Supporting Authorize-then-Authenticate for Wi-Fi access based on an electronic identity infrastructure

Diana Berbecaru*, Antonio Lioy, and Cesare Cameroni
Politecnico di Torino, Torino, Italy
{diana.berbecaru, lioy, cesare.cameroni}@polito.it

Received: February 20, 2020; Accepted: June 2, 2020; Published: June 30, 2020

Abstract

Federated electronic identity systems are increasingly used in commercial and public services to let users share their electronic identities (eIDs) across countries and providers. In Europe, the eIDAS Regulation and its implementation - the eIDAS Network - allowing mutual recognition of citizen's eIDs in various countries, is now in action. We discuss authorization (before authentication), named also *authorize-then-authenticate* (AtA), in services exploiting the eIDAS Network. In the eIDAS Network, each European country runs a national eIDAS Node, which transfers in other Member State countries, via the eIDAS protocol, some personal attributes, upon successful authentication of a person in his home country. Service Providers in foreign countries typically use these attributes to implement authorization decisions for the requested service. We present a scenario where AtA is required, namely Wi-Fi access, in which the service provider has to implement access control decisions `before` the person is authenticated through the eIDAS Network with his/her national eID. The Wi-Fi access service is highly required in public and private places (e.g. shops, hotels, a.s.o.), but its use typically involves users' registration at service providers and is still subject to security attacks. The eIDAS Network supports different authentication assurance levels, thus it might be exploited for a more secure and widely available Wi-Fi access service to the citizens with no prior registration, by exploiting their national eIDs. We propose first a model that discusses AtA in eIDAS-based services, and we consider different possible implementation choices. We describe next the implementation of AtA in an eIDAS-based Wi-Fi access service leveraging the eIDAS Network and a Zeroshell captive portal supporting the eIDAS protocol. We discuss the problems encountered and the deployment issues that may impact on the service acceptance by the users and its exploitation on large scale.

Keywords: authorization, electronic identity infrastructures, eIDAS Network, Wi-Fi access service

1 Introduction

The electronic identity (eID) (sometimes called also "electronic identification") is a digital solution allowing to prove the identity of citizens or organizations when they try to access remotely various services provided by public or private organizations. On one hand, the eIDs have started to be largely deployed to the citizens, e.g., the electronic identity cards are provided to the citizens in several European countries. On the other hand, the eIDs are still scarcely used in practical scenarios although some improvements can be seen at the national level, e.g., in Italy, the public administrations must support eIDs issued by SPID [1] Identity Providers (IdPs) to the citizens, such as InfoCert SpA or Poste Italiane SpA [2].

Although awareness is constantly rising, in cross-border scenarios, the eIDs are still barely recognized in services provided in countries different from the one where the eID was issued. In the past, several projects addressed this problem and put the grounds for electronic identification and recognition in

Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA), 11(2):34-54, June 2020
DOI:*10.22667/JOWUA.2020.06.30.034*

*Corresponding author: Dip. di Automatica e Informatica, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129, Torino, Italy, Tel: +39-011-090-7081, Web: `https://security.polito.it/~diana/`

Europe. For example, the European pilot projects STORK and STORK2 [3] [4] created a pan-European eID interoperability framework, which allowed authentication means from one country to be accepted in applications outside of the origin country's jurisdiction, such as to register a student at foreign universities [5] or even to provide wireless roaming access [6]. The results of the STORK project were used in the definition of the eIDAS (electronic IDentification, Authentication and trust Services) Regulation [7], which allows nowadays legal recognition of eIDs across the European Union (EU). Moreover, the European Commission also maintains the eIDAS technical specification [8] and releases periodically the updated code [9] for the eIDAS Nodes, which exploits the Security Assertion Markup Language (SAML). Thus, in the last years, many European countries have set up eIDAS Nodes that are part of a Circle of Trust (CoT), named the eIDAS Network. Each eIDAS Node is connected to the national IdPs implementing the notified eID scheme - further referred as notified IdPs - to authenticate (legal or natural) persons and to provide basic identification attributes for them, and to the national Service Providers (SPs) to allow foreign citizens to authenticate in their home country when accessing eIDAS-enabled services. We observe that only national eID schemes that will be notified by the countries need to be connected to the eIDAS Network [10]. Examples of notified eID schemes are SPID (Sistema Pubblico di Identità Digitale) in Italy, Spanish ID card (DNIe) in Spain, Key card/token (OTP) and Mobile app in Denmark, Digital Mobile Key in Portugal, or the National identity card in Germany. The key point, however, is that if we consider only the eID cards, it is foreseen that by 2020 more than 30% of the 500 million citizens in Europe will own one, not considering the other types (more user friendly) credentials issued by the national eID schemes. So, a huge amount of users have or will own very soon such credentials (with different levels of assurance) issued by their governments or notified IdPs, but they are still scarcely exploited in online scenarios. Our work addresses the creation of possible widely used services leveraging a secure eID infrastructure (like the eIDAS Network) and user's authentication credentials.

In our previous work on this subject [11], we observed that eIDAS responds to authentication requirements of on-line web-based applications especially in cross-border scenarios, but additional work is needed to cover those services requiring authorization before authentication. For example, in the Wi-Fi access scenario, the user does not have access to the network, but the service itself consists of getting access to the network, provided that the user can demonstrate to the SP that he has been authenticated by some (recognized) IdP in eIDAS. We called this service *eAccess* (eIDAS-based Wi-Fi access service). This resembles the chicken-and-egg problem: to provide access, the SP requires authentication data from the IdP, but to obtain this data the user must be first authorized to access the remote IdP. We propose an architecture based on the eIDAS Network, which supports "authorize-then-authenticate" (AtA) and we discuss design options for its main components. Next, we present an AtA implementation for the eAccess service, by designing and implementing a SAML-enabled captive portal based on Zeroshell [12], which we have integrated with the eIDAS Network. A citizen that has an authentication credential released by an IdP of an eID scheme notified under eIDAS, e.g. a username-password, a national eID card or a mobile token, he/she may use it to get Wi-Fi access to the network across EU, provided the SP has been integrated with the eIDAS Network and it supports AtA through a dedicated solution, such as the one proposed and described further below. However, as discussed in our work, several issues have to be faced when implementing the service, which might compromise the functionality, the scalability, or even the acceptance of the service by the users.

Organisation. The paper is organised as follows: Sect. 2 discusses the related works, Sect. 3 presents the eIDAS Network in brief, Sect. 4 describes our architecture, the design issues for the various components in scenarios requiring AtA, and two main design models. We detail our implementation scenario for the eIDAS-based Wi-Fi access service in Sect. 5, we provide additional implementation details in Sect. 6, and we discuss the main problems encountered and the open issues in Sect. 7. Finally, Sect. 8 concludes the paper and indicates future work.

2 Related work

We reviewed some solutions proposed for Wi-Fi access, in particular the ones exploiting Shibboleth [13] (i.e. SAML-based), and others based on other technologies that are most widely used nowadays, especially in the academic environment.

One of the first and simple solutions exploiting Shibboleth for user's network access control was proposed in [14], and worked as follows. The user connected to the docking network (e.g. by activating her WLAN card) and got an IP address via DHCP. All traffic was initially blocked except the one for the port TCP/443 (HTTPS) on the server hosting the Shibboleth 'Where Are You From' (WAYF) service allowing the selection of the IdP, and the traffic for all the IdPs in the federation. The initial HTTP request from the UA was intercepted by the NAC (Network Access Controller), which was a web server equipped with Shibboleth SP component. NAC constructed a SAML authentication request and redirected the UA to the WAYF service where the user selected the IdP from a drop-down list. The WAYF application redirected the UA to the HTTPS URL of the IdP where the authentication process took place. After successful authentication, the IdP redirected the UA back to the SP with a message containing a signed SAML assertion. Based on user's attributes in the SAML assertion, the NAC decided if the user was authorized to access the network. If access was granted, NAC updated the firewall rules so that traffic could flow between the client and the network. This solution assumed that the IP address of the IdP was known by SP and could be pre-configured on the NAC. In a real scenario, the SP could obtain the IdP's IP address/DNS name during the federation setup phase, but if the IdP's name was changed, the SP had to be notified in advance securely (e.g. via an out-of-band channel). This simple approach has been adopted also in the Zeroshell captive portal, with similar SAML messages flowing between the SP and IdP and the configuration of the IdP's DNS name or IP address on (Zeroshell) SP side.

On wide scale, nowadays also other solutions address the Wi-Fi access by exploiting different technologies. For example, Eduroam [15] is a world wide roaming service for education and research, it allows educational users (teacher, researchers, and students) who are members of an educational institution to log on to the WLAN (Wireless LAN) of a visited institution by using the same credentials (username and password) that they have at their home institution.

Eduroam authentication is based on a hierarchical RADIUS [16] infrastructure (which typically consists of three levels: organizational, national, and global) and uses 802.1X standard. When the local RADIUS server can't authenticate the visiting user because he/she is not among the registered users, user's credentials are forwarded to the visiting user's home institution RADIUS server passing through the national level Eduroam RADIUS server and, if needed, the top level Eduroam RADIUS server. In this way, users don't need new credentials to access the network of the visited institution, but on the other hand, it is not possible to use authentication credentials stronger than username and password.

Nowadays, Eduroam is largely used in the academic environment, and in some countries, the service is also available in other places like libraries, public buildings, and airports, but it cannot be used if a person is not part of such an environment (student, teacher, etc.). In contrast, our proposed solution can be used even by persons that temporarily visit a university (for a meeting, conference a.s.o.) **without** necessarily being part of the academic staff. In our approach, to get access to the network the citizen uses his national authentication credential, e.g. a static password, a smartcard or a (one time) password combined with the use of his device (e.g. a smartphone) instead of exploiting his university authentication credentials, which might not exist or might be expired.

Although widely deployed and used, Eduroam presents some drawbacks. In RFC 7593 [17] Wierenga et al. outlines some privacy and security issues such as service providers collusion, user credentials exposure, user location tracking, MITM (Man-in-the-Middle) attacks, and DoS (Denial of Service) attacks. Brenza et al. [18] investigated how client device misconfiguration and poor software implementation can lead the user to trust a counterfeit access point allowing identity theft and MITM attack. Wierenga et al.

proposed to enhance Eduroam security via a new trust hierarchy [17] based on RADIUS/TLS [19], while Liu et al. [20] proposed a scheme to improve Eduroam stability and performance.

Despite possible problems, Eduroam is largely exploited nowadays. For example, Govroam (government roaming) [21] implements roaming for government agencies and it is developed on the same architecture as Eduroam. It provides connectivity to the Internet and availability of resources present at home organization by connecting from the network of a visiting organization. Govroam started from The Netherlands and it is based on policies and principles provided by the "NL Service Policy" document [22], which has to be agreed and followed by all the organizations participating in the roaming network.

When implementing the model illustrated in Section 4, also other possible tools may be employed on the SP side. Other alternatives to Zeroshell for the captive portal are for example PacketFence [23], an open-source tool for network access control (NAC) solutions providing also a captive portal, and NoDogSplash [24], which is a small footprint captive portal that provides an API that can be integrated with authentication applications.

3 Short Review of the eIDAS Network: components, workflows, attributes

The eIDAS Network is an eID infrastructure being supported by several countries across Europe in order to allow: 1) the authentication of citizens from different EU countries with credentials issued by their national eID scheme(s) when accessing services in a foreign country; 2) the transfer of a set of basic identification attributes (about a natural or legal person) in other countries at the authentication time, to be exploited in eIDAS-enabled services. The Innovation and Networks Executive Agency of the European Commission, through its Connecting Europe Facility (CEF) programme, and in particular via the CEF eID Building Block, is performing several activities to support eIDAS implementation and adoption. In particular, it maintains the eIDAS specification, it maintains and distributes the reference code for the eIDAS Node (currently two versions [9] exist, i.e. version 1.4.5 and version 2.4), and it promotes and finances initiatives that facilitate the integration of public and private service providers in the eIDAS infrastructure. In terms of privacy, eIDAS complies with the OECD (Organization for Economic Cooperation and Development) privacy principles of user data collection, limitation and data quality, ensuring user data sovereignty.

The eIDAS Network exploits the SAML protocol [25] and supports two models, proxy (currently supported by many countries, e.g. Italy, Spain, Portugal, Belgium) and middleware (currently supported by Germany). In the proxy model, a country runs a single gateway called eIDAS Node, which is composed of two elements, an eIDAS Proxy Service (in short, eIDAS Proxy) and an eIDAS Connector, although several eIDAS Connectors are allowed to be deployed in a country, e.g. one per specific sector. These components establish trust relationships with the national IdPs and SPs, so if SAML is used also in their communication, they share national SAML metadata. Optionally there may also be two further elements, a Specific Connector, and a Specific Proxy, in charge of the translation between eIDAS SAML and national SAML messages, as shown in Fig. 1. The eIDAS Nodes also are in a CoT, and their SAML metadata is distributed through a dedicated mechanism. Cross-border authentication is delegated from an SP to its national eIDAS Connector (or to its Specific Connector), which acts as a gateway and subsequently forwards the eIDAS authentication request (*eIDAS_auth_req*) to the eIDAS Proxy in the country in which the person will be authenticated. The *eIDAS_auth_req* is handled by the eIDAS Proxy according to Member-State (MS) specific approach. Typically, a new authentication request is constructed by the Specific Proxy and is sent (through the user's browser) to the national IdP (part of the National eID scheme) where the citizen is asked to authenticate with a national eID. For example in Italy, the *eIDAS_auth_req* is converted to an authentication request in SPID [1] format by the Specific Proxy, which in Italy is called IdP Proxy. Upon successful authentication, the eIDAS authentication re-

sponse (*eIDAS_auth_resp*) containing also the (personal) attributes that have been requested are returned through the eIDAS Network back to the requesting SP. Each eIDAS Node communicates with the other eIDAS Nodes through the eIDAS communication protocol [26], which is based on SAML 2.0 WebSSO Profile [27].

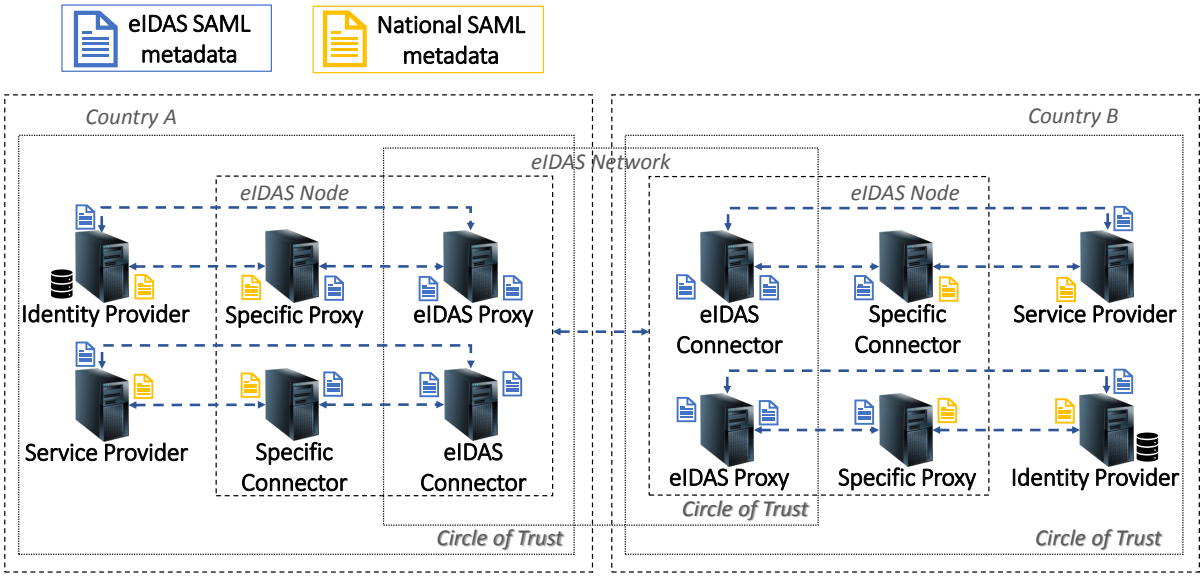

Figure 1: Possible connection of the eIDAS Nodes (version 1.4.x), employing Specific Connector and Specific Proxy components in two countries adopting the proxy model.

The eIDAS Network supports only a limited number of natural person attributes to be exchanged through the eIDAS Nodes, some of them are mandatory, i.e. FamilyName, FirstName, DateOfBirth, and PersonIdentifier, others instead are optional, e.g. BirthName, CurrentAddress, PlaceOfBirth, and Gender. Mandatory attributes are always transferred through the eIDAS Network and they are referred to as eIDAS Minimum Data Set (MDS). As explained in [28] [29], these attributes are not typically sufficient to build more advanced services. Nevertheless, in eAccess they might be enough to implement network access decisions, but in some cases, e.g., for users that have strict requirements on providing such extremely sensitive data to SPs, the user could forgo network access altogether rather than providing their personal data.

The Italian eIDAS Node currently uses the eIDAS code version 1.4.4 [30] but it is expected to migrate to the 2.x branch of the code this year, which implies a redesign and implementation of some parts of the Specific Connector and Specific Proxy modules currently used.

4 Architecture and design models proposed

Federated identity management (FIM) solutions do not natively provide authorize-then-authenticate feature, which is required for example in the proposed service. All practical FIM architectures require the user to access **first** its own IdP to authenticate and afterward the SP provides access to the services (upon successful authentication) based on the authentication token (such as a SAML response) generated and signed by the IdP, which is transparently sent and processed by the SP. On the other hand, in the network access service, the SP must grant the user limited network resource access before the authentication process has been actually completed. On wide scale, e.g., if the SP and IdP are in different countries, this turns to be a challenging task: the network access provider cannot be assumed to know and trust

every possible IdP, but external interactions belonging to the authentication process should be carefully authorized by the SP to avoid covert channels.

4.1 Generic architecture for AtA support in eIDAS-enabled services

To discuss the authorize-then-authenticate support in eIDAS-enabled services, we use the generic architecture shown in Fig. 2. The user accesses an eIDAS-enabled service through various devices (e.g. PC, tablet, smartphone) and owns one or more authentication credentials - e.g. a username/password used possibly in combination with the personal device, a One Time Password (OTP), an eID card - issued by the notified IdPs under eIDAS.

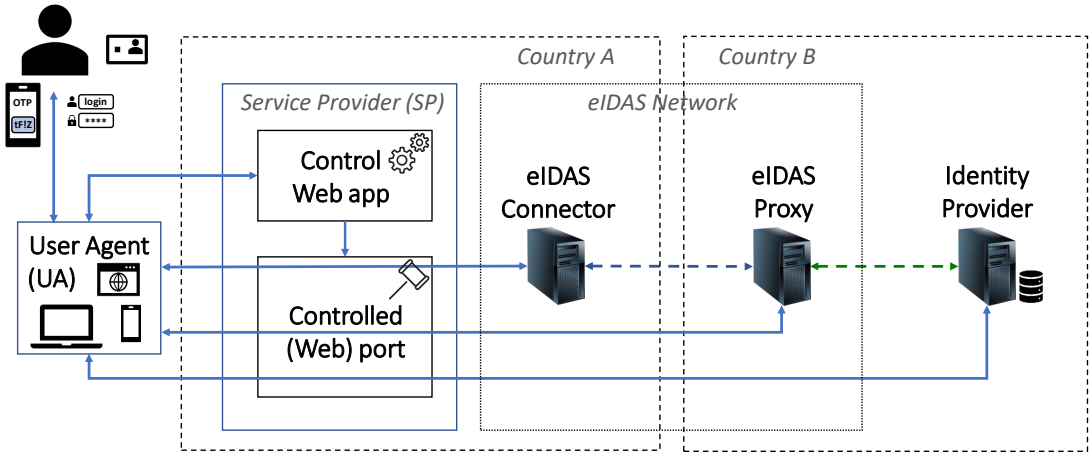

Figure 2: Elements for authorize-then-authenticate support at Service Provider in services exploiting the eIDAS Network.

The user's web browser named also User Agent (*UA*) interacts with SP's *Control Web app* (named also *Control service*) to allow user authenticate as well as his authorization to access external resources, both before the authentication process and afterward. The *Control service* defines the authorization policy for the *Controlled (Web) port*, which allows or denies UA with the right to communicate with external resources. Before full access is granted, *UA* is required to interact with the eIDAS Network to collect the required authentication response and the attributes employed in the authorization decisions. The protocol is user-centric in compliance with the most modern FIM concepts: *UA* switches tokens among the various involved servers, including the eIDAS Nodes in the SP and IdP countries, the user's IdP, and the SP *Control Web app*. No direct server-to-server interaction is assumed inline. *Control service* verifies the authentication and authorization credentials collected by *UA* and takes the ultimate decisions enforced by the *Controlled (Web) port*. The eIDAS Connector translates externally collected credentials into data that the SP can directly verify, thus masking to SP the complexity of the eID infrastructure.

4.2 Design models

In [11], we considered two main design models for our architecture (shown in Fig. 2), one working exclusively at the application level, the other one involving also the transport level, and we assumed that the eIDAS Nodes and the IdPs may issue authorization tokens. Unfortunately, this functionality is not supported by default in the eIDAS Nodes, which are only in charge of transferring authentication request and response messages through the eIDAS Network but they cannot currently create and/or process

authorization tokens. In this work, we assume that the functionality of the IdP and of the eIDAS Node cannot be easily changed for authorization purposes at SP side.

For simplicity, we considered only simple rules for authorization policies at SP site:
($\mathscr{R}1$) "UA X is not allowed to connect to any server";
($\mathscr{R}2$) "UA X is allowed to connect to server (app) Y";
($\mathscr{R}3$) "UA X is allowed to connect to any server (app)".

The initial policy for every user contains $\mathscr{R}1$ only, while $\mathscr{R}2$ rules are added dynamically by SP upon obtaining correct authorization tokens from the local eIDAS Connector, whereas $\mathscr{R}3$ is applied also dynamically by SP upon successful authentication of the user. In general, $\mathscr{R}3$ can be further extended to require specific attributes (e.g. the role Z) or to restrict access to specific classes of external resources.

However, in a simpler case, *static configuration* may be performed on SP to enable the $\mathscr{R}2$ rules. For example, if the DNS names of the eIDAS Connector, of the eIDAS Proxy, and of the IdP are static and trustworthy, they could be statically configured on the SP site, so that the user can reach the IdP. In this case, the user authenticates to the IdP, and upon the receipt of the SAML authentication response token, the SP applies the authorization policy $\mathscr{R}3$. This case is considered in the implementation scenario presented further below. Finally, we restrict our study to web-based scenarios, i.e., services using HTTP and HTTPS (HTTP over TLS channel [31], and HTTP then TLS [32]).

Control Web Application. This application is the internal authorization decision point at SP and performs two tasks: (1) it verifies authentication tokens retrieved through eIDAS Network, including the binding data preventing attackers to intercept and use tokens in place of the real owner; (2) it configures the controlled port, i.e. it defines the authorization policies to be applied. In a practical design, the application is a Web application supporting the specific authorization token format along with the specific control mechanism to configure the controlled port. For example, the application could use the Shibboleth SAML-aware interface to support SAML-based tokens. Alternatively, for HTTP-based tokens, the application could use the APIs and documentation provided by OAuth supporters to enable OAuth into web applications [33]. Finally, an ad-hoc application may be developed in case the authorization token is a X.509 v3 certificate, as proposed in the TLS-federation approach [34].

Controlled (Web) port. This module is in charge of enforcing the authorization policy by allowing or denying connections from the UA to the external Web applications/servers. In practice, it must verify the authentication and authorization data associated to each UA and allow/deny its connections toward specific external servers. In our previous work [11], we have identified two design alternatives that could be adopted at different levels: (1) at the transport-level, where the HTTP server at SP plays the *Controlled (Web) port* role, and (2) at the application-level, where a dedicated Web application or protocol may be used.

The application-level option is more flexible under this perspective. We could exploit TLS protocol to protect HTTPS connections between UA and *Controlled (Web) port* and between the *Controlled (Web) port* and the external servers. We can instead rely on secure Web service protocols, e.g. WS-SecureConversation [35], to protect end-to-end message exchange between UA and the external servers. Unfortunately, this requires the re-design of the applications running on the external servers, including the eIDAS Node(s) and the IdP(s). Moreover, it is difficult to grant user awareness without dedicated features in the browser allowing to manage secure web service protocols. Though the *application-level* solution is appealing in the long term, nevertheless, the transport-level solution integrates better into the existing Web-based infrastructure.

In the *transport-level* solution, the HTTP server at SP acts as *Controlled (Web) port.* In this case, the server must be capable of proxying HTTPS channels from UA to the external servers. The Apache Web server provides the "mod_proxy", 'mod_ssl'', and "mod_connect" modules to proxy HTTP and HTTPS sessions, but a couple of technical issues need to be addressed. First, an appropriate mechanism is needed to allow the *Control Web app* signal the current authentication and authorization policy to

the HTTP server. In other words, the *Control Web app* should translate the SAML-based credentials collected by the UA into access credentials understood by the HTTP server.

5 Implementation scenario

In the practical implementation scenario of the architecture shown in Sect. 4, we focused on the SP providing the Wi-Fi access service and the eIDAS Nodes, to analyze their interaction. We assumed that the IdPs connected to the eIDAS Nodes are correctly performing authentication and attribute retrieval, so their functionality is not part of the described work. Furthermore, we assumed also that the DNS names of the eIDAS Connector, eIDAS Proxies and the IdPs are static and trustworthy.

In the implemented scenario, the SP provides Wi-Fi network access service to users from different countries, by exploiting the eIDAS Network and the national credentials issued to the citizens by the IdPs supporting notified eID schemes under eIDAS. We designed and deployed a captive portal integrated with eIDAS, which, to the best of our knowledge, has not been proposed and implemented yet. We used Zeroshell because it already implements a captive portal allowing to authenticate users against an IdP SAML 2.0 by using Shibboleth. To apply the $\mathscr{R}2$ rules, we performed the *static configuration* on the SP site, in particular, the DNS names of the eIDAS Connector, eIDAS Proxies, and of the notified IdPs were configured in Zeroshell.

Thus, on the SP side, we installed firstly Zeroshell as described in Section 6.1. Next, we modified Shibboleth SP's configuration to generate an *eIDAS_auth_req* message instead of a plain SAML request, which is sent through the eIDAS Network. From the technical point of view, the Shibboleth SP must support the eIDAS protocol and the WAYF service allowing the user to select the country in which he will be authenticated. Moreover, the Shibboleth SP needs to be configured with the SAML metadata of the eIDAS Connector, so that the eIDAS request and response messages exchanged with it can be cryptographically verified.

We have installed the eIDAS code version 1.4.4 in an experimental testbed hosting a dedicated academic eIDAS Node, which acts both as eIDAS Connector and as eIDAS Proxy Service. The test eIDAS Node was connected to an IdP ((in a controlled environment)) issuing real Italian SPID credentials (so-called SPID IdP), namely the Infocert IdP operating in Italy. In practice, in the frame of the eID4U project [36], we installed in addition the IdP Proxy in the testbed environment, since the eIDAS Node communicates with the national IdPs through this component.

Authentication sequence diagram. We consider an Italian citizen trying to use the Wi-Fi access service at Politecnico di Torino (Polito) by authenticating himself with his national eID, issued by a notified IdP, e.g. a SPID credential. Thus, the SP at Polito is connected to the Italian eIDAS Node. Note that, in case of a foreign citizen, the eIDAS Proxy and the Specific Proxy component (if any) is abroad in the citizen's country.

The steps performed are illustrated in Fig. 3: The user tries to open a web page, e.g. `http://www.abc.com` (step 1). The HTTP request is intercepted by the captive portal, which automatically redirects the user browser (referred also as UA in Fig. 2) to the captive portal's start web page (step 2). The user selects the country he will authenticate in (i.e. "Italy" in the depicted scenario) and then he clicks on the eIDAS authentication button (step 3).

Next, the user browser is redirected to the eIDAS Connector along with the *eIDAS_auth_req* generated by the SP (step 4). The eIDAS Connector receives the *eIDAS_auth_req* message through the user browser, it processes it internally, then it generates a new *eIDAS_auth_req* (digitally signed by the eIDAS Connector), and sends it through the UA to the eIDAS Proxy (steps 5,6). The eIDAS Proxy receives the *eIDAS_auth_req* message through the UA (step 7) and processes it internally. At this step, the eIDAS Proxy requests user's consent to perform attribute retrieval. In particular, the user sees the attributes being

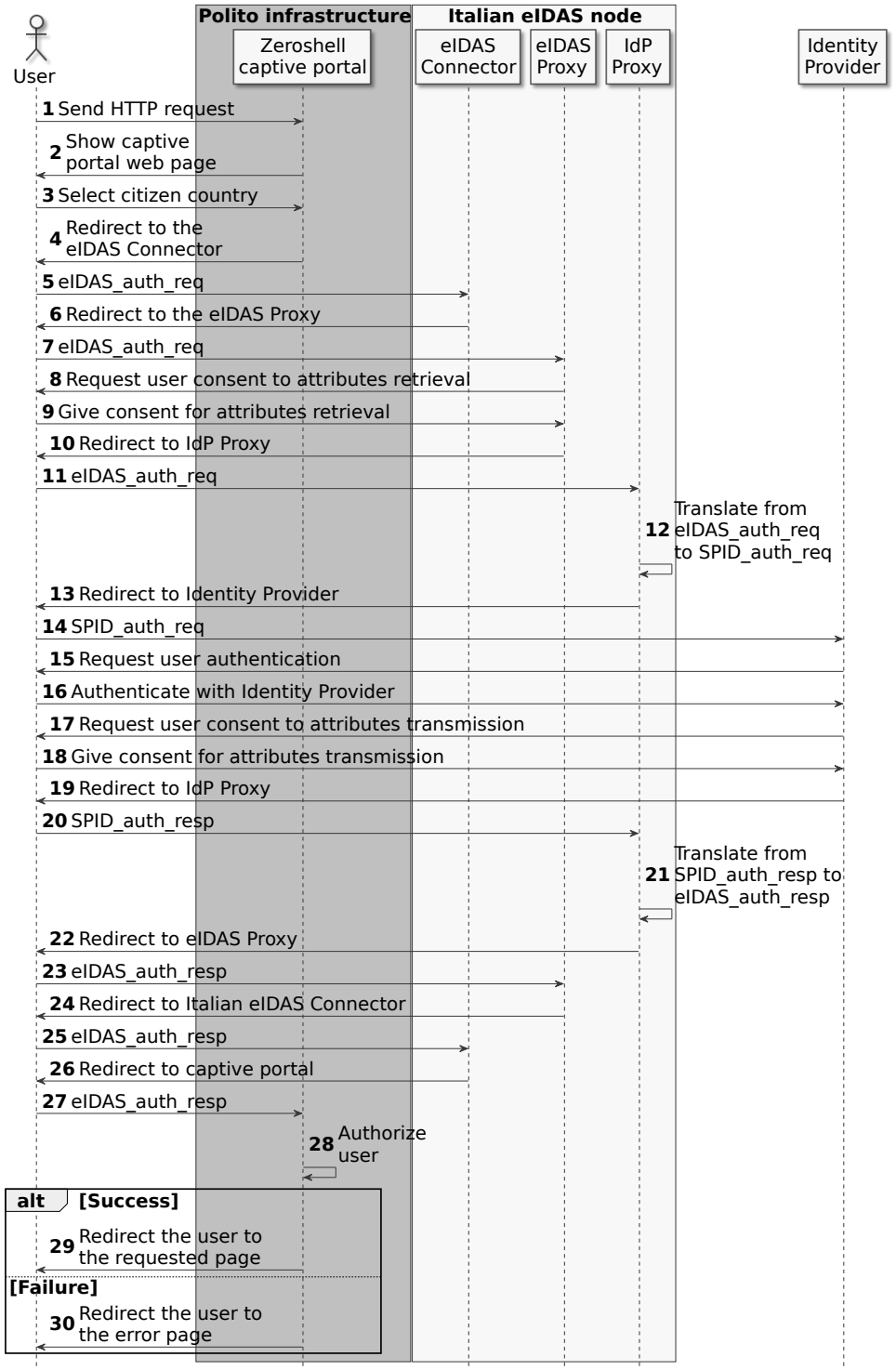

Figure 3: Authentication sequence diagram for Wi-Fi access for a citizen with an Italian SPID credential, by exploiting an eIDAS-enabled Zeroshell captive portal.

requested and he agrees to be valued (steps 8-9), otherwise the authentication workflow stops. This might be a critical point because some users might become suspicious, as they are not typically used to grant

personal data consent in such service. In Italy, the additional element named IdP Proxy acts as Specific Proxy in the eIDAS Node (in Fig. 1). Thus, the eIDAS Proxy sends the *eIDAS_auth_req* through the UA to the IdP Proxy (steps 10,11). Note that this *eIDAS_auth_req* is not the one received from the eIDAS Connector but it's generated based on the one received previously and it's digitally signed by using a dedicated asymmetric key corresponding to the certificate shared with the IdP Proxy in the national SAML metadata. Subsequently, the IdP Proxy translates the *eIDAS_auth_req* to *SPID_auth_req*, which may be processed by a SPID IdP and sends it through the UA to the IdP (steps 12-14). We observe that in this step another type of SAML metadata is used, namely the SPID metadata, which is shared between the IdP Proxy and the SPID IdP(s). This type of metadata contains typical SAML elements like the X.509 certificates for signing SAML messages, but additionally, it contains also other data specific to the SPID protocol, e.g., the attribute sets. These sets contain different identification attributes for the persons, allowing the implementation of the principle of data minimization, which means the IdP should provide the minimum set of attributes required so that it should not overshare as default [37].

When receiving an *SPID_auth_req*, the IdP processes it internally, e.g., it validates its signature by using the corresponding X.509 certificate in the SPID metadata of the IdP Proxy, and extracts the information from the request. Next, the user authenticates with his SPID credentials and the IdP requests user consent to transfer the valued personal attributes (steps 15-17). If the user agrees to transfer the data, the *SPID_auth_resp* is generated and is sent back through the UA to the IdP Proxy (steps 18-20), which translates it into an *eIDAS_auth_resp* and sends it back through the UA to the eIDAS Proxy (steps 21,22). Next, the eIDAS Proxy processes the response with the help of the SAML metadata shared with the IdP Proxy, then it generates a new *eIDAS_auth_resp* signed by the eIDAS Proxy, which is sent to the eIDAS Connector through the UA (steps 23,24). The eIDAS Connector receives the *eIDAS_auth_resp* message, validates it and it generates a new *eIDAS_auth_resp* which is sent to the Zeroshell captive portal (steps 25-27). Here the response is processed by exploiting the SAML metadata shared with the eIDAS Connector and the SAML-aware logic in Zeroshell (step 28). If the processing has completed correctly, the captive portal provides Internet access to the user, and the user browser is redirected to the web page requested in the first step (step 29).

6 Implementation details

6.1 Experimental testbed setup

We set up a dedicated machine for the eIDAS-enabled SP providing the Wi-Fi access service through a captive portal. On this machine, we installed Zeroshell in a Virtual Machine (VM) by using VirtualBox [38] and we modified the configuration of the Shibboleth SP (in Zeroshell) to generate an *eIDAS_auth_req* (instead of a plain SAML request).

The infrastructure used in our test environment is depicted in Figure 4. The Guest Network can be a LAN connected through a switch to the management network, or a WLAN connected to the management network through a wireless access point.

6.2 Zeroshell setup

First, we ran the Zeroshell 3.9.0 VM by using VirtualBox hypervisor on an Ubuntu machine. We provided the VM with 2048 MB of RAM and 10 GB of space dynamically allocated on a VDI hard drive. We used the Zeroshell ISO image downloaded from the official Zeroshell website at URL `https://zeroshell.org/download/`. Subsequently, we configured Zeroshell with two new interfaces, a Host-only Adapter for guest clients to connect to the internal network and a Bridged Adapter to connect to the Internet. We created the Host-only Adapter "vboxnet0" with network 192.168.56.0/24.

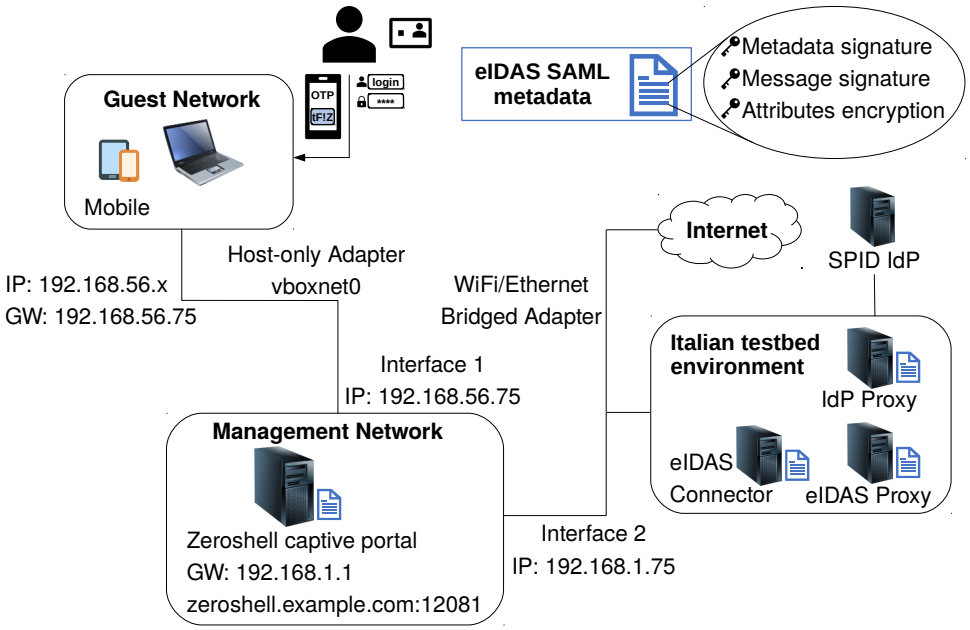

Figure 4: Zeroshell captive portal connected to the eIDAS Node in our experimental testbed environment. To obtain Wi-Fi access, the user exploits web browser on different devices (e.g. PC, smartphone), and various types of authentication credentials.

Afterwards, we accessed Zeroshell Command Line Interface (CLI), which is used to configure the IP address of the web interface and some other basic parameters. The Zeroshell interface is set by default to 192.168.0.0/24 network, more precisely with IP address 192.168.0.75. To access the network, either a static IP address can be set within the 192.168.0.0/24 network, or the IP address of Zeroshell can be changed by using the CLI. As we have two interfaces on Zeroshell, we decided to use the Host-only Adapter "vboxnet0" network to access the web interface, and we changed the IP address of the Zeroshell VM eth0 interface to 192.168.56.75 so that to access the web interface from the web browser at `https://192.168.56.75/`.

Once we were able to access the web interface, we created a profile to save our work and we configured the captive portal. The profile is created by using `Profiles` tab, which is accessible from the web interface at `Setups` > `Profiles`. By selecting the hard disk where the profile will be saved, the creation button is shown. After clicking that button Zeroshell opens a new window for the profile configuration. We selected our internal network interface in the Ethernet interface and set IP address to *192.168.56.75*. After the profile creation, we selected it and clicked the `Activate` button. This reboots Zeroshell and requires to login again by using the created credentials.

Thereafter, we created the network interfaces through the web interface, by selecting `Setups` > `Network`. We have created two interfaces, one for the internal network and another one for the Internet. On *ETH00* we have set our IP *192.168.56.75* on "vboxnet0" and on *ETH01* we have added an IP on the Ethernet or Wi-Fi network. In our case, we used *192.168.1.75* to get access to Internet. Moreover, we set the default gateway of *ETH01*, which in our case is *192.168.1.1*, and we also enabled DHCP on *ETH00* interface to provide dynamic IP address to the guest users. After that, we enabled the Zeroshell captive portal, which intercepts the requests from the guest clients on the gateway. In our case, the interface of the internal network is *ETH00*, so from the captive portal tab we selected the interface *ETH00* from the drop down menu and then checked the "GW" check-box to enable the captive portal.

6.3 Shibboleth configuration

We changed Shibboleth SP configuration to generate the *eIDAS_auth_req* used by the eIDAS Network.

Zeroshell provides a basic "Web File Editor" for Shibboleth configuration, which is accessible by selecting: `Captive Portal > Authentication > Shibboleth Authentication > Config`.

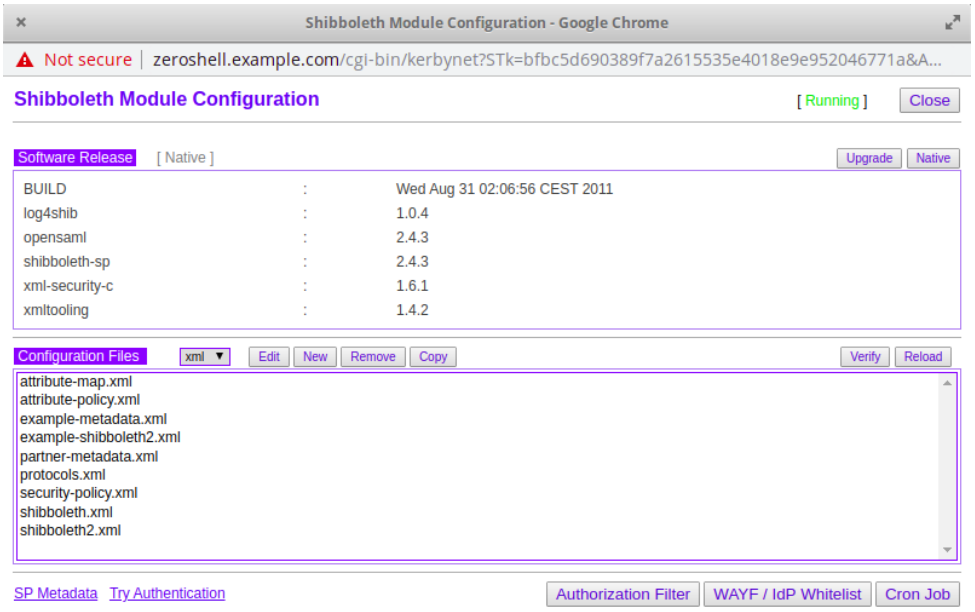

Figure 5: Configuration of Shibboleth files.

To make Zeroshell captive portal interoperable with the eIDAS Network, either it communicates with the Specific Connector (typically via a specific protocol), or it communicates directly with eIDAS Connector via the eIDAS protocol. We used the last option above. For this purpose, we set the `entityID` in the `Host` element to `zeroshell.example.com` for metadata.

```
<Host name="zeroshell.example.com" port="12081" scheme="https">
  <Path name="secure" authType="shibboleth" requireSession="true"/>
</Host>
```

Then, in the `ApplicationDefaults`, the `entityID` was set to `https://zeroshell.example.com:12081/shibboleth`. The `Remote_User` parameter, which is used to identify the user after authentication, was set to the following attributes: `FirstName, PersonIdentifier, and DateOfBirth`. We have also enabled the signing and encryption for the eIDAS messages handled.

```
<ApplicationDefaults entityID="https://zeroshell.example.com:12081/shibboleth"
  REMOTE_USER="FirstName PersonIdentifier DateOfBirth"
  signing="true" encryption="true">
```

The eIDAS protocol requires also an agreement between system entities regarding identifiers, binding support and endpoints, certificates and keys, and so forth, which is achieved through the SAML metadata. To communicate with the eIDAS Connector, the SP (i.e. Zeroshell) needs the eIDAS Connector's SAML metadata and its X.509 certificate for SAML metadata validation, which is used to verify the signature on the SAML metadata. Thus, we configured the eIDAS Connector's SAML metadata file and the X.509 SAML metadata certificate locally in the *MetadataProvider* element:

```
<MetadataProvider type="XML"  file="eIDASConnector-metadata.xml">
  <MetadataFilter type="Signature" certificate="eIDASConnectorsigner.pem" />
</MetadataProvider>
```

The keys and certificates used for signature and encryption of the eIDAS messages were added in the *CredentialResolver* element in the `shibboleth2.xml` file. It is worth mentioning that we assumed the certificates (for signing and encryption) to be valid, but various solutions exist nowadays (and should be adopted by Zeroshell or by the eIDAS Nodes) to validate the certificates via OCSP [39], as discussed in [40] [41], or through various alternative methods, e.g., [42].

We have defined two *CredentialResolver* elements, one used for decrypting the attributes in the received *eIDAS_auth_resp*, and the other one used for signing the *eIDAS_auth_req* sent to the eIDAS Connector.

```
<CredentialResolver type="File" key="zeroshell-saml-signature.key"
  certificate="zeroshell-saml-signature.pem" use="signing"/>
<CredentialResolver type="File" key="zeroshell-saml-encryption.key"
  certificate="zeroshell-saml-encryption.pem" use="encryption"/>
```

Finally, to generate an *eIDAS_auth_req*, which contains also the attributes of the eIDAS MDS, we are using `AuthnRequest` element, which is used as a template for the request issued, as shown below. This template can be used to supply advanced request content that cannot be configured in a simpler way.

```
<saml2p:AuthnRequest
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:eidas="http://eidas.europa.eu/saml-extensions"
  xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion" ForceAuthn="true"
  IsPassive="false" Consent="urn:oasis:names:tc:SAML:2.0:consent:unspecified"
  Destination="https://connector-test-eid4u.polito.it/EidasNode/ServiceProvider"
  ProviderName="zeroshell-SP"
  ID="foo" Version="2.0" IssueInstant="2012-01-01T00:00:00Z">
  <saml2:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
    https://zeroshell.example.com:12081/shibboleth</saml2:Issuer>
  <saml2p:Extensions>
    <eidas:SPType>public</eidas:SPType>
    <eidas:RequestedAttributes>
      <eidas:RequestedAttribute FriendlyName="FamilyName"
      Name="http://eidas.europa.eu/attributes/naturalperson/CurrentFamilyName"
      NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri" isRequired="true"/>
      <eidas:RequestedAttribute FriendlyName="FirstName"
      Name="http://eidas.europa.eu/attributes/naturalperson/CurrentGivenName"
      NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri" isRequired="true"/>
      <eidas:RequestedAttribute FriendlyName="DateOfBirth"
      Name="http://eidas.europa.eu/attributes/naturalperson/DateOfBirth"
      NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri" isRequired="true"/>
      <eidas:RequestedAttribute FriendlyName="PersonIdentifier"
      Name="http://eidas.europa.eu/attributes/naturalperson/PersonIdentifier"
      NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri" isRequired="true"/>
    </eidas:RequestedAttributes>
  </saml2p:Extensions>
  <saml2p:NameIDPolicy AllowCreate="true"
    Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified"/>
  <saml2p:RequestedAuthnContext Comparison="minimum">
    <saml2:AuthnContextClassRef>http://eidas.europa.eu/LoA/low
```

```
    </saml2:AuthnContextClassRef>
  </saml2p:RequestedAuthnContext>
</saml2p:AuthnRequest>
```

Whitelisting configuration. In general, since the user has not been authenticated yet, the captive portal denies access to Internet, but we need to configure it to allow access to eIDAS Nodes and IdPs, while all the other Internet traffic would not be allowed until the user has successfully authenticated. This problem is solved by adding the eIDAS Nodes and IdPs endpoints statically into the so-called "whitelist", which is shown in Figure 6. The whitelist is accessible at: `Captive Portal > Authentication > Shibboleth Authentication > Config > WAYF/IDP Whitelist`

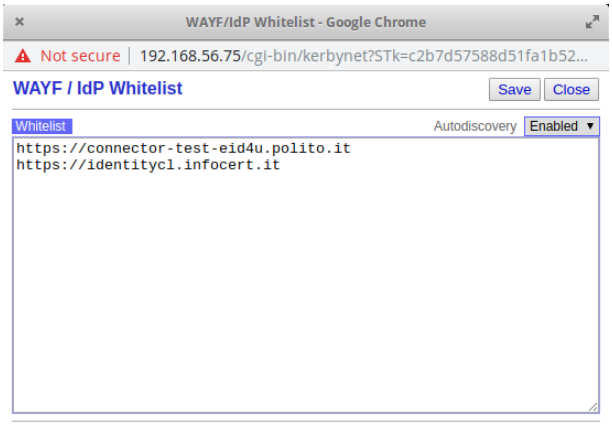

Figure 6: Whitelisting of the Italian eIDAS Connector (in our testbed) and the Infocert IdP in Zeroshell captive portal.

6.4 Step-by-step detailed authentication flow

The user starts the authentication by connecting to the internal network of Zeroshell, which dynamically assigns an IP address to the user via DHCP. When the user tries to access an HTTP URL, such as `www.abc.com`, Zeroshell detects that the user is not authorized and redirects him to the captive portal login page, e.g. at the URL `https://192.168.56.75:12081/cgi-bin/zscp` in our test environment. The captive portal login page is then shown to the user. To allow authentication via eIDAS Network, a dedicated button is shown to the user. When clicked, Zeroshell creates an *eIDAS_auth_req* and sends it to the eIDAS Connector at the URL `https://connector-test-eid4u.polito.it/EidasNode/ServiceProvider` via the user's browser. The following parameters are passed in the request:

```
country: IT
sendmethods: POST
postLocationUrl:
    https://connector-test-eid4u.polito.it/EidasNode/ServiceProvider
redirectLocationUrl:
    https://connector-test-eid4u.polito.it/EidasNode/ServiceProvider
RelayState: ss:mem:17419df78809c9d7b59631cf6edab9ad
SAMLRequest: <SAMLRequest>
```

The eIDAS Connector receives the *eIDAS_auth_req* and verifies its signature by using the SAML metadata (which has to be validated also with the X.509 SAML metadata signing certificate stored locally in the dedicated store of the eIDAS Connector) and the X.509 certificate for the verification of signature on messages found in the SAML metadata itself. Then, it creates the *eIDAS_auth_req* for the eIDAS Proxy selected based on the country chosen by the citizen. In this case, it creates an *eIDAS_auth_req* for the Italian eIDAS Proxy at URL `https://service-test-eid4u.polito.it/EidasNode/ColleagueRequest` and sends it through the user's browser. The following parameters are passed in the request.

```
RelayState: ss:mem:a1bf8bbf23f2a688b8e57e6372bbdc3f
token: 2sjjhlvPMkUkTZCM-mEGUIUAl5k$
SAMLRequest: <SAMLRequest>
```

The eIDAS Proxy receives the *eIDAS_auth_req* and verifies its signature by using the SAML metadata of the eIDAS Connector (downloaded on the fly) and the X.509 certificate for the signature on the messages found in the SAML metadata itself. Note that the signature of the SAML metadata has to be validated as well with the X.509 (SAML metadata) signing certificate of the eIDAS Connector, which is stored locally on eIDAS Proxy and has to be confronted with the one found in the SAML metadata file. After that, eIDAS Proxy creates the *eIDAS_auth_req* for the IdP Proxy, the component used in the Italian infrastructure to translate between eIDAS SAML messages and SPID SAML messages. The eIDAS Proxy sends the *eIDAS_auth_req* to the IdP Proxy at the URL `https://idp-proxy-test-eid4u.polito.it/idpproxy/idpeurequest` via user's browser. The following parameters are passed in the request:

```
messageFormat: eidas
SAMLRequest: <SAMLRequest>
```

The IdP Proxy receives the *eIDAS_auth_req* and verifies its signature by using the eIDAS Proxy's SAML metadata (downloaded on the fly) and the X.509 certificate for the signature on the messages found in the SAML metadata itself. Note that the verification of the eIDAS Proxy's SAML metadata is done as well as explained for the eIDAS Connector metadata. If the validation of the *eIDAS_auth_req* proceeds correctly, the IdP Proxy asks the user to select one Identity Provider (IdP) from a list as shown in Figure 7. In our tests, we used the *InfoCert* IdP. Subsequently, the IdP Proxy creates the *SPID_auth_req* for *InfoCert* IdP and sends it to the URL `https://identitycl.infocert.it/spid/samlsso` through the user's browser, along with the following parameters:

```
RelayState: SPID_REQUEST_RELAYSTATE
SAMLRequest: <SAMLRequest>
```

The *InfoCert* IdP receives the *SPID_auth_req* and processes it internally, including the checks on the SAML metadata of the IdP Proxy and the verification of the signature on the message itself. After that, it asks for user credentials by using HTML page at URL `https://identitycl.infocert.it/spid/basicauth.page`. The user provides his credentials and clicks on "Entra con SPID" button. After verifying user's credentials and receiving user consent for attributes transmission, the IdP creates the *SPID_auth_resp* for the IdP Proxy and sends it to the URL `https://idp-proxy-test-eid4u.polito.it/idpproxy/spidresponse` by using user's browser. The following parameters are passed in the response:

```
RelayState: SPID_REQUEST_RELAYSTATE
SAMLResponse:  <SAMLResponse>
```

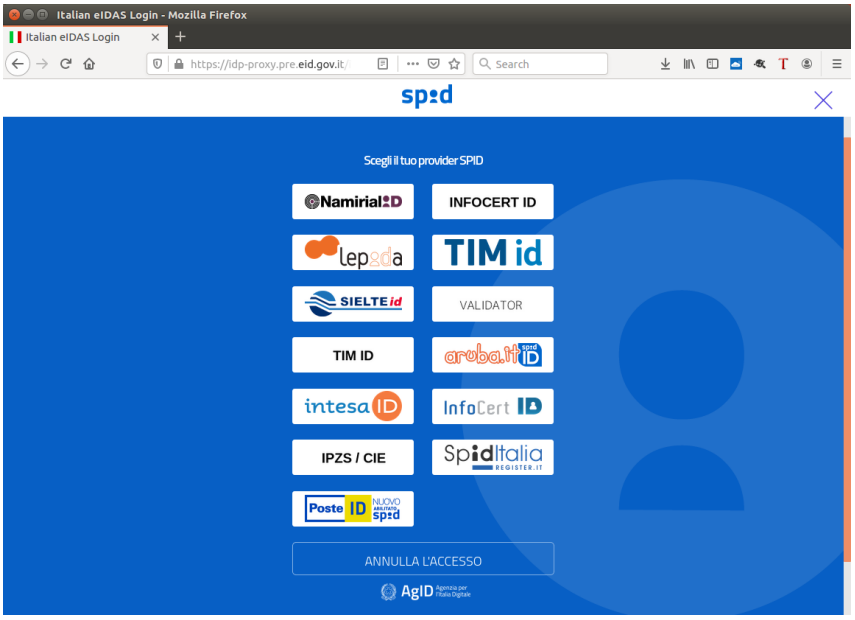

Figure 7: The list of IdPs shown on the IdP Proxy interface. All the IdPs support the notified eID scheme SPID, which must be recognized by the SPs across European countries.

The IdP Proxy receives the *SPID_auth_resp* and verifies its validity by using IdP's SAML metadata and its digital certificate used for signing SAML metadata. Note that the signature on the SAML metadata has to verified as well, as describe above. Next it gets the attributes from the *SPID_auth_resp*, it creates the *eIDAS_auth_resp* for the eIDAS Proxy and sends it at URL `https://service-test-eid4u.polito.it/EidasNode/IdpResponse` through the user's browser. The eIDAS Proxy receives the *eIDAS_auth_resp* and verifies it with IdP Proxy's SAML metadata, the IdP Proxy's certificate for signing found in the SAML metadata itself (to check the signature on the response) and its own X.509 certificate for encryption (to decrypt the attribute values). Note that, as explained above, the signature on the IdP Proxy's SAML metadata has to be verified as well. The attributes are encrypted by using a symmetric algorithm and a session key. The session key is encrypted with the public key of the eIDAS Proxy. The certificate containing the "encryption" public key and the accepted encryption algorithms are published in the eIDAS Proxy's SAML metadata.

The eIDAS Proxy decrypts the session key by using its "encryption" private-key and decrypts the attributes by using the session key. Then it creates the *eIDAS_auth_resp* for the eIDAS Connector and sends it to the URL `https://connector-test-eid4u.polito.it/EidasNode/ColleagueResponse` through the user's browser. The eIDAS Connector receives the *eIDAS_auth_resp* and verifies its validity by exploiting eIDAS Proxy's SAML metadata and the digital certificate of the eIDAS Proxy for SAML metadata signing, which is stored in a local dedicated keystore. Next, it decrypts the session key by using its "encryption" private-key and decrypts the attributes by using the session key. Then it creates the *eIDAS_auth_resp* for the captive portal SP and sends it at the URL `https://zeroshell.example.com:12081/Shibboleth.sso/SAML2/POST` via user's browser. The captive portal SP receives the *eIDAS_auth_resp* and verifies its validity by using eIDAS Connector's SAML metadata and the corresponding X.509 certificate for SAML metadata signing. Next it decrypts the session key by using its "encryption" private-key and decrypts the attributes by using the session key. Then it authenticates the user and redirects him to the URL `http://www.abc.com` selected in the first step. It also creates another window to manage the session of the user at `https://zeroshell.example.com:12081/cgi-bin/zscp`.

7 Evaluation results and discussion

We tested the proposed scenario in the illustrated testbed with a set of valid user credentials received from Infocert SPID IdP, i.e., username and password. In a real environment, involving official eIDAS Nodes, various different types of user credentials may be used, including OTPs and smartcards. The proposed solution has several advantages, but also some disadvantages to be tackled. One advantage is that the user neither has to be registered in advance on the SP side, nor it has to send his authentication credential to some intermediary point or server. The user already owns authentication credentials issued at national level, and he can exploit them in such eIDAS-aware services. Moreover, after being re-directed through the eIDAS Network, the user authenticates directly with his IdP, as this is one of the core idea(s) in the eIDAS Network. Furthermore, an SP may require (if necessary) the user to authenticate with a stronger level of authentication. In eIDAS, three Levels of Assurance (LoA) exist - i.e. Low, Substantial and High- and in some cases the SP might require the users to authenticate with the highest level credential, e.g. an eID card. This might prevent some security attacks which could occur in case weak authentication credentials are employed, such as username and (inadequate, e.g. simple or too short) passwords. In general, since Shibboleth is widely used to implement federated authentication across domains, our solution can be adapted in those environments that already run Shibboleth and want to connect to the eIDAS Network. An analysis of whether and how our solution may be adopted also in other environments, such as collaborative networks [43] and wireless sensor networks with limited resources [44] that might require AtA, is part of future work.

Privacy and usability issues. One of the "conceptual" disadvantages could be that the user might disagree to transfer personal data to get access to the service. It is not intuitive and, although a user consent is required and needs to be provided, many users might not understand the necessity to transfer such sensitive data to the SP and they might require additional measures and/or guarantees for data processing and storage on SP side. From the usability point of view, some users might find not convenient to use national eID cards for this types of services and might also disagree with the several intermediate steps to be followed through the eIDAS Nodes and Specific components to reach the IdP that might be time consuming as well because cryptographic operations and internal data processing is performed in each step.

Technical issues. Technically speaking, we faced several problems during integration of the Zeroshell captive portal with the eIDAS Network:

$\mathscr{I}$**1.** We detected incompatibilities between the encryption algorithms supported by the Zeroshell (which supports for example only AES 256-CBC [45]) and the security requirements of the eIDAS Node [46] that are stricter, e.g., the eIDAS Nodes are required to use AES 128-GCM, AES 192-GCM, AES 256-GCM [47]. To face this problem, in our testbed, we configured the eIDAS Node with AES 256-CBC support, but if adopted on wide scale, the eIDAS Node would not be downgraded, thus Zeroshell has to be updated with support for stronger cryptographic algorithms.

$\mathscr{I}$**2.** Whitelisting (auto-discovery). There is no automatic discovery of all the endpoints of the components to be reached by the user's web browser, i.e., the eIDAS Nodes, other specific elements in the eIDAS Node like the IdP Proxy, or the IdPs. Thus, such endpoints have to be found through an alternative manner, and then they need to be whitelisted on the captive portal, e.g. by configuring statically their DNS names. While the eIDAS Node's endpoints are typically public and known, the authentication endpoints might be hidden and/or limited to public exposure.

$\mathscr{I}$**3.** Citizen country selection. By default, Zeroshell does not allow to select a citizen country where the user will be authenticated. When enabled with eIDAS, it is required to modify the captive portal interface to allow the selection of the country in which the citizen will be authenticated.

$\mathscr{I}$**4.** Publishing the captive portal's SAML metadata. In the default installation, Zeroshell SP metadata is statically configured on the IdP. When enabled with eIDAS, the eIDAS Connector requires a

public URL from where the captive portal's SAML metadata has to be downloaded. However, since Zeroshell does not expose at a public URL its SAML metadata, this would require a modification of Zeroshell source code. To address this problem, we configured the captive portal's SAML metadata locally on the eIDAS Connector, but in a real scenario the support for publishing the captive portal's SAML metadata to a public URL must be added.

8 Conclusions

Many governments or dedicated IdPs across Europe provide eIDs to the citizens to use them for authentication in more and more services nowadays: to register children at public schools, to access health data or to pay some taxes. To allow mutual and legal cross-border recognition of eIDs, the eIDAS Regulation and its implementation (the eIDAS Network) are nowadays in place, and it's expected to be increasingly supported in more scenarios in Europe. Our work addressed a practical use case, that is how a citizen could exploit his eID for Wi-Fi network access in his country or abroad. When integrated with the eIDAS Network, such service requires AtA, that is the SP needs to process and enforce some authorization decisions before the citizen has been authenticated. We presented first a generic AtA model and the design choices. We implemented a prototype for the eIDAS-enabled Wi-Fi access, based on Zeroshell and eIDAS code, and we discussed several practical issues encountered, as well as the main problems to be addressed to adopt such a solution on wide scale. Our approach is extensible to other usage scenarios where access to sensible Web resources is required before full user authentication can take place.

Acknowledgement

This work was developed in the eID4U project, co-funded by the European Union's Connecting Europe Facility, under the grant agreement no. INEA/CEF/ICT/ A2017/1433625. We thank also Muhammad Ali Anjum (student at Politecnico di Torino) for his work in implementing the presented scenario.

References

[1] "Sistema Pubblico di Identità Digitale (SPID)," https://www.spid.gov.it/ [Online; accessed on June 15, 2020].

[2] "SPID Identity Providers notified under eIDAS," https://ec.europa.eu/cefdigital/wiki/display/EIDCOMMUNITY/Italy+-+SPID [Online; accessed on June 15, 2020].

[3] "Secure Identity Across Borders Linked (Stork) project - Towards pan-European recognition of electronic IDs (eIDs)," https://ec.europa.eu/digital-single-market/en/content/stork-take-your-e-identity-you-everywhere-eu [Online; accessed on June 15, 2020], 2008-2011.

[4] D. Berbecaru, A. Lioy, and C. Cameroni, "Providing digital identity and academic attributes through european eid infrastructures: Results achieved, limitations, and future steps," *Software: Practice and Experience*, vol. 49, no. 11, pp. 1643–1662, August 2019.

[5] D. Berbecaru, A. Lioy, M. Mezzalama, G. Santiano, E. Venuto, and M. Oreglia, "Federating e-identities across Europe, or how to build cross-border e-services," in *Proc. of AICA-2011: Smart Tech and Smart Innovation conference, Torino (Italy)*, November 2011.

[6] D. Berbecaru, A. Lioy, and M. D. Aime, "Exploiting proxy-based federated identity management in wireless roaming access," in *Proc. of the 8th International Conference on Trust, Privacy and Security in Digital Business (TrustBus'11), Toulouse, France*, ser. Lecture Notes in Computer Science, vol. 6863. Springer Berlin Heidelberg, August-September 2011, pp. 13–23.

[7] "Regulation (EU) No 910/2014 of the European Parliament and of the Council of 23 july 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing directive 1999/93/ec," August 2014.

[8] "eIDAS Technical Specifications v1.1," https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/eIDAS+eID+Profile [Online; accessed on June 15, 2020].

[9] "eIDAS-Node software releases," https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/eIDAS-Node+Integration+Package [Online; accessed on June 15, 2020].

[10] "How to integrate an eid scheme," https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/How+to+integrate+an+eID+scheme [Online; accessed on June 15, 2020].

[11] D. Berbecaru, A. Lioy, and C. Cameroni, "Authorize-then-Authenticate: Supporting Authorization Decisions Prior to Authentication in an Electronic Identity Infrastructure," in *Proc. of the 2019 International Symposium on Intelligent Distributed Computing (IDC'19), Petersburg, Russia*, vol. 868. Springer, Cham, October 2019, pp. 313–322.

[12] "Zeroshell," https://zeroshell.org/ [Online; accessed on June 15, 2020].

[13] "Shibboleth Consortium," https://www.shibboleth.net/ [Online; accessed on June 15, 2020].

[14] M. Linden and V. Viitanen, "Roaming Network Access Using Shibboleth," in *TERENA Networking Conference*, 2004, https://www.terena.org/publications/tnc2004-proceedings/papers/linden.pdf [Online; accessed on June 15, 2020].

[15] "Eduroam, a RADIUS based authentication system for network access," https://www.eduroam.org [Online; accessed on June 15, 2020].

[16] C. Rigney, S. Willens, A. Rubens, and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)," https://tools.ietf.org/html/rfc2865 [Online; accessed on June 15, 2020], 2000.

[17] K. Wierenga, S. Winter, and T. Wolniewicz, "The Eduroam Architecture for Network Roaming," https://tools.ietf.org/html/rfc7593 [Online; accessed on June 15, 2020], September 2015.

[18] S. Brenza, A. Pawlowski, and C. Pöpper, "A Practical Investigation of Identity Theft Vulnerabilities in Eduroam," in *Proc. of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks (WiSec'15), New York, New York, USA*. ACM, June 2015, pp. 1–11.

[19] S. Winter, M. McCauley, S. Venaas, and K. Wierenga, "Transport layer security (TLS) encryption for RADIUS," https://tools.ietf.org/html/rfc6614 [Online; accessed on June 15, 2020], May 2012.

[20] H. Liu and G. H., "Certificate-Based, Disruption-Tolerant Authentication System with Automatic CA Certificate Distribution for Eduroam," in *Proc. of the 38th IEEE International Computer Software and Applications Conference Workshops, Vasteras, Sweden*. IEEE, July 2014, pp. 169–173.

[21] "Govroam," https://govroam.nl/english/ [Online; accessed on June 15, 2020].

[22] "Govroam NL Service Policy," https://govroam.nl/wp-content/uploads/2015/02/govroam-NL-service-policy-jan2015.pdf [Online; accessed on June 15, 2020].

[23] "PacketFence Overview," https://packetfence.org/about.html [Online; accessed on June 15, 2020].

[24] "NoDogSplash Overview," https://nodogsplashdocs.readthedocs.io/en/stable/overview.html [Online; accessed on June 15, 2020].

[25] S. C. et al., "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0 - Errata Composite. Working Draft 07," http://www.oasis-open.org/committees/download.php/56776/sstc-saml-core-errata-2.0-wd-07.pdf [Online; accessed on June 15, 2020], 2015.

[26] "eIDAS SAML Message Format, version 1.1," https://ec.europa.eu/cefdigital/wiki/download/attachments/82773108/eIDAS%20Message%20Format_v1.1-2.pdf.

[27] J. H. et al., "Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS Standard," http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf [Online; accessed on June 15, 2020], March 2005.

[28] D. Berbecaru, A. Lioy, and C. Cameroni, "Electronic Identification for Universities: Building Cross-Border Services Based on the eIDAS Infrastructure," *Information*, vol. 10, no. 6, pp. 210:1–210:17, June 2019.

[29] D. Berbecaru and A. Lioy, "On integration of academic attributes in the eIDAS infrastructure to support cross-border services," in *Proc. of the 22nd International Conference on System Theory, Control and Computing (ICSTCC'18), Sinaia, Romania*, October 2018, pp. 691–696.

[30] "eIDAS-Node version 1.4.4," https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/eIDAS-Node+version+1.4.4 [Online; accessed on June 15, 2020].

[31] E. Rescorla, "HTTP Over TLS. RFC 2818," https://tools.ietf.org/html/rfc2818 [Online; accessed on June 15, 2020], May 2000.

[32] R. Khare and S. Lawrence, "Upgrading to TLS Within HTTP/1.1. rfc 2817," https://tools.ietf.org/html/rfc2817 [Online; accessed on June 15, 2020], May 2000.

[33] E. Bidelman, "Google Data APIs team: Using OAuth with the Google Data APIs," https://developers.google.com/gdata/articles/oauth, September 2008.

[34] B. Bruegger, D. Hühnlein, and J. Schwenk, "TLS-federation–a secure and relying-party-friendly approach for federated identity management," in *Proc. of the 7th International Conference of the Biometrics Special Interest Group on Biometrics and Electronic Signatures (BIOSIG'08), Darmstadt, Germany*, September 2008, pp. 93–104.

[35] "Web Services Secure Conversation Language 1.4, OASIS Standard," http://docs.oasis-open.org/ws-sx/ws-secureconversation/v1.4/ws-secureconversation.html [Online; accessed on June 15, 2020], 2009.

[36] "eID4U project - eID for University (2018-2019)," https://ec.europa.eu/inea/en/connecting-europe-facility/cef-telecom/2017-eu-ia-0051 [Online; accessed on June 15, 2020].

[37] NSTIC (National Strategy for Trusted Identities in Cyberspace), "Pilot common considerations 4: Attributes," https://www.nist.gov/blogs/cybersecurity-insights/nstic-pilot-common-considerations-4-attributes [Online; accessed on June 15, 2020], 2013.

[38] "VirtualBox," https://www.virtualbox.org/ [Online; accessed on June 15, 2020].

[39] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP," https://tools.ietf.org/html/rfc2560 [Online; accessed on June 15, 2020], 1999.

[40] D. Berbecaru, M. M. Casalino, and A. Lioy, "FcgiOCSP: A scalable OCSP-based certificate validation system exploiting the FastCGI interface," *Software: Practice and Experience*, vol. 43, no. 12, pp. 1489–1518, August 2012.

[41] T. Chung, J. Lok, B. Chandrasekaran, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, J. Rula, N. Sullivan, and C. Wilson, "Is the Web Ready for OCSP Must-Staple?" in *Proc. of the 2018 Internet Measurement Conference (IMC'18), Boston, Massachusetts, USA*. ACM, October 2018, pp. 105–118.

[42] M. Naor and K. Nissim, "Certificate revocation and certificate update," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 4, pp. 561–570, April 2000.

[43] D. G. Reina, V. Sharma, I. You, and R. Kumar, "Energy Efficient Data Dissemination in Multi-UAV Coordinated Wireless Sensor Networks," *Mobile Information Systems*, vol. 2016, pp. 8 475 820:1–8 475 820:13, June 2016.

[44] R. Sun, J. Yuan, I. You, X. Shan, and Y. Ren, "Energy-aware weighted graph based dynamic topology control algorithm," *Simulation Modelling Practice and Theory*, vol. 19, no. 8, pp. 1773 – 1781, September 2011.

[45] National Institute of Standards and Technology, "Recommendation for Block Cipher Modes of Operation: Modes and techniques," https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf [Online; accessed on June 15, 2020], December 2001, nIST Special Publication 800-38A.

[46] "eIDAS Crypto Requirements, version 1.1," https://ec.europa.eu/cefdigital/wiki/download/attachments/82773108/eidas_-_crypto_requirements_for_the_eidas_interoperability_framework_v1.0.pdf [Online; accessed on June 15, 2020].

[47] National Institute of Standards and Technology, "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC," https://www.nist.gov/publications/recommendation-block-cipher-modes-operation-galoiscounter-mode-gcm-and-gmac [Online; accessed on June 15, 2020], November 2007, nIST Special Publication 800-38D.

Author Biography

Diana Berbecaru is Senior Research Assistant in the Department of Control and Computer Engineering at Politecnico di Torino. She holds a Ph.D. in Computer Science from Politecnico di Torino and a M.Sc. from University of Craiova, Romania. She is a member of TORSEC Cybersecurity Research Group. She is the author of more than 30 refereed publications and participated in several projects on developing new security solutions. Her research interests include digital certificates, network forensics, multicast authentication, and electronic identification.

Antonio Lioy received the M.Sc. degree (summa cum laude) in electronic engineering and the Ph.D. degree in computer engineering from the Politecnico di Torino. He is currently a Full Professor with the Politecnico di Torino, where he leads the TORSEC Cybersecurity Research Group. His research interests include network security, policy-based system protection, trusted computing, and electronic identity.

Cesare Cameroni received the M.Sc. degree in computer engineering in 2010 from Politecnico di Torino. From 2011 to 2014 and from 2018 to date he has been a Research Assistant with Department of Control and Computer Engineering at Politecnico di Torino. He is a member of the TORSEC Cybersecurity Research Group and his research interest includes security and dependability of ICT systems, electronic identities and federated identity infrastructures.