </param-value>
99    </init-param>
100   <init-param>
101    <param-name>specialAttributeNameSpace</param-name>
102    <param-value>
103     urn:mace:shibboleth:1.0:attributeNamespace:uri
104    </param-value>
105   </init-param>
106  </servlet>
107
108  <servlet-mapping>
109     <servlet-name>PDPServlet</servlet-name>
110     <url-pattern>/PDP</url-pattern>
111   </servlet-mapping>
112
113 </web-app>
```

Listing 5.6: Classe Java di utilità per firmare oggetti OpenSAML.

```
1  package org.openspcoop.sec;
2
3  import java.io.FileInputStream;
4  import java.io.IOException;
5  import java.security.KeyStore;
6  import java.security.KeyStoreException;
7  import java.security.NoSuchAlgorithmException;
8  import java.security.PrivateKey;
9  import java.security.UnrecoverableKeyException;
10 import java.security.cert.Certificate;
11 import java.security.cert.CertificateException;
12 import java.security.cert.X509Certificate;
13 import java.util.Arrays;
14
15 import org.apache.xml.security.signature.XMLSignature;
16 import org.opensaml.SAMLException;
17 import org.opensaml.SAMLSignedObject;
18
19
20 /**
21  * Classe di utilita' per firmare oggetti SAML
22  */
23 public class SignUtil {
24
25
26  private String keyStorePath = null;
27  private String keyStoreType = null;
28  private String keyStorePass = null;
29  private String keyPass = null;
30  private String keyStoreUser = null;
31
```

```
32
33  public
34  SignUtil(String path,
35       String type,
36       String privatePass,
37       String pass,
38       String user)
39  {
40   super();
41
42   keyPass = privatePass;
43   keyStorePass = pass;
44   keyStorePath = path;
45   keyStoreType = type;
46   keyStoreUser = user;
47  }
48
49
50  /**
51   * Effettua la XML Digital Signature di un qualsiasi blocco SAML [...]
         firmabile
52   * Utilizza la chiave privata del mittente
53   * Include il certificato del mittente
54   * @param toSign, il blocco da firmare
55   * @return lo stesso blocco dopo la firma
56   * @throws Exception in caso di eccezioni catturate
57   */
58  public void
59  sign(SAMLSignedObject toSign)
60   throws Exception
61  {
62   //recupero delle chiavi e dei certificati dai Java Keystores
63   //forniti nella configurazione della webapp
64
65
66   if(!keyStoreType.equals("JKS"))
67    throw new
68     Exception("formato del keystore non supportato");
69   try {
70
71    KeyStore localKeyStore =
72     KeyStore.getInstance(keyStoreType);
73
74    localKeyStore.load(new FileInputStream(keyStorePath),
75              keyStorePass.toCharArray());
76
77    PrivateKey localPrivateKey =
78     (PrivateKey) localKeyStore.getKey(keyStoreUser,
79                      keyPass.toCharArray());
80    if (localPrivateKey == null)
81     throw new
82      Exception("nessuna chiave privata trovata nel Java KeyStore");
83
84    Certificate[] certificates =
85     localKeyStore.getCertificateChain(keyStoreUser);
86    if (certificates == null)
87     throw new
88      Exception("nessun certificato trovato nel Java KeyStore");
89
```

```
90     X509Certificate[] x509Certs =
91      new X509Certificate[certificates.length];
92     for (int i = 0; i < certificates.length; i++) {
93      if (certificates[i] instanceof X509Certificate) {
94       x509Certs[i] = (X509Certificate) certificates[i];
95      } else {
96       throw new
97        Exception("solo i certificati di tipo X509 sono supportati");
98      }
99     }
100
101
102    //firma vera e propria (usando OpenSAML)
103
104    toSign.sign(XMLSignature.ALGO_ID_SIGNATURE_RSA_SHA1,
105          localPrivateKey,
106          Arrays.asList(x509Certs));
107
108   } catch(KeyStoreException kse) {
109    throw new
110     Exception("errore nell'accesso al Java KeyStore: "
111          + kse);
112   } catch(NoSuchAlgorithmException nsae) {
113    throw new
114     Exception("impossibile trovare l'appropriato provider JCE: "
115          + nsae);
116   } catch(CertificateException ce) {
117    throw new
118     Exception("impossibile caricare il certificato dal Java KeyStore: "
119      + ce);
120   } catch(IOException ioe) {
121    throw new
122     Exception("errore in lettura del Java KeyStore: "
123          + ioe);
124   } catch(UnrecoverableKeyException uke) {
125    throw new
126     Exception("impossibile leggere la chiave privata dal Java KeyStore: "
127          + uke);
128   } catch(SAMLException se) {
129    throw new
130     Exception("errore nell'esecuzione della firma: "
131          + se);
132   }
133
134  }
135
136 }
```