POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea Magistrale

Realizzazione di un Attribute Provider eIDAS con Shibboleth

Relatore
prof. Antonio Lioy

Candidato
Daniele Pellone

Anno accademico 2016-2017

Alla mia famiglia

Sommario

Il presente lavoro analizza la possibilità di estendere il sistema di identità digitale europeo eIDAS attraverso la realizzazione di un Attribute Provider mediante il software di identità federata Shibboleth. Verranno studiati gli strumenti messi a disposizione dal sistema eIDAS per permettere l'integrazione tra i diversi sistemi di identità digitale dei paesi dell'Unione Europea, con particolare attenzione al sistema eID italiano, SPID; saranno poi valutati altri sistemi di gestione degli attributi, come Shibboleth. Verranno illustrate alcune possibili soluzioni alternative per includere degli Attribute Provider all'interno della rete eIDAS, analizzandone vantaggi e limiti e presentando anche un esempio di implementazione per una di esse. In particolare, saranno analizzati i possibili rischi che tali soluzioni comportano per la sicurezza delle informazioni e le tecniche per eliminare o attenuare tali problematiche. Infine, saranno presentati possibili miglioramenti ed estensioni al sistema modificato.

Indice

Introduzione

Negli ultimi dieci anni, in Europa, vi è stata una rapida digitalizzazione dei servizi offerti da istituzioni sia pubbliche che private; questo ha portato la necessità di avere un sistema di autenticazione riconosciuto a livello nazionale nei diversi Paesi, che consentisse di identificare i cittadini in modo facile e sicuro. Nell'ambito della sempre più stretta collaborazione tra gli Stati membri dell'Unione Europea sono stati varati alcuni progetti per favorire l'interoperabilità tra i diversi sistemi di identità digitale. L'ultimo in ordine di tempo è il regolamento eIDAS (*electronic IDentification Authentication and Signature*) promulgato nel 2014 dal Parlamento Europeo [1], che fissa una base normativa comune per i servizi fiduciari e i mezzi di identificazione elettronica degli Stati membri. Per raggiungere il suo scopo, il regolamento prevede la creazione di una rete di nodi che permettano ai cittadini europei l'uso della propria identità digitale per accedere a servizi che si trovano in Stati diversi dal proprio.

Allo stato attuale, il sistema consente ad un fornitore di servizi (*Service Provider* o *SP*) di ottenere informazioni sull'identità di un utente straniero tramite una richiesta inoltrata attraverso la rete eIDAS e destinata all'ente (*Identity Provider* o *IdP*) presso cui l'utente si è registrato; quest'ultimo dovrà, quindi, autenticarsi secondo le procedure di tale IdP. Se l'autenticazione riesce, nella risposta inviata al SP vengono incluse, se disponibili, le informazioni richieste sotto forma di attributi.

Un esempio di sistema di identità digitale nazionale è rappresentato da SPID, il Sistema Pubblico di Identità Digitale promosso dal governo italiano; attraverso la registrazione presso uno degli IdP autorizzati, i cittadini possono ottenere delle credenziali in grado di fornire accesso ad una vasta gamma di servizi, in particolare nell'ambito della pubblica amministrazione. Gli utenti italiani possono usare le loro credenziali SPID anche per accedere a servizi in altri paesi dell'Unione Europea, grazie al progetto FICEP (First Italian Crossborder eIDAS Proxy) per la realizzazione del primo nodo eIDAS italiano, che consentirà anche ai cittadini europei di usare le loro identità digitali nazionali per utilizzare i servizi italiani.

Prima dell'introduzione del regolamento eIDAS sono stati sviluppati alcuni progetti pilota, finanziati dall'Unione Europea, il cui scopo era di rendere interoperabili i sistemi di identità digitale degli Stati europei; in particolare va ricordato il progetto STORK (*Secure idenTity acrOss boRders linKed*) e la sua evoluzione STORK 2.0. Queste sperimentazioni hanno fornito la base per l'architettura del network eIDAS basata su nodi nazionali. L'infrastruttura STORK, sebbene facente parte di un progetto pilota, risulta ancora essere in parte utilizzata e, per questo, nell'ambito del progetto e-SENS (*electronic Simple European Networked Services*), è stato sviluppato un plug-in che consente di includere i nodi STORK nella rete eIDAS.

I sistemi di identità digitale dei Paesi dell'Unione usano diversi insiemi di attributi, pertanto il network eIDAS fornisce un'infrastruttura capace di fare da tramite e di tradurre le richieste e le risposte usando un set di attributi standard; tuttavia, in alcuni casi

esso potrebbe non essere sufficiente, in quanto il SP potrebbe aver bisogno di informazioni aggiuntive sull'utente per fornire il servizio richiesto. Si pensi all'ammissione di uno studente italiano ad un corso di dottorato di ricerca in un'università straniera: in questo caso tale università è interessata ad avere informazioni sulla carriera accademica dello studente, per poter valutare i requisiti di ammissione a tale corso. Queste informazioni non possono essere fornite direttamente dall'IdP, perché sono mantenute da enti diversi, in questo caso l'università presso cui lo studente si è laureato.

Per far fronte a tali situazioni, è possibile estendere l'infrastruttura inserendo delle entità capaci di fornire informazioni aggiuntive su un utente già autenticato, che prendono il nome di *Attribute Provider* (*AP*): essi verrebbero contattati nel caso in cui ci si renda conto che l'IdP non è stato in grado di fornire tutti gli attributi richiesti. Il regolamento eIDAS prevede la possibilità, per uno Stato membro, di integrare le informazioni fornite dal suo schema di identità digitale con quelle provenienti da uno o più AP, tuttavia non specifica come tale integrazione debba avvenire, garantendo la massima flessibilità tra le diverse soluzioni che è possibile adottare. Anche nell'ambito del progetto STORK 2.0 è stato affrontato il problema della progettazione e inclusione di AP all'interno del network [2]; i risultati di tali studi possono essere usati come punto di partenza per lo sviluppo di soluzioni simili per la rete eIDAS.

Questa tesi mira ad analizzare i diversi metodi per includere uno o più AP nel processo di autenticazione mediante la rete eIDAS, valutando vantaggi e svantaggi dei diversi approcci, cercando di ridurre al minimo gli interventi sull'infrastruttura già esistente e attendendosi, per quanto possibile, ai protocolli attualmente in uso. Nel Capitolo 1 verrà analizzato lo schema di autenticazione usato dal servizio eID eIDAS, esaminando il sistema SPID come esempio concreto di un sistema di identità digitale nazionale. Nel Capitolo 2 sarà valutato Shibboleth, il sistema di identità federata che verrà utilizzato per implementare l'AP da integrare nel sistema: sarà valutato il suo funzionamento, confrontandolo con altri software per la gestione di attributi. Nel Capitolo 3 si analizzeranno in dettaglio le problematiche e le soluzioni relative all'estensione della rete eIDAS con uno o più AP: quali sono i possibili schemi di comunicazione tra i diversi attori in gioco, a quali possibili rischi per la sicurezza e la riservatezza delle informazioni si va incontro e quali misure adottare per mitigarli; verrà quindi presentata una soluzione la cui implementazione sarà oggetto del Capitolo 4, in cui sarà fornita una descrizione approfondita del codice sviluppato. Nel Capitolo 5 saranno presenti informazioni e istruzioni su come installare ed usare il software sviluppato; infine, nel Capitolo 6 saranno presentate le conclusioni del lavoro svolto e alcuni spunti per ulteriori miglioramenti e sviluppi.

Capitolo 1

Il servizio eID eIDAS

1.1 Concetti generali

Il regolamento eIDAS è stato introdotto dal Consiglio dell'Unione Europea al fine di fornire una base normativa comune per le interazioni digitali sicure tra cittadini, imprese e pubbliche amministrazioni dei diversi Stati europei. Il Consiglio ha scelto di affrontare questa materia delicata attraverso lo strumento del regolamento: a differenza delle direttive, che lasciano ai singoli Stati la libertà di definire le disposizioni per raggiungere gli obiettivi fissati, un regolamento è un atto legislativo vincolante che deve essere applicato in tutte le sue parti nell'intera Unione Europea. Il regolamento eIDAS stabilisce le norme relative ad alcuni strumenti utilizzati nell'ambito di tali interazioni, tra cui firme e validazioni temporali elettroniche e certificati di autenticazione. Un aspetto fondamentale nell'erogazione di servizi di questo tipo è l'identificazione certa delle parti in causa; il SP e l'utente devono fornire l'un l'altro una prova della propria identità, tipicamente attraverso una terza parte fidata che la certifichi. Per garantire l'identità del SP viene di solito utilizzato un certificato firmato da un'autorità fidata, tramite una PKI (Public Key Infrastructure), mentre esistono diversi meccanismi di autenticazione per i singoli utenti, il più comune dei quali consiste nell'uso di una password associata al nome utente. Negli anni i singoli Stati si sono adoperati per progettare e costruire sistemi in grado di fornire ai cittadini un'identità digitale che consentisse loro di accedere ad un numero sempre crescente di servizi sia pubblici che privati. Questo lavoro è proseguito per diverso tempo in modo parallelo e disgiunto nei diversi Paesi, portando alla creazione di sistemi differenti e tra loro incompatibili sia sul piano tecnico che normativo. Per questo motivo un punto centrale del regolamento eIDAS consiste nell'equiparare i mezzi di identificazione elettronica dei diversi Stati che soddisfano alcuni requisiti e nel garantire che tali mezzi siano interoperabili ([1, artt. 6 e 12]).

Una componente importante di cui i legislatori hanno tenuto conto per uniformare i diversi regimi di identificazione sono i "livelli di garanzia" offerti da questi ultimi [3, art. 1]: essi definiscono diversi gradi di sicurezza riguardo all'identità dichiarata di una persona da parte del sistema di identità digitale. Tutti i regimi di identificazione facenti parte della rete eIDAS devono specificare almeno uno dei tre livelli previsti (basso, significativo, elevato). Ognuno di questi livelli richiede l'impiego di tecniche e procedure che riducano il rischio di uso abusivo o alterazione dell'identità, che deve essere tanto più basso quanto più è alto il livello di garanzia. L'uso di tali livelli consente di equiparare identità digitali generate da regimi diversi ma che abbiano lo stesso livello di garanzia, ovvero garantiscano lo stesso grado di sicurezza. Il regolamento eIDAS [3, allegato] stabilisce procedure e specifiche

tecniche richieste per specificare ed implementare i livelli di garanzia; esse riguardano diversi aspetti del ciclo di vita delle identità elettroniche, tra cui la loro emissione, la gestione dei mezzi di identificazione e l'autenticazione mediante questi ultimi.

Tra i diversi parametri specificati dal regolamento vi è il meccanismo di autenticazione scelto dal sistema di identità digitale: un esempio di ciò può essere riscontrato in SPID, che definisce tre diverse modalità di accesso, ognuna con un grado si sicurezza diverso. Il primo livello prevede un sistema di autenticazione a singolo fattore (password associata all'ID), che comporta un rischio moderato in relazione alla complessità della password e alla possibilità che questa venga compromessa. Il secondo livello richiede un sistema di autenticazione a due fattori attraverso l'uso di One Time Password (OTP), ovvero codici da usare una sola volta e che tipicamente vengono generati attraverso un token o un'applicazione per smartphone: l'uso di un secondo codice rende il sistema più robusto in caso di compromissione della password, anche se permangono alcuni rischi in caso di furto o violazione del dispositivo di generazione di OTP, in particolar modo se si tratta di una semplice app per smartphone. Il terzo livello consiste nell'impiegare, come secondo fattore di autenticazione, certificati digitali, le cui chiavi private devono essere opportunamente protette e memorizzate su dispositivi che ne garantiscano la riservatezza; tale livello consente di ridurre al minimo i rischi legati all'uso dell'identità digitale, fintanto che l'utente mantiene il controllo della chiave privata.

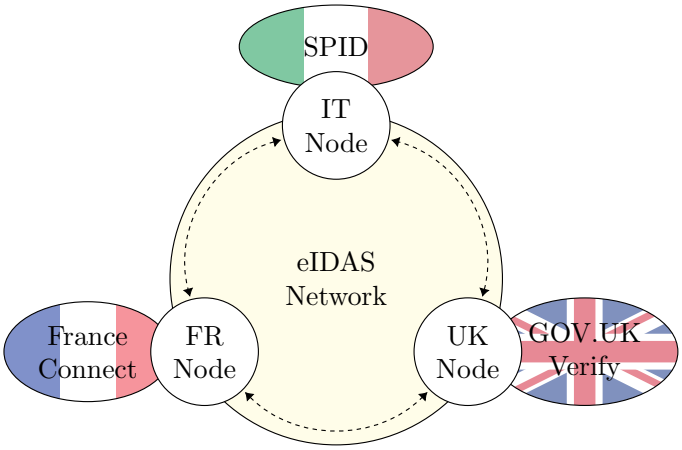


Figura 1.1: Esempio di interoperabilità mediante nodi

L'interoperabilità dei diversi schemi di identità digitale è oggetto del Regolamento di esecuzione (EU) No 2015/1501 [4], che stabilisce anche l'insieme minimo di dati per l'identificazione. In questo documento viene definito il nodo eIDAS come l'elemento base dell'architettura di interoperabilità, in grado di interfacciarsi sia con l'infrastruttura di identificazione elettronica di uno Stato che con gli altri nodi della rete [4, art. 2]; questo gli consente di fare da tramite fra un particolare schema e gli altri. Attraverso la connessione di questi nodi si vene a creare un rete che consente di tradurre le informazioni di autenticazione da uno schema all'altro senza necessità di intervenire direttamente su tali sistemi (Figura 1.1).

La normativa prevede che i nodi siano in grado di trattare degli insiemi minimi di attributi per le persone fisiche e quelle giuridiche (Tabella 1.1) Alcuni di questi attributi sono obbligatori per poter identificare correttamente un utente, mentre altri possono essere forniti se richiesti dal SP, previa autorizzazione dell'utente. Il regolamento eIDAS consente agli schemi di identità digitale di esporre anche attributi diversi da quelli previsti, la

Obbligatori	*Opzionali*
Cognome attuale	Cognome alla nascita
Nome attuale	Nome alla nascita
Data di nascita	Luogo di nascita
Identificativo univoco	Indirizzo attuale
	Sesso

(a) Persona fisica

Obbligatori	*Opzionali*
Ragione sociale	Indirizzo attuale
Identificativo univoco	Numero di partita IVA
	Numero di registrazione fiscale
	Identificativo direttiva 2012/17/EU
	Altri codici (LEI, EORI, SEED, SIC)

(b) Persona giuridica

Tabella 1.1: Insiemi minimi di dati per persone fisiche e giuridiche [4, art. 11 e allegato]

cui interoperabilità, tuttavia, non è garantita e dipende dalla possibilità di tradurre tali attributi nello schema eID del Paese in cui si trova il SP.

1.2 Dettagli implementativi

Al fine di definire le specifiche tecniche della rete eIDAS è stata creata una rete di cooperazione che riunisce i soggetti dei singoli Stati deputati allo sviluppo di tale infrastruttura. Le specifiche vengono elaborate congiuntamente con la Commissione europea e pubblicate sotto forma di pareri dalla rete di cooperazione; viene fornita, inoltre, un'implementazione di esempio del nodo eIDAS, che gli Stati possono adottare dopo averla modificata. Nel seguito si farà riferimento alla versione 1.1 delle specifiche, approvata dal parere No 2/2016 della rete di cooperazione il 16 Dicembre 2016.

Nelle specifiche di interoperabilità [5, sezione 1.1] i nodi eIDAS finora menzionati vengono separati in due moduli distinti: il modulo che dialoga con il SP viene detto *eIDAS-Connector*, mentre quello che si interfaccia con lo schema di identità digitale di uno specifico Stato viene chiamato *eIDAS-Service*. Per ogni Paese possono esistere diversi eIDAS-Connector, alcuni dei quali gestiti direttamente da SP. L'implementazione degli eIDAS-Service è a carico degli Stati, che possono adottare due possibili soluzioni: nello scenario *Proxy-based* lo Stato gestisce un proprio eIDAS-Service, eventualmente in combinazione con un eIDAS-Connector, che si occupa delle richieste di autenticazione provenienti dai nodi degli altri Paesi. Nel secondo scenario, lo Stato non opera nessun eIDAS-Service, ma sviluppa un *Middleware* che sarà fornito agli altri Paesi e incluso nei loro eIDAS-Connector, i quali potranno comunicare direttamente con lo schema eID del primo.

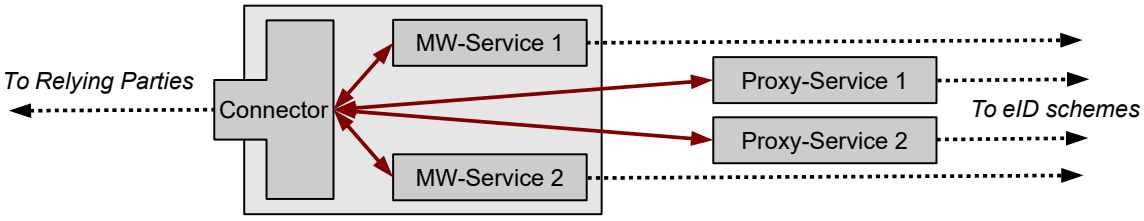


Figura 1.2: Interfacce dei nodi eIDAS (fonte: Interoperability Architecture [5, fig. 3])

La comunicazione tra i nodi avviene utilizzando il linguaggio SAML 2.0 (Security Assertion Markup Language) [5, sezione 2.3], un linguaggio derivato da XML che consente lo scambio di informazioni di autenticazione mediante l'uso di messaggi contenenti asserzioni, ossia affermazioni sul soggetto da identificare. Per garantire che le autenticazioni inoltrate attraverso la rete siano fidate è necessario che ogni nodo sia in grado di identificare correttamente e in modo sicuro gli altri prima di trasmettere o ricevere informazioni. I dettagli relativi ai nodi della rete vengono distribuiti usando documenti XML chiamati Metadati SAML [5, sezione 2.4]; questi devono essere disponibili pubblicamente attraverso una URL HTTPS presso tutti gli eIDAS-Connector e gli eIDAS-Service di tipo Proxy. I Metadati dei Middleware vengono forniti direttamente dagli Stati insieme a questi ultimi. L'autenticità e l'integrità dei Metadati devono essere garantite mediante l'uso di una firma digitale e certificati a chiave pubblica secondo l'RFC-5280 [6]: ogni Stato deve rilasciare un certificato associato ad una chiave privata mantenuta dallo Stato stesso, che sarà usata per firmare direttamente i Metadati oppure come radice di una PKI usata per firmarli [5, sezione 6]. L'utilizzo di certificati firmati non è richiesto per i Middleware, in quanto essi vengono scambiati direttamente tra gli Stati. I Metadati devono contenere le URL degli end-point per l'invio e la ricezione dei messaggi, i certificati delle chiavi utilizzate dal nodo in questione per la cifratura e la firma dei messaggi e i relativi algoritmi supportati, il massimo livello di garanzia fornito e la lista di attributi supportati.

Le specifiche tecniche richiedono che tutti i messaggi SAML scambiati tra i nodi debbano essere firmati [5, sezione 3.2.1 e 3.2.2]; le funzioni e gli algoritmi da usare per la firma dei messaggi, così come per quella dei Metadati, sono specificati nel documento relativo ai requisiti crittografici [7]. Viene richiesto l'uso della funzione di hash SHA-2 con output da almeno 256 bit vietando l'uso di altre funzioni [7, sezione 3.1.1], mentre gli algoritmi di firma supportati sono RSASSA-PSS con chiave da almeno 3072 bit e DSA con curve ellittiche e chiave da almeno 256 bit [7, sezione 3.3]. Le asserzioni contenute all'interno delle risposte SAML possono essere a loro volta firmate, anche se questo non è obbligatorio; devono, invece, essere cifrate, per proteggere la confidenzialità delle informazioni contenute al loro interno sotto forma di attributi. La cifratura delle asserzioni avviene tramite crittografia simmetrica con l'algoritmo AES-GCM usando una chiave lunga 128, 192 o 256 bit, generata in maniera casuale per ogni trasmissione; la chiave deve essere poi cifrata usando l'algoritmo RSA e la chiave pubblica contenuta nel certificato del ricevente. In alternativa, può essere usato l'algoritmo Diffie-Hellman effimero con curve ellittiche (ECDHE) per concordare la chiave di cifratura simmetrica [7, sezione 3.2].

1.3 Comunicazione tra i nodi

La comunicazione tra i diversi attori del sistema eIDAS avviene indirettamente attraverso l'utente; i messaggi SAML, infatti, vengono trasportati all'interno di richieste HTTP effettuate dal browser di quest'ultimo. Il protocollo SAML prevede diverse modalità di trasporto dei messaggi, chiamate *binding*; il sistema eIDAS richiede che le richieste di autenticazione siano trasportare mediante HTTP Redirect come *query parameter* all'interno dell'URI oppure tramite HTTP POST, mentre le risposte contenenti asserzioni devono utilizzare il binding HTTP POST [5, sezione 3.2]. In entrambi i casi la richiesta viene codificata in base64 per permetterne l'utilizzo in un protocollo testuale come HTTP.

Tutte le comunicazioni tra utente e nodi devono avvenire utilizzando il protocollo HTTPS, ovvero messaggi HTTP incapsulati all'interno di una connessione TLS [7, sezione 2]. Per mitigare i rischi legati ad una serie di attacchi possibili contro le varie versioni di TLS, le specifiche richiedono che venga usata le versione 1.2 del protocollo oppure, solo se

quest'ultima non dovesse essere disponibile nel browser dell'utente, la versione 1.1. Viene richiesto che i nodi debbano supportare solo cipher suite che forniscono *perfect forward secrecy*, per evitare che la violazione di una chiave comprometta tutte le comunicazioni del nodo interessato. Nelle specifiche viene indicata una lista di cipher suite raccomandate: tutte prevedono come algoritmo di cifratura l'utilizzo di AES-CBC o AES-GCM con chiave da 128 o 256 bit, mentre come funzione di hash viene indicata SHA-2, anche se in mancanza di supporto da parte del browser dell'utente può essere adoperata la funzione SHA-1. Per garantire l'identità del nodo eIDAS all'utente, dal 2018 i certificati utilizzati per negoziare il canale TLS dovranno essere *qualificati*, ovvero rispettare i requisiti indicati nel regolamento [1, allegato IV]: dovranno indicare tutte le informazioni che permettano di identificare sia nodo che l'ente responsabile della sua gestione.

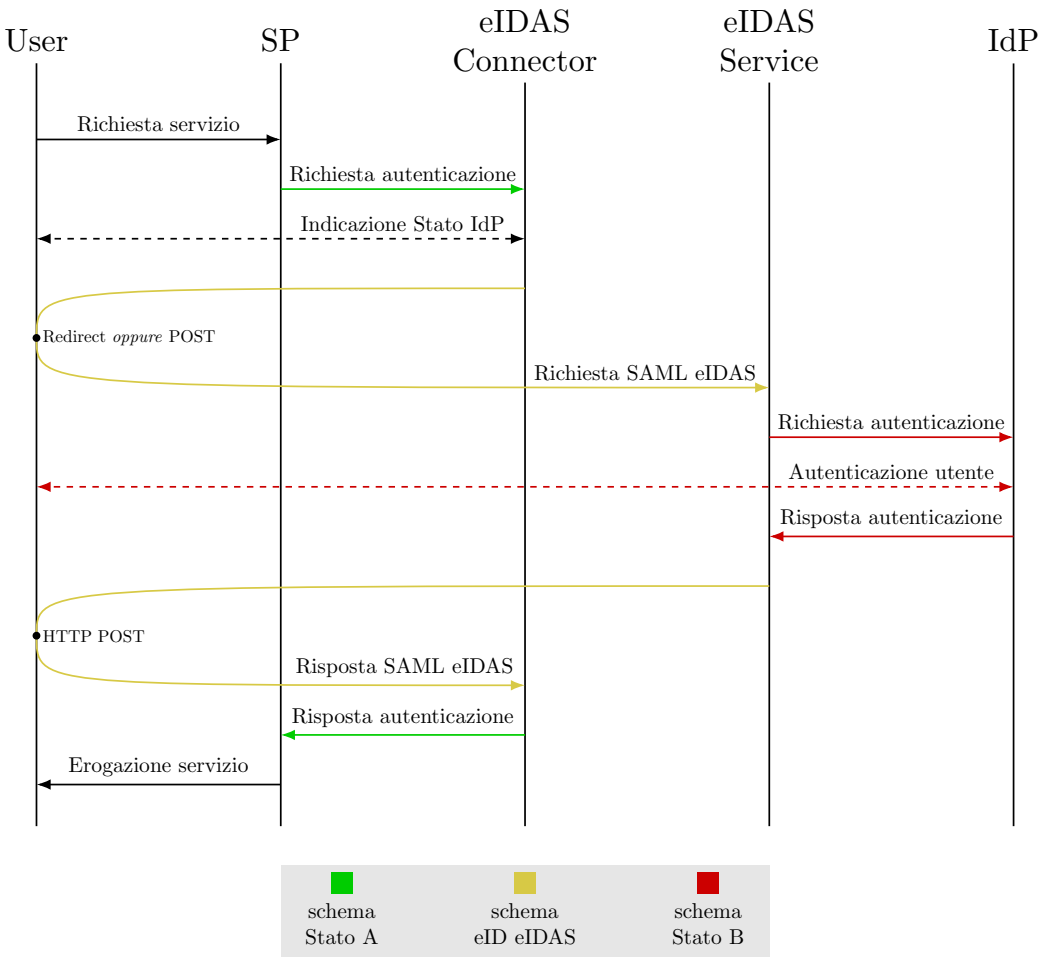


Figura 1.3: Sequenza dello scambio di messaggi nel sistema eID eIDAS

La Figura 1.3 mostra un esempio di identificazione di un utente, cittadino dello stato B, che desidera accedere ad un servizio nello stato A; in questo esempio si suppone che l'eIDAS-Service dello stato B sia di tipo Proxy, in quanto è il più diffuso, e che l'autenticazione vada a buon fine. Le specifiche descrivono dettagliatamente il protocollo [5, sezione 5] ed il formato dei messaggi [8, sezione 2] da utilizzare per le comunicazioni tra i nodi, ma non forniscono alcuna indicazione in merito alla comunicazione tra l'eIDAS-Connector e il SP né tra l'eIDAS-Service e lo schema di identità digitale dell'utente: lo sviluppo di questi protocolli è di competenza dei singoli Stati, che dovranno anche implementare le interfacce adatte per i nodi eIDAS.

Nel momento in cui un eIDAS-Connector riceve una richiesta di identificazione da parte di un SP, deve fornire all'utente una lista di schemi eID tra i quali scegliere quello presso cui desidera autenticarsi, a meno che tale indicazione non sia contenuta nella richiesta del SP; la lista rappresenta l'insieme di eIDAS-Service di cui il nodo può recuperare Metadati validi e che quindi è autorizzato a contattare. Per permettere ai nodi di poter recuperare e verificare tali Metadati, ad ognuno di essi vengono forniti i certificati radice dei diversi Stati e le URL da contattare per ottenere i Metadati. L'eIDAS-Connector genera un messaggio SAML di tipo `AuthnRequest` indirizzato all'eIDAS-Service scelto, includendo la lista degli attributi richiesti e il livello di garanzia minimo necessario. La richiesta viene quindi firmata con la chiave privata dell'eIDAS-Connector e inviata tramite il browser dell'utente: se il messaggio è abbastanza piccolo, le specifiche suggeriscono di usare il binding HTTP Redirect, in quanto generalmente è più veloce di quello POST. L'eIDAS-Service che riceve il messaggio deve validarlo recuperando i Metadati del nodo mittente, verificando che siano firmati dalla PKI dello Stato che lo gestisce e usando i certificati contenuti al loro interno per verificare la firma del messaggio; se uno qualsiasi di questi passaggi fallisce, il messaggio non può essere autenticato e deve essere ignorato. Nel caso in cui la verifica avvenga correttamente, l'eIDAS-Service inoltra la richiesta di autenticazione allo schema di identità digitale del Paese, dove l'utente dovrà autenticarsi.

L'esito dell'autenticazione verrà notificato all'eIDAS-Service che, in ogni caso, preparerà una risposta attraverso un messaggio di tipo SAML `Response`: se l'autenticazione è fallita o se ci sono altri tipi di errore, come la mancanza di attributi indicati come obbligatori, l'elemento `Status` conterrà un codice che descrive il problema che si è verificato. In caso di avvenuta autenticazione, verrà costruita una asserzione contenete l'identità del utente e gli attributi richiesti; l'eIDAS-Service ha la facoltà di firmare l'asserzione, anche se non è obbligatorio. Questa sarà poi cifrata usando uno degli algoritmi previsti e inserita all'interno della risposta, che sarà poi firmata. L'utente, attraverso l'uso di una richiesta HTTP POST, inoltrerà il messaggio all'eIDAS-Connector, il quale provvederà a verificarne l'autenticità utilizzando i certificati presenti nei Metadati dell'eIDAS-Service per validare le firme. Le risposte non validate o per cui non era stata precedentemente inviata una richiesta non devono essere accettate e devono essere ignorate. Se tutte le verifiche vanno a buon fine, l'asserzione viene decifrata e l'autenticazione dell'utente viene inoltrata al SP, insieme con gli attributi recuperati.

1.4 Analisi del sistema

Osservando le specifiche e il protocollo utilizzato si apprezzano i due obiettivi che il sistema eID eIDAS si prefigge: garantire una catena di fiducia e responsabilità lungo tutto il processo di autenticazione e fornire la massima interoperabilità con qualsiasi sistema di identità digitale. L'uso di una catena di certificati, la cui radice è una mantenuta direttamente dallo Stato responsabile del nodo, unito ad algoritmi robusti per cifrare e firmare le informazioni, garantisce un buon livello di integrità e riservatezza fintanto che tutti gli elementi della catena mantengono il controllo delle proprie chiavi private. Ogni Stato deve garantire che i nodi da lui autorizzati siano effettivamente degni di fiducia e deve vigilare per rilevare immediatamente eventuali condotte scorrette che possono minare il meccanismo di fiducia su cui si basa il sistema. Per evitare che la violazione di uno di questi nodi comprometta tale fiducia è necessario che venga mantenuto un monitoraggio costante e che i certificati compromessi vengano revocati entro il più breve tempo possibile; un'ulteriore precauzione sarebbe quella di utilizzare certificati con durata ridotta per firmare i Metadati.

L'uso di algoritmi robusti, come quelli a curve ellittiche e le funzioni di hash della famiglia SHA-2, unito all'utilizzo di chiavi con lunghezza minima adeguata, mette il sistema al riparo dagli attacchi contro le vulnerabilità note per gli altri algoritmi e dà una ragionevole sicurezza che le informazioni trasmesse non possano essere compromesse o manomesse nel loro transito sulla rete. Grazie all'uso di una chiave di cifratura casuale e diversa per ogni comunicazione, l'eventuale violazione di un singolo messaggio non impatta sulla sicurezza complessiva del sistema. Tuttavia, per permettere la compatibilità con browser datati, in alcune circostanze particolari le specifiche permettono l'uso di TLS 1.1 e dell'hash SHA-1, entrambi vulnerabili e ormai deprecati. Le informazioni sono comunque al riparo da modifiche non autorizzate, in quanto i messaggi SAML vengono firmati utilizzando algoritmi più efficaci; allo stesso modo, le informazioni contenute nelle asserzioni vengono protette con cifratura adeguata e pertanto rimarrebbero riservate anche in presenza di violazione del canale TLS. Come ulteriori misure di sicurezza, le specifiche forniscono delle raccomandazioni per l'uso di TLS: alcune funzionalità, come la compressione, l'estensione di heartbeat e la rinegoziazione della sessione, vengono fortemente sconsigliate, per evitare una serie di attacchi ben documentati [9, sezione 2].

Un fattore che riveste un grande peso nella sicurezza del sistema, ma che risulta essere difficilmente valutabile, è il servizio di identità digitale dei singoli Stati: anche se non rientra a pieno titolo nella trattazione dell'architettura eID eIDAS, sarebbe sbagliato non tenere in considerazione anche questo elemento nella valutazione dei rischi. Le comunicazioni tra eIDAS-Service e IdP e la fase di autenticazione dell'utente attraverso quest'ultimo sono elementi molto sensibili che devono essere attentamente valutate all'atto di implementazione dei diversi schemi di identità digitale. Nel caso sia presente un eIDAS-Connector centralizzato, bisogna prevedere un adeguato protocollo di comunicazione tra quest'ultimo e i SP, che soddisfi i requisiti di sicurezza del massimo livello di garanzia fornito dal Connector stesso, che altrimenti risulterebbe vanificato da possibili attacchi sull'ultimo tratto della comunicazione.

1.5 eID in Italia: SPID

Il Sistema Pubblico di Identità Digitale, noto come SPID, costituisce l'infrastruttura eID italiana, consentendo ai cittadini di registrarsi presso uno degli IdP autorizzati e usare le proprie credenziali per accedere a numerosi servizi sul territorio nazionale e, grazie all'integrazione con il sistema eID eIDAS, in tutta l'Unione Europea. Così come eIDAS, anche SPID utilizza il linguaggio SAML per lo scambio di richieste e autenticazioni; la comunicazione tra SP e IdP avviene attraverso il browser dell'utente con i binding HTTP Redirect e HTTP POST [10, p. 2]. Oltre agli IdP, il sistema prevede che gli attributi possano essere recuperati o verificati contattando uno o più AP; in questo caso la comunicazione avviene direttamente tra SP e AP utilizzando il binding SOAP over HTTP, ovvero incapsulando le richieste SAML all'interno di messaggi SOAP trasportati mediante HTTP POST [10, p. 22].

A differenza di eIDAS, SPID prevede un ente centrale, l'Agenzia per l'Italia Digitale, che supervisiona il corretto funzionamento del sistema e funge da terza parte fidata che certifica gli enti (SP, IdP, AP) cha appartengono al sistema stesso; a questo scopo, l'AgID mantiene un registro contenente, per ogni ente, alcune informazioni, come l'URL del servizio di reperimento dei Metadati e la lista degli attributi certificabili [10, p. 31]. L'AgID stabilisce, inoltre, i protocolli e le specifiche tecniche che i provider devono implementare per poter aderire al sistema.

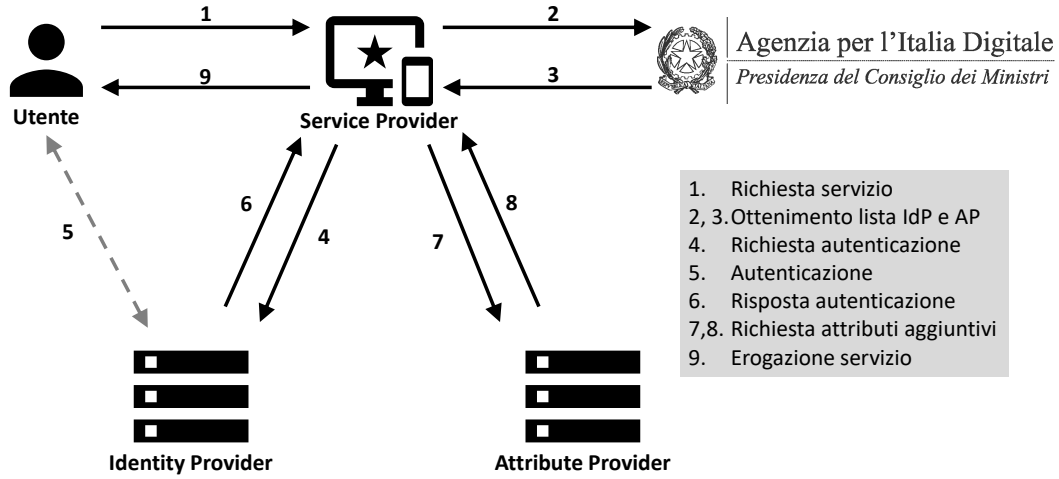


Figura 1.4: Esempio di autenticazione tramite SPID

Per consentire l'integrazione di SPID con il sistema eID eIDAS, è attualmenteattivo il progetto FICEP (First Italian Crossborder eIDAS Proxy) per lo sviluppo del primo nodo eIDAS italiano, che fungerà sia da eIDAS-Connector che da eIDAS-Service di tipo Proxy. Il progetto è sviluppato sulla base dell'implementazione fornita dalla rete di cooperazione e portato avanti dall'AgID con il contributo di alcuni IdP e del Politecnico di Torino.

La procedura di autenticazione, illustrata in Figura 1.4, viene avviata dall'utente che richiede un servizio ad un SP e lo informa di volersi autenticare tramite SPID; l'utente deve anche specificare quale IdP facente parte della rete SPID vuole utilizzare per la procedura di autenticazione. Il SP dovrà quindi esaminare il registro SPID per individuare l'URL dell'IdP a cui richiedere i Metadati; tale registro può essere ottenuto tramite un'interfaccia REST messa a disposizione dall'AgID e può esserne mantenuta una copia locale fino alla fine della sua validità. Il SP prepara una `AuthnRequest` SAML firmandola con la propria chiave privata e la invia all'IdP tramite l'utente. L'IdP provvederà a verificare la firma del messaggio usando i Metadati del SP e autenticherà l'utente in base al livello di garanzia richiesto; verrà, quindi, preparata una risposta con l'esito dell'autenticazione ed eventualmente l'asserzione contenente l'identità e gli attributi dell'utente. A differenza di quanto avviene nella risposta dell'eIDAS-Service, in SPID si richiede solamente che l'asserzione venga firmata con la chiave privata dell'IdP, senza cifrare alcunché. La risposta viene inoltrata tramite l'utente al SP, che, in caso di avvenuta autenticazione, può decidere di contattare uno o più AP per ottenere attributi aggiuntivi: per fare questo, viene preparato un messaggio SAML di tipo `AttributeQuery`, elencando gli attributi di cui si vuole conoscere il valore e indicando come identificativo dell'utente il suo codice fiscale. Questa richiesta, firmata con la chiave privata del SP, viene inviata direttamente da quest'ultimo all'AP tramite una richiesta HTTP POST contenente un messaggio SOAP che incapsula la suddetta richiesta; l'AP verificherà la firma e risponderà tramite una asserzione contenuta in un messaggio di tipo `Response`, a sua volta incapsulato in un messaggio SOAP. Ottenuta l'autenticazione insieme con tutti gli attributi necessari, il SP potrà decidere se autorizzare o meno la fruizione del servizio da parte dell'utente.

Per garantire integrità e riservatezza del canale di comunicazione, SPID richiede che tutte le comunicazioni tra i diversi attori avvengano utilizzando un canale TLS nella versione più recente disponibile [10, p. 16]. Questo implica che versioni obsolete dei browser, che non supportano tale versione, non possono essere utilizzate, a tutto vantaggio della

sicurezza del canale. La presenza di un canale TLS sicuro non mette al riparo da eventuali compromissioni degli end-point; in particolare, un eventuale attaccante che avesse il controllo del dispositivo usato dall'utente potrebbe accedere alle informazioni contenute nell'asserzione una volta che il messaggio sia uscito dal canale TLS, in quanto non viene impiegata alcuna tecnica di cifratura delle informazioni contenute nell'asserzione SAML. Le regole tecniche rilasciate dall'AgID non specificano quali siano le cipher suite da utilizzare per le comunicazioni, lasciando tale scelta ai gestori dei nodi della rete: il livello di sicurezza offerto dipenderà dalle cipher suite offerte all'utente e l'uso di alcune di esse potrebbe non fornire la robustezza sufficiente al livello di garanzia richiesto dal SP. Le specifiche rilasciate dall'AgID richiedono che le firme dei Metadati e dei messaggi SAML siano generate utilizzando chiavi RSA da almeno 1024 bit e algoritmo di hash SHA-256 o superiore; questi requisiti minimi sono insufficienti, in quanto chiavi RSA da 1024 bit sono ormai troppo corte per garantire un adeguato livello di sicurezza. La violazione di tali chiavi consentirebbe ad un'attaccante di generare false asserzioni, che verrebbero poi firmate e rese valide agli occhi del SP.

Capitolo 2

Gestione degli attributi con Shibboleth

2.1 Struttura e funzionalità

Shibboleth è uno dei più diffusi software per la gestione di identità federate basato sullo standard SAML; esso è un esempio di sistema Single Sign-On (SSO), fornendo agli utenti una sola credenziale per accedere a più servizi. Il software è diviso in due componenti principali: l'IdP e il SP. Lo scopo dell'IdP è quello di autenticare gli utenti e fornire le informazioni su di essi al SP; è, quindi, responsabile per la gestione degli attributi e fornisce diversi metodi standard di autenticazione, con la possibilità di implementarne di nuovi. Il SP è incaricato di proteggere una risorsa accessibile online, richiedendo all'IdP l'autenticazione e gli attributi degli utenti interessati; il suo compito è negare o consentire l'accesso alla risorsa in base all'esito dell'autenticazione e al valore degli attributi. È disponibile un ulteriore modulo, chiamato Embedded Discovery Service, che può essere presente accanto al SP e fornisce all'utente la possibilità di scegliere quale IdP utilizzare per l'autenticazione nel caso ne siano previsti diversi. L'uso dello protocollo SAML consente a Shibboleth di essere interoperabile con qualsiasi software di identità federata che usi lo stesso standard, senza bisogno di apportare modifiche alla configurazione dei moduli.

Shibboleth è un software web-based ed è compatibile con i più diffusi web server, come Apache HTTPD e Microsoft IIS; mentre il SP viene distribuito come modulo per tali web server, l'IdP consiste in una applicazione web Java da eseguire in un web container, ad esempio Apache Tomcat o Jetty. Uno dei maggiori vantaggi di Shibboleth è la sua natura open source: il codice di entrambi i moduli è liberamente accessibile e modificabile e questo consente un notevole controllo sulla qualità del prodotto, in quanto eventuali problemi hanno maggiore probabilità di emergere. Ciò rende il software estremamente personalizzabile, operazione ulteriormente semplificata dalla struttura del programma, modulare e facilmente espandibile. La configurazione dei due moduli avviene per mezzo di file XML che contengono, ad esempio, le risorse per cui chiedere l'autenticazione oppure gli attributi da rilasciare ad un certo SP.

Un uso tipico di Shibboleth prevede la creazione di una *federazione* composta da molteplici risorse protette da altrettanti SP, i quali fanno affidamento su uno o più IdP per autenticare gli utenti. Per permettere a tali moduli di conoscere i dettagli sul resto della federazione, per ognuno di essi vengono generati dei Metadati SAML; la loro creazione e distribuzione può essere a carico dei singoli nodi, ma più spesso è presente una terza parte che, in aggiunta, certifica le informazioni contenute al loro interno.

2.1.1 Profili supportati

Shibboleth supporta casi d'uso diversi dal semplice SSO grazie anche ai profili SAML 2.0 [11, sezione 1.1]: questi ultimi sono insiemi di regole che garantiscono l'interoperabilità del protocollo in determinati contesti. Il profilo più comunemente utilizzato è chiamato **Web Browser SSO** [11, sezione 4.1], il quale consente ad un utente di autenticarsi tramite browser dopo aver tentato di accedere ad una risorsa presso un SP. Il profilo prevede che l'utente venga rediretto dal SP all'IdP trasportando un messaggio `AuthnRequest`, all'interno del quale sono presenti diverse informazioni, tra cui: un proprio ID e l'istante di tempo della propria emissione, per evitare attacchi di tipo *replay*; l'identità del SP che ha generato il messaggio e opzionalmente l'URL dove inviare la risposta, in modo che l'IdP possa verificare tali informazioni, tipicamente grazie ai Metadati; diversi elementi opzionali, come i requisiti richiesti per l'autenticazione. Il messaggio può essere firmato in modo da garantirne l'integrità e autenticarne il mittente. È possibile anche che il SP invii tramite l'utente solo piccolo messaggio chiamato *artifact* che contiene un riferimento che l'IdP userà per recuperare la richiesta direttamente dal SP. In seguito, l'IdP autentica l'utente e produce un messaggio `Response` che contiene il risultato dell'operazione: se l'autenticazione è fallita, il messaggio conterrà un elemento `Status` con le informazioni sul motivo del fallimento. In caso di successo, l'IdP inserirà all'interno della risposta una asserzione che attesta l'identità dell'utente attraverso un identificativo, spesso opaco, e un insieme di attributi. L'asserzione può essere sia firmata che cifrata, per mantenere la riservatezza delle informazioni sensibili contenute al suo interno. La risposta viene inviata al SP tramite l'utente o resa disponibile per il recupero tramite *artifact*.

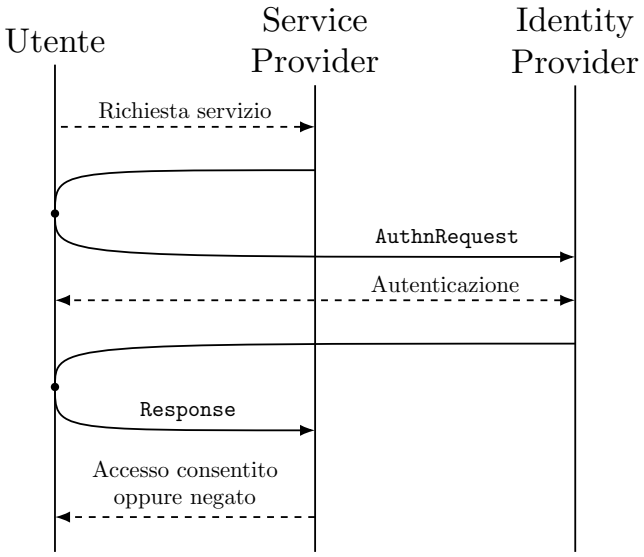


Figura 2.1: Profilo Web Browser SSO

Un secondo profilo per SSO è quello **Enhanced Client/Proxy** [11, sezione 4.2], sfruttato soprattutto per l'autenticazione di client diversi da un browser (applicazioni desktop, applicazioni web ...). Anche questo profilo si basa sullo scambio di messaggi `AuthnRequest` e relative risposte, ma in questo caso il client è in grado di determinare l'IdP da contattare senza bisogno di essere rediretto dal SP. Quest'ultimo risponde alla richiesta di accesso alla risorsa con un messaggio `AuthnRequest` veicolato tramite SOAP inverso (PAOS) e destinato al client stesso; il client stabilisce quale IdP contattare e invia ad esso la richiesta incapsulata in un messaggio SOAP. Dopo aver autenticato l'utente, l'IdP

risponde al client con un messaggio `Response`, il quale sarà inoltrato al SP nuovamente attraverso il binding SOAP inverso. La differenza che intercorre tra i binding SOAP e PAOS risiede nel fatto che nel primo le richieste SAML sono inviate tramite richieste HTTP, mentre nel secondo vengono inserite all'interno di una risposta ad un servizio; in quest'ultimo caso è necessario che il client dichiari di supportare tale binding.

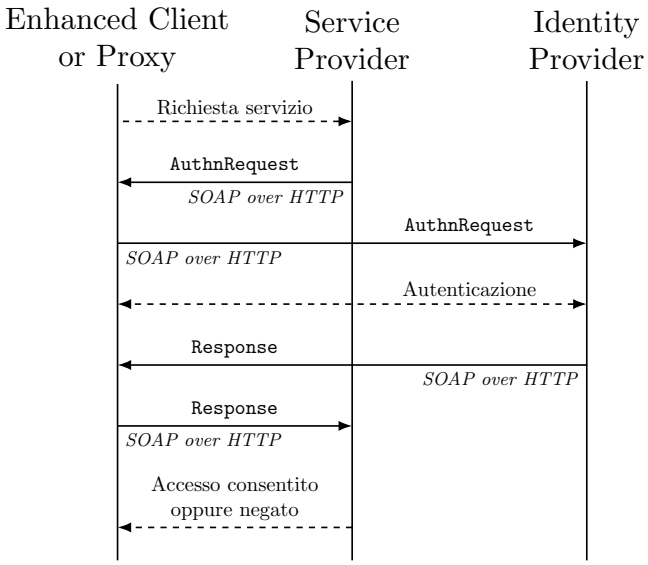


Figura 2.2: Profilo Enhanced Client/Proxy

Shibboleth supporta anche il profilo **Single Logout** [11, sezione 4.2], che fornisce all'utente un modo per terminare la sessione su tutti i SP presso cui si è precedentemente autenticato; ciò avviene attraverso l'invio, da parte dell'IdP, di messaggi di tipo `LogoutRequest` rivolti ai SP che abbiano in precedenza ricevuto un'asserzione di identificazione per il soggetto in questione. Le richieste di logout, così come le risposte di conferma, dovrebbero essere firmate, per garantirne autenticità e integrità; esse possono essere scambiate direttamente tra i provider oppure attraverso l'utente. Il profilo può essere attivato direttamente presso l'IdP oppure attraverso uno dei SP, il quale invierà una richiesta di logout all'IdP; sarà quest'ultimo, in ogni caso, ad inviare le richieste ai diversi SP.

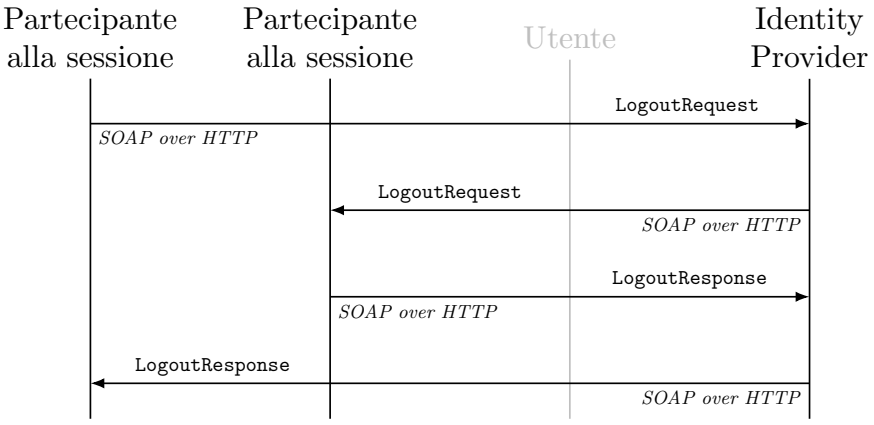


Figura 2.3: Profilo Single Logout

Come già accennato in precedenza, Shibboleth consente il recupero di messaggi SAML da parte di un provider utilizzando il profilo **Artifact Resolution** [11, sezione 5]: in alcuni casi, per evitare che un messaggio venga inoltrato tramite il browser dell'utente, il suo mittente può sostituirlo con un riferimento detto *artifact*. Il destinatario, per recuperare il messaggio originale, invierà una richiesta `ArtifactResolve` contenente il suo riferimento attraverso un binding sincrono come SOAP; la risposta conterrà il messaggio originale. Il profilo richiede che il richiedente dimostri la propria identità, firmando la richiesta o tramite altri mezzi supportati dal binding, come, ad esempio, tramite autenticazione client a livello TLS.

L'ultimo profilo SAML 2.0 supportato da Shibboleth è chiamato **Attribute Query**, un caso particolare del più generico *Assertion Query/Request Profile* [11, sezione 6]: quest'ultimo, nello standard SAML, consente di recuperare asserzioni già esistenti riguardanti un soggetto. Le richieste e le relative risposte vengono scambiate direttamente tra i provider tramite binding SOAP; esistono diversi tipi di richieste e query che è possibile inviare per ottenere informazioni come attributi aggiuntivi o decisioni di autorizzazione. Shibboleth supporta esclusivamente le richieste di tipo `AttributeQuery`, che consentono di richiedere ad un IdP determinati attributi per un soggetto. La richiesta contiene l'identificativo dell'utente in questione e, facoltativamente, una lista di attributi; in caso di successo, la risposta conterrà i valori degli attributi richiesti, se questi possono essere rilasciati al richiedente.

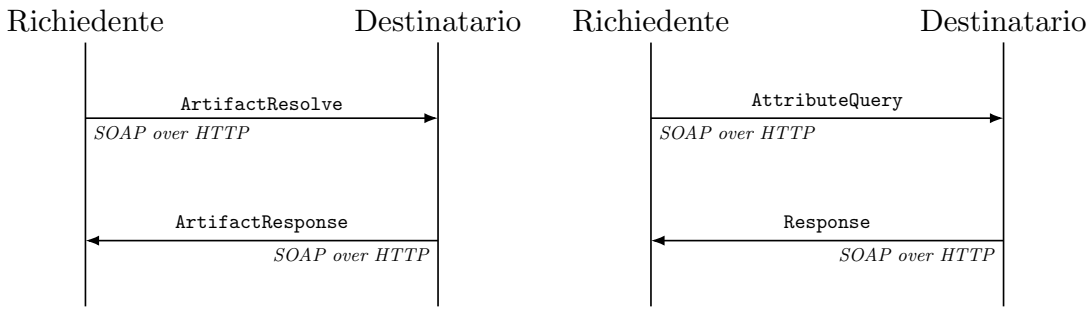


Figura 2.4: Profili Artifact Resolve e Attribute Query

Oltre ai profili SAML 2.0, gli IdP Shibboleth supportano anche quelli della versione 1.1; viene supportato anche il protocollo CAS ed è in sviluppo un plug-in che consenta il supporto a OpenID Connect. Questo consente a Shibboleth di adattarsi alla tutti gli scenari più comuni e di interfacciarsi con decine di prodotti per la gestione di identità federate diversi.

2.1.2 Service Provider

Il SP Shibboleth è costituito da una coppia di componenti: un modulo caricato dal web server che gestisce le risorse da proteggere e un servizio che lavora in background chiamato *"shibd"*. Questa soluzione viene adottata principalmente in quanto la maggior parte dei web server vengono eseguiti tramite un pool di processi, per i quali il SP deve mantenere una cache di sessioni di autenticazione; il modo più semplice per gestire tale cache è quella di mantenerla in un processo separato. Un ulteriore vantaggio di questo sistema è la possibilità di restringere l'accesso alle eventuali chiavi crittografiche usate dal SP al solo processo *shibd*, incrementando in tal modo la sicurezza.

La configurazione del SP avviene mediante file XML opportunamente modificati; oltre ai file specifici del web server, la maggior parte delle modifiche si concentra nel file `shibboleth2.xml`. Questo documento consiste in un elemento `SPConfig` il quale contiene diversi altri elementi, ognuno dei quali rappresenta una categoria di configurazioni del SP. Uno degli elementi da configurare è quello che contiene i servizi che devono essere supportati dal SP; ognuno di essi è rappresentato da un elemento che identifica il tipo di servizio, come SSO o Logout, insieme con degli attributi che specificano le proprietà di quel particolare servizio. Nel caso di un elemento SSO è possibile configurare l'IdP da usare per l'autenticazione, l'eventuale Discovery Service da interrogare e il protocollo da utilizzare per lo scambio di messaggi. Per ogni IdP con cui il SP dovrà comunicare è necessario fornire un modo per recuperare i suoi Metadati: essi possono essere ottenuti da un file locale o tramite una richiesta ad un server remoto. Tra gli elementi da configurare obbligatoriamente rientra anche quello che specifica come recuperare le credenziali del SP, costituite dal certificato a chiave pubblica e la corrispondente chiave privata usata per firmare i messaggi inviati e decifrare quelli ricevuti.

Lo scopo del SP è quello di garantire che l'accesso a una o più risorse avvenga esclusivamente da parte di utenti autenticati e autorizzati; questo avviene per mezzo di sessioni create all'atto di autenticazione di un utente il quale sarà in possesso di una chiave di sessione, tipicamente memorizzata all'interno di un cookie web. Il SP Shibboleth offre due tipologie di protezione dei contenuti: la modalità attiva blocca l'accesso di tutti gli utenti sprovvisti di una sessione valida, mentre quella passiva permette l'accesso alla risorsa indistintamente. Quest'ultima modalità può essere utile per quelle applicazioni dinamiche che offrono contenuti anche ad utenti non registrati, oppure se sono previste altre forme di autenticazione, delegando all'applicazione il compito di attivare il modulo Shibboleth. Mentre la modalità attiva necessita che siano specificate le risorse da proteggere per entrare in funzione, quella passiva viene abilitata all'avvio di Shibboleth: le applicazioni che operano sul web service possono ottenere informazioni sulla sessione corrente attraverso delle variabili CGI (Common Gateway Interface) condivise che vengono valorizzate da Shibboleth a seguito dell'autenticazione. Se un'applicazione desidera autenticare un utente e cominciare una sessione in modalità passiva deve redirigere l'utente presso una delle risorse che identificano i gestori Shibboleth per l'inizializzazione delle sessioni, le quali possono essere specificate nella configurazione dei servizi supportati.

Le risorse protette in modalità attiva possono essere definite direttamente nella configurazione del web service oppure all'interno del SP; nel primo caso la procedura da seguire dipende ovviamente dal web service su cui viene eseguito Shibboleth, mentre la configurazione all'interno del SP avviene tramite l'elemento `RequestMapper`. Al suo interno è possibile specificare nomi host e percorsi, definendo le proprietà e i vincoli per l'accesso a questi ultimi. Ogni elemento all'interno di tale configurazione rappresenta una risorsa o un gruppo di esse, mentre le proprietà consentono di definire se per l'accesso a tali risorse è necessaria la presenza di una sessione attiva e diversi altri parametri che possono sovrascrivere quelli specificati nella configurazione generale; tra essi troviamo l'IdP da contattare e la richiesta di autenticazione forzata anche in presenza di una sessione attiva. Si possono definire, inoltre, delle semplici politiche di accesso basate sulla presenza e sul valore di determinati attributi dell'utente.

Nella configurazione è possibile specificare quali attributi estrarre delle asserzioni e le eventuali regole usate per filtrarli. All'interno dell'elemento `AttributeExtractor` è presente la lista di attributi che devono essere recuperati, se presenti; per ognuno di essi si deve specificare il decoder da utilizzare per processare il valore dell'attributo, a meno che non si tratti di una semplice stringa, e il suo identificativo all'interno del SP. Gli

attributi possono essere filtrati attraverso delle politiche, che contengono delle condizioni che ne decidono l'attivazione; le politiche sono costituite da un insieme di regole, ognuna delle quali agisce su un determinato attributo per stabilire se esso debba essere eliminato o messo a disposizione. Le regole possono essere costituite da semplici comparazioni o operazioni più complesse. Tutti gli attributi che superano la fase di filtraggio vengono resi disponibili come variabili CGI il cui nome è dato dal loro identificativo.

Quando esso viene attivato, in base alla sua configurazione, il SP determina quale IdP contattare e quale protocollo utilizzare; la scelta può avvenire interrogando l'utente, contattando un Discovery Service oppure in modo automatico in base alla risorsa richiesta. In questa fase il SP utilizza un meccanismo chiamato *"relay state"* per mantenere alcune informazioni riguardo la richiesta, in particolare la risorsa a cui l'utente voleva inizialmente accedere. Grazie alle informazioni contenute nei Metadati dell'IdP scelto e alle proprietà della configurazione relativa alla risorsa richiesta, viene generata una richiesta di autenticazione che viene, se possibile, firmata e inoltrata attraverso l'utente, che viene rediretto, tramite GET o POST, all'end-point dell'IdP apposito.

Dopo aver terminato il processo di autenticazione, l'utente verrà rediretto dall'IdP all'end-point del SP corrispondente al suo Assertion Consumer Service (ACS), il cui compito è interpretare la risposta; questa può contenere un'asserzione con l'identità dell'utente ed eventualmente alcuni attributi, nel caso l'autenticazione sia andata a buon fine. L'ACS dovrà decifrare tale asserzione, se necessario, ed effettuare una serie di verifiche: dovrà convalidare, utilizzando il certificato dell'IdP, la firma del messaggio e assicurarsi che esso non sia scaduto al momento della verifica. Se tutti i controlli danno esito positivo il SP crea un nuova sessione per l'utente inserendo al suo interno gli attributi estratti dall'asserzione, dopo averli opportunamente filtrati; tali informazioni vengono rilasciate sotto forma di variabili CGI. Infine, l'utente viene riportato alla risorsa richiesta, utilizzando le informazioni contenute nel *relay state* memorizzato.

2.1.3 Identity Provider

Il ruolo dell'IdP Shibboleth in un sistema di identità federata è duplice: il suo scopo primario è quello di verificare l'identità degli utenti che desiderano accedere ai servizi offerti dai SP della federazione; può, inoltre, gestire delle informazioni su tali utenti le quali, se utili alla fruizione del servizio richiesto, possono essere condivise con i SP.

L'IdP Shibboleth è un'applicazione web in linguaggio Java basata sul framework Spring, costituito da singoli blocchi di codice chiamati *componenti*, ognuno dei quali svolge una particolare funzione; questi vengono legati assieme attraverso il framework per creare componenti più complessi, come un modulo per la risoluzione degli attributi, e possono essere configurati attraverso file XML. Il framework Spring fa uso del pattern di *inversione di controllo*, per cui, durante l'esecuzione, i componenti dell'applicazione vengono richiamati dal framework, al contrario di quanto accade nella tradizionale programmazione procedurale; questo pattern aumenta notevolmente la modularità e l'estensibilità dell'applicazione, consentendo di aggiungere nuovi componenti semplicemente aggiornando i file di configurazione.

I passi necessari per lo svolgimento dei compiti dell'IdP vengono incapsulati all'interno di un flusso di azioni gestito da Spring; questo viene sempre attivato dalla ricezione di una richiesta proveniente da un SP, la quale viene analizzata da un dispatcher che ha il compito di riconoscere il tipo di messaggio, in base al quale verrà attivato uno dei flussi detti *Profile Handler*. I diversi profili gestiscono le richieste attraversi una serie di azioni

che a loro volta fanno uso dei componenti dell'IdP per svolgere i compiti più semplici. Shibboleth contiene già al suo interno una serie di Profile Handler predefiniti per gestire i profili SAML standard. L'uso dei Web Flow Spring consente agli sviluppatori di estendere facilmente le funzionalità dell'IdP implementando nuove azioni o creando nuovi flussi.

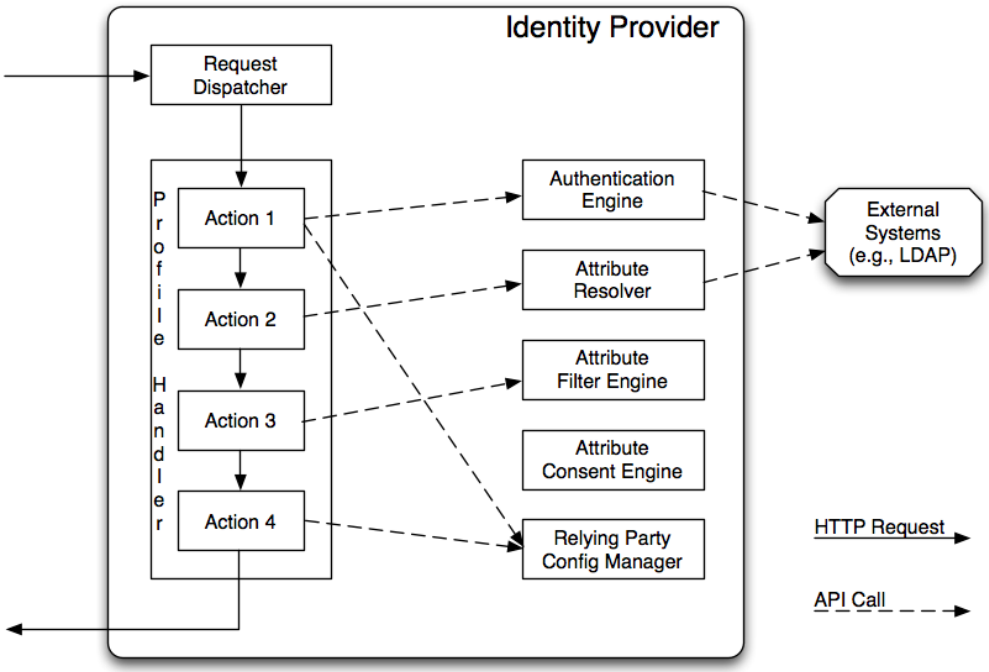


Figura 2.5: Architettura dell'IdP Shibboleth (fonte: wiki.shibboleth.net)

Il modo più semplice per estendere le funzionalità di un flusso sarebbe quello di modificare il file che racchiude la sequenza di azioni che lo compongono e aggiungervene di nuove. Nel caso dei flussi predefiniti questa strategia renderebbe difficoltoso l'aggiornamento del sistema, dovendo riapplicare tutte le modifiche alla nuova versione. Per evitare questo problema, la maggior parte dei flussi prevede che in alcuni punti dell'esecuzione il controllo possa passare a uno o più flussi secondari, i quali possono modificare lo stato della richiesta prima di ritornare al flusso principale; in questo modo le azioni aggiuntive rimangono separate dal flusso principale. I flussi secondari vengono chiamati *interceptor* e ogni Profile Handler offre almeno due punti di inserzione, ovvero durante le elaborazioni iniziali e finali del messaggio; per quei profili che gestiscono le richieste di autenticazione se ne aggiunge un terzo nella fase successiva all'autenticazione, durante il elaborazione degli attributi. Quest'ultimo punto è spesso il più utilizzato, in quanto consente di intervenire su un maggior numero di informazioni: ad esempio è possibile implementare una richiesta di consenso per l'utente, in modo che possa decidere quali attributi effettivamente rilasciare.

La configurazione dell'IdP Shibboleth avviene sulla base di molteplici file, ognuno dei quali riguarda un diverso aspetto del software. Sebbene essi siano, per la maggior parte, documenti XML, si differenziano per la sintassi utilizzata: molti utilizzano il linguaggio di configurazione Spring, dato che intervengono direttamente sulla gestione degli oggetti istanziati dal framework, chiamati *"beans"*, mentre altri utilizzano una sintassi propria di Shibboleth. Per configurare correttamente l'IdP occorre, come prima cosa, specificare le sorgenti di Metadati da utilizzare per ottenere informazioni riguardo i SP con cui dovrà interagire: possono essere forniti diversi tipi di sorgenti, come file locali o server HTTP esterni. È poi necessario definire quali funzioni l'IdP deve supportare in base all'identità del SP con cui comunica; tali configurazioni avvengono nel file `relying-party.xml`, nel

quale sono definiti tre elementi: il primo consente di specificare i flussi attivi e le relative proprietà per quei SP di cui l'IdP è in grado di recuperare i Metadati, mentre il secondo riguarda tutti gli altri. Tipicamente in quest'ultimo non viene attivato alcun flusso, per evitare che Shibboleth risponda a SP sconosciuti. L'ultimo elemento contiene delle configurazioni particolari che possono essere attivate in base all'identità del SP, sostituendosi a quelle degli elementi precedenti. Ogni flusso specificato all'interno di tali configurazioni può essere a sua volta configurato impostando delle proprietà che variano in base al tipo di flusso: per quello SSO SAML, ad esempio, è possibile definire, tra i vari parametri, se cifrare e firmare le asserzioni, la loro durata di validità e la durata massima della sessione. Tra le proprietà più importanti che è possibile valorizzare ci sono la lista dei metodi di autenticazione da adoperare e gli interceptor da attivare a seguito dell'autenticazione.

Tra i flussi già implementati, i più importanti sono quelli che si occupano di gestire i profili SSO SAML, in quanto sono maggiormente usati. Dopo aver selezionato, se possibile, il flusso corretto, l'IdP verifica la provenienza e l'integrità del messaggio grazie ai Metadati del mittente, i quali possono essere recuperati in diversi modi, come accade per il SP; se le verifiche vanno a buon fine, viene attivato il flusso adeguato per gestire la richiesta. Nel caso in cui venga avviato un flusso SSO, il primo componente ad essere attivato è, tipicamente, quello che si occupa di autenticare l'utente: Shibboleth fornisce *out of the box* il supporto ad alcuni dei metodi più utilizzati, tra cui l'autenticazione con password, tramite certificato X.509, a due fattori e attraverso Kerberos. Tipicamente, il componente di autenticazione fa uso di moduli di back-end che implementano la logica di verifica delle credenziali, ad esempio sfruttando il protocollo LDAP (Lightweight Directory Access Protocol) o quello Kerberos. Ogni flusso di autenticazione si basa sulla configurazione contenuta in un file Spring, la quale varia in base al tipo di autenticazione scelta.

La seconda fase nella gestione di una richiesta SSO è il recupero degli attributi; Shibboleth consente di specificare diverse sorgenti da cui attingere. Questi attributi saranno usati come input per creare quelli da esporre ai SP; l'elaborazione può essere molto semplice, come presentare l'attributo senza modifiche, oppure prevedere diverse manipolazioni per mezzo di espressioni regolari o script; in alcuni casi non è necessario indicare alcun attributo di input, come nel caso si voglia esporre l'identificativo dell'utente autenticato. Dopo aver generato la lista di attributi da esporre ai SP, l'IdP effettua un filtraggio di tale lista in base a determinate regole specificate nella configurazione. In questa fase si inseriscono la maggior parte dei flussi secondari configurati per estendere le funzionalità dell'IdP; sono già inclusi alcuni esempi come la richiesta di consenso dell'utente al rilascio degli attributi oppure la possibilità di avvisare l'utente della prossima scadenza delle sue credenziali. L'ultimo passaggio prevede la costruzione della risposta che, tipicamente, conterrà un'asserzione con le informazioni sull'identità dell'utente. La risposta e il suo contenuto possono essere cifrati e firmati in base alla configurazione specificata per il SP destinatario.

I flussi eseguiti per la gestione delle richieste sono dinamici, essendo composti da molti sotto-flussi che possono essere eseguiti o meno in base a diversi fattori, come il tipo di richiesta, il SP mittente o il risultato di un'operazione precedente. Questo comportamento consente di gestire molteplici situazioni con il minimo sforzo, in quanto nella maggior parte dei casi basta agire sui file di configurazione, senza la necessità di modificare il codice e ricompilare l'applicazione. Anche qualora fosse necessario estendere le funzionalità con nuovi flussi secondari, la modularità del Web Flow Spring incoraggia il riuso dei sotto-flussi già presenti, i quali, essendo stati estensivamente testati dagli sviluppatori del progetto Shibboleth, riducono sensibilmente il rischio di introdurre nuovi bug che inficerebbero la sicurezza del sistema.

2.2 Sicurezza

Essendo Shibboleth un software open source, il suo sviluppo viene portato avanti da un gran numero di volontari, i quali sottopongono dei contributi che vengono esaminati dagli altri partecipanti al progetto e, in particolar modo, dal comitato di gestione, che ha il compito di valutare quali siano quelli che meritano di essere integrati nelle nuove release. Questa struttura consente di ottenere codice ampiamente analizzato e testato, garantendo trasparenza e riducendo drasticamente il rischio di bug e vulnerabilità.

La profonda configurabilità del software, benché lo renda estremamente versatile, comporta una grande responsabilità per coloro incaricati della sua messa in opera, in quanto, per funzionare correttamente, è necessario che i file di configurazione vengano modificati con coscienza. Un errore in uno di questi file può avere effetti importanti che vanno dal mancato funzionamento del sistema fino a problemi ben più gravi come il rilascio indiscriminato di attributi. Uno dei punti fondamentali per il corretto funzionamento della struttura di identità federata è l'autenticità delle richieste e delle risposte: è necessario che i SP così come gli IdP siano in grado di identificare eventuali messaggi provenienti da fonti esterne alla federazione. Per garantire ciò, sono fondamentali le informazioni contenute nei Metadati e, conseguentemente, la distribuzione di questi ultimi. Il modo più sicuro ed efficiente per la loro gestione consiste nell'approntare un server centrale da cui i componenti della federazione possano scaricarli; per garantirne l'autenticità e l'integrità, i Metadati dovranno essere firmati con la chiave privata dal server e verificati con il corrispondente certificato, che deve essere fornito ad ogni componente. All'interno dei Metadati è possibile specificare i certificati associati alle chiavi usate dai SP e IdP per firmare i messaggi SAML, in modo che il destinatario di questi ultimi sia in grado di verificarne il mittente.

Le comunicazioni tra il software Shibboleth e l'utente dovrebbe avvenire utilizzando un canale sicuro TLS, il quale, se usato nell'ultima versione disponibile e con cipher suite adeguate, garantisce integrità e riservatezza delle informazioni scambiate, in particolar modo nel caso di dati sensibili come le asserzioni. L'uso di TLS per le connessioni in ingresso deve essere configurato nel web server su cui viene eseguito Shibboleth; tuttavia, ci sono dei casi in cui avviene una comunicazione diretta tra due diversi provider, come caso di una Attribute Query da parte di un SP. In questi casi è possibile configurare la connessione TLS in modo da richiedere la *client authentication* mediante certificato X.509: questo consente di avere un ulteriore livello di affidabilità, anche se necessita di una configurazione aggiuntiva.

Per quanto si è detto finora, è fondamentale che le chiavi private usate dai provider vengano mantenute segrete, al fine di garantire l'integrità e la riservatezza dei messaggi scambiati. Di default, Shibboleth prevede che le chiavi private siano mantenute all'interno di file codificati con lo standard PKCS #8 non cifrati. È necessario assicurarsi che l'accesso a tali file sia limitato al web server su cui viene eseguito Shibboleth, gestendo attentamente i permessi assegnati dal sistema operativo. Risulta evidente che chiunque riuscisse ad avere accesso a tali informazioni potrebbe emettere richieste o, ancor peggio, asserzioni spacciandosi per un provider legittimo; bisogna, quindi, che gli amministratori mettano in atto misure di prevenzioni efficaci e che venga monitorata costantemente la situazione per rilevare tempestivamente eventuali violazioni, in modo da poter rapidamente prendere provvedimenti, come la revoca dei certificati compromessi. Purtroppo non è possibile configurare Shibboleth affinché faccia uso di HSM (Hardware Security Module) i quali, mantenendo le chiavi private al loro interno, garantirebbero un livello di sicurezza maggiore.

2.3 Gestione e uso degli attributi

L'identità di un soggetto, all'interno di Shibboleth, è costituita dall'insieme di attributi ad esso associati: essi sono costituiti da coppie chiave-valore che rappresentano delle caratteristiche del soggetto. Ogni attributo fornisce un'informazione che può essere usata per classificare l'individuo, come sesso e nazionalità, o per poterlo identificare rispetto agli altri, ad esempio tramite mail o numero di telefono. La totalità degli attributi di una persona può essere suddivisa in gruppi che definiscono un suo singolo aspetto: basti pensare ad attributi come gruppo sanguigno o risultati degli esami clinici, che costituiscono parte dell'ambito medico, oppure la lista delle nostre relazioni sui social network, che invece fanno parte della sfera sociale. L'accesso alle informazioni contenute negli attributi deve essere regolato sulla base del soggetto che li richiede e dell'uso che se vuole fare: nell'esempio precedente, le informazioni mediche saranno accessibili solo al medico che ha in cura la persona in oggetto.

Per poter essere trasmessi al SP che ne ha fatto richiesta, in Shibboleth, dopo l'autenticazione dell'utente, gli attributi attraversano quattro passaggi: il loro recupero dalle fonti, la loro elaborazione da parte dell'IdP con relativo filtraggio, la trasmissione al modulo SP Shibboleth e infine la loro disponibilità alla risorsa che li aveva richiesti. Durante queste fasi, gli attributi e i rispettivi valori possono cambiare, alcuni di essi potrebbero essere eliminati mentre altri attributi potrebbero essere generati; tutto ciò dipende principalmente dal SP che ha richiesto l'autenticazione e dall'utente a cui potrebbe essere chiesto il consenso per il rilascio di alcuni di essi.

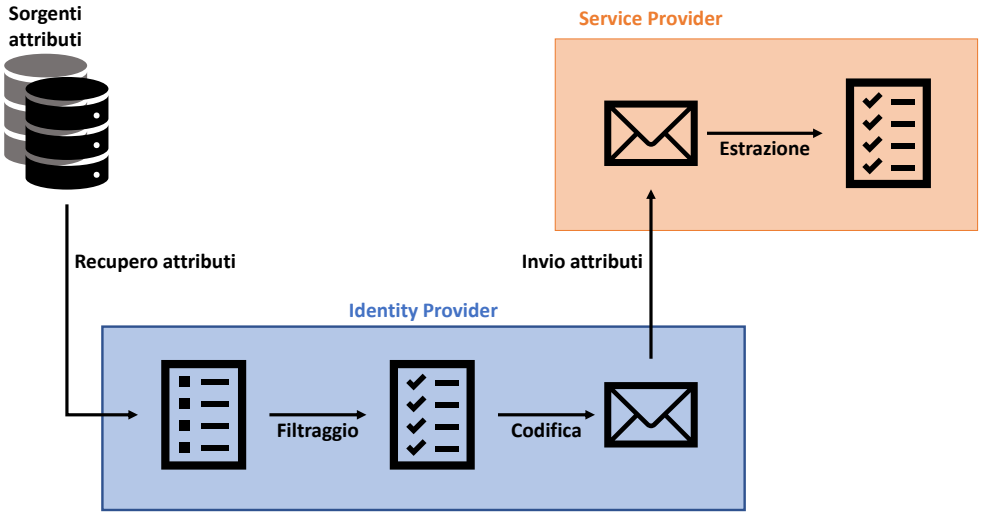


Figura 2.6: Fasi della gestione degli attributi con Shibboleth

La configurazione delle prime tre fasi avviene nell'IdP attraverso la modifica di due file: `attribute-resolver.xml` e `attribute-filter.xml`. Nel primo vengono definite le sorgenti da cui recuperare gli attributi e le definizioni degli stessi. Le fonti vengono specificate mediante dei componenti detti `DataConnector`, che definiscono il tipo di sorgente da contattare e nei quali vengono indicati i parametri utili al recupero degli attributi, come ad esempio la query da effettuare. I componenti più usati per questo scopo sono quelli che consentono di interrogare un database relazionale o una directory usando rispettivamente i protocolli JDBC e LDAP; è anche possibile codificare staticamente i valori di

alcuni attributi direttamente nel file di configurazione, oppure generarli tramite script Java, attingendo dalle informazioni del contesto e da quelle contenute nella richiesta. Nelle prossime versioni è previsto anche un componente che consentirà di interrogare una risorsa HTTP per ottenere informazioni da un *web service*. Gli attributi recuperati da queste fonti sono poi usati come input per crearne di nuovi a partire dalle definizioni presenti nel file; lo scopo principale di queste elaborazioni è disaccoppiare gli attributi presenti nei database e nelle directory interrogate da quelli forniti ai SP. Gli attributi definiti in questa fase possono essere ad uso interno dell'IdP, usati come passo intermedio per la generazione di altri attributi, oppure pronti per essere inviati al SP; a questi ultimi vengono associati uno o più codificatori, che convertono la rappresentazione interna dell'attributo in un particolare formato del protocollo SAML; i codificatori maggiormente utilizzati sono *String* e *Base64*, che codificano rispettivamente una stringa e un insieme di dati binari in un attributo SAML.

Prima della codifica vera e propria gli attributi vengono filtrati utilizzando le regole definite nel file `attribute-filter.xml`; Shibboleth mette tutte le informazioni di contesto a disposizione del componente incaricato di questo compito, incluse le informazioni sul SP, sulla richiesta e sull'utente. Le regole consentono di specificare le condizioni che le attivano e gli attributi da rilasciare o bloccare; si creano quindi due liste contenenti gli attributi permessi e quelli negati, che saranno usate per popolare e poi sfoltire la lista finale di attributi da inviare nella risposta. Le condizioni di attivazione sono altamente configurabili e possono consistere in semplici comparazioni, conteggio del numero di attributi, fino all'uso di espressioni regolari e di script; come già detto, i valori su cui testare le condizioni possono essere recuperati dalle informazioni del contesto. La lista degli attributi da rilasciare può essere ulteriormente modificata attraverso gli interceptor configurati in questa fase: ad esempio, all'utente può essere richiesto quali di questi attributi effettivamente inviare.

Gli attributi che hanno superato la fase di filtraggio vengono codificati tramite uno dei codificatori loro assegnati e sono inviati al SP. È importante che IdP e SP concordino sui nomi assegnati agli attributi e sui significati da assegnare ai loro valori: bisogna stabilire un vocabolario comune che includa, possibilmente, uno standard per identificare univocamente i diversi attributi. Shibboleth suggerisce, per gli attributi standard come e-mail e dati anagrafici, di adottare il profilo SAML V2.0 LDAP/X.500 [12, sezione 2.3], che specifica di usare come nome degli attributi uno *Uniform Resource Name* (URN) appartenente al namespace definito dall'RFC-3061 [13]; questo, a sua volta, fa riferimento al concetto di *Object Identifier* (OID), una codifica standardizzata da ITU e ISO per identificare univocamente generici oggetti. Gli attributi più comuni sono codificati da un codice univoco, in modo da evitare ambiguità: ad esempio, l'OID `0.9.2342.19200300.100.1.3` corrisponde ad un indirizzo e-mail secondo l'RFC-822. Nel caso un attributo non abbia un corrispondente OID, il suggerimento è quello di assegnare ad esso una URL che lo identifichi; si tratta di una pratica diffusa in ambito XML per identificare degli oggetti, dato che una URL è spesso autodescrittiva dell'elemento che rappresenta. Nel precedente esempio con la e-mail, si potrebbe adottare come nome dell'attributo `http://example.org/attributes/mail`, dove `example.org` rappresenta il dominio all'interno del quale l'attributo viene scambiato: è importante che quest'ultimo sia controllato dal gestore della federazione SAML, come un dominio aziendale.

Tramite SAML, gli attributi possono essere ottenuti dal SP principalmente in due modi: il primo consiste nella loro ricezione all'interno dell'asserzione inviata dall'IdP in seguito all'autenticazione dell'utente. Questa metodologia è la più comune e, a partire dalla versione 2.0 del protocollo, anche quella standard, in quanto riduce al minimo le comunicazioni tra i provider e fornisce all'utente maggiore controllo sul rilascio dei suoi

attributi. Il secondo metodo consiste nella richiesta, da parte del SP, di uno o più attributi specificando l'identificativo dell'utente tramite un messaggio `AttributeQuery`; si tratta di un meccanismo che deriva dalle prime versioni dello standard, quando gli attributi non erano inclusi di default all'interno dell'asserzione di autenticazione. A differenza dei messaggi scambiati durante la procedura di SSO, che vengono trasportati dal browser dell'utente tramite il protocollo HTTP, le richieste di attributi effettuate con il secondo metodo vengono scambiate direttamente tra i provider e vengono incapsulate in messaggi SOAP all'interno delle richieste HTTP. Poiché questo metodo non richiede l'interazione dell'utente, deve essere usato con la dovuta cautela, in quanto un SP potrebbe, in determinate circostanze, essere in grado di ottenere attributi senza l'autorizzazione dell'utente o, peggio, a sua insaputa; è necessario, quindi, che venga posta particolare attenzione nella configurazione dell'IdP, in modo da garantire l'uso di questa funzione solo quando non è possibile farne a meno e implementando correttamente le regole per il filtraggio degli attributi.

Una volta ricevuti dal SP, gli attributi vengono decodificati; nel file `attribute-map.xml` devono essere indicati gli identificativi degli attributi attesi, con gli eventuali decoder da usare per la loro estrazione. Gli attributi e le altre informazioni memorizzate nella sessione del SP vengono resi disponibili alle altre applicazioni presenti sul web server mediante variabili o header HTTP esposti tramite CGI; ogni attributo viene inserito in una variabile con il suo nome, in aggiunta ad altre predefinite del SP che forniscono informazioni circa la sessione e il processo di autenticazione.

2.4 Alternative

L'uso di attributi per caratterizzare un soggetto non è una prerogativa dello standard SAML: esistono decine di protocolli e sistemi diversi che fanno uso, in diverse forme, della nozione di attributo per memorizzare e distribuire informazioni. La famiglia di standard **X.500** sviluppata dall'ITU-T risale a trent'anni fa, ma ancora oggi è parzialmente utilizzata, sia pure con i dovuti aggiornamenti: di questa suite fa parte lo standard X.509 che, nella sua ultima versione [6, sezione 4], descrive la struttura dei certificati a chiave pubblica usati tutt'oggi. Un'altra eredità di questo standard è la struttura con cui possono essere organizzate le informazioni relative a dei soggetti all'interno di una directory: tale struttura è ancora usata nel protocollo **LDAP** (Lightweight Directory Access Protocol), oltre che in altri standard come il suddetto X.509. In tali directory ogni entità, che si tratti di una persona o di un soggetto diverso, viene classificata in modo gerarchico ed è rappresentata da un insieme di attributi, i quali hanno un tipo e uno o più valori; ogni entità viene poi identificata da un *Distinguished Name*, composto da uno o più attributi del soggetto e dal DN dell'entità genitore. Per garantire interoperabilità tra directory diverse, l'ITU-T ha rilasciato uno standard che contiene una lista di definizioni degli attributi più comuni [14, sezione 6]; questi sono usati anche per definire il soggetto all'interno dei certificati a chiave pubblica tramite un DN. LDAP consente di collegarsi alla directory in modo anonimo o tramite autenticazione: quest'ultima può avvenire attraverso l'invio di un nome utente nella forma di un DN e una password, che verrà confrontata con il valore dell'attributo `userPassword` dell'entità corrispondente. Sono previsti meccanismi di autenticazione più sicuri, come Kerberos o usando l'autenticazione client TLS. Sebbene LDAP e le directory X.500 non facciano attivamente parte di alcun meccanismo SSO, il loro uso come fonte di attributi li rende ancora molto usati; quasi tutti i gestori di attributi e gli IdP, Shibboleth compreso, supportano la connessione e il recupero di attributi da queste sorgenti.

Il protocollo SAML, sebbene molto diffuso in ambito desktop e server web-based, non è particolarmente adatto per applicazioni eseguite in ambiente mobile, a causa della sua complessità e dell'uso del linguaggio XML per la codifica dei messaggi, i quali risultano essere lunghi e pesanti da gestire. Per questi motivi, nel tempo si è affermato un diverso standard chiamato **OpenID Connect**, il quale si basa sull'architettura REST e sull'uso di messaggi JSON per lo scambio di informazioni di autenticazione. OpenID Connect prevede un'architettura molto simile a quella di SAML, con un SP a cui l'utente richiede un servizio e un IdP sul quale quest'ultimo viene indirizzato per l'autenticazione. Al fine di rendere il protocollo più leggero e adatto per dispositivi con ridotta capacità computazionale, come gli smartphone, OpenID Connect prevede, nella maggior parte dei casi, che la risposta inoltrata al SP attraverso l'utente contenga solo un *Authorization code*, un codice che il SP può utilizzare per richiedere direttamente le informazioni di autenticazione all'IdP; queste vengono rilasciate all'interno di un JSON Web Token (JWT), uno standard codificato dall'RFC-7519 [15] per lo scambio di asserzioni attraverso oggetti JSON compatti e firmati.

In OpenID Connect l'identità dell'utente è costituita da una serie di "dichiarazioni" (*claims*) che svolgono le funzioni che in SAML erano a carico degli attributi; come questi ultimi, i *claims* consistono in coppie nome-valore che forniscono informazioni sul soggetto a cui si riferiscono. Poiché i documenti JSON sono nativamente costituiti da coppie chiave-valore, tale rappresentazione risulta molto più agevole e compatta in OpenID Connect di quanto non fosse con gli attributi SAML. I *claims* sono costrutti propri di JWT e, nello standard, vengono classificati in base al loro nome: infatti, viene consigliato di non usare nomi arbitrari, per facilitare l'interoperabilità ed evitare collisioni. Per questo motivo, presso lo IANA (Internet Assigned Numbers Authority), sono registrati i *Public Claim Names*, una serie di nomi standard che coprono le informazioni più comuni per descrivere una persona, come i dati anagrafici, il numero di telefono e la mail; se fosse necessario descrivere un'informazione per cui non è presente un nome pubblico, lo standard consente di usare nomi privati. Infine, sono presenti dei nomi chiamati *Registered Claim Names*, che identificano *claims* che forniscono informazioni riguardo il token stesso piuttosto che il soggetto: chi ha emesso il token, a chi è destinato, l'istante di emissione e il periodo di validità.

Un esempio di utilizzo di OpenID Connect in ambiente server è **Active Directory Federation Services** (AD FS), sviluppato da Microsoft per fornire funzionalità SSO ai suoi sistemi operativi per server. Anche se questo software si basa sul protocollo OpenID Connect consente di partecipare a federazioni che si basano sullo standard SAML, grazie alla sua compatibilità con tale protocollo. AD FS può essere usato solo su sistemi operativi Windows e, essendo una soluzione proprietaria, non è possibile estenderne le funzionalità intervenendo sul codice sorgente, tuttavia può dialogare con software di terze parti e gestire la condivisione dell'autenticazione tra reami di sicurezza differenti. AD FS è in grado di raccogliere informazioni da una grande varietà di fonti, in particolare dai servizi Active Directory tramite LDAP, per generare *claims* da inoltrare al SP; è anche in grado, tramite un componente chiamato Access Control Service, di ottenere informazioni tramite *claims* rilasciati da servizi fidati, che si fanno carico di autenticare l'utente. Tra i servizi che possono essere inclusi vi sono Google Identity Platform, che utilizza i token OpenID Connect, e Facebook Login, il quale si basa su uno standard simile basato su JWT.

Un ulteriore standard usato per la descrizione di soggetti per mezzo di attributi è **SCIM** (System for Cross-domain Identity Management), una specifica studiata per facilitare la condivisione e lo scambio di informazioni di utenti tra sistemi differenti. Lo standard, descritto nella RFC-7643 [16], permette di descrivere generiche "risorse" per mezzo di documenti JSON; esistono diversi tipi di risorse, come quelle che descrivono

utenti o gruppi di soggetti, ognuna delle quali è descritta da uno schema che specifica quali attributi presenta. Gli attributi possono essere semplici o complessi, nel qual caso il loro valore consiste in uno o più attributi semplici, garantendo la possibilità di raggrupparli e dando quindi maggiore potenza espressiva al protocollo. Esistono attributi comuni a tutte le risorse, come un identificatore univoco e una serie di metadati che forniscono informazioni sulla risorsa; gli schemi specificano diversi altri attributi che aggiungono informazioni in base al tipo di risorsa. Uno degli schemi più usati è quello che identifica una risorsa di tipo *User*, che contiene molti attributi già visti per descrivere un individuo, come informazioni anagrafiche, recapiti telefonici e indirizzi e-mail; la possibilità di usare attributi multi valore e complessi consente di fornire informazioni aggiuntive rispetto a quelle veicolate dal solo valore, come nel caso del nome, che comprende eventuali titoli o secondi nomi, oppure dell'indirizzo. Lo standard prevede una estensione di questo schema che aggiunge alcuni attributi specifici per gli utenti aziendali, come la posizione nella gerarchia aziendale e il numero di matricola. È possibile costruire nuovi schemi o estendere quelli già esistenti, per aggiungere attributi speciali non presenti. Per la comunicazione tra due provider SCIM, lo standard specifica un protocollo REST HTTP [17, sezione 3] che consente di effettuare tutte le operazioni necessarie per la gestione delle risorse (creazione, lettura, aggiornamento, cancellazione) semplicemente usando i metodi standard HTTP.

Una ulteriore alternativa a Shibboleth come IdP è **$\mathbf{WSO_2}$ Identity Server**, un software per la gestione delle identità digitali open source con molteplici funzionalità. È in grado di fornire supporto per la creazione, l'amministrazione e l'utilizzo di account su molteplici piattaforme, interagendo con numerosi protocolli, tra cui SCIM, SAML e OpenID Connect; oltre a poter essere configurato come un semplice IdP, può fungere da bridge tra diversi protocolli, consentendo di usare applicazioni esterne per l'autenticazione degli utenti. Grazie al supporto del protocollo XACML, può fornire decisioni di autorizzazione, agendo sia come Policy Access Point che come Policy Decision Point. WSO_2 Identity Server viene distribuito come applicativo Java che può essere configurato tramite browser grazie ad una console di gestione. Il componente che integra l'IdP supporta i profili SAML *Web SSO* e *Artifact Resolution*, includendo anche la possibilità di effettuare il Single Logout; tuttavia, i metodi di autenticazione locali supportati comprendono esclusivamente l'uso di username/password, Integrated Windows Authentication (autenticazione tramite credenziali Windows) e FIDO (Fast IDentity Online). Per usufruire di altri sistemi di autenticazione è necessario utilizzare un IdP separato a cui inoltrare le richieste. Il software è molto complesso, dati i numerosi compiti che assolve, per cui risulta meno facile da estendere rispetto a Shibboleth o SimpleSAMLphp, anche se è possibile modificare i componenti che lo costituiscono. Grazie al supporto per SCIM è possibile gestire facilmente gli attributi degli utenti registrati presso l'IdP, così come è possibile recuperate tali attributi tramite la stessa interfaccia.

Oltre alle alternative precedentemente menzionate, ne esistono diverse altre, con funzionalità e complessità differenti. In definitiva, la scelta di usare Shibboleth per questo lavoro è motivata da una serie di ragioni diverse, prima fra tutte la profonda configurabilità e estensibilità, che consente, come si vedrà, di adattare il software alle necessità riscontrate con poche modifiche. Shibboleth rappresenta un punto di riferimento nel panorama delle soluzioni SSO, vantando una notevole base di installazioni e un'esperienza decennale che garantiscono al software maturità ed affidabilità. Un dettaglio importante è rappresentato dalla comunità di sviluppatori che collabora al suo sviluppo e che è in grado di fornire un notevole supporto tecnico a fronte di una documentazione non sempre dettagliata e aggiornata. Infine, risulta essere uno dei software più completi per quanto riguarda la gestione dei profili SAML, anche di quelli poco usati come *Attribute Query*, particolarmente utile per implementare un possibile AP.

Capitolo 3

Sviluppo di un Attribute Provider eIDAS

3.1 Ubicazione e informazioni necessarie

Per poter integrare degli AP all'interno dell'architettura eID eIDAS è necessario definire la loro collocazione nel sistema e le relazioni con gli elementi già presenti. Appare evidente che gli AP siano da collocarsi al fianco degli IdP all'interno dei sistemi di identità digitale collegati attraverso la rete eIDAS. Molti di questi sistemi già prevedono la possibilità, durante il processo di autenticazione, di integrare le informazioni fornite dagli IdP attraverso l'interrogazione di enti diversi utilizzando tecniche differenti: nel caso di SPID, ad esempio, i SP possono, tramite il messaggi `AttributeQuery` SAML, richiedere attributi aggiuntivi agli AP registrati, previo consenso dell'utente. Nel sistema eID eIDAS, le entità che avranno il compito di comunicare con gli AP saranno gli eIDAS-Service, sia Middleware che Proxy, in quanto fungono da punto di contatto tra la rete eIDAS e i sistemi di identità digitale dei singoli Paesi.

La prima cosa di cui occorre tenere conto per l'integrazione di un AP nel sistema eID eIDAS è lo Stato in cui esso si trova rispetto a quello in cui è avvenuta l'autenticazione: il caso in cui essi non coincidano presenta problematiche più complesse, una su tutte la necessità di modificare l'eIDAS-Connector per consentire la scelta del Paese presso cui l'AP si trova e comunicare all'eIDAS-Service l'esigenza di recuperare degli attributi aggiuntivi. Il protocollo di comunicazione tra nodi eIDAS [5, sezione 3.2] prevede esclusivamente l'uso del profilo SAML Web SSO, pertanto non è possibile trasmettere richiesta di soli attributi all'interno della rete. Sarebbe necessario modificare tale protocollo per includere anche messaggi come `AttributeQuery` o adattare in qualche modo il profilo utilizzato; entrambe le soluzioni richiedono, però, una revisione dello standard. Nel seguito si farà l'assunzione che l'autenticazione dell'utente e l'eventuale recupero di attributi aggiuntivi avvenga all'interno dello stesso Stato: sebbene ciò possa essere limitante in alcuni casi, rappresenta un caso d'uso comune e un buon punto di partenza per poter analizzare le diverse possibilità di integrazione.

Avendo ipotizzato che i diversi AP si trovano nello stesso Paese dell'IdP, le modifiche alla rete eID eIDAS riguardano esclusivamente l'eIDAS-Service, in particolare il componente che si occupa di gestire la risposta del sistema di identità digitale contenente l'asserzione di autenticazione prima che quest'ultima sia inoltrata all'eIDAS-Connector. Nella richiesta ricevuta dall'eIDAS-Service è contenuta la lista di attributi da recuperare [8, sezione 2.3.2],

insieme con l'indicazione di quali siano obbligatori: il nodo in questione, dopo aver ricevuto dall'IdP l'asserzione di autenticazione dell'utente, potrà stabilire autonomamente se è necessario recuperare ulteriori attributi interrogando uno o più AP. È indispensabile che l'eIDAS-Service possa recuperare una lista degli AP autorizzati a rilasciare attributi sul soggetto; tale lista può essere fornita sotto forma di un documento presente all'interno del nodo stesso, anche se questa soluzione richiede l'aggiornamento manuale di tale documento ogni volta che ve ne sia bisogno, operazione complessa se lo Stato fornisce un eIDAS-Service di tipo Middleware. Un sistema più efficiente consiste nell'uso di un server a cui l'eIDAS-Service può richiedere la lista aggiornata, ad esempio tramite un'interfaccia REST. Tale server deve essere gestito separatamente dall'eIDAS-Service, richiedendo quindi investimenti aggiuntivi, ed è indispensabile che le comunicazioni tra i due siano protette mediante un canale TLS sicuro, per garantire integrità e riservatezza; la lista, inoltre, dovrebbe essere firmata dal server, come ulteriore garanzia della sua integrità e della sua origine. Una precauzione aggiuntiva potrebbe essere l'uso di autenticazione client da parte dell'eIDAS-Service al momento dell'instaurazione del canale TLS: sebbene questa soluzione aggiunga ulteriore complessità, consentirebbe di garantire l'accesso alla lista ai soli nodi eIDAS autorizzati. Per ogni AP, la lista dovrà almeno contenere le informazioni necessarie all'eIDAS-Service per contattarlo, sotto forma di Metadati SAML; un'alternativa consiste nell'inserire al loro posto una URL presso cui recuperarli, insieme con un certificato X.509 da usare per verificare la loro autenticità e integrità. Quest'ultima soluzione garantisce maggiore flessibilità, in quanto gli AP sono gestiti da soggetti differenti che, in questo modo, possono modificare facilmente le informazioni rilasciate all'eIDAS-Service; naturalmente, è necessario che la chiave privata usata per firmare i Metadati corrisponda al certificato presente nella lista.

All'interno dei Metadati di ogni AP dovrebbe essere inserita la lista di attributi che esso è autorizzato a rilasciare, in modo da permettere all'eIDAS-Service di stabilire se tale AP può essere contattato per recuperare gli attributi mancanti. Nel caso in cui i Metadati vengano recuperati in altro modo, gli attributi rilasciati da ciascun AP potrebbero essere inseriti all'interno della lista recuperata dall'eIDAS-Service, per evitare che quest'ultimo debba recuperare i Metadati di tutti gli AP; questa soluzione, tuttavia, aggiunge ridondanza e limita l'utilità di avere i Metadati separati dalla lista. Una soluzione alternativa consisterebbe nel permettere all'eIDAS-Service di mantenere i Metadati in una cache fino alla loro scadenza, in modo da velocizzare le operazioni e rendere disponibile la lista di attributi. Avere a disposizione tale lista per i diversi AP consentirebbe all'eIDAS-Service di presentare all'utente, per ogni attributo mancante, solo i provider che possono fornirlo; questo ridurrebbe al minimo l'interazione richiesta all'utente nel caso in cui gli attributi vadano recuperati da AP diversi.

Un ulteriore problema che può essere risolto tramite il web service che fornisce la lista di AP è la corrispondenza tra l'identità dell'utente ricavata dall'IdP e il modo con cui lo stesso utente è identificato sugli AP; questi sono gestiti da enti diversi, i quali avranno modalità differenti per poter identificare i loro utenti. Basti pensare alle università, che utilizzano delle matricole con formato differente tra i diversi atenei, oppure la Pubblica Amministrazione che, in Italia, spesso usa il codice fiscale come identificativo. Se il sistema di identità digitale utilizza lo standard SAML 2.0, quando l'eIDAS-Service riceve dall'IdP l'asserzione contenete l'identità dell'utente esso viene identificato tramite il campo `NameID`, che molto spesso contiene solo un codice temporaneo: bisogna quindi usare le informazioni contenute negli attributi per poter costruire l'identificativo con il quale l'utente è registrato presso gli AP. La soluzione più banale sarebbe quella di inoltrare tutti gli attributi agli AP al momento della richiesta, in modo che siano essi stessi a ricavare tale ID; questo, però, porterebbe potenziali problemi di riservatezza e di privacy, tenendo anche conto che nella

maggior parte dei casi bastano solo alcuni degli attributi per poter ottenere l'identificativo. Per evitare che le informazioni sull'utente vengano trasmesse indiscriminatamente agli AP è necessario che sia l'eIDAS-Service stesso a generare l'ID corretto per ogni richiesta: bisogna, quindi, istruirlo su quali attributi combinare e come comporli. Queste istruzioni possono essere fornite dal web service insieme con la lista, per semplificare l'inserimento di nuovi AP con le loro regole. È necessario che l'identificativo dell'utente presso l'AP sia ricavabile a partire dagli attributi forniti dall'IdP, in particolare quelli che ricadono nell'elenco degli attributi obbligatori previsti dalla normativa eIDAS (vedi Tabella 1.1).

3.2 Protocollo di comunicazione

Dopo aver stabilito le modalità con cui l'eIDAS-Service "scopre" quale AP contattare e dove contattarlo, è necessario definire come avviene la comunicazione tra i due. Nella scelta del protocollo da adottare, si devono tener presente quali sono i requisiti da garantire:

autenticità la fonte dei dati recuperati deve poter essere verificata con certezza;

integrità eventuali tentativi di modifica non autorizzati durante la trasmissione dei dati devono essere individuabili;

confidenzialità le informazioni sull'utente, in particolare i valori degli attributi recuperati, devono essere protetti da accessi non autorizzati durante la trasmissione;

controllo l'utente deve avere il pieno controllo sulle informazioni rilasciate dagli AP, decidendo quali attributi rilasciare e quali omettere;

interoperabilità il protocollo utilizzato deve essere compatibile con il resto del sistema e, per quanto possibile, minimizzare l'impatto relativo alla sua implementazione.

Poiché la maggior parte dei sistemi di identità digitale ed eIDAS stesso si affidano al protocollo SAML per la comunicazione tra i diversi attori, risulta naturale che esso venga scelto come potenziale protocollo per permettere all'eIDAS-Service di dialogare con gli AP; SAML è uno standard *de facto* e, in più, fornisce un profilo specifico per la richiesta e lo scambio di attributi. È inoltre possibile usare i Metadati SAML nel modo descritto in precedenza per condividere informazioni sugli AP e sugli attributi da loro rilasciati. Sarebbe certamente possibile sfruttare altri protocolli tra quelli descritti nella sezione 2.4, tuttavia questo richiederebbe ulteriori sforzi per convertire gli attributi e i loro valori secondo le regole del protocollo SAML per poter essere gestiti all'interno della rete eID eIDAS; inoltre, sarebbe necessario rimpiazzare o estendere Shibboleth come software di gestione degli attributi presso l'AP, in quanto è nativamente progettato per il protocollo SAML.

3.2.1 Attribute Query

Per effettuare richieste di asserzioni esistenti o interrogazioni sulla base di un soggetto, SAML mette a disposizione il profilo **Assertion Query/Request**, il quale prevede due fasi: nella prima, il richiedente invia una interrogazione per mezzo di un messaggio inviato direttamente al provider tramite binding SOAP. Il tipo della richiesta dipende dalle informazioni da recuperare, come un'asserzione di autenticazione, una lista di attributi o una decisione di autorizzazione. Dopo aver processato la richiesta, il provider risponde con un

messaggio di tipo `Response` con le informazioni richieste o con un codice di errore, nel caso la richiesta non possa essere evasa. Per poter ottenere i valori di certi attributi, il richiedente deve inviare un messaggio di tipo `AttributeQuery` nel quale specificare il soggetto a cui si riferisce la richiesta, sotto forma di NameID; è poi possibile specificare una lista di attributi da recuperare oppure lasciare che il provider rilasci tutti gli attributi disponibili per quel soggetto. L'invio dei messaggi `AttributeQuery` avviene mediante l'uso del metodo POST HTTP su di uno specifico end-point tipicamente collocato su una diversa porta del provider rispetto a quella usata per le richieste di autenticazione. Il motivo dell'uso di questo secondo canale, denominato *"back-channel"*, deriva dal meccanismo storicamente usato per autenticare le richieste di questo tipo: la connessione viene protetta mediante l'uso di un canale TLS e viene richiesta l'autenticazione tramite certificato client per poter identificare il mittente del messaggio. Questo tipo di autenticazione sarebbe impossibile da attuare se le richieste dirette avvenissero sulla stessa porta a cui si collegano gli utenti per inoltrare le richieste di autenticazione.

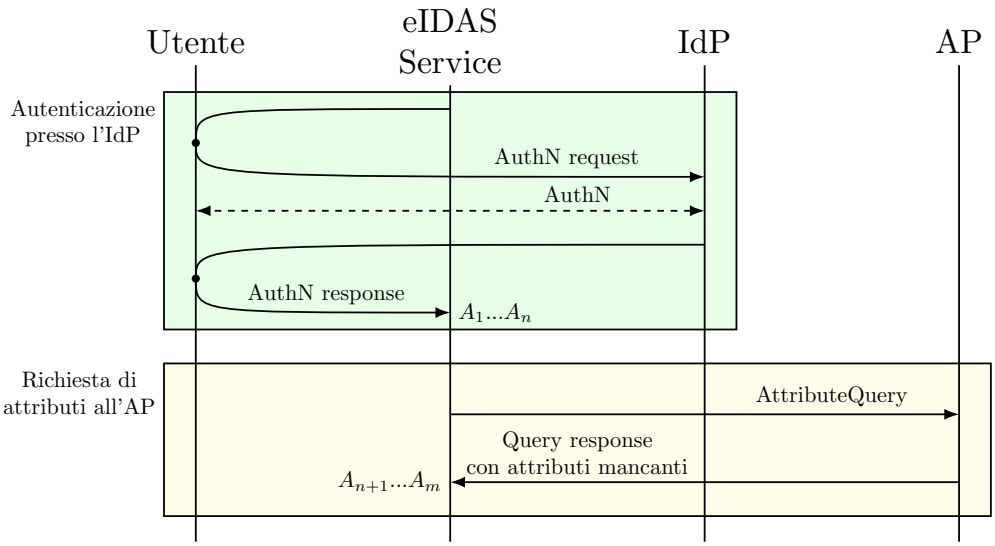


Figura 3.1: Interrogazione AP mediante `AttributeQuery`

Le richieste SAML per gli attributi attraverso il back-channel sono state introdotte fin dalla prima versione dello standard, quando l'invio di attributi durante la fase di autenticazione era sconsigliato e spesso disabilitato; tale scelta era dovuta alla mancanza della possibilità di cifrare il contenuto del messaggio, che è stata introdotta a partire dalla versione 2.0. Questo comportava che le informazioni trasportare, anche utilizzando dei canali TLS cifrati, passassero in chiaro attraverso il client, che risultava essere un punto altamente vulnerabile del sistema. Con l'avvento della nuova versione del protocollo, che risolve tale problema, l'uso di `AttributeQuery` è caduto in disuso, anche se tali richieste continuano ad essere supportate sia dal protocollo che da molti software, compreso Shibboleth. La riservatezza delle informazioni può quindi essere garantita cifrando l'asserzione contenuta nella risposta e utilizzando una chipher suite adeguata per il canale TLS; sia la richiesta che la risposta dovrebbero, poi, essere firmate dai provider, per garantirne l'integrità.

L'utilizzo di `AttributeQuery` come protocollo di comunicazione tra eIDAS-Service e AP, sebbene sembri logicamente la scelta migliore, porta con sé alcune problematiche che ne minano l'utilità e spingono a prendere in considerazione strade alternative. Il profilo richiede che, per lo scambio dei messaggi, venga adottato il binding SOAP invece di

quello HTTP POST [11, sezione 6.3]: questo comporta che i messaggi SAML vengano incapsulati all'interno di un messaggio SOAP, il quale sarà trasportato nel corpo di una richiesta o di una risposta HTTP. Il messaggio SOAP è scritto in linguaggio XML e, quindi, aggiunge complessità alla comunicazione; l'ulteriore elaborazione richiesta per estrarre il messaggio SAML necessita di un componente aggiuntivo, rischiando di aumentare la superficie di attacco. Alcuni software, come SimpleSAMLphp, non supportano tale binding, riducendo l'interoperabilità di questa soluzione. Una problematica minore riguarda l'uso del NameID come mezzo per identificare l'utente: molto spesso i software che forniscono servizi SSO tramite SAML generano un NameID temporaneo per identificare un utente appena autenticato. Questo comporta che l'interrogazione tramite `AttributeQuery` possa avvenire solo a seguito dell'autenticazione dell'utente in questione presso l'AP e la successiva assegnazione di un NameID temporaneo, cosa che richiederebbe ulteriori passaggi e complicherebbe notevolmente il protocollo. La soluzione più efficace è quella di usare, presso l'AP, identificativi non temporanei che possano essere ricavati dagli attributi in possesso dell'eIDAS-Service.

Una caratteristica fondamentale della richiesta di attributi tramite `AttributeQuery` è che essa avviene senza alcun coinvolgimento dell'utente: la richiesta, essendo inoltrata tramite il back-channel, arriva all'AP direttamente e quest'ultimo la gestisce senza bisogno di interazione con l'utente. Questo meccanismo ha il vantaggio di essere semplice e veloce, ma non lascia all'utente nessun controllo sul rilascio dei suoi attributi: l'eIDAS-Service, una volta generato l'identificativo dell'utente, può richiedere agli AP qualsiasi attributo, anche se non richiesto dal SP. La mancanza della necessità di autorizzazione nel rilascio degli attributi genera negli eIDAS-Service una grande responsabilità e richiede che l'utente accordi la propria fiducia in essi; sarebbe meglio, perciò, che nel protocollo di comunicazione tra eIDAS-Service e AP fosse prevista una qualche forma di autorizzazione che garantisca a questi ultimi che l'utente in questione acconsenta al rilascio degli attributi richiesti.

3.2.2 Authentication Request

Un AP ha il compito di rilasciare attributi richiesti da un SP (nel nostro caso l'eIDAS-Service) dopo aver correttamente identificato l'utente associato alla richiesta; tale scopo è simile a quello di un IdP, con la differenza che l'AP non necessita di autenticare l'utente, in quanto la sua identità è già stata provata precedentemente. Questa considerazione suggerisce che i messaggi `AuthnRequest` e le rispettive risposte, tipicamente usati per richiedere l'autenticazione di un utente, possano fornire un mezzo per richiedere e trasportare gli attributi presenti su un AP. L'uso del profilo **Browser SSO** di SAML presuppone, però, che l'AP emetta una asserzione di autenticazione nei confronti dell'utente, diventando un IdP a tutti gli effetti. Poiché l'utente è già stato autenticato presso il sistema di identità digitale del suo Paese, si può sfruttare il meccanismo SSO di tale sistema per provare la sua identità anche presso l'AP: quest'ultimo, alla ricezione dell'`AuthnRequest`, può "delegare" l'autenticazione del soggetto emettendo a sua volta un messaggio analogo indirizzato all'IdP del sistema di identità digitale che ha autenticato precedentemente l'utente. L'IdP emetterà, quindi, una asserzione in cui dichiara l'identità del soggetto, in modo che l'AP possa rilasciare a sua volta una asserzione di autenticazione nella quale inserirà gli attributi richiesti.

Utilizzando questa strategia, l'AP si comporterebbe in maniera analoga ad un IdP agli occhi dell'eIDAS-Service, in quanto esso riceverebbe una nuova asserzione che attesta l'identità dell'utente accompagnata da un nuovo insieme di attributi; questo consentirebbe

di ridurre al minimo le modifiche da apportare all'eIDAS-Service, in quanto è già predisposto per lavorare con il profilo Web Browser SSO. Verso il sistema di identità digitale, invece, l'AP apparirebbe come un SP che richiede la verifica dell'identità di un utente, il quale, essendosi autenticato poco tempo prima sullo stesso IdP, potrebbe non essere costretto a provare nuovamente la propria identità, sfruttando la sessione già aperta.

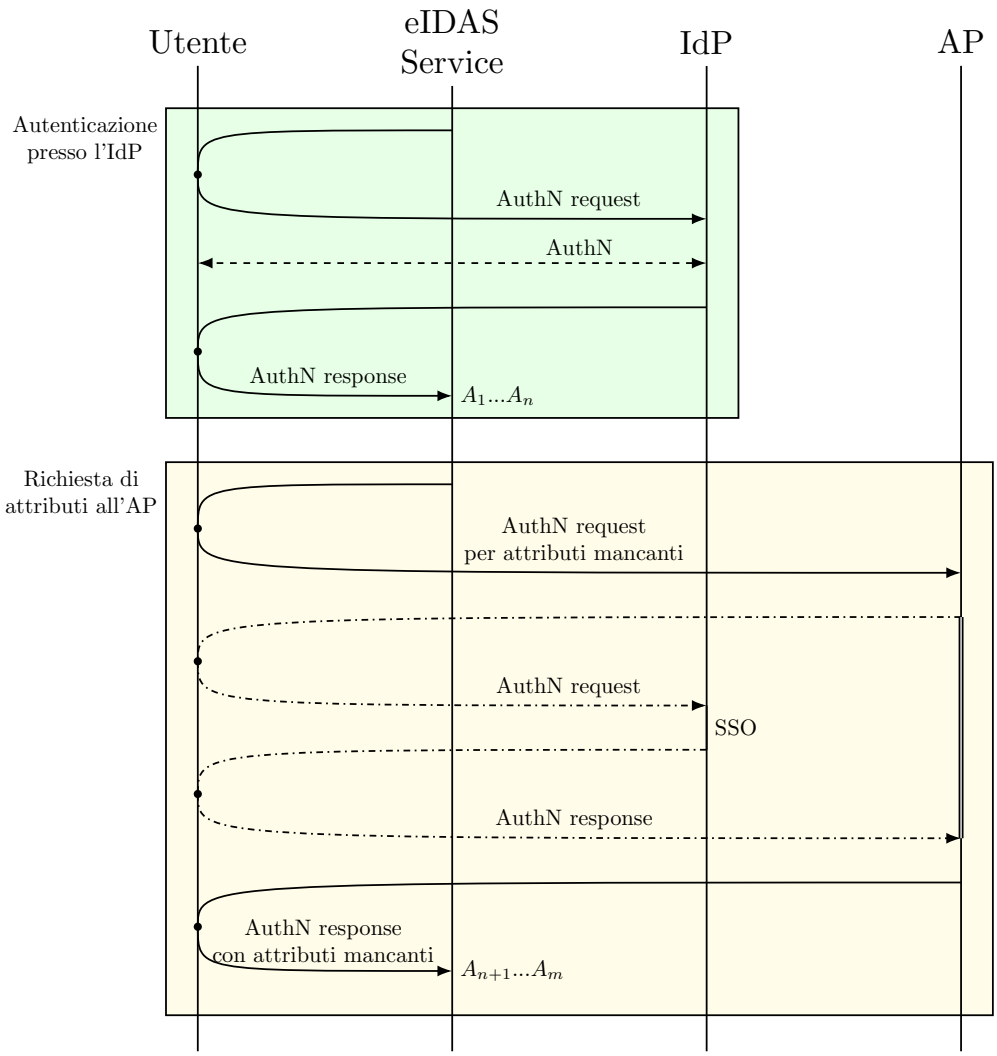


Figura 3.2: Interrogazione AP mediante `AuthnRequest`

Uno dei flussi di autenticazione messi a disposizione dall'IdP Shibboleth, chiamato **External**, consente di usare un qualsiasi meccanismo di autenticazione esterno all'IdP stesso definendo la risorsa presso cui tale meccanismo si trova e sulla quale redirigere l'utente. Usando le API fornite da Shibboleth si possono gestire i risultati ottenuti dall'autenticazione esterna, tipicamente memorizzati in delle variabili CGI usate per recuperare le informazioni necessarie da trametterе ai passaggi successivi. Una delle variabili più importanti in questa fase è *REMOTE_USER*, la quale viene tipicamente valorizzata con l'identificativo o il nome utente del soggetto; tale variabile è fondamentale per capire se l'autenticazione è avvenuta correttamente e se è possibile stabilire una sessione per l'utente. Un AP che utilizzi i messaggi di `AuthnRequest` attraverso Shibboleth può essere ottenuto implementando sullo stesso server un IdP e un SP: il primo riceverà la richiesta dall'eIDAS-Service e, attraverso il flusso di autenticazione External, si affiderà al SP per

l'autenticazione. Quest'ultimo farà affidamento sul sistema di identità digitale del Paese e, ottenuta l'asserzione, comunicherà al primo modulo il risultato, in modo che questo possa propagarlo, con i nuovi attributi, all'eIDAS-Service.

Il vantaggio di questa soluzione rispetto all'utilizzo del profilo Attribute Query è la possibilità, per l'AP, di stabilire se l'eIDAS-Service è autorizzato a richiedere gli attributi per un certo utente: potendo interagire con esso, è possibile informarlo di quali attributi vengono effettivamente rilasciati. Inoltre, avendo a disposizione l'asserzione fornita dall'IdP, l'AP può verificare che essa si riferisca allo stesso soggetto per cui vengono richiesti gli attributi, in modo che non sia possibile usare l'autenticazione di un utente per ottenere gli attributi di un'altro. L'uso di questo protocollo consente di sfruttarne le tecniche di integrità e riservatezza derivanti dalla cifratura delle asserzioni e dalla firma dei messaggi, oltre all'uso, consigliabile, di un canale TLS sicuro per la trasmissione.

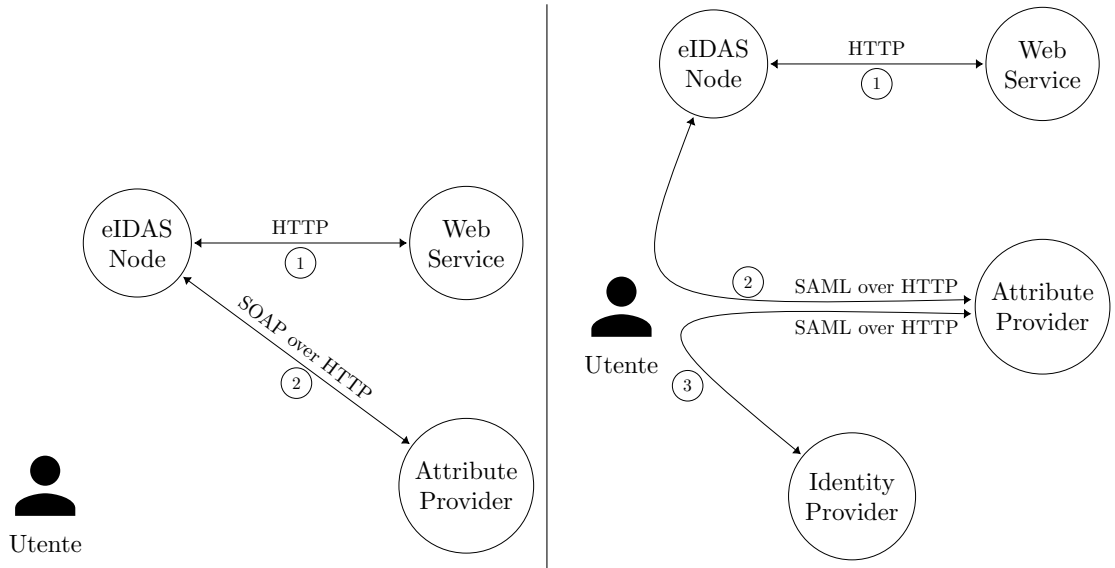
La comunicazione tra eIDAS-Service e AP attraverso il profilo Web Browser SSO richiede un maggior numero di passaggi per poter essere completata rispetto all'uso di Attribute Query, dato che, per autenticare l'utente, è necessario coinvolgere il sistema di identità digitale presso cui esso è registrato. L'utente viene rediretto più volte e, nella peggiore delle ipotesi, ad ogni passaggio può essere richiesto il suo intervento: l'eIDAS-Service può richiedere quale AP contattare, il quale potrebbe domandare se rilasciare gli attributi; infine, il sistema di identità digitale potrebbe richiedere una nuova autenticazione. Bisogna anche tenere presente che le cose si complicano ancora di più nel caso in cui l'AP si trovi in uno Stato diverso da quello in cui l'utente si è autenticato: sarebbe necessario usare la rete eIDAS per trasportare la richiesta di autenticazione e la successiva risposta, aggiungendo ulteriori passaggi e aumentando così la superficie di attacco.

I messaggi `AuthnRequest` non consentono di specificare quali attributi vengano richiesti insieme all'autenticazione: questi sono stabiliti nella configurazione dell'IdP a partire dall'identità del mittente della richiesta. Nel protocollo SAML adottato da eIDAS, gli attributi richiesti da un eIDAS-Connector vengono inseriti all'interno di un elemento `Extensions` [18, serzione 3.2.1], previsto dallo standard per aggiungere informazioni ai messaggi. È possibile seguire lo stesso metodo per inoltrare gli attributi richiesti all'AP, avendo cura di adattare Shibboleth in modo che possa gestire tale estensione e rilasci solo gli elementi richiesti. Secondo lo standard SAML nel messaggio `AuthnRequest` è possibile specificare il soggetto per il quale si richiede l'autenticazione [18, sezione 3.4.1]; questo campo può essere usato per indicare l'identificativo dell'utente costruito a partire dagli attributi a disposizione.

3.3 Possibili attacchi

Come già evidenziato in precedenza, tra le proprietà da soddisfare nell'integrare un AP nella rete eID eIDAS vi sono l'autenticità e la riservatezza dei dati degli utenti; in generale, bisogna fare in modo che la relazione di fiducia che intercorre tra l'utente, il SP, la rete eIDAS e l'IdP possa essere estesa anche agli AP. È necessario, quindi, analizzare i possibili attacchi che possono essere portati ai diversi elementi in gioco per poter predisporre adeguate contromisure. Le modifiche da apportare all'eIDAS-Service devono essere studiate in modo da non interferire con le normali funzionalità e non introdurre falle che possano portare ad un'eventuale compromissione del nodo. Allo stesso modo, nel progettare ed implementare l'AP e l'eventuale Web Service di supporto devono essere rispettati i fondamenti dell'ingegneria del software, prestando particolare attenzione alla fase di testing e riusando il più possibile software preesistente di comprovata affidabilità.

Il Web Service, il cui compito è fornire all'eIDAS-Service la lista con le informazioni sugli AP, sebbene a prima vista possa sembrare un elemento secondario, riveste un ruolo fondamentale nell'instaurare la relazione di fiducia tra gli AP e l'eIDAS-Service: quest'ultimo, infatti, riconoscerà come validi solo e soltanto gli AP inclusi in tale lista. Un'eventuale compromissione del Web Service potrebbe comportare l'accreditamento di un AP malevolo come fidato, con la possibilità, per un eventuale attaccante, di rilasciare qualsiasi attributo egli desideri, senza la necessità di compromettere un AP già esistente. Lo stesso effetto si avrebbe se un attaccante riuscisse a dirottare verso un Web Service da lui controllato le richieste dell'eIDAS-Service, nel caso in cui quest'ultimo non verificasse la sua identità; ancora, se non viene implementata una soluzione per verificare l'integrità dei dati ricevuti, si può portare avanti un attacco Man in the Middle per alterare la lista durante il suo transito nella Rete.



(a) Attribute Query

(b) Web Browser SSO

Figura 3.3: Comunicazioni successive all'autenticazione

Il Web Service dovrebbe esser quindi implementato in modo da impedire accessi non autorizzati e mantenendo un monitoraggio costante, in modo da rilevare immediatamente ogni possibile attacco. Per garantire l'identità del Web Server nella comunicazione con l'eIDAS-Service (Figura 3.3, punto 1) può essere usato un certificato a chiave pubblica, la cui corrispondente chiave privata viene mantenuta dal Web Service stesso; tale certificato deve essere firmato tramite una PKI la cui radice sia nota all'eIDAS-Service, in modo che esso possa essere usato come certificato server per instaurare un canale TLS. L'uso di TLS sarebbe utile anche per altri motivi, primo fra tutti l'integrità e la riservatezza delle informazioni inviate dal Web Service: per far questo, è necessario che le cipher suite consentite durante la negoziazione prevedano algoritmi di firma e cifratura robusti e con chiavi di lunghezza adeguata. A tal proposito, i requisiti presenti in [7, sezione 2] rappresentano un buon esempio delle richieste da soddisfare per ottenere un buon grado di sicurezza per il canale. La comunicazione deve prevedere anche l'autenticazione dell'eIDAS-Service, in modo che i dati relativi agli AP non vengano rilasciati a soggetti non autorizzati; per far questo, la soluzione più efficace consiste nell'uso dell'autenticazione client TLS, in modo che il Web Service possa verificare l'identità del suo interlocutore ancor prima di ricevere la richiesta. Per garantire ulteriormente l'integrità e l'autenticità della lista trasmessa all'eIDAS-Service, questa può essere firmata utilizzando un secondo certificato che abbia

una durata di validità ridotta, il quale andrebbe aggiornato con cadenza giornaliera o settimanale: questo consentirebbe di ridurre il rischio di compromissione della chiave usata per la firma. A seconda del formato con cui è codificata la lista, potrebbe essere necessario incapsularla all'interno di una busta che consenta di apporvi la firma, ad esempio seguendo lo standard RFC-5652 Cryptographic Message Syntax (CMS) con contenuto di tipo *Signed-Data* [19, sezione 5].

Nel caso in cui venga utilizzato il profilo Attribute Query (Figura 3.3a), la comunicazione tra eIDAS-Service e AP avviene attraverso uno scambio di messaggi diretto tra i due (punto 2): la query e la successiva risposta SAML vengono incapsulate all'interno di un *envelope* SOAP e trasportate tramite il protocollo HTTP. Per tale comunicazione possono essere fatte considerazioni simili a quelle espresse per quella tra eIDAS-Service e Web Service: anche in questo caso è indispensabile proteggere la comunicazione con l'uso di un canale TLS sicuro, possibilmente utilizzando anche l'autenticazione client per autenticare l'eIDAS-Service. I messaggi SAML scambiati dovrebbero essere firmati e l'eventuale asserzione prodotta dall'AP cifrata, per garantire un'ulteriore livello di integrità e riservatezza. In questa soluzione, l'autenticazione del mittente è essenziale, in quanto l'AP non ha alcun modo di verificare il consenso dell'utente al rilascio degli attributi, dovendosi fidare della buona fede dell'eIDAS-Service; nel caso in cui un attaccante riesca a comprometterlo o a sostituirsi ad esso, sarebbe in grado di ottenere gli attributi di un qualsiasi utente semplicemente conoscendone l'identificativo presso l'AP. La comunicazione avviene utilizzando il binding SOAP over HTTP, nel quale il messaggio SAML viene incapsulato all'interno del body di *envelope* SOAP; quest'ultimo non ha alcuna funzione se non quella di fungere da contenitore in quanto lo standard SAML non richiede alcun header SOAP particolare per tale binding. Sebbene l'aggiunta dell'imbustamento SOAP aumenti la complessità ed esponga un ulteriore punto sensibile dei provider, il limitato uso che si fa del protocollo rende difficile l'ipotesi che possa essere sfruttato per compromettere l'integrità del sistema.

L'uso del profilo Web Browser SSO (Figura 3.3b) richiede più passaggi e attori rispetto al precedente, aumentando in questo modo la superficie di attacco ma, contemporaneamente, aggiungendo una verifica aggiuntiva sull'identità dell'utente e sulla sua autorizzazione al rilascio degli attributi. La comunicazione tra eIDAS-Service e AP (punto 2) avviene indirettamente tramite l'utente che trasporta i messaggi all'interno di richieste POST HTTP oppure tramite Redirect: per questo motivo non è possibile contare sulla autenticazione client TLS per identificare l'eIDAS-Service ma è necessario che la richiesta sia firmata. Le comunicazioni tra l'utente e i vari attori devono essere protette tramite TLS con cipher suite adeguate; nel caso in cui il browser dell'utente non sia sufficientemente aggiornato da supportare i requisiti di sicurezza richiesti, sarebbe opportuno negare l'accesso al servizio, spiegando il motivo del fallimento dell'operazione, anche se l'usabilità del sistema verrebbe ridotta. Il terminale dell'utente rappresenta un punto critico del sistema, in quanto è il nodo più vulnerabile e, essendo l'end-point dei canali TLS, le informazioni scambiate tra gli attori vi passano in chiaro; per questi motivi, oltre a firmare i messaggi SAML, è opportuno che l'asserzione rilasciata dall'AP venga cifrata, in modo da mantenere la riservatezza degli attributi anche al di fuori del canale TLS.

Nel caso della comunicazione tra l'AP e il sistema di identità digitale (punto 3), il primo si comporta a tutti gli effetti come un SP che richiede l'autenticazione dell'utente; nella maggior parte dei casi, dato che in precedenza quest'ultimo si è autenticato presso lo stesso IdP, verrà rilasciata una asserzione senza bisogno, per il soggetto, di provare nuovamente la sua identità. L'AP dovrà rispettare le normative in vigore nello Stato in cui si trova per poter interagire correttamente con il sistema di identità digitale. L'AP, ricevendo l'asserzione dell'IdP, deve verificare che questa riguardi lo stesso soggetto indicato nella

richiesta dell'eIDAS-Service, in modo da evitare il rilascio di attributi appartenenti ad un utente diverso; per far questo, il sistema di identità digitale deve includere nell'asserzione gli attributi necessari per generare l'identificativo dell'utente presso l'AP, in modo che quest'ultimo possa ricostruirlo e confrontarlo con quello presente nella richiesta.

Capitolo 4

Implementazione e analisi della soluzione

4.1 Concetti generali

Prendendo in considerazione quanto si è detto nel capitolo precedente si può delineare una soluzione complessiva che permetta di combinare diversi elementi al fine di consentire alla rete eID eIDAS il recupero di attributi da fonti diverse dall'IdP presso cui l'utente si autentica. Il fulcro di tale soluzione è, senza dubbio, l'AP, che deve essere progettato tenendo conto del suo ruolo all'interno del sistema di identità digitale del Paese in cui si torva; in base a quanto detto precedentemente, l'AP comunicherà con l'eIDAS-Service attraverso messaggi `AuthnRequest`, perciò dovrà essere in grado di richiedere l'autenticazione dell'utente presso tale sistema. La scelta di usare questo meccanismo di comunicazione è data in primo luogo dal fatto che ciò consente di garantire il rilascio degli attributi solo in presenza dell'utente a cui si riferiscono; inoltre, essendo l'eIDAS-Service già progettato per gestire il profilo Web Browser SSO, l'integrazione richiede minori modifiche con un minor rischio di introdurre problemi nel codice che possano influire sulla sicurezza. Accanto agli AP sarà presente un Web Service il cui compito sarà quello di rilasciare le informazioni su di essi e, in particolare, su come costruire gli identificativi degli utenti a partire dagli attributi rilasciati dall'IdP. Tale Web Service dovrebbe essere gestito dall'ente che ha in carico il servizio di identità digitale del Paese, in modo che essa possa raccogliere le informazioni di tutti gli AP e servire tutti gli eIDAS-Service autorizzati.

La base da cui partire per creare il nodo eIDAS a cui aggiungere le funzionalità desiderate è il software di esempio messo a disposizione dalla rete di cooperazione europea[1], il quale consiste in un package Java contenente l'implementazione di un nodo che funga sia da eIDAS-Service che da eIDAS-Connector. All'interno di tale package sono inclusi anche degli strumenti di testing sotto forma di un SP e un IdP che consentono di verificare la corretta installazione e configurazione del nodo. Nell'implementare la soluzione, l'IdP fornito è stato sostituito con un server Shibboleth adeguatamente configurato in modo da simulare un semplice sistema di identità digitale, mentre il SP di prova è stato mantenuto. Per implementare l'AP è stato usato un server sul quale sono stati configurati entrambi i moduli Shibboleth, sia IdP che SP; il primo riceverà le richieste da parte dell'eIDAS-Service, mentre il secondo richiederà l'autenticazione dell'utente all'IdP menzionato in

[1] disponibile all'indirizzo:
https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/eIDAS-Node+-+Current+release

precedenza. Infine è stato implementato un semplice Web Service REST usando la piattaforma Java Spring, che consente all'eIDAS-Service di richiedere la lista di AP disponibili e, per ognuno di essi, le informazioni necessarie a contattarlo.

Per valutare il corretto funzionamento del sistema e verificare i messaggi scambiati tra le parti, i diversi servizi sono stati distribuiti su quattro macchine virtuali diverse, con sistema operativo Debian, messe in comunicazione attraverso una rete LAN virtuale. Sulla prima macchina è stato installato Apache Tomcat, un *application server* per permettere l'esecuzione del SP fornito dalla rete di cooperazione sotto forma di applicazione web; discorso analogo per la seconda macchina virtuale, dove Tomcat viene usato per eseguire il software dell'eIDAS-Connector e dell'eIDAS-Service, i quali, quindi, saranno raggiungibili allo stesso indirizzo. La terza macchina virtuale funge da IdP, sulla quale viene eseguito il web server Jetty che, a sua volta, permette l'esecuzione dell'IdP Shibboleth. L'AP è stato installato sull'ultima macchina, su cui viene eseguito il web server Apache che funge da proxy per l'IdP Shibboleth eseguito all'interno di Jetty; inoltre, tramite il modulo *shibd*, consente all'AP di intercettare le richieste per una particolare risorsa in modo da attivare l'autenticazione tramite SSO presso l'IdP che simula il sistema di identità digitale. La macchina host, anch'essa connessa alla LAN virtuale, contiene il Web Service eseguito all'interno di un server Tomcat; per verificare il funzionamento del sistema, l'host assumerà anche il ruolo della macchina dell'utente che, attraverso un browser web, invierà la richiesta iniziale al SP.

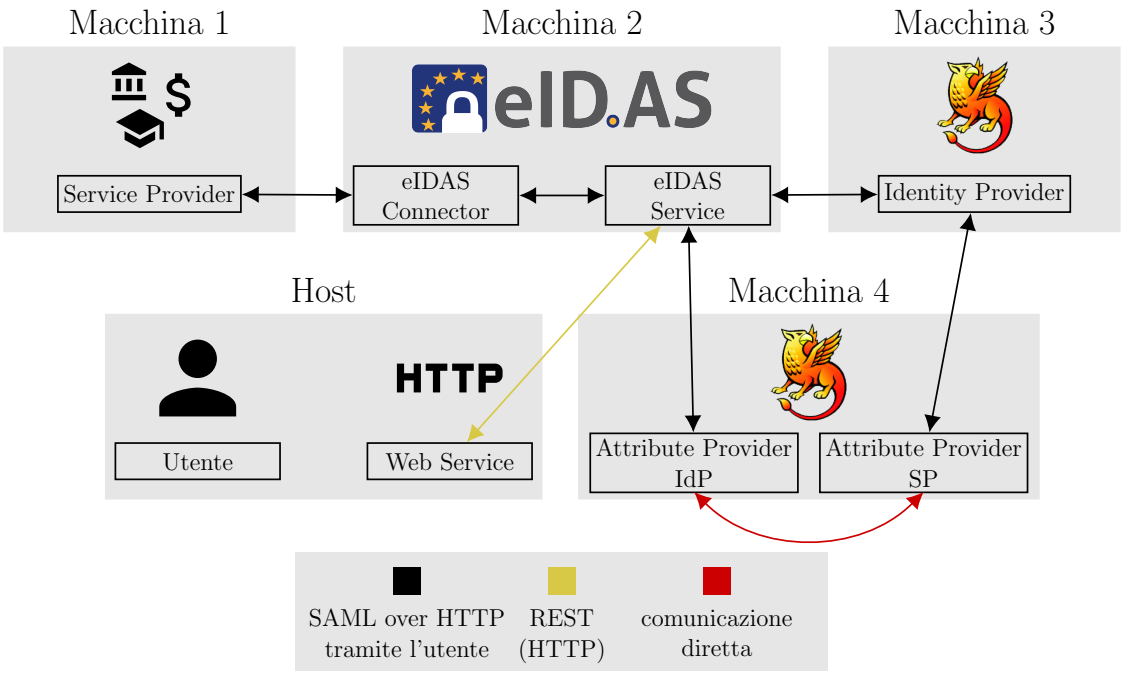


Figura 4.1: Struttura della configurazione implementata

Per garantire l'integrità dei Metadati dei provider SAML e delle informazioni scambiate, oltre alla riservatezza di queste ultime, è stato necessario rilasciare dei certificati X.509 collegati alle chiavi usate negli algoritmi di firma e cifratura asimmetrica. Ogni provider è stato dotato di due coppie di chiavi: la prima coppia viene usata per firmare i propri messaggi e permettere ai destinatari di verificarne la firma, la seconda per consentire agli altri provider di cifrare i messaggi ad esso diretti. I certificati associati a queste chiavi possono essere self-signed, in quanto essi vengono rilasciati all'interno dei Metadati del provider, i quali devono essere firmati da una Certification Authority riconosciuta come

fidata dai provider che ne fanno uso. Nello scenario della rete eIDAS ogni Stato mantiene una propria CA, la quale fa da radice per la catena di certificazione dei Metadati dei suoi nodi; i certificati di tali Authority vengono distribuiti tra gli Stati in modo che possano essere usati dagli altri nodi come certificati radice per la verifica dei Metadati. Per semplificare la distribuzione e la messa in opera del software di test, i Metadati di tutti i provider SAML sono stati firmati usando un unico certificato self-signed. Oltre alle chiavi usate per i messaggi SAML, è necessario che ogni provider sia fornito di un certificato da usare per instaurare il canale TLS con l'utente, in modo da proteggere la comunicazione e dimostrare la propria identità.

4.2 Configurazione preliminare

4.2.1 Service Provider e nodo eIDAS

Per poter applicare le modifiche software in grado di includere l'AP nel sistema, è stato necessario innanzitutto installare e configurare correttamente i diversi programmi. Per prima cosa, su tutte le macchine virtuali sono stati installati i file del Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy, che consentono di rimuovere le limitazioni agli algoritmi crittografici contenuti nella versione base di JCE. Il software eIDAS, una volta compilato seguendo le istruzioni fornite insieme ad esso, genera tre pacchetti: `EidasNode.war`, `SP.war` e `IdP.war`. I primi due file sono stati distribuiti all'interno degli application server sulle macchine virtuali, mentre l'ultimo è stato sostituito dall'IdP Shibboleth. Assieme al software viene fornito anche un gruppo di file di configurazione da copiare in una directory sul disco; per consentire al nodo e al SP di accedervi, il suo percorso è stato assegnato ad una particolare variabile di sistema. Oltre ad i file di configurazione generici, ve ne sono altri che si riferiscono in particolare ai diversi moduli; anch'essi sono stati messi a disposizione del modulo a cui si riferiscono.

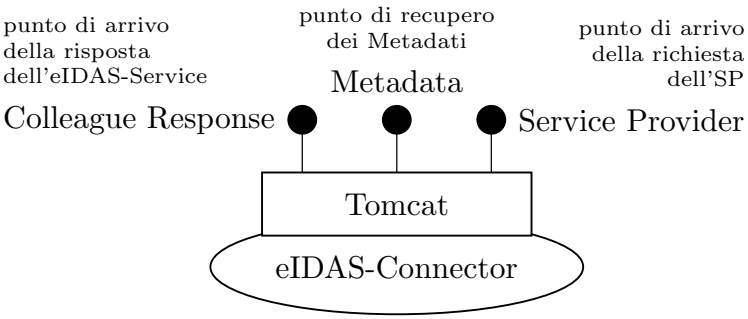


Figura 4.2: Risorse esposte dall'eIDAS-Connector

Dopo aver copiato tutti i file necessari, è stato necessario modificare la configurazione del nodo per adattarla alle nostre esigenze: per far questo, il primo file modificato è `eidas.xml`, che contiene le proprietà generiche e quelle relative alle parti eIDAS del nodo, ovvero quelle che non dipendono dal sistema di identità digitale del Paese. In tale file sono state specificate le URL presso cui il Service e il Connector dovranno essere contattati: in particolare sono stati indicati gli end-point per il recupero dei Metadati da parte dei SP e degli IdP, gli indirizzi a cui inviare le richieste SAML e quelli per le risposte. Nel file sono stati definiti i nomi che identificano Connector e Service, oltre a diverse impostazioni per la sicurezza delle comunicazioni SAML, come gli algoritmi consentiti per la firma e la

cifratura dei dati. Per il Connector sono anche state indicate le informazioni sui Service che è autorizzato a contattare, in modo che possa recuperare i loro Metadati e proporli agli utenti.

Per configurare il SP fornito con il nodo è stato utilizzato il file `sp.properties`, nel quale sono stati indicati i valori delle proprietà che saranno utilizzate durante l'esecuzione: oltre al nome e all'identificativo del SP, è stato indicato il file dal quale recuperare i propri Metadati e l'indirizzo presso il quale renderli disponibili all'eIDAS-Connector. È stato necessario, inoltre, specificare tutti i Connector che il SP è autorizzato a contattare, includendo, per ognuno di essi, l'URL presso cui recuperare i Metadati e quella verso cui inviare la richiesta SAML.

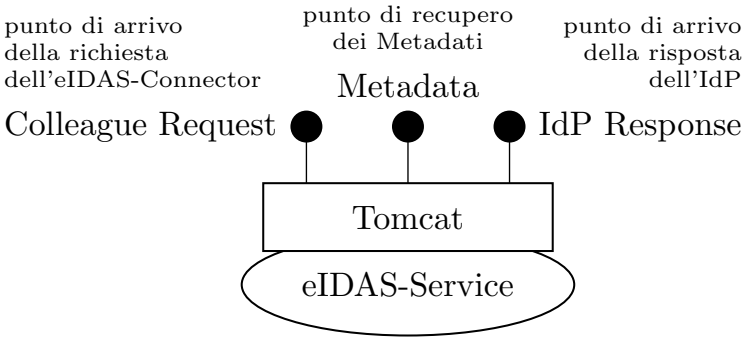


Figura 4.3: Risorse esposte dall'eIDAS-Service

Per consentire al SP, all'eIDAS-Connector e all'eIDAS-Service di firmare i propri messaggi e decifrare le asserzioni ricevute, è stato necessario indicare dove recuperare le chiavi private da usare negli algoritmi. Tali chiavi sono conservate all'interno di KeyStore Java (JKS) protetti da password, i quali, nei nodi eIDAS, contengono anche i certificati fidati rilasciati dai Paesi della rete eIDAS ed utilizzati per verificare le firme dei Metadati ricevuti. Il percorso in cui si trova il file JKS è stato specificato nei file `SignModule_*.xml` e `EncryptModule_*.xml`. Per ogni interfaccia esposta dai tre software sono presenti una o più copie di tali file, in modo da poter specificare diverse chiavi o KeyStore da usare: sul SP è presente una singola coppia che riguarda la comunicazione con i Connector, mentre sull'eIDAS-Connector si trova una coppia con le chiavi da usare all'interno della rete eIDAS, cioè verso gli altri nodi, e altri due file per la comunicazione all'interno dello Stato con i SP. Anche sull'eIDAS-Service si trovano due coppie, una per la connessione con altri nodi e l'altra con le chiavi per la comunicazione con gli IdP del sistema di identità digitale. Poiché in ogni KeyStore possono essere presenti più chiavi, nei file di configurazione, oltre a specificare il loro percorso, è stato necessario indicare anche il numero di serie e l'autorità emittente del certificato associato alla chiave da utilizzare. Nella configurazione del nodo eIDAS è presente un file (`encryptionConf.xml`) che consente di specificare, per ogni altro nodo della rete, se abilitare o meno la cifratura delle risposte SAML; inoltre, quest'ultima può essere abilitata globalmente andando ad agire sul parametro `response.encryption.mandatory` nel file `eidas.xml`.

4.2.2 Identity Provider

Come già sottolineato, l'IdP di prova fornito insieme al nodo eIDAS è stato sostituito dal software Shibboleth, in modo da lavorare con un ambiente quanto più possibile vicino al contesto reale; tale IdP viene eseguito attraverso un'istanza dell'application server Jetty.

Le credenziali degli utenti, insieme con gli attributi eIDAS obbligatori, sono memorizzati all'interno di un database MySQL: per semplicità, tutti i dati sono contenuti all'interno di un'unica tabella denominata "users".

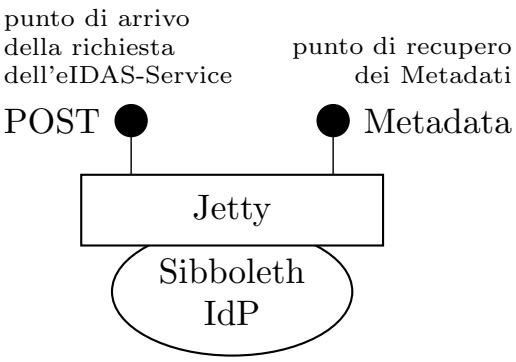


Figura 4.4: Risorse esposte dall'IdP

La configurazione dell'IdP avviene per mezzo di file XML presenti nella cartella `conf` all'interno della directory di installazione. Come primo passaggio è stato specificato come recuperare i Metadati dell'eIDAS-Service e dell'AP: per far questo, si interviene sul file `metadata-providers.xml`, indicando come tipo di provider *"FileBackedHTTPMetadataProvider"*, che consente di ottenere i Metadati da una risorsa HTTP e di memorizzarli su disco. Per consentire all'IdP di verificare i Metadati ottenuti, nella configurazione è stato indicato il percorso del certificato radice della loro catena di certificazione. I Metadati dell'IdP devono essere conservati all'interno di un file chiamato `idp-metadata.xml` nella cartella `metadata`, in modo che Shibboleth li esponga automaticamente presso una risorsa HTTP predefinita. Le chiavi private relative ad i certificati di firma e cifratura che dovranno essere usati dall'IdP per i messaggi SAML sono memorizzate secondo lo standard PKCS #1 [20, appendice A.1.2], indicando la loro directory nel file `idp.properties`.

In seguito è stato indicato all'IdP quale meccanismo adoperare per autenticare gli utenti e dove recuperare i loro attributi. La modalità di autenticazione viene configurata all'interno del file `relying-party.xml`: qui è possibile specificare e configurare, per singolo SP o per tutti quelli riconosciuti, i diversi profili abilitati, indicando, per ognuno di essi, le eventuali proprietà, come i meccanismi di autenticazione supportati. Nella configurazione di prova verrà abilitato il profilo `SAML2.SSO` con meccanismo di autenticazione tramite username e password, i quali dovranno essere recuperati dal database: questo tipo di login, infatti, consente di sfruttare diversi back-end per il recupero e la verifica delle credenziali, ad esempio tramite directory LDAP. Nella configurazione di prova viene usato un modulo di terze parti[2] che fa uso della tecnologia Java Authentication and Authorization Service (JAAS): per abilitarlo, dopo averlo incluso tra le librerie a disposizione dell'application server, è stato necessario indicare l'uso di JAAS nel file `password-authn-config.xml`, per poi configurare le impostazioni del modulo nel file `jaas.config`.

Il recupero degli attributi è stato configurato nel file `attribute-resolver.xml`, dove sono state indicate le sorgenti da contattare e gli attributi da ottenere: le fonti sono rappresentate da elementi di tipo `DataConnector`, che, nel caso in esame, è stato configurato per la connessione ad un database relazionale tramite il driver JDBC. È stata specificata la query SQL da effettuare sul database e le colonne da prendere in esame per generare

[2] jaas-rdbms (https://github.com/tauceti2/jaas-rdbms)

gli attributi; questi saranno ottenuti fornendo in input i valori estratti dalle colonne dalla query ai codificatori standard SAML, come `SAML2String`. Per recuperare solo gli attributi relativi all'utente autenticato, viene aggiunto un filtro alla query sulla chiave primaria, costituita dal nome utente: esso viene fatto corrispondere a quello inserito in fase di login dall'utente mediante la variabile `resolutionContext.principal`.

Infine, l'IdP è stato istruito in modo che rilasci tali attributi all'eIDAS-Service attraverso la configurazione del file `attribute-filter.xml`: qui sono state specificate le policy utilizzate da Shibboleth per filtrare gli attributi inseriti nelle asserzioni sulla base di diversi parametri, primo fra tutti l'identità del SP. Per i nostri scopi è stato sufficiente aggiungere una policy che abbia come regola di attivazione la corrispondenza tra il mittente della richiesta e l'ID dell'eIDAS-Service; in tale policy è stata aggiunta una regola per ogni attributo che ne permetta il rilascio. Per quanto riguarda l'AP, è necessario aggiungere una nuova policy che contenga la lista di attributi che esso è autorizzato a ricevere.

4.3 Implementazione dell'Attribute Provider

Dopo aver configurato correttamente il sistema SP-Connector-Service-IdP, si può passare alla progettazione e implementazione dell'AP. Come è stato già accennato, l'AP è stato realizzato mediante l'unione di un IdP e un SP, entrambi implementati tramite Shibboleth: l'IdP riceve la `AuthnRequest` da parte dell'eIDAS-Service e, attraverso il SP, delega l'autenticazione dell'utente al sistema di identità digitale; ricevuta la risposta, emetterà una propria asserzione di autenticazione contenente gli attributi richiesti.

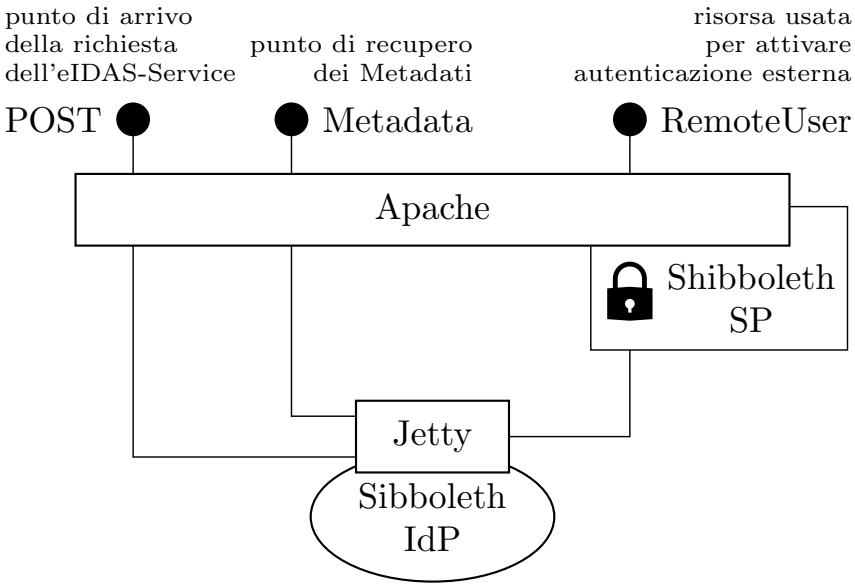


Figura 4.5: Risorse esposte dall'AP

4.3.1 Web server Apache

Per consentire al modulo SP di lavorare al fianco dell'IdP, quest'ultimo è stato posizionato dietro un reverse proxy Apache che, oltre ad avere il compito di inoltrare le richieste verso

l'application server, consente di proteggere una particolare risorsa con il modulo SP Shibboleth; per autenticare l'utente, l'AP lo reindirizzerà su tale risorsa in modo che il modulo possa richiedere l'autenticazione al sistema di identità digitale. Dopo aver installato il web server è stato necessario configurarlo affinché potesse inoltrare le richieste dell'utente verso le risorse corrette, oltre ad abilitare il modulo SP Shibboleth: tale configurazione avviene nel file presente nella cartella `sites-available`. Per consentire il corretto funzionamento dell'IdP bisogna specificare, attraverso una lista di direttive `ProxyPass`, quali sono le risorse da esporre e a quali percorsi sull'application server fanno riferimento: ad esempio, la direttiva

```
ProxyPass /idp/profile/SAML2/POST/SSO
   http://127.0.0.1:8080/idp/profile/SAML2/POST/SSO
```

consente di specificare che le richieste verso tale risorsa devono essere inoltrate sulla porta TCP 8080, dove è in ascolto il server Jetty. Per evitare che Apache sostituisca il nome host presente nelle richieste con il proprio durante l'inoltro, è stato necessario specificare anche la direttiva `ProxyPreserveHost On`. Nello stesso file è stata anche indicata la risorsa da proteggere attraverso il SP e che verrà usata per l'autenticazione tramite RemoteUser:

```
<Location /idp/Authn/RemoteUser>
        AuthType shibboleth
        ShibRequestSetting requireSession 1
        require valid-user
</Location>
```

Questo codice specifica che l'accesso alla risorsa indicata deve essere protetto tramite il modulo Shibboleth SP e richiede una sessione valida; l'ultima direttiva indica che l'accesso deve essere garantito a tutti gli utenti autenticati.

4.3.2 Identity Provider

L'installazione e la configurazione dell'IdP Shibboleth all'interno dell'AP segue gli stessi passaggi compiuti in precedenza sulla macchina del vero IdP; ovviamente, in questo caso, alcune impostazioni dovranno essere modificate per consentire all'AP di sfruttare il sistema di identità digitale del Paese dell'utente per autenticarlo; le informazioni dettagliate sulla configurazione dell'AP sono oggetto del Capitolo 5. La differenza maggiore tra le due configurazioni consiste nel meccanismo di autenticazione che viene attivato nei confronti dell'utente: mentre l'IdP del sistema di identità digitale è stato configurato per identificare l'utente tramite username e password, in questo caso verrà sfruttato il flusso di autenticazione denominato `RemoteUser`. Esso è basato sul più generico flusso `External`, il quale consente di affidarsi ad una qualsiasi tecnica di autenticazione che non si basi sui flussi Spring: se selezionato, al momento dell'autenticazione l'IdP Shibboleth redirige l'utente su una risorsa specificata, presso la quale dovrà essere presente un servlet esterno che dovrà, in qualche modo, autenticare l'utente, usando un'interfaccia fornita da Shibboleth per notificare il risultato dell'operazione. Il flusso `RemoteUser` fornisce una semplice implementazione di tale servlet che si basa sull'estrazione dell'identità dell'utente dalla richiesta HTTP che gli viene inoltrata; questo implica che l'autenticazione debba avvenire attraverso il web server, ad esempio tramite HTTP digest authentication, LDAP o, come nel nostro caso, un altro sistema SSO.

Alla ricezione della richiesta di autenticazione presso l'end-point esposto dall'IdP Shibboleth (Figura 4.6, punto 1), questo risponderà all'utente con il codice HTTP 302 (Found),

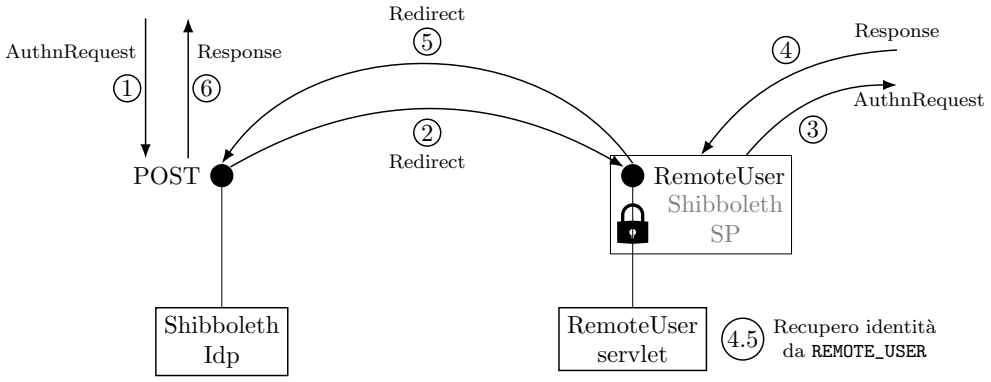


Figura 4.6: Passaggi eseguiti dal flusso `RemoteUser`

redirigendolo verso la risorsa presso cui si trova il servlet (punto 2). Tale risorsa, tuttavia, sarà protetta tramite il modulo SP, pertanto l'utente verrà inviato presso l'IdP del sistema di identità digitale per ottenere un'asserzione di autenticazione (punti 3 e 4). Dopo aver autenticato l'utente, il SP inserirà la sua identità all'interno dell'attributo `REMOTE_USER` della richiesta e autorizzerà il web server ad inoltrarla al servlet; quest'ultimo recupererà le informazioni dall'attributo e le invierà, attraverso altri attributi, all'IdP. L'utente verrà infine rediretto nuovamente presso l'end-point (punto 5) in modo che l'IdP possa recuperare tali informazioni e emettere la propria asserzione.

Per poter attivare tale flusso di autenticazione per le richieste provenienti dall'eIDAS-Service è stato modificato il file `idp.properties`: esso contiene una proprietà che definisce i flussi di login abilitati, alla quale bisogna aggiungere `RemoteUser`. Dopo aver apportato tale modifica, è stato aggiornato il file `relying-party.xml` in modo da rendere tale metodo di autenticazione quello predefinito per i SP autorizzati. Naturalmente, così come nel caso dell'IdP, l'AP deve essere configurato per poter recuperare e validare i Metadati dell'eIDAS-Service, in modo che esso rientri tra tali SP. Successivamente il flusso di autenticazione è stato configurato per indicare presso quale risorsa dovesse essere istanziato il servlet, la stessa da proteggere tramite il modulo SP: il percorso deve essere specificato sia nel file di configurazione del flusso, `remoteuser-authn-config.xml`, sia nel file `web.xml` nella directory dell'applicazione web. Quest'ultimo contiene i nomi e i parametri di inizializzazione dei servlet che compongono l'applicazione; per quanto riguarda quello relativo al flusso `RemoteUser` è necessario anche specificare come recuperare l'identità dell'utente dalla richiesta. Il servlet, infatti, può essere configurato per recuperare l'identità dell'utente attraverso un header della richiesta o un suo attributo; la configurazione base prevede che l'identità sia recuperata attraverso l'attributo standard `REMOTE_USER`, che sarà valorizzato dal SP Shibboleth in caso di avvenuta autenticazione.

Dopo aver autenticato l'utente, l'IdP provvede a recuperare gli attributi necessari: anche in questo caso essi sono memorizzati all'interno di un database MySQL, la cui connessione a Shibboleth è stata configurata come sull'IdP del sistema di identità digitale. Nell'implementazione di prova, per ogni utente, sono conservati un identificato nella forma `[iniziale nome].[cognome]` e uno degli attributi opzionali del sistema eIDAS, il sesso. La forma del nome utente è stata scelta in modo da differire da quella utilizzata presso l'IdP ma, allo stesso tempo, ricavabile attraverso alcuni degli attributi obbligatori rilasciati da quest'ultimo, nello specifico il nome e il cognome. In questo modo, per indicare l'utente del quale si richiede il recupero degli attributi, l'eIDAS-Service, dopo aver costruito il nome utente seguendo le indicazioni ottenute dal Web Service, lo inserisce come NameID del

Subject all'interno della `AuthnRequest` da inviare all'AP, così come suggerito dalle specifiche SAML [18, sezione 3.4.1]. Tale identificativo viene estratto dalla richiesta in fase di risoluzione degli attributi attraverso uno script presente nel file `attribute-resolver.xml`:

```
authr = profileContext.getInboundMessageContext().getMessage();
value = authr.getSubject().getNameID().getValue();
requestedSubject.addValue(value);
```

Questo codice, che recupera l'`AuthnRequest` attraverso la gerarchia di contesti estraendo poi il valore del NameID, è stato inserito all'interno della definizione di un attributo chiamato `requestedSubject`, il quale viene usato come filtro nella query effettuata sul database per recuperare l'attributo *Gender*.

4.3.3 Interceptor

Al fine di aggiungere all'AP delle funzionalità aggiuntive rispetto a quelle fornite da Shibboleth, si è fatto uso dello strumento degli intercept per modificare ed estendere il comportamento del modulo IdP durante la gestione dei messaggi `AuthnRequest` provenienti dall'eIDAS-Service. Gli interceptor sono costituiti da flussi Spring, ovvero serie di azioni da eseguire specificate all'interno di documenti XML, che fanno spesso uso di classi e funzioni Java per l'elaborazione dei dati. Per definire un nuovo interceptor, oltre ad implementare il codice per il suo funzionamento, è necessario aggiungere dei file di configurazione e modificare alcuni di quelli già presenti: nel file `intercept/profile-intercept.xml` è presente una lista degli interceptor disponibili nel sistema, a cui va aggiunto quello da implementare. Il flusso che costituisce l'interceptor deve essere definito in un file XML posizionato nella directory `flows/intercept/[nome_interceptor]` insieme con la definizione dei *beans* Spring di cui fa uso. Infine, l'interceptor deve essere abilitato nella configurazione del flusso di autenticazione presente nel file `relying-party.xml`, facendo uso della proprietà `postAuthenticationFlows` per indicare che esso deve essere attivato successivamente all'autenticazione del soggetto.

Rilascio degli attributi da parte dell'IdP

Nell'implementazione del nodo eIDAS fornita dalla rete di cooperazione, i messaggi SAML `AuthnRequest` inviati dall'eIDAS-Service e diretti ad un IdP seguono le specifiche della rete eIDAS, come se fossero scambiati tra due nodi; in particolare, viene emessa la lista degli attributi richiesti attraverso un elemento `RequestedAttributes` presente nelle estensioni della richiesta [8, sezione 2.3.2]. Tale comportamento deriva dal fatto che il codice relativo alla connessione tra l'eIDAS-Service e il sistema di identità digitale deve essere implementato dai singoli Stati e quello già presente è progettato per riusare le funzioni già implementate per la generazione di richieste eIDAS. Per sfruttare al meglio il codice fornito e simulare ciò che avverrebbe in un'implementazione reale si è deciso di sviluppare un'estensione, chiamata **EidasReleaseAttributes** che sia in grado di interpretare la lista di attributi richiesti e aggiungerli alla lista di quelli da rilasciare.

Il flusso Spring implementato per tale interceptor è costituito da una sola azione di tipo `action-state`:

```
<evaluate
    expression="AttrFilter.apply(opensamlProfileRequestContext)"/>
```

Essa consiste nel richiamare una funzione che esamini la richiesta e selezioni gli attributi da rilasciare. A tale funzione viene passato un riferimento ad un oggetto che rappresenta il contesto nel quale avviene l'autenticazione, il quale da accesso alle informazioni della richiesta gestita e della risposta che sarà inviata. L'interceptor lavora basandosi sui nomi descrittivi degli attributi eIDAS: per poter confrontare quelli presenti nella richiesta con quelli recuperati dall'IdP, è necessario che questi ultimi abbiano come identificativo il nome descrittivo del corrispondente attributo eIDAS. Il cuore dell'interceptor è costituito dalla classe `EidasReleaseAttributes` la quale, ereditando dalla classe base degli interceptor, fornisce il metodo `apply`, che viene richiamato nel flusso. Nella prima fase, a partire dal contesto fornito, viene recuperata la richiesta che sta venendo gestita e si verifica se essa contiene l'estensione `RequestedAttributes`: in caso positivo viene costruita una lista con i nomi descrittivi degli attributi richiesti. Se l'estensione non fosse presente l'interceptor terminerebbe subito, facendo proseguire il flusso principale, altrimenti si procede alla seconda fase, ovvero l'inclusione degli attributi presenti nella lista tra quelli da rilasciare. Dal contesto vengono recuperate due nuove liste, quella contenente tutti gli attributi risolti dall'IdP e quella degli attributi da rilasciare. Per ogni elemento della prima lista si verifica se questo sia già presente nella seconda e, in caso contrario, se il suo identificativo corrisponda ad uno di quelli degli attributi richiesti: se necessario, tale attributo viene aggiunto alla lista di quelli da rilasciare. In questo modo eventuali attributi richiesti ma non presenti tra quelli recuperati dall'IdP vengono semplicemente ignorati e sarà compito dell'eIDAS-Service gestire tale situazione.

Poiché questa estensione aggiunge gli attributi richiesti alla lista di quelli rilasciati, essa sovrasta gli effetti di eventuali regole che riguardano tali attributi: se nel file di configurazione `attribute-filter.xml` fosse presente una regola che proibisce il rilascio all'eIDAS-Service di un attributo richiesto, esso verrebbe incluso comunque nell'asserzione. Questo comportamento deriva dal fatto che l'interceptor viene eseguito successivamente al filtraggio interno di Shibboleth; in questo modo è possibile evitare di specificare delle regole che riguardino gli attributi degli utenti, dato che, in mancanza di esse, l'IdP include nella risposta esclusivamente quelli rilasciati dall'interceptor. Eventuali regole su attributi non richiesti vengono normalmente applicate, consentendo, ad esempio, di rilasciare degli attributi di servizio utili al sistema di identità digitale.

Controllo del nome utente

Per verificare che l'utente autenticato tramite il sistema di identità digitale corrisponda a quello per i quali l'eIDAS-Service richiede gli attributi, è possibile fare in modo che l'IdP al quale l'AP richiede l'autenticazione rilasci gli attributi necessari per ricavare il nome utente. In questo modo, il modulo SP potrà ricostruire tale identificativo una volta ottenuta l'asserzione e inoltrarlo tramite RemoteUser al software IdP; questo potrà confrontarlo con quello inviato dall'eIDAS-Service per verificarne la corrispondenza. Per effettuare tale verifica, è stato sviluppato un secondo interceptor da eseguire prima di rilasciare la risposta al nodo eIDAS. A differenza di quanto accade con l'interceptor impiegato per il rilascio degli attributi, il quale consiste in un'unica azione che non ha alcun impatto sull'esito dell'autenticazione, in questo caso la verifica della corrispondenza tra gli identificativi può portare ad un'interruzione del flusso con il rilascio di una risposta negativa verso l'eIDAS-Service.

```
<decision-state id="IdControl">
       <if test="APVerifyId.apply(opensamlProfileRequestContext)"
          then="proceed" else="IdentitySwitch"/>
```

```
</decision-state>
```

Il flusso dell'interceptor è costituito da un elemento `decision-state` il quale esegue un'azione (verifica della corrispondenza tra gli identificativi) che produce un risultato booleano: se l'azione è andata a buon fine, l'interceptor terminerà correttamente facendo proseguire il flusso principale, mentre in caso di un risultato negativo verrà generata una risposta SAML con status `AuthnFailed`. Si è scelto di non usare un errore più specifico per evitare di rilasciare all'esterno informazioni, riducendo il rischio che esse possano essere usate per progettare degli attacchi.

Il confronto degli identificativi viene effettuato all'interno del metodo `apply` della classe `APVerifyId` che è stata implementata per questo scopo. Utilizzando l'oggetto di tipo `ProfileRequestContext` che viene passato alla funzione, viene recuperato il messaggio contenete la richiesta SAML e da essa si estrae il valore del NameID, se presente; in seguito, dallo stesso oggetto si estrae il nome del soggetto autenticato, recuperato in fase di autenticazione. Nel caso i due identificativi non coincidano o ci siano stati dei problemi nel loro recupero, come la mancanza del Subject nella richiesta, la funzione ritorna il valore *false*, altrimenti restituisce *true*. Il risultato del confronto verrà valutato dal flusso dell'interceptor che terminerà nello stato `proceed`, in caso di avvenuta corrispondenza, oppure `IdentitySwitch` se si sono verificati errori.

4.3.4 Service Provider

Il modulo SP Shibboleth è disponibile sia come pacchetto nelle repository delle maggiori distribuzioni di Linux, sia come codice sorgente presso il sito web, da compilare ed installare manualmente; per il test sulla macchina virtuale si è preferito usare il primo metodo, essendo più semplice e, per gli scopi del progetto, equivalente ad una installazione manuale. Per includere il modulo tra quelli caricati dal web server Apache è sufficiente aggiungere una direttiva `LoadModule` tra quelle già presenti, indicando il percorso di installazione della libreria.

Avendo già configurato Apache per proteggere la risorsa del servlet RemoteUser tramite il SP, la configurazione di quest'ultimo riguarda l'IdP da contattare e gli attributi ricevuti da rendere disponibili; nel momento in cui il SP riceverà l'asserzione dal sistema di identità digitale, esso combinerà gli attributi ricevuti per formare l'identificativo che verrà passato al software IdP Shibboleth attraverso l'attributo `REMOTE_USER`. È stato scelto di elaborare gli attributi nel SP e non inoltrarli al software IdP principalmente per due ragioni: la trasmissione di un unico attributo tramite RemoteUser è estremamente semplice, mentre trasportarne diversi avrebbe richiesto diverse modifiche non facilmente realizzabili; inoltre, dalla versione 2.5 del software, il SP Shibboleth mette a disposizione un modo molto semplice per combinare diversi attributi e applicare semplici trasformazioni agendo solo sui file di configurazione.

La lista contenente gli attributi da recuperare dall'asserzione è stata specificata nel file `attribute-map.xml`, dove ad ognuno di essi viene assegnato un ID per farvi riferimento nel resto della configurazione. Poiché la forma dell'identificativo scelta per rappresentare l'utente sull'AP è `[iniziale nome].[cognome]`, gli attributi da recuperare saranno solo il nome ed il cognome del soggetto, ai quali si aggiunge il *PersonIdentifier* per scopi di debug.

Le altre impostazioni sono state configurate nel file `shibboleth2.xml`; l'elemento più importante in tale file è `ApplicationDefaults`, che definisce la maggior parte dei comportamenti del software. Sono stati definiti due suoi attributi: l'identificativo del SP e

il nome dell'attributo da usare per valorizzare `REMOTE_USER`. Configurando l'elemento `SSO` è stato indicato l'ID dell'IdP del sistema di identità digitale da utilizzare per l'autenticazione dell'utente; per permettere il recupero dei suoi Metadati è stato configurato un elemento `MetadataProvider` specificando come sorgente l'end-point di distribuzione sull'IdP. I Metadati recuperati in questo modo saranno memorizzati su disco e verificati attraverso la convalida della firma usando il certificato radice specificato. Per permettere al SP di firmare le proprie richieste è stato indicato, mediante un elemento di tipo `CredentialResolver`, dove recuperare la chiave privata e il corrispondente certificato, i quali saranno gli stessi usati dal modulo IdP per comunicare con l'eIDAS-Service.

Al fine di costruire l'identificativo dell'utente dagli attributi estrapolati dall'asserzione ricevuta, sono stati adoperati alcuni elementi `AttributeResolver` per elaborare i valori di tali attributi, generandone anche alcuni aggiuntivi:

```
<AttributeResolver type="LowerCase" source="FirstName" />
<AttributeResolver type="LowerCase" source="FamilyName" />
<AttributeResolver type="Transform" source="FirstName">
       <Regex match="^(.)(.*)$" dest="NameInitial">$1.</Regex>
</AttributeResolver>
<AttributeResolver type="Template" sources="NameInitial FamilyName"
    dest="Username">
        <Template>$NameInitial$FamilyName</Template>
</AttributeResolver>
```

I primi due elementi non fanno altro che portare il nome ed il cognome in caratteri minuscoli; l'`AttributeResolver` che presenta il tipo `Transform` genera un nuovo attributo, chiamato *NameInitial*, che consiste nel primo carattere del nome seguito da un punto. Infine, l'ultimo elemento, di tipo `Template`, costruisce l'attributo *Username* concatenando quello appena generato con il cognome normalizzato; esso sarà usato per valorizzare `REMOTE_USER` e identificare l'utente nella query sul database.

4.4 Modifica dell'eIDAS-Service per supportare l'AP

4.4.1 Progettazione e implementazione del Web Service

Per consentire all'eIDAS-Service di recuperare le informazioni riguardanti gli AP disponibili, è stato progettato ed implementato un Web Service REST basato su Java Spring; si tratta di un prototipo di ciò che potrebbe essere offerto dal gestore del sistema di identità digitale al fine di distribuire una lista degli AP autorizzati insieme con i dati necessari a contattarli. L'API del servizio consiste in due risorse: la prima restituisce un array JSON in cui ogni elemento rappresenta un'AP disponibile, di cui viene fornito un nome descrittivo ed il suo identificativo; la seconda richiede di specificare un ID della lista precedente e fornisce, sempre in formato JSON, l'URL da cui recuperare i Metadati dell'AP corrispondente insieme con le indicazioni per generare il nome utente a partire dagli attributi obbligatori da utilizzare nella richiesta.

Tali indicazioni sono fornite per mezzo di un array di oggetti JSON, chiamati *token*, ognuno dei quali rappresenta un attributo o una stringa da concatenare per formare l'identificativo dell'utente da adoperare sull'AP corrispondente. Ogni token è formato da quattro valori:

isAttribute un valore booleano che indica se il token, nel formare il nome utente, deve essere sostituito con il valore di un attributo o con una stringa prefissata;

string se il valore di *isAttribute* è pari a *true* indica il nome descrittivo dell'attributo da utilizzare, altrimenti rappresenta la stringa da inserire;

characters opzionale, presente solo se il token rappresenta un attributo, indica quanti caratteri del suo valore, a partire dal primo, devono essere usati;

upperOrLower opzionale, presente solo se il token rappresenta un attributo, può assumere i valori "AllUpper" o "AllLower" per indicare come normalizzare il valore.

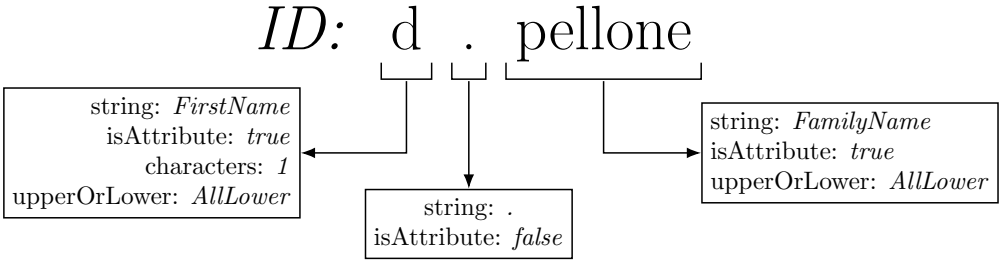


Figura 4.7: Esempio dei token usati nella configurazione di prova

Il seguente esempio mostra un token che indica l'attributo *FirstName*, il cui valore dovrà essere convertito in caratteri minuscoli e di cui dovrà essere preso solo il primo carattere:

```
{"string": "FirstName", "isAttribute": true, "characters": 1,
   "upperOrLower": "AllLower"}
```

È possibile aggiungere nuovi valori per estendere il numero di trasformazioni che possono essere richieste, anche se questo comporterebbe l'aggiornamento del codice deputato all'interpretazione dei token. Al momento di generare il nome utente da inserire nella richiesta all'AP, ogni token dovrà essere sostituito con la stringa contenuta al suo interno o con il valore, opportunamente modificato, dell'attributo indicato; le stringhe così ottenute andranno concatenate nell'ordine in cui appaiono i rispettivi token, ottenendo così l'identificativo.

4.4.2 Aggiunta delle funzionalità al Service

Contestualmente all'implementazione dell'AP è stato modificato il codice dell'eIDAS-Service fornito dalla rete di cooperazione, al fine di permettere ad esso di recuperare gli attributi mancanti. Sono state aggiunte due funzionalità: la possibilità di interrogare il Web Service per ottenere informazioni sugli AP disponibili e la modifica del flusso di gestione delle risposte dell'IdP per consentire l'emissione di richieste verso l'AP e la gestione delle successive risposte. Il codice del nodo è costituito da una parte relativa alla comunicazione con la rete eIDAS, indipendente dallo Stato in cui viene distribuito, e una che dialoga con il sistema di identità digitale e i SP, la quale dovrà essere modificata dai singoli Paesi per adattarla ai loro protocolli. Nell'intervenire sul codice, si è cercato di limitare le modifiche a questa seconda parte, anche se sono stati necessari alcuni piccoli cambiamenti nel resto del programma.

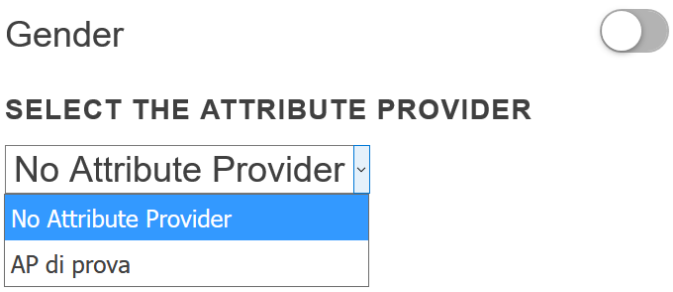


Figura 4.8: Lista a cascata aggiunta nella pagina web dell'eIDAS-Service

La prima modifica da apportare è l'inserimento della possibilità, per l'utente, di scegliere quale AP contattare nel caso vengano richiesti degli attributi non obbligatori. Per far questo, è stato necessario recuperare la lista degli AP dal Web Service, per poi modificare la pagina che viene visualizzata per richiedere all'utente il consenso al rilascio degli attributi opzionali; in tale pagina è stato aggiunto una lista a cascata che consente di indicare uno degli AP o selezionare *Nessun AP*. Il recupero della lista dal Web Service avviene nella funzione `doPost` del servlet che gestisce l'arrivo di una richiesta da parte di un eIDAS-Connector, nella classe `ColleagueRequestServlet`: la lista viene recuperata grazie ad una richiesta GET presso la risorsa del Web Service deputata al rilascio di tali informazioni. La richiesta viene effettuata tramite la classe `HttpsURLConnection`, che viene istruita per instaurare una connessione TLS con autenticazione client. Le chiavi da usare per l'autenticazione e i certificati fidati per la verifica dell'identità del Web Service vengono recuperate da un keystore JKS il cui percorso deve essere specificato nel file `eidas.xml`; anche le URL presso le quali contattare il Web Service, sia per la lista che per le informazioni sullo specifico AP, vengono recuperate dalle proprietà presenti in tale file, in modo da poterle aggiornare facilmente all'occorrenza. La lista recuperata viene inserita all'interno di un attributo della richiesta, in modo che possa essere accessibile alla pagina JSP che richiede il consenso al rilascio degli attributi. Qui l'utente selezionerà uno degli AP proposti, il cui identificativo sarà inserito in una variabile chiamata `__apSelector` e recuperato in seguito; se viene selezionato *Nessun AP*, la variabile varrà `null`.

Dopo aver richiesto il consenso dell'utente, l'eIDAS-Service prepara la richiesta da inviare all'IdP del sistema di identità digitale; questa operazione viene effettuata dal metodo `sendRequest` della classe `SpecificProxyServiceImpl`, la quale rientra nel novero di quelle da adattare allo specifico protocollo utilizzato dallo Stato. Nell'implementazione fornita, per memorizzare la richiesta ricevuta dall'eIDAS-Connector e poterla poi collegare a quella che sarà inviata all'IdP e alla sua risposta, viene adoperata una mappa nella quale la richiesta originale viene inserita insieme con l'ID di quella nuova. Seguendo questa strategia, è stata aggiunta una mappa denominata *requestAttributeProviderCorrelationMap*, che ha lo scopo di memorizzare l'identificativo dell'AP selezionato dall'utente, associato all'ID della richiesta SAML inviata all'IdP, in modo che possa essere recuperato durante la gestione della risposta. Dopo aver preparato la richiesta, l'eIDAS-Service la inoltra all'IdP utilizzando il binding appropriato.

Per gestire i messaggi SAML contenenti le risposte che vengono inoltrati tramite binding HTTP POST presso l'end-point definito viene invocato il metodo `processResponse` della stessa classe. Poiché anche gli AP utilizzeranno tale end-point per inviare le loro risposte, è stato necessario modificare profondamente tale funzione, sia per gestire tali risposte che per distinguerle da quelle dell'IdP. Alla classe è stata aggiunta un'ulteriore

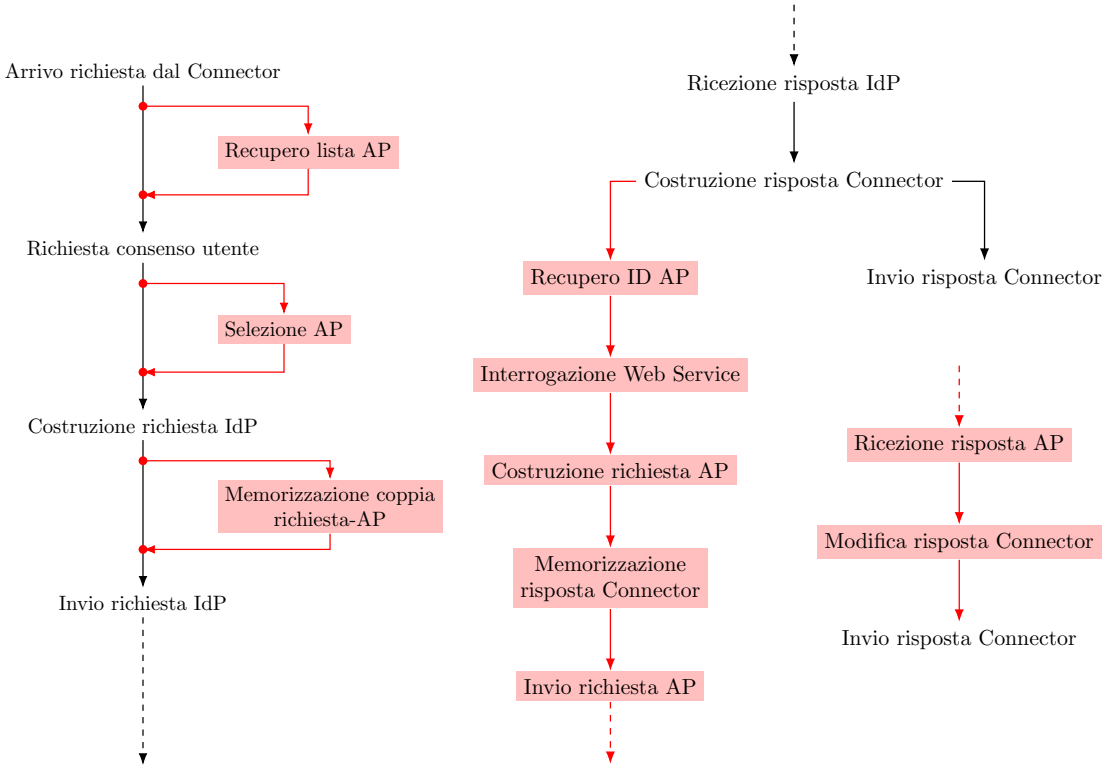


Figura 4.9: Sequenza di azioni dell'eIDAS-Service (in rosso quelle aggiunte)

mappa, chiamata *incompleteResponses*, nella quale, nel caso in cui fosse necessario contattare un AP, vengono memorizzate le risposte da inviare agli eIDAS-Connector contenenti gli attributi recuperati fino a quel momento. Ogni risposta incompleta viene associata all'ID della richiesta inviata all'AP, in modo che sia possibile recuperarla in seguito. All'arrivo di una risposta, per determinare se essa provenga dall'IdP o da un AP basta verificare se la mappa contiene un elemento la cui chiave coincide con l'ID presente nel campo `InResponseTo` del messaggio SAML: in caso affermativo esso proviene da un AP.

Dopo aver verificato la provenienza della risposta si effettua, mediante un metodo appositamente implementato, un controllo per verificare se ci sono attributi mancanti: nel caso in cui il messaggio provenga da un IdP si considerano solo gli attributi al suo interno, mentre in caso di risposta da parte di un AP questi vengono integrati con quelli presenti nella risposta parziale memorizzata. La funzione, chiamata *getMissingAttributes*, riceve in ingresso due liste di attributi e ne restituisce una che contiene tutti gli attributi della seconda che non sono presenti nella prima. Se tutti gli attributi necessari sono stati recuperati il metodo restituisce `null` e viene costruita la risposta da inviare all'eIDAS-Connector; se gli attributi sono stati recuperati solo dall'IdP il messaggio si baserà sulla risposta appena ottenuta, mentre in caso contrario verrà costruita basandosi sulla risposta parziale memorizzata aggiungendo i nuovi attributi. In ogni caso, la risposta viene costruita adoperando il *builder* fornito dalla classe `AuthenticationResponse` della libreria eIDAS e, dopo aver rimosso i messaggi memorizzati nelle mappe, viene passata al servlet che si occuperà di inoltrarla, tramite l'utente, all'eIDAS-Connector destinatario.

Se il metodo che elenca gli attributi mancanti restituisce una lista non vuota si rende necessario contattare un AP per recuperare tali attributi. Come prima operazione viene recuperato l'ID dell'AP memorizzato in precedenza nell'apposita mappa; se esso è pari a

`null` significa che l'utente non ha indicato alcun AP quando gli è stato richiesto, pertanto viene costruita la risposta per l'eIDAS-Connector con i soli attributi disponibili. Con l'ID dell'AP recuperato si interroga nuovamente il Web Service presso l'URL indicata nella configurazione per ottenere il documento JSON contenente le informazioni su tale AP; attraverso le funzioni della libreria *Jackson* si effettua il parsing del documento per ottenere l'URL da cui recuperare i Metadati dell'AP e una lista di oggetti *StringToken* per costruire l'ID dell'utente da inserire nella richiesta SAML. Per generare tale identificativo è stata implementata una classe chiamata *IDBuilder* la quale espone il metodo statico *getID* che, a partire dalla lista di *StringToken* e un insieme di attributi, restituisce l'ID dell'utente, costruito interpretando i *token*.

Viene quindi costruita la `AuthnRequest` da inviare all'AP sfruttando le funzioni offerte dalla libreria *openSAML*; per questo scopo è stata implementata una funzione chiamata *createSamlAuthNRequest*, che ha anche il compito di recuperare i Metadati dell'AP. Questi sono ottenuti attraverso la classe *CachingMetadataFetcher* presente nel nodo, che si occuperà anche di validarli usando il certificato presente nel KeyStore. Dai Metadati viene estratta l'URL dell'end-point verso il quale redirigere l'utente per trasportare la richiesta tramite binding HTTP POST. La `AuthnRequest` viene costruita a partire dai suoi elementi, tra cui l'*Issuer*, il *Subject* in cui viene inserito l'ID generato in precedenza e gli attributi mancanti ottenuti dal metodo *getMissingAttributes* e inseriti tra le estensioni, come in una richiesta eIDAS. Infine, il messaggio viene firmato con la chiave privata dell'eIDAS-Service e restituito sotto forma di array di byte da codificare successivamente in *base64*.

Dopo aver ottenuto il messaggio codificato, esso viene inserito, insieme con l'indicazione di usare il binding HTTP POST e l'URL dell'end-point dell'AP, all'interno di un attributo della richiesta HTTP che sta venendo gestita; il controllo viene poi restituito al servlet che aveva invocato il metodo `processResponse`. Nei casi precedenti in cui era stata preparata una risposta per l'eIDAS-Connector, il metodo la restituisce al servlet che si occupa di costruire il messaggio SAML che l'utente deve trasportare; poiché la richiesta per l'AP è già stata preparata ed è pronta per essere inoltrata, in questo caso il metodo `processResponse` restituisce `null`. Il servlet è stato modificato in modo che, quando si verifica questa eventualità, rediriga l'utente su una risorsa chiamata `apRedirect.jsp`; al suo interno è presente un form nascosto che, utilizzando i dati contenuti negli attributi della richiesta, genera il metodo POST che l'utente eseguirà presso l'AP.

Capitolo 5

Manuale del software

Nel seguito si suppone che il Sistema Operativo utilizzato si basi sul kernel Linux, come ad esempio Debian (https://www.debian.org/); i programmi utilizzati sono disponibili anche per il sistema Microsoft Windows e le istruzioni possono essere usate anche in quel caso, con opportune variazioni sopratutto per quanto riguarda i percorsi delle directory.

5.1 Attribute Provider

Prima di procedere con l'installazione è necessario avere a disposizione due coppie di chiavi crittografiche con relativi certificati per consentire la firma e la cifratura dei messaggi SAML; è richiesta, inoltre, una ulteriore coppia di chiavi per le connessioni TLS dai client. Devono anche essere disponibili i Metadati del modulo IdP, da rilasciare all'eIDAS-Service, e del modulo , per il servizio di identità digitale; essi dovranno contenere i certificati relativi alle chiavi usate dall'AP e dovranno essere firmati tramite una PKI riconosciuta sia dal Service che dall'IdP.

5.1.1 Requisiti

Apache HTTP Server versione 4.2, disponibile presso il repository ufficiale Debian e installabile con il comando `apt-get install apache2` o tramite il sito https://httpd.apache.org/.

Shibboleth SP versione 2.6, disponibile presso la repository ufficiale del sistema Debian e installabile attraverso il comando `apt-get install shibboleth-sp2`; in alternativa, è possibile scaricare e compilare il codice sorgente seguendo le istruzioni del sito ufficiale (https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPLinuxSourceBuild).

Shibboleth IdP versione 3.3, disponibile tramite il sito https://wiki.shibboleth.net/; per poter essere installato, tale software richiede i due programmi di seguito menzionati.

Oracle Java Development Kit versione 8, disponibile attraverso il sito http://www.oracle.com/technetwork/java/index.html; sarà necessario installare i file JCE Unlimited Strength Jurisdiction Policy disponibili presso lo stesso sito per abilitare l'uso di tutti gli algoritmi crittografici.

Jetty versione 9.3, disponibile presso https://www.eclipse.org/jetty/; in alternativa è possibile usare Apache Tomcat versione 8 (https://tomcat.apache.org/).

Apache Maven versione 3.5, usato per gestire e compilare il codice degli interceptor, è disponibile presso https://maven.apache.org/.

5.1.2 IdP Shibboleth

Per installare Jetty, fare riferimento alla guida fornita nella documentazione di Shibboleth; dopo aver scaricato il pacchetto contenente l'application server, estrarne il contenuto in una directory e configurarlo secondo quanto descritto nella guida alla sezione *Required Configuration*. Alcune impostazioni devono essere modificate:

1. non copiare i file `https.ini`, `ssl.ini` e `jetty-ssl-context.xml`, in quanto il canale TLS sarà gestito da Apache;
2. nel file `http.ini` lasciare la porta di default (8080);
3. nel file `http.ini` aggiungere la riga "`--module=http-forwarded`" per poter usare Apache come reverse proxy;

Successivamente scaricare il pacchetto contenente l'IdP Shibboleth ed estrarlo in una cartella temporanea; portarsi nella sotto cartella `bin` ed eseguire il file `install.sh`. Durante l'installazione viene richiesta la directory dove collocare i file di Shibboleth: scegliere una cartella appropriata a cui ci si riferirà come `idp-home`. Nella nuova directory saranno presenti tutti i file necessari per la configurazione; prima di procedere, copiare il file `idp-home/war/idp.war` all'interno della cartella `webapps` presente nella directory contenente la configurazione di Jetty. Per configurare l'IdP devono essere modificati i file contenuti nella cartella `idp-home/conf/`, seguendo le istruzioni seguenti.

1. Fornire un modo per recuperare i Metadati dell'eIDAS-Service, specificandolo nel file `metadata-providers.xml`: il modo più semplice consiste nell'utilizzare un provider di tipo `FileBackedHTTPMetadataProvider` specificando l'URL del Service da cui recuperare i Metadati e il certificato con cui validarli.
2. Copiare il file contenenti i Metadati del modulo IdP nella cartella `metadata` e rinominarlo *idp-metadata.xml*, in modo che Shibboleth li esponga automaticamente presso una risorsa HTTP predefinita.
3. Nel file `idp.properties` modificare le righe relative alle chiavi ed ai certificati da usare per firma e cifratura, specificando i file preparati in precedenza.
4. All'interno dello stesso file modificare il valore della proprietà `idp.authn.flows` in "RemoteUser", per abilitare tale flusso di login.
5. Nel file `relying-party.xml` aggiungere il seguente flusso alla lista presente nell'elemento `DefaultRelyingParty`, in modo da abilitarlo per i SP riconosciuti, come l'eIDAS-Service:

```
<bean parent="SAML2.SSO" p:signAssertions="true"
    p:encryptAssertions="true"
    p:authenticationFlows="#{{'RemoteUser'}}"/>
```

6. Nel file `attribute-resolver.xml` configurare uno o più `DataConnector` per il recupero degli attributi dalle sorgenti appropriate; è possibile utilizzare l'attributo definito al punto 8 inserendo una dipendenza.

7. Aggiungere al file le definizioni degli attributi recuperati con i loro codificatori.

8. Nello stesso file inserire il seguente attributo che recupererà il nome utente inviato nella la richiesta:

```
<AttributeDefinition id="requestedSubject"
    xsi:type="ScriptedAttribute" dependencyOnly="true">
        <Script>
        <![CDATA[authr =
            profileContext.getInboundMessageContext().getMessage();
        value = authr.getSubject().getNameID().getValue();
        requestedSubject.addValue(value);]]>
        </Script>
</AttributeDefinition>
```

9. Se non si fa uso dell'interceptor *EidasReleaseAttributes*, inserire all'interno del file `attribute-filter.xml` le politiche per il rilascio degli attributi all'eIDAS-Service.

5.1.3 Interceptor per Shibboleth

APVerifyId e *EidasReleaseAttributes* sono estensioni per il software Shibboleth IdP che consistono in flussi Spring da inserire come interceptor all'interno del profilo di autenticazione SAML2 SSO. Lo scopo di *APVerifyId* è quello di verificare la corrispondenza tra il nome utente presente nella richiesta pervenuta con quello recuperato per mezzo del flusso di autenticazione External oppure RemoteUser. Il nome utente da estrarre dalla richiesta viene ricercato nell'elemento `Subject`, se presente. *EidasReleaseAttributes*, invece, consente di aggiungere alla lista di attributi da rilasciare ad un eIDAS-Service tutti quelli presenti all'interno della richiesta: in particolare, dall'elemento `Extensions` viene estratto, se presente, l'elemento `RequestedAttributes`, il quale dovrà contenere la lista di attributi richiesti come elementi `RequestedAttribute`. Entrambi i software vengono forniti come progetti Java gestiti tramite Maven: per ottenere il pacchetto JAR è sufficiente portarsi nella cartella di uno di essi ed eseguire da riga di comando `mvn clean package`.

Per installare i plugin in Shibboleth IdP occorre copiare i file JAR generati al passo precedente all'interno della cartella `edit-webapp` nella directory di installazione di Shibboleth, poi ricompilare l'IdP e ridistribuirlo sull'application server. Per poter usare l'interceptor è necessario copiare il contenuto della cartella `intercept` presente nei due progetti all'interno della directory `idp-home/flows/intercept`; al termine dell'operazione verranno aggiunte due sotto cartelle, denominate `APVerifyId` e `eidasReleaseAttributes`, che conterranno due file: uno contiene la definizione del bean Spring che rappresenta la classe Java dell'interceptor, mentre l'altro descrive il flusso Spring da eseguire all'attivazione dell'interceptor. Aggiungere poi le seguenti righe all'elemento `list` del file `idp-home/conf/intercept/profile-intercept.xml`:

```
<bean id="intercept/APVerifyId" parent="shibboleth.InterceptFlow"/>
<bean id="intercept/eidasReleaseAttributes"
    parent="shibboleth.InterceptFlow"/>
```

È possibile abilitare gli interceptor aggiungendoli tra i `postAuthenticationFlows` di un flusso di autenticazione `SAML2.SSO` nel file `idp-home/conf/relying-party.xml`; per maggiori informazioni sugli interceptor e il loro funzionamento, consultare la documentazione del software Shibboleth.

5.1.4 SP Shibboleth

L'installazione del web server Apache e del SP Shibboleth può essere effettuata usando il gestore di pacchetti presente su molte distribuzioni Linux; per compilare ed installare i sorgenti, si rimanda alla documentazione dei due progetti. Completata l'installazione, si può passare alla configurazione di Apache, in modo da abilitare le sue funzionalità di reverse proxy e attivare il SP; per impostazione predefinita, i file di configurazione si trovano nella cartella `/etc/apache2/`.

1. Abilitare il modulo Shibboleth eseguendo da terminale "`a2enmod shib2`".

2. Abilitare il modulo Headers eseguendo da terminale "`a2enmod headers`"; questo modulo verrà usato per effettuare aggiungere degli header alle richieste quando queste transitano per il proxy.

3. Nel file `sites-available/default-ssl.conf` attivare le funzionalità di proxy verso le risorse da esporre:

```
<VirtualHost>
...
ProxyPreserveHost On
ProxyPass /idp/profile/SAML2/Redirect/SSO
    http://127.0.0.1:8080/idp/profile/SAML2/Redirect/SSO
ProxyPass /idp/profile/SAML2/POST/SSO
    http://127.0.0.1:8080/idp/profile/SAML2/POST/SSO
ProxyPass /idp/shibboleth http://127.0.0.1:8080/idp/shibboleth
ProxyPass /idp/Authn/RemoteUser
    http://127.0.0.1:8080/idp/Authn/RemoteUser
RequestHeader set X-Forwarded-Proto "https" env=HTTPS
RequestHeader set REMOTE-USER %{REMOTE_USER}s
...
</VirtualHost>
```

4. Nello stesso file abilitare la protezione del SP Shibboleth sulla risorsa `RemoteUser`:

```
<Location /idp/Authn/RemoteUser>
        AuthType shibboleth
        ShibRequestSetting requireSession 1
        ShibUseHeaders On
        ShibUseEnvironment On
        require valid-user
</Location>
```

5. Sempre in `sites-available/default-ssl.conf` configurare i valori delle proprietà `SSLCertificateFile` e `SSLCertificateKeyFile` in modo che puntino al certificato e alla chiave privata da usare per il canale TLS.

Dopo aver configurato il web server è il turno del modulo SP Shibboleth: se è stato installato nel percorso predefinito, i file di configurazione si troveranno in `/etc/shibboleth/`. Laddove non indicato, le modifiche vanno apportate al file `shibboleth2.xml`.

1. Nel file `attribute-map.xml` devono essere specificati gli attributi da recuperare dalle asserzioni ricevute dall'IdP: questo viene fatto per mezzo di elementi `Attribute` in cui specificare il nome dell'attributo nell'asserzione e l'identificativo da assegnargli.

2. Prima dell'elemento `ApplicationDefaults` inserire la libreria da caricare per effettuare operazioni sugli attributi:

```
<OutOfProcess>
        <Extensions>
                <Library path="plugins.so" fatal="true"/>
        </Extensions>
</OutOfProcess>
```

3. Modificare l'elemento `ApplicationDefaults` specificando l'ID da mostrare all'IdP e il nome dell'attributo che conterrà l'identità dell'utente da trasmettere:

```
<ApplicationDefaults entityID="https://ap-example.org"
    REMOTE_USER="Username">
```

4. All'interno dell'elemento `Sessions` aggiungere l'elemento `SSO` con l'ID dell'IdP del sistema di identità digitale da contattare:

```
<SSO entityID="https://idp-example.it">SAML2</SSO>
```

5. All'interno dell'elemento `ApplicationDefaults` deve essere inserito un elemento `MetadataProvider` per il recupero dei Metadati dell'IdP da URL o da file.

6. Sempre nell'elemento `ApplicationDefaults` inserire l'elemento `CredentialResolver` specificando i file della chiave e del certificato da usare per firma e cifratura:

```
<CredentialResolver type="File" key="/path/to/key.key"
    certificate="/path/to/cert.crt"/>
```

7. Aggiungere, tramite elementi `AttributeResolver`, le istruzioni per la costruzione del nome utente da passare al modulo IdP; sono disponibili vari tipi di elaborazione, tra cui l'applicazione di espressioni regolari, la combinazione di due o più attributi, la conversione dei caratteri in maiuscoli e minuscoli.

5.1.5 Avvio dell'AP

Per attivare l'AP devono essere avviati i suoi vari componenti: assicurarsi prima che il server Apache sia in esecuzione (`service apache2 start`), poi avviare il SP Shibboleth (`service shibd start`). Infine, portarsi nella directory di Jetty dove è stato distribuito l'IdP ed eseguire da riga di comando `java -jar ../start.jar`.

5.2 Web Service

Prima di procedere con l'installazione è necessario avere a disposizione una coppia di chiavi crittografiche con relativo certificato per instaurare le connessioni TLS con l'eIDAS-Service. Per poter identificare correttamente il Service tramite autenticazione client TLS, deve anche essere disponibile il certificato che esso presenterà durante la procedura di handshake, in modo da poterlo inserire tra i certificati fidati.

5.2.1 Requisiti

Oracle Java Development Kit versione 8, disponibile attraverso il sito `http://www.oracle.com/technetwork/java/index.html`; sarà necessario installare i file JCE Unlimited Strength Jurisdiction Policy disponibili presso lo stesso sito per abilitare l'uso di tutti gli algoritmi crittografici.

Apache Maven versione 3.5, usato per gestire e compilare il codice, è disponibile presso `https://maven.apache.org/`.

5.2.2 Installazione

Il software del Web Service viene distribuito come progetto Java gestito tramite Maven: per ottenere il pacchetto JAR è sufficiente portarsi nella cartella del progetto, dove si trova il file `pom.xml`, ed eseguire da riga di comando `mvn clean package`. Questo genererà il file `APMapping-1.0.0.jar`, che può essere spostato in una cartella più adatta ed eseguito con il comando `java -jar ./APMapping-1.0.0.jar`, in modo da avviare il Web Service.

5.2.3 Configurazione

Prima di avviare il Web Service è necessario aggiungere i file necessari al suo funzionamento, che andranno collocati all'interno della stessa cartella in cui è stato spostato il pacchetto JAR.

1. Copiare il contenuto della cartella `examples` presente nella directory del progetto, che consiste nel file di configurazione `application.properties` e in due file JSON di esempio.

2. Nel file `application.properties` modificare le proprietà `server.ssl.key*` in base alla propria configurazione della chiave privata TLS, che dovrà essere memorizzata in un Keystore JKS:

   ```
   server.ssl.key-store = path/to/keystore.jks
   server.ssl.key-store-password = keystore_password
   server.ssl.key-alias = key_alias_in_keystore
   server.ssl.key-password = key_password
   ```

3. All'interno dello stesso file modificare le proprietà `server.ssl.trust-store*` in base alla propria configurazione del Keystore JKS in cui sono presenti i certificati client TLS fidati:

   ```
   server.ssl.trust-store = path/to/truststore.jks
   server.ssl.trust-store-password = truststore_password
   ```

4. Sempre in `application.properties`, se necessario, modificare il valore della proprietà `server.port` con la porta su cui il server deve essere in ascolto.

5. Modificare il file `ap.json` seguendo l'esempio fornito in modo da includere gli identificativi e i nomi descrittivi degli AP di cui si devono rilasciare le informazioni.

6. Per ogni AP aggiunto al passo precedente, aggiungere una file `id-ap.json`, dove `id-ap` è l'identificativo presente nel file `ap.json`. All'interno di ognuno dei file inserire la URL da cui recuperare i Metadati dell'AP e la lista dei token da usare per costruire il nome utente per tale AP; il file `myAP.json` può essere usato come esempio.

Quando viene eseguito, il Web Server espone un'interfaccia REST con due risorse:

`/DPellone/APMapping/1.0.0/attributeProviders` risorsa alla quale richiedere la lista di AP mediante una richiesta GET;

`/DPellone/APMapping/1.0.0/attributeProviders/mapping` risorsa alla quale richiedere le informazioni su uno specifico AP mediante una richiesta GET e il *query parameter* `apid` con il suo identificativo.

5.3 Nodo eIDAS

Il software del nodo eIDAS distribuito dalla rete di cooperazione ed usato come base in questo lavoro racchiude al suo interno sia le funzionalità di Connector che quelle di Service. Il codice sorgente modificato include la guida di installazione e configurazione fornita dall'Unione Europea [21], in quanto gran parte della procedura di messa in opera è rimasta invariata. Prima di procedere con l'installazione è bene avere a disposizione le chiavi crittografiche e i certificati da utilizzare per la firma e la cifratura delle asserzioni SAML.

5.3.1 Requisiti

Oracle Java Development Kit versione 8, disponibile attraverso il sito `http://www.oracle.com/technetwork/java/index.html`; sarà necessario installare i file JCE Unlimited Strength Jurisdiction Policy disponibili presso lo stesso sito per abilitare l'uso di tutti gli algoritmi crittografici.

Apache Maven versione 3.5, usato per gestire e compilare il codice, è disponibile presso `https://maven.apache.org/`.

Application Server tra quelli supportati; per i test è stato usato Apache Tomcat versione 8 (`https://tomcat.apache.org/`).

5.3.2 Installazione

Come già accennato, è possibile seguire la guida fornita insieme al codice per compilare ed installare il software del nodo eIDAS; tuttavia, essendoci alcuni passaggi aggiuntivi, nel seguito verranno ricapitolati i punti salienti.

1. Copiare il contenuto della cartella `config`, presente nel codice sorgente, all'interno di una directory apposita: questi file costituiscono la configurazione del nodo.

2. Definire all'interno del Sistema Operativo utilizzato due variabili d'ambiente denominate `$EIDAS_CONFIG_REPOSITORY` e `$SPECIFIC_CONFIG_REPOSITORY`, i cui valori saranno rispettivamente il percorso della directory di configurazione e della sotto cartella `specific` presente al suo interno. È anche possibile definire tali valori attraverso la JVM con il parametro `-D`.

3. Compilare il software portandosi nella cartella `EIDAS-Parent` ed eseguendo da riga di comando `mvn clean install -P appl-server`, sostituendo ad *appl-server* il nome dell'application server utilizzato.

4. Distribuire il file `EIDAS-Node/target/EidasNode.war` appena generato nel application server.

5. Modificare i file di configurazione seguendo le istruzioni della guida [21, Capitolo 4], adattando le impostazioni alle proprie esigenze.

6. Oltre alla configurazione di base, nel file `eidas.xml` aggiungere gli indirizzi a cui contattare il Web Service e le informazioni sul keystore contenente le chiavi ed il certificato da usare nell'autenticazione client TLS verso di esso:

```
<entry key="webservice.listURL">
https://indirizzo.webservice/attributeProviders
</entry>
<entry key="webservice.mappingURL">
https://indirizzo.webservice/attributeProviders/mapping?apid=
</entry>
<entry key="webservice.keystore">/path/to/keystore.jks</entry>
<entry key="webservice.keystore.passw">keystore-password</entry>
```

Capitolo 6

Estensioni e sviluppi futuri

6.1 Risultati del lavoro svolto

Il progetto sviluppato e implementato, comprendente l'AP con i suoi interceptor, il Web Service e le modifiche al nodo eIDAS, dimostra che è possibile permettere al sistema eIDAS il recupero di attributi da fonti diverse dal servizio di identità digitale del Paese dell'utente. È stato sviluppato un meccanismo sicuro per mantenere e distribuire all'eIDAS-Service le informazioni sugli AP autorizzati e su come ricavare, a partire dagli attributi a disposizione, il nome utente da usare per interrogare ognuno di essi. Si è scelto di adattare il profilo SAML Web Browser SSO per consentire il recupero degli attributi in modo da fornire all'AP un modo per verificare che il nodo sia autorizzato a richiedere informazioni su un dato utente.

Sebbene sia stata implementata attraverso determinati programmi, la soluzione sviluppata non è legata necessariamente ad essi, ma può essere adattata e generalizzata in base al software utilizzato per implementare l'IdP e il nodo eIDAS. La scelta dell'uso di comunicare tramite `AuthnRequest` consente all'AP sviluppato di dialogare con qualsiasi provider SAML 2.0, essendo questo il profilo più diffuso e supportato. Nel caso in cui il sistema di identità digitale presso cui si trova l'IdP dovesse essere basato su un protocollo di comunicazione differente, come OpenID Connect, è possibile adattare l'AP semplicemente sostituendo il modulo SP Shibboleth con un modulo Apache che protegga la risorsa *RemoteUser* sfruttando tale protocollo.

Il codice sviluppato si basa su meccanismi e protocolli affidabili per salvaguardare la riservatezza e l'integrità dei dati nei diversi passaggi. Grazie al canale TLS con autenticazione client, se configurato correttamente, le informazioni sugli AP vengono rilasciate solo agli eIDAS-Service autorizzati e ne viene garantita l'integrità durante il passaggio nella Rete. I Metadati di tali AP devono essere firmati con un certificato facente parte di una PKI la cui radice sia considerata fidata dal Service; per ogni provider deve essere usato un certificato diverso, in modo che, in caso di compromissione del provider, tale certificato possa essere immediatamente revocato. L'uso di TLS deve essere esteso anche alle comunicazioni tra l'utente e i provider, in modo che il primo possa verificare l'identità di questi ultimi; l'uso di cypher suite adeguate garantisce, inoltre, integrità e riservatezza del canale. Sfruttando la possibilità di firmare i messaggi SAML e cifrare le asserzioni contenenti gli attributi, tali dati non possono essere alterati o estrapolati neppure dall'utente, che estrae i messaggi dai canali TLS per poterli inoltrare.

6.2 Possibili sviluppi

Il sistema sviluppato consente ad un eIDAS-Service di integrare le informazioni di un utente con quelle recuperate tramite un AP ed è stato sviluppato avendo come obiettivo la realizzazione di un servizio sicuro ed affidabile; esso costituisce un punto di partenza a cui aggiungere funzionalità che possano renderlo adatto all'impiego in ogni situazione.

Le modifiche apportate al software dell'eIDAS-Service fornito dalla rete di cooperazione danno la possibilità all'utente di scegliere un solo AP da cui recuperare gli attributi mancanti Tale scelta è stata fatta tenendo conto di uno scenario in cui, oltre agli attributi recuperati tramite il servizio di identità digitale, sia necessario recuperarne altri appartenenti ad uno specifico ambito: in tale situazione, è verosimile che gli attributi mancanti vengano recuperati da un'unica fonte. Tuttavia, possono esserci molti casi in cui gli attributi mancanti riguardino settori differenti; sarebbe utile, perciò prevedere la possibilità di contattare diversi AP nell'ambito di un'unica richiesta di autenticazione. Per far questo, si potrebbe intervenire sulla richiesta di consenso presentata all'utente dal Service e consentire di specificare un AP per ogni attributo aggiuntivo; sarebbe poi necessario raggruppare gli attributi da richiedere ai singoli AP e apportare alcune modifiche alla fase di gestione della risposta SAML, in modo da emettere più richieste in sequenza ad AP differenti.

Una funzionalità da aggiungere nel caso di richieste ad AP differenti sarebbe quella di verificare gli attributi che è possibile recuperare da ognuno di essi e presentare per ogni attributo richiesto, al momento della scelta, solo gli AP che possono effettivamente fornire tale attributo. Questo richiederebbe che i provider inseriscano, all'interno dei propri Metadati, la lista degli attributi che essi sono autorizzati a rilasciare; in tal modo l'eIDAS-Service potrebbe contattarli per recuperare i loro Metadati ed estrarre tale lista, così da costruire un elenco di attributi aggiuntivi, insieme con gli AP da cui recuperarli. Un'altra soluzione potrebbe essere quella di inserire queste informazioni tra quelle rilasciate dal Web Service nel momento in cui l'eIDAS-Service richiede la lista degli AP disponibili.

Il Web Service incluso nel progetto è solamente un prototipo, realizzato per testare il funzionamento del sistema e illustrare quale sia il suo ruolo come *repository* certificata di informazioni sugli AP. Nel distribuire il sistema nella realtà, esso deve essere implementato tenendo conto dell'importanza cruciale che ha per il corretto funzionamento dell'apparato: se esso dovesse venir meno, per l'eIDAS-Service sarebbe impossibile contattare gli AP, anche se essi fossero in funzione. In caso di compromissione del Web Service un attaccante potrebbe rilasciare false informazioni che porterebbero l'eIDAS-Service a fidarsi di AP malevoli, con tutte le conseguenze che questo porta. È necessario, quindi, adottare misure adeguate per proteggere il Web Service e i dati che esso contiene, prevedendo anche precauzioni contro attacchi Denial of Service, per garantire continuità al servizio.

La metodologia adottata per descrivere come combinare gli attributi rilasciati dall'IdP per ottenere l'identificativo dell'utente usato dall'AP si adatta a numerose situazioni e può essere ulteriormente estesa per prevedere nuove forme di manipolazione degli attributi, ad esempio aggiungendo la possibilità di attivare alcuni token solo in presenza di una certa condizione. Il sistema proposto funziona solo se l'ID utente usato dall'AP è interamente ricavabile dagli attributi rilasciati dall'IdP, elaborati e composti insieme con parti statiche, ovvero uguali per tutti gli utenti. Potrebbe accadere che il nome utente sia completamente slegato da altri attributi (si pensi alla matricola universitaria) o che comprenda parti scelte dall'utente (come, ad esempio, un indirizzo mail): in questi casi, anche con il sistema proposto, l'eIDAS-Service non sarebbe in grado di costruire l'ID senza interagire con l'utente. Si potrebbe prevedere, quindi, uno speciale token che informi il nodo della necessità di richiedere al soggetto l'inserimento del proprio identificativo; tale soluzione,

tuttavia, ha lo svantaggio di impedire all'AP di verificare la corrispondenza tra tale ID e l'utente autenticato presso l'IdP. Una possibile soluzione a tale problema potrebbe essere quella di informare l'eIDAS-Service, tramite un token, di adoperare l'identificativo rilasciato dall'IdP come nome utente per interrogare l'AP e lasciare che quest'ultimo, tramite una lista di corrispondenze, estragga l'ID corretto; questo meccanismo, tuttavia, risulta di difficile attuazione e richiede che l'AP conservi ulteriori informazioni sugli utenti, cosa non sempre possibile.

All'inizio del Capitolo 3 è stata fatta l'assunzione che l'AP si trovi nello stesso Stato del sistema di identità digitale presso cui l'utente è registrato: questa ipotesi è resa necessaria dal fatto che il protocollo di comunicazione della rete eIDAS non prevede alcun modo per inoltrare richieste di attributi tra i diversi Stati. Anche se l'eIDAS-Service contattato per autenticare l'utente constatasse che alcuni degli attributi richiesti devono essere recuperati da un AP fuori dai confini nazionali, esso non avrebbe alcun mezzo per notificare la richiesta di tali attributi all'eIDAS-Service dello Stato appropriato; né l'eIDAS-Connector potrebbe richiederli dopo aver ricevuto la risposta del Service con l'autenticazione. Per consentire un tale scenario sarebbe necessaria quanto meno una revisione del protocollo di interoperabilità al fine di includere, ad esempio, il profilo Attribute Query tra quelli supportati dai nodi eIDAS. In ogni caso, si dovrebbe intervenire anche sull'eIDAS-Connector per fare in modo che sia esso e non più il Service a verificare la mancanza di attributi e a chiedere all'utente quale Stato contattare.

Nel caso in cui fosse possibile scambiare le richieste di attributi tra i nodi della rete, il protocollo implementato dalla soluzione presentata dovrebbe essere adattato alla nuova situazione, in quanto è previsto che l'AP richieda l'autenticazione dell'utente al sistema di identità digitale presso cui è registrato: tale autenticazione non potrebbe avvenire direttamente se l'AP si trovasse in uno stato diverso. Sarebbe opportuno riprogettare il protocollo includendo la possibilità, per l'AP, di usare la stessa rete eIDAS per ottenere l'autenticazione, come se fosse un Relying Party qualunque; una soluzione alternativa consisterebbe nel progettare lo scambio di richieste di attributi tra i nodi eIDAS in modo che esse trasportino anche le autorizzazioni al rilascio di questi ultimi, magari all'interno di un'apposita estensione SAML.

Le specifiche della rete eID eIDAS e la rete stessa sono in fase avanzata di sviluppo: come previsto dal regolamento del Consiglio [1, art. 52], il riconoscimento obbligatorio da parte dei Paesi sottoscrittori dei mezzi di autenticazione elettronici degli altri Stati entrerà in vigore nel settembre del 2018. Entro tale data la rete di nodi eIDAS dovrà essere pienamente funzionante; pertanto, risulta improbabile che, nel prossimo futuro, possano essere apportate modifiche radicali come la possibilità di recuperare attributi da Paesi diversi da quello in cui l'utente viene autenticato. Tuttavia, è verosimile che i nodi eIDAS implementati dai diversi Stati sfruttino le tecniche descritte in questo lavoro per consentire al sistema di integrare le informazioni recuperate in fase di autenticazione dell'utente con quelle messe a disposizione da enti terzi, per ampliare le possibilità di applicazione del sistema di interoperabilità.

Bibliografia

[1] Council of European Union, "Regulation (EU) No 910/2014", Official Journal L 257, August 28, 2014, pp. 73-114, ELI http://data.europa.eu/eli/reg/2014/910/oj

[2] D.Berbecaru, A.Lioy, "On the design, implementation and integration of an Attribute Provider in the Pan-European eID infrastructure", 2016 IEEE Symposium on Computers and Communication (ISCC), Messina (Italy), June 27-30, 2016, pp. 1263-1269, DOI 10.1109/ISCC.2016.7543910

[3] Council of European Union, "Commission Implementing Regulation (EU) No 2015/1502", Official Journal L 235, September 9, 2015, pp. 7-20, ELI http://data.europa.eu/eli/reg_impl/2015/1502/oj

[4] Council of European Union, "Commission Implementing Regulation (EU) No 2015/1501", Official Journal L 235, September 9, 2015, pp. 1-6, ELI http://data.europa.eu/eli/reg_impl/2015/1501/oj

[5] eIDAS Expert Group, "eIDAS - Interoperability Architecture", version 1.00, November 6, 2015, URL https://ec.europa.eu/cefdigital/wiki/pages/viewpage.action?pageId=37750723

[6] D.Cooper, S.Santesson, S.Farrell, S.Boeyen, R.Housley, W.Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC-5280, May 2008, DOI 10.17487/RFC5280

[7] eIDAS Expert Group, "eIDAS - Cryptographic requirements for the Interoperability Framework - TLS and SAML", version 1.0, November 6, 2015, URL https://ec.europa.eu/cefdigital/wiki/pages/viewpage.action?pageId=37750723

[8] eIDAS Expert Group, "eIDAS Message format", version 1.1, October 22, 2016, URL https://ec.europa.eu/cefdigital/wiki/pages/viewpage.action?pageId=37750723

[9] Y.Sheffer, R.Holz, P.Saint-Andre, "Summarizing Known Attacks on Transport Layer Security (TLS) and Datagram TLS (DTLS)", RFC-7457, February 2015, DOI 10.17487/RFC7457

[10] Agenzia per l'Italia Digitale, "Regolamento recante le regole tecniche (articolo 4, comma 2, DPCM 24 ottobre 2014)", July 28, 2015, URL http://www.agid.gov.it/sites/default/files/circolari/spid-regole_tecniche_v1.pdf

[11] S.Cantor et al, "Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0", OASIS Standard, March 15, 2005, URL https://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf

[12] H.Lockhart, B.Campbell, S.Cantor, "SAML V2.0 X.500/LDAP Attribute Profile", March 27, 2008, URL http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-attribute-x500-cs-01.html

[13] M.Mealling, "A URN Namespace of Object Identifiers", RFC-3061, February 2001, DOI 10.17487/RFC3061

[14] ITU-T Recommendation G.114, "Information technology - Open Systems Interconnection - The Directory: Selected attribute types", October 2016, URL http://handle.itu.int/11.1002/1000/13036

[15] M.Jones, J.Bradley, N.Sakimura, "JSON Web Token (JWT)", RFC-7519, May 2015, DOI 10.17487/RFC7519

[16] P.Hunt, Ed., K.Grizzle, E.Wahlstroem, C.Mortimore, "System for Cross-domain Identity Management: Core Schema", RFC-7643, September 2015, DOI 10.17487/RFC7643

[17] P.Hunt, Ed., K.Grizzle, M.Ansari, E.Wahlstroem, C.Mortimore, "System for Cross-domain Identity Management: Protocol", RFC-7644, September 2015, DOI 10.17487/RFC7644

[18] S.Cantor et al, "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0", OASIS Standard, March 15, 2005, URL https://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf

[19] R.Housley, "Cryptographic Message Syntax (CMS)", RFC-5652, September 2009, DOI 10.17487/RFC5652

[20] K.Moriarty, Ed., B.Kaliski, J.Jonsson, A.Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2", RFC-8017, November 2016, DOI 10.17487/RFC8017

[21] European Commission, DIGIT Unit D3, "eIDAS-Node Installation and Configuration Guide Version 1.4", October 6, 2017, URL https://ec.europa.eu/cefdigital/wiki/download/attachments/46992189/eIDAS-Node Installation and Configuration Guide.pdf