An efficient user-centric consent management design for multiservices platforms

Mikaël Ates, Maryline Laurent, Paul Marillonnet, Nesrine Kaaniche

Hindawi
Security and Communication Networks
Volume 2021, Article ID 5512075, 19 pages
https://doi.org/10.1155/2021/5512075

Research Article

An Efficient User-Centric Consent Management Design for Multiservices Platforms

Paul Marillonnet,[1,2] **Mikaël Ates**,[1] **Maryline Laurent**,[2] **and Nesrine Kaaniche**[2]

[1]*Entr'ouvert, Paris 75014, France*
[2]*SAMOVAR, Télécom SudParis, Institut Polytechnique de Paris, Évry 91000, Paris, France*

Correspondence should be addressed to Paul Marillonnet; pmarillonnet@entrouvert.com

Received 6 February 2021; Revised 22 April 2021; Accepted 10 June 2021; Published 22 June 2021

Academic Editor: Ahmad Samer Wazan

This paper presents an efficient user-centric consent management system to access online services of the Territorial Collectivities and Public Administration (TCPA) as well as user-authorized third parties. It defines a novel PII manager that supports a set of sources obeying to different authorization and PII retrieval protocols. This contribution is motivated by the necessity to interface TCPA services with remote sources that provide Personally Identifiable Information (PII). Hence, the originality of our solution is multifold. First, the burden for enforcing the interoperability between the sources and the TCPA services collecting the PII is reduced from the point of view of the user, the administrator of the User-Relationship Management (URM) platform, and the territorial agent responsible for processing the user's queries. Second, it defines a unified consent model supporting four types of sources. Third, it goes into details of practical implementations. Fourth, the relevance of the proposed PII manager for a relevant TCPA use case is demonstrated through a functional analysis.

1. Introduction

Our contribution pertains to the field of access control of Territorial Collectivities and Public Administration (TCPA) online services to their citizens. These TCPA are local and national official entities providing online services to citizens. Users of TCPA are requested to submit some regulated administrative requests, e.g., official document renewal, various allowance requests, and registrations to local services.

To benefit from these services, the user must provide Personally Identifiable Information (PII). As an example, the user issuing a passport renewal request is asked to fill in a web form including his PII fields, uploading scanned documents and retrieving PII from third-party sources.

Scanned documents and third-party-issued PII enable the user to provide the TCPA with validated data as valuable input for processing their requests. For instance, the France Connect official identity service provides user identity information, complying with the OIDC [1] identification protocol. Implicit grant OAuth 2.0 [2] authorization is also at experimental stage for tax and children allowance information.

User-centric architectures that provide consent management aim at allowing the user to have the governance of their PII through their lifecycle, that is, the consent to collection, the usage control over time, and the visibility of past collections. These capabilities should happen even when the PII has been provided by third-party sources.

However, many shortcomings of user-centric PII management within TCPA have been identified over the previous years in Chapter 1.2.2 in [3]. The problem of the reunification of personal data has been well identified in the literature in Chapter IV in [4] and remains relevant. In fact, academic and industrial solutions, either they be (i) personal data stores [5–8], (ii) identity managers [9–13], (iii) anonymous certificate systems [3, 14, 15], or (iv) delegation architecture [16, 17], do not address the specific needs of PII management within TCPA.

Indeed, (i) personal data stores only support simple PII management scenarios where PII is collected on an "all-or-nothing" basis.

Similarly, (ii) identity managers by nature have limited support of third-party PII sources nor do they provide thorough delegation capabilities.

Furthermore, (iii) anonymous certificate systems, although a powerful solution for PII certification in a privacy-compliant manner, do not appear entirely suitable for PII management within TCPA, as too many core features of such management are out of scope of this category of solutions.

Finally, (iv) the same scope issue applies to delegation architectures when it comes to PII validation and to the support of third-party PII sources.

More precisely, none of these solutions addresses the four critical functional requirements identified for a standard TCPA's use case, namely, (a) the need to manage the various consent information given by the user over time, (b) a wide extent of delegation on behalf of the user, (c) the possibility to validate PII, and (d) the support of remote PII sources.

In the European Union, functional requirements regarding user data management in TCPA have recently changed as part of the international regulations. As a matter of fact, the General Data Protection Regulation (GDPR) [18], applied by each member of the European Union since May 2018, requires that data controllers adopt new strategies regarding PII management of their users. Those strategies must address the need to give online users a thorough governance of their PII whether these PII be, for instance, service-provider transactional data or user profile data (additionally, some TCPA still partly rely on scanned documents). The necessity to deal with remote sources providing user PII led to propositions regarding identity-matching issues [19].

However, existing solutions did not discuss three other concerns that arise in the specific context of the TCPA.

First, user consent must be enforced consistently regardless of the PII's actual location on any remote source. This paper aims at providing a way for the user to define consent to offline PII processing, wherever the PII. The fact that the PII is provided by remote sources does not hinder the consent management capabilities of the contribution.

Second, the interoperability concerns, that arise when dealing with such an heterogeneous system involving different sources, must be addressed.

Finally, the level of trust granted to remote sources for their role in providing user's PII must be formalized, with the possibility for a PII provided by an untrusted source to be relevant for our TCPA use case, but less relevant than a PII originating from a fully-trusted source. This paper aims at specifying the levels of trust granted to sources and their impact on the TCPA use case.

This article describes a case-study architecture for TCPA services with the primary concern of enforcing users' informational governance. The main proposition of this paper is a PII manager which supports the TCPA requirements with regard to PII management.

This paper is structured as follows:

Section 2 defines the use case of our contribution. Section 3 defines the system model, i.e., the actors, the functional requirements, as well as the environment and technical hypotheses of our contribution. Section 4 describes the related works, i.e., the academic or industrial solutions that are closely related to the use case. Section 5 introduces the PII manager within an existing architecture. Sections 6–8, respectively, present the PII Query Interface, the source backend, and the PII Management User Interface. Section 9 provides a functional analysis of the PII manager, proving its adequacy to the initial use case. Eventually, Section 10 presents a software proof of concept for our contribution and also provides implementation guidelines, before concluding in Section 11.

2. Use Case

This section proposes a use case for motivating our contribution and functional requirements. Our use case takes place in the Territorial Collectivities and Public Administration (TCPA) with which citizens interact. A number of PII are transferred through the TCPA and require a clear management.

In that context, our precise use case is about a user owning a proof of postal address, which is a PII, and delivering the proof to the TCPA for a procedure. In our contribution, a PII manager is responsible for managing the user's data on his behalf when completing online procedures with their TCPA. Indeed, whenever needed for a procedure, e.g., the user's child school registration or the user's identity documents renewal, this data is directly made available by this service, only in case the user has previously agreed so by an explicit consent.

Their proof of address and other pieces of PII are managed for that service. The sources for this PII may be multiple, but the management features offered by this service remain unchanged.

This PII is used by various TCPA services possibly belonging to various collectivities at different scales such as town, department, region, and country. The user can visualize the past collections of their PII by the TCPA services and can revoke any further collection at any time.

3. System Model

This section defines the system model for our contribution. First, the actors of the use case are defined, along with their roles in the environment. Second, environment hypotheses are defined. Third, functional requirements, which our contribution must enforce, are also detailed. Last, the technical hypotheses of our industrial environment are listed.

3.1. Actors. The use case involves the four following actors:

(i) The user of the online TCPA services submits one or several requests to the administrative or territorial

services. The requests are tracked through the user's account on the platform.

(ii) The PII manager is responsible for enforcing the user consent and for evaluating the trust level of remote sources.

(iii) The TCPA User-Relationship Management (URM) platform acts as a service provider for the user and relies on the PII manager service.

(iv) The data sources may be official, i.e., maintained or acknowledged as such by TCPA, or private, i.e., maintained by a third-party service provider.

3.2. Environment Hypotheses. The PII managers are assumed to be dynamically discovered. This is made possible thanks to the user selecting the PII manager among a list for the TCPA platform to interconnect with.

The PII managers are also assumed to be regulated. This enables the TCPA URM platforms to establish direct trust with the PII managers, the latter having the critical duties to trustfully select PII sources and validate the retrieved PII.

Regulation can be enforced in two different hypothetical ways. First, there might be a regulation for a passlist of PII operators hosting several PII managers. The users would then be asked freely to choose the operator of their PII manager. Second, a PII manager authority, trusted by the TCPA, might organize PII managers based on a hierarchical certification architecture (i.e., a public-key infrastructure).

Finally, there might be an interest for the users to rely on several PII managers instead of only one. The direct advantage of distributing the responsibility for managing the PIIs over several entities would be higher availability of the service and distributed knowledge about their PII.

3.3. Functional Requirements. A noncomprehensive list of functional requirements relevant to PII management includes (i) user governance requirements, such as consent management, privacy/usability tradeoff, extent of delegation, and PII sharing capabilities and (ii) data exchange flow requirements, e.g., the supported PII types, the possibility to validate PII, the reusability of previously uploaded PII, and the support of remote PII sources.

As a result, the following requirements should be fulfilled by the PII manager.

(1) Usage definition: the user can define the purposes justifying the PII collection.

(2) Consent management: the PII manager keeps track of the authorizations given by the user for each piece of PII.

(3) Usage monitoring: the user is given clear metrics of the PII consumption by any TCPA service. This monitoring facility offers a view of the user's PII usage by the TCPA services.

(4) Delegation capabilities: the PII manager is able to decide whether or not to grant access to the PII, even when the user is not connected to the platform. In that way, the access granting process is asynchronous from the authorizations granted by the user.

(5) PII location abstraction: the PII manager ensures PII management regardless of the original source of the PII.

(6) Protocol standardization: through standard interfaces, the PII manager can be queried with a common interface relying on standard PII management protocols.

(7) Access uniformization: the multiple PII data sources are accessible in the same way.

(8) Authorization protocol interoperability: the main identity management protocols, access mechanisms and authorization schemes, are supported with the heterogeneous remote sources to achieve interoperability.

3.4. Technical Hypotheses. The four following types of sources are considered:

(1) Plain OAuth 2.0 resource servers based on OAuth 2.0 providers or OIDC providers

(2) SAML 2.0 identity providers [20]

(3) Plain read-only REST [21] sources accessible after an HTTP basic authentication [22]

(4) Sources acting as resource servers according to the Kerberos [23] protocol

These sources have been chosen for their actual use in production environments. Additionally, they have not been designed to be interoperable, challenging the PII location abstraction requirement identified earlier in Section 3.3.

Interoperability concerns are addressed at the PII exchange protocol layer. The PII formats used in those protocols are (mainly) JSON for OAuth, OIDC, and REST and XML for SAML. We take the hypothesis that such nomenclatures are used for every given type of PII by all the sources. These nomenclatures are already in use in the TCPA environments. For instance, date and datetime information rely on ISO 8601 [24] and its use on the Internet, as covered by the RFC 3339 [25]. Similarly, the standardization of phone numbers as URIs is covered in RFC 3966 [26].

Additionally, this paper assumes that the different acting entities' clocks are loosely synchronized, which is common and considered easy to achieve.

Additionally, it assumes that the URM platforms do not permit the same identifier to be assigned to several users time after time. This is a loose requirement as most of the identifiers are either reversible or irreversible high-entropy pseudonyms, where reversible pseudonyms rely on symmetrical cryptographic functions, whereas irreversible pseudonyms rely on hash functions. In both cases, provided that the user PII being used as input to the pseudonym function varies, the risk of collision is considered negligible.

4. Related Work

This section presents the related work published in the literature. The literature provides several industrial or academic solutions, implemented or only provided with implementation guidelines. Table 1 gives an overview of existing solutions and provides a comprehensive comparison between them with respect to their support of various functional requirements and other identified technical considerations.

INDIGO [17] provides a token translation system that supports interoperability among sources supporting different protocols.

Its coverage of the functional requirements as well as the technical considerations is thorough; however, it does not give information about usage definition capabilities.

User-Managed Access (UMA) [16], specified by the Kantara Initiative consortium, provides the delegation capabilities of interest and benefits from protocol interoperability as specified in the OAuth 2.0 authorization protocol; for instance, see the OAuth 2.0 assertion framework [2].

The UMA use case does not cover authorization protocol interoperability nor does it provide an abstraction of the PII location (however, compatible with OAuth 2.0 token exchange [27]). PII usage definition is not strictly covered by this solution, as it depends on the actual implementation of an UMA architecture.

Additionally, the Kantara Initiative provides a consent receipt model [27] for generic applications where user consents need tracking. For specific applications where this model needs to be reduced, the Kantara Initiative also provides a Minimal Viable Consent Receipt (MVCR) [28].

Databox [7] supports PII sources and their respective drivers that provide an interesting architecture for our use case.

It does not support information about whether usage definition and protocol standardization are supported. Authorization protocol interoperability depends on the presence of drivers for these protocols.

The Fargo [8] document storage service has a rather limited functional coverage.

In spite of supporting interoperability, through a direct authenticated API or through OAuth 2.0, it shows many shortcomings either in the identified functional requirements or the technical considerations, such as the unability to handle raw PII or the absence of delegation capabilities. It is therefore not suited for our use case.

5. Our PII Manager as Part of the TCPA Architecture

5.1. Overview. This section details the way the PII manager acts within our architecture. The PII manager enforces the use case of Section 2, while maintaining the functional requirements of Section 3.3. The components of the PII manager are later described in subsequent sections, i.e., Sections 6–8.

5.2. Presentation of the Solution. The overall architecture is depicted in Figure 1. Figure 1(a) illustrates the global architecture involving our PII manager interfacing to the URM system and the remote sources.

Figure 1(b) depicts the interaction between the PII manager and the URM platform(s). This figure illustrates the need for a user identifier mapping service, presented in Section 7.3.2, as the user already has its own local identifier. For organizational reasons, the URM platforms maintain their own local identity manager. Indeed, each URM platform is maintained by a TCPA. These local identity managers act as local authorization servers within definite sectors. The term of "sector border" in this figure denotes the logical separation of identifiers between the URM platforms.

Figure 1(c) shows the interactions between the PII manager and the sources. The drivers, as part of the PII manager's source backend presented in Section 7, make it possible to interface with several remote sources. This figure illustrates the use of a third-party authorization server (AS), as part of the OAuth authorization process for OAuth-based sources, labelled (a) on the figure, for getting the adequate access token for a given resource. (i) for instance, the France Connect data providers obeying to the OAuth 2.0 implicit authorization grant in Chapter 4.2 in [2]. Generic REST sources, labelled (b) on the figure, simply rely on base HTTP authentication mechanisms directly performed by the source, see for instance [22, 29]. (ii)for instance, the Particulier API (https://particulier.api.gouv.fr/ (resource in French)). Alternatively, sources acting as Kerberos management resource servers, labelled (c) on the figure, refer to permission tickets that are granted in a two-step authorization procedure requiring first to get a ticket-granting ticket (TGT) from the key distribution center (KDC) and second to get a permission ticket from the ticket-granting server (TGS). (iii) it happens in the collectivities information system where the Kerberos protocol is used to access to network resources.

Eventually, Figure 1(d) depicts the user-centric PII management zone, through which the user manages their PII, the authorized sources, and their consent to URM platforms. It corresponds to the direct interactions between the user and the PII Management User Interface of our contribution, presented in Section 8.

To better explain the PII collection process, we also present the interactions between entities along with four sequence diagrams as follows:

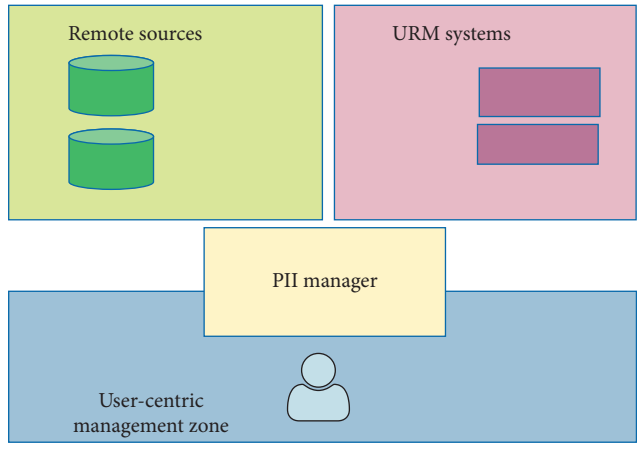
(i) A simple sequence diagram, depicted in Figure 2, describes the PII manager's discovery by the TCPA URM platforms.

(ii) Figure 3 describes the user authentication and consent obtention on the PII manager. Unauthorized PII access requests result in the obtention of a permission ticket. After user authentication and consent obtention, this ticket enables the issuance of an access token with the adequate authorization scopes on the requested resource(s).

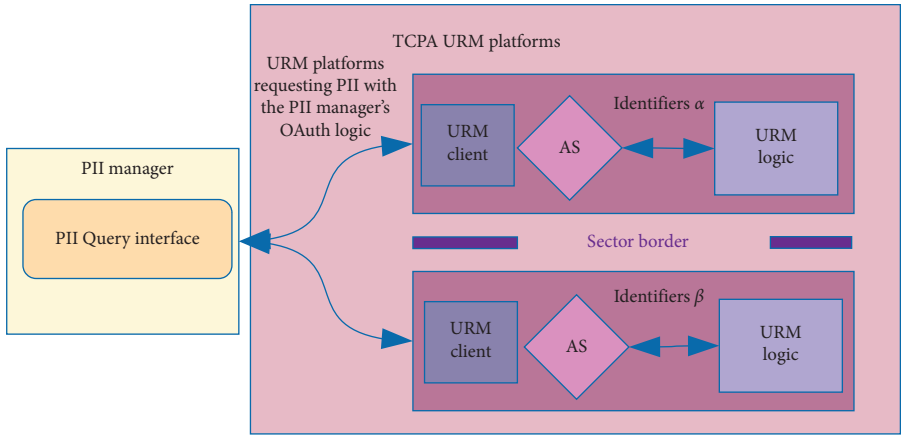
Table 1: Related personal data management solutions comparison.

Criterion		Solution				
		Related work				This contribution
		INDIGO architecture [17]	UMA [16]	Databox architecture [7]	Fargo [8]	PII manager
Functional requirements	Usage definition	?	●	?	✗	●
	Consent monitoring	✓	✓	✓	✗	✓
	Usage monitoring	✓	✓	✓	✗	✓
	Delegation capabilities	✓	✓	✓	✗	✓
	PII location abstraction	✓	✗	✓	✗	✓
	Protocol standardization	✓	✓	?	✓	✓
	Access uniformization	✓	✓	✓	✗	✓
	Authorization protocol interoperability	✓	✗	●	✓	✓
Technical considerations	Identified consent model	?	✓	?	✗	✓
	Available implementations	✓	✓	✓	✓	✗
	Open specifications	✓	✓	✓	✓	✓

✓: yes, ✗: no, ●: depends on implementation, and ?: no information available.



(a)



(b)

Figure 1: Continued.

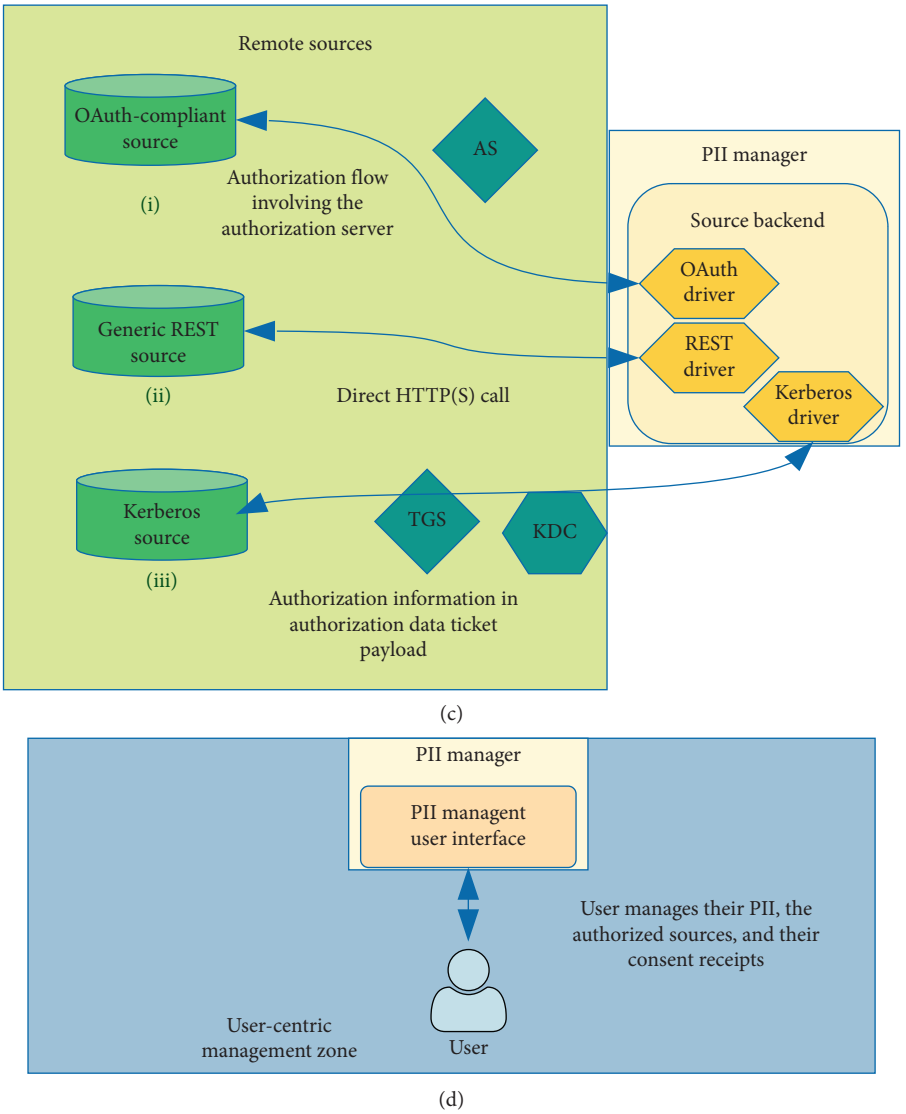


Figure 1: Complementarity in the PII manager's roles regarding our use case entities. (a) General overview. (b) At URM platform side. (c) At remote sources side. (d) In the user-centric management zone.

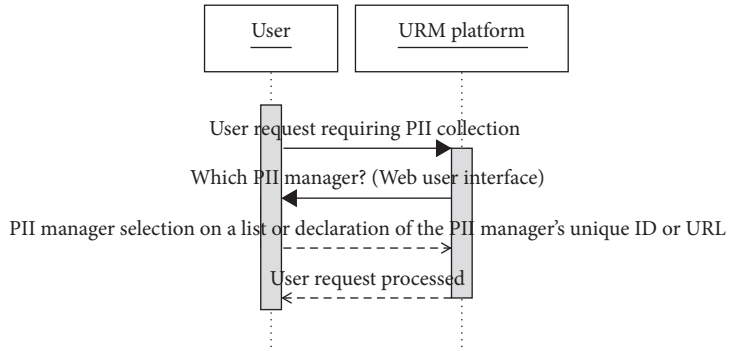


Figure 2: Discovery of the PII manager.

(iii) A sequence diagram for a typical PII collection scenario is provided in Figure 4. It shows the TCPA URM platform interacting directly with our PII manager, regardless of the data sources' location. In a two-step process, PII are collected by the TCPA URM platform. First, a request is sent by the TCPA URM platform to our PII manager through the PII retrieval endpoint. Second, the source backend at

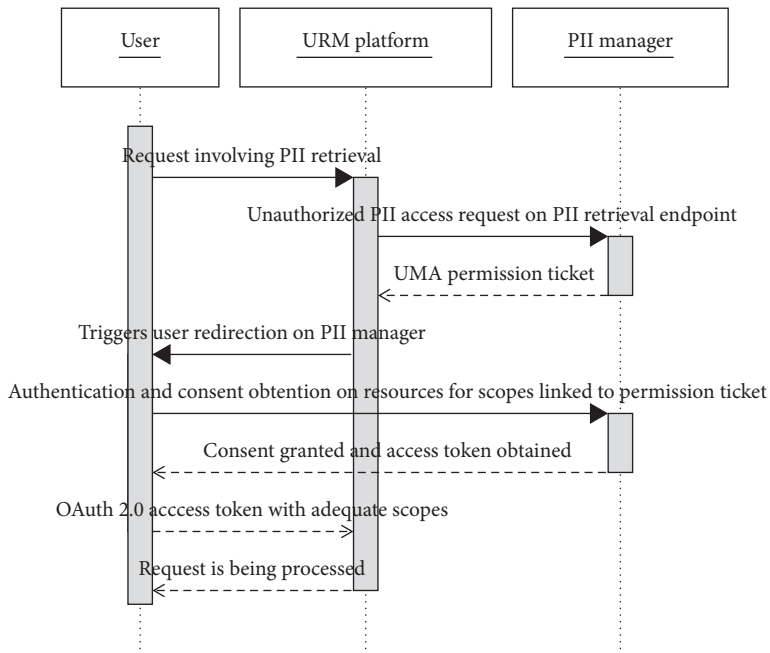


Figure 3: User authentication and consent obtention on the PII manager.

the PII manager selects the adequate driver for collecting the needed PII from the appropriate remote source. The authorization, which is a part of the second step, is either synchronous or asynchronous, unbeknownst to the requesting TCPA URM platform.

(iv) Figure 5 describes the PII collection process once the source-side user authorization has been obtained. The PII manager stores his authorization information granted by the user. As long as this authorization information still has a valid time-to-live value and the user has not revoked the authorization, user interaction is no longer required.

The three different components of our PII manager, i.e., the PII Query Interface (PQI), the source backend (SB), and the PII Management User Interface (PMUI), and their mutual articulation are discussed in Sections 6–8.

First, Section 6 presents the PQI as a means to support PII location abstraction and consent management. Section 6 describes a set of endpoints, respectively, for registration, retrieval, and introspection of PII, the standard usage of this interface, and the consent management at this interface level.

Second, Section 7 introduces the SB which serves to enforce the authorization protocol interoperability. This section tackles the consent management, and it describes the support of OAuth token exchange.

Third, Section 8 describes the PMUI for supporting consent management and usage definition. User-definable parameters as part of this interface are also discussed in Section 8.2.

6. PII Query Interface (PQI)

6.1. Overview. The PII Query Interface is used by the TCPA URM services to retrieve the user's PII on the PII manager. This section presenting the PQI is organized as follows: first, an overview of the PQI endpoints is given. Second, the registration endpoint and its ability to handle different types of registration information are discussed. Third, the usage of the PQI as part of our PII manager in the federated-identity environment of our use case is further described. Fourth, the role of the PQI in enforcing user consent at the PII manager's level is also defined.

6.2. PQI Endpoints. The PQI is used by the URM platform when issuing requests to the PII manager. It exposes three endpoints:

(i) Client registration as defined in Section 6.3.

(ii) The PII retrieval endpoint complies with standard OAuth 2.0 scenarios for a resource server. Additionally, it performs the reduction of OAuth scopes, as mentioned in Section 3.3.4 in [16]. Eventually, the authorization process that is part of the PQI relies on the consent receipts generated by the PII manager.

(iii) Access to the PII metadata introspection endpoint is legitimate to services that do not need the actual PII content. Metadata about this PII can be provided to them instead. Metadata include creation and modification informations and user consents granted to the requesting service for that PII.

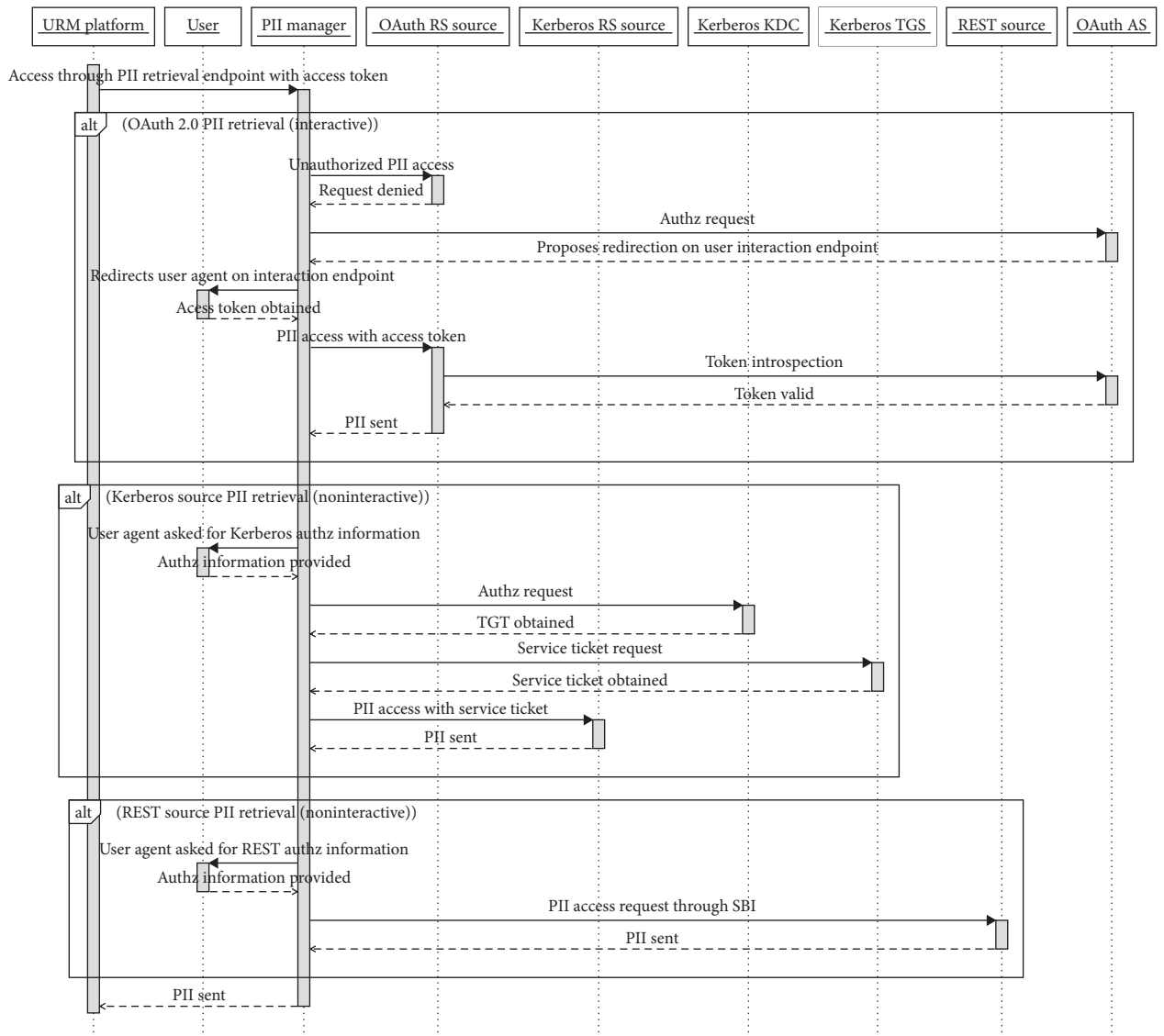


Figure 4: PII collection sequence diagram for getting the source-side user authorization.

In particular, user consent management is necessary to ensure that later offline authorization flows can be granted to the service.

6.3. The Registration Endpoint. The URM platforms must be registered beforehand.

The registration endpoint operates according to the specifications provided in [30]. To enable a service provider to access the registration endpoint, an access token must be delivered by the URM platform administrator. The issuance of the access token is a manual process and is not covered in this document.

Registering a platform is performed by a dedicated endpoint of the PQI, i.e., the registration endpoint. The following information needs to be provided while performing the registration:

(i) *Functional Registration Information.* This information includes terms of the policy, version of the policy terms, categories of PII that will be collected, and purpose of the collection. It details all the elements that will be used when generating a consent receipt, if the user decides to give his consent.

(ii) *Technical Registration Information.* This information includes a set of redirection URIs. These URIs will be later used by the PII manager during the PII authorization and PII access process.

In return, the platform is given an identifier (*"client ID"*) and a password (*"client secret"*), necessary for all the future PII access requests. The platform is supposed to securely store these two registration elements, as they are required when issuing PII access requests to the PII manager.

When deploying the PII manager in our URM platform, each URM subservice (agenda, content-managed system, identity manager, Web form manager, etc.) is preregistered. This preregistration spares the user or agent from manually registering these trustworthy URM subservices. It is

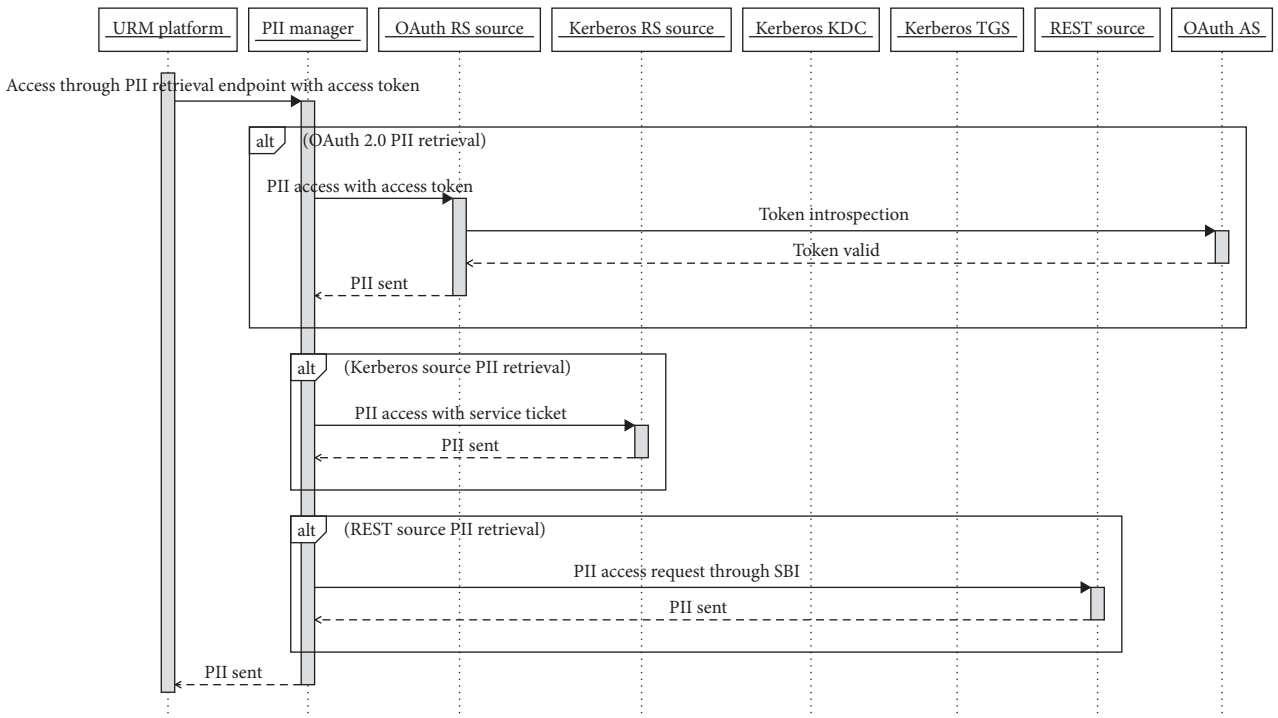


Figure 5: PII collection sequence diagram after source-side user authorization has been obtained.

performed by generating a long-lived registration API token that only these URM subservices know.

Alternatively, the URM platforms are issued a registration token to perform dynamic platform registration according to [30]. In return, these providers are able to access the registration endpoint as an API, issuing all the information necessary to register the platform.

6.4. Usage in a Federated-Identity Environment. The PQI for collecting PII from several remote sources is illustrated in Figure 6.

These three figures illustrate the different HTTPS redirect flows happening between the user and the PII manager when collecting PII from remote sources.

Figure 6(a) depicts the PII access process when the requesting URM client does not possess an access token. As shown in this figure, the PII access process provides an abstraction of the resource location.

Figures 6(b) and 6(c) describe the two possibilities for client-initiated PII collection, after a URM client access token has been issued, as depicted in Figure 6(a), i.e., (a) the direct PII collection for which the PII manager is doing the token verification on its own prior to send the PII to the URM client and (b) the indirect PII collection where the identity provider, through its token introspection endpoint, is asked to verify the token.

6.5. User Consent Enforcement at PQI Level. User consent enforcement at the PQI level enables (i) the delegation of access decisions on the PII manager and (ii) the generation of consent receipts for traceability purposes. Properties (i) and (ii) have both undergone specification efforts by the Kantara Initiative.

An access decision delegation, based on OAuth 2.0 scope-based access requests, is shown in Section 3.3.4 in [16]. UMA specifies (i) the delegation of access decisions. The relevant algorithm, presented in Section 3.3.4 in [16], deals with the way the UMA server should decide on whether to grant access to users. Given an OAuth client C, the input information for this algorithm to be run is preregistration scopes for C, recently requested scopes for C, and OAuth scopes associated with the resources requested by C.

With this input information, the authorization server evaluates which scopes of authorization can be granted to the client. Therefore, three cases are possible: the authorization server to either grant the authorization with the actual scopes as requested by the client, to restrict the authorization grant to a smaller set of scopes, or to completely deny the authorization request.

This algorithm ensuring properties (i) and (ii) is detailed in Section 8.2.

7. The Source Backend

7.1. Overview. The source backend makes it possible to deal with multiple sources, with a backend for each source type. It enables interoperability through the support of multiple authorization and PII access protocols. Indeed, the multiplicity of such authorization protocols makes it hard to identify a unified consent model.

An OAuth scope is an authorization unit, characterizing a resource or an action to be performed on it. In OAuth terms, it is a character string of blank space-separated

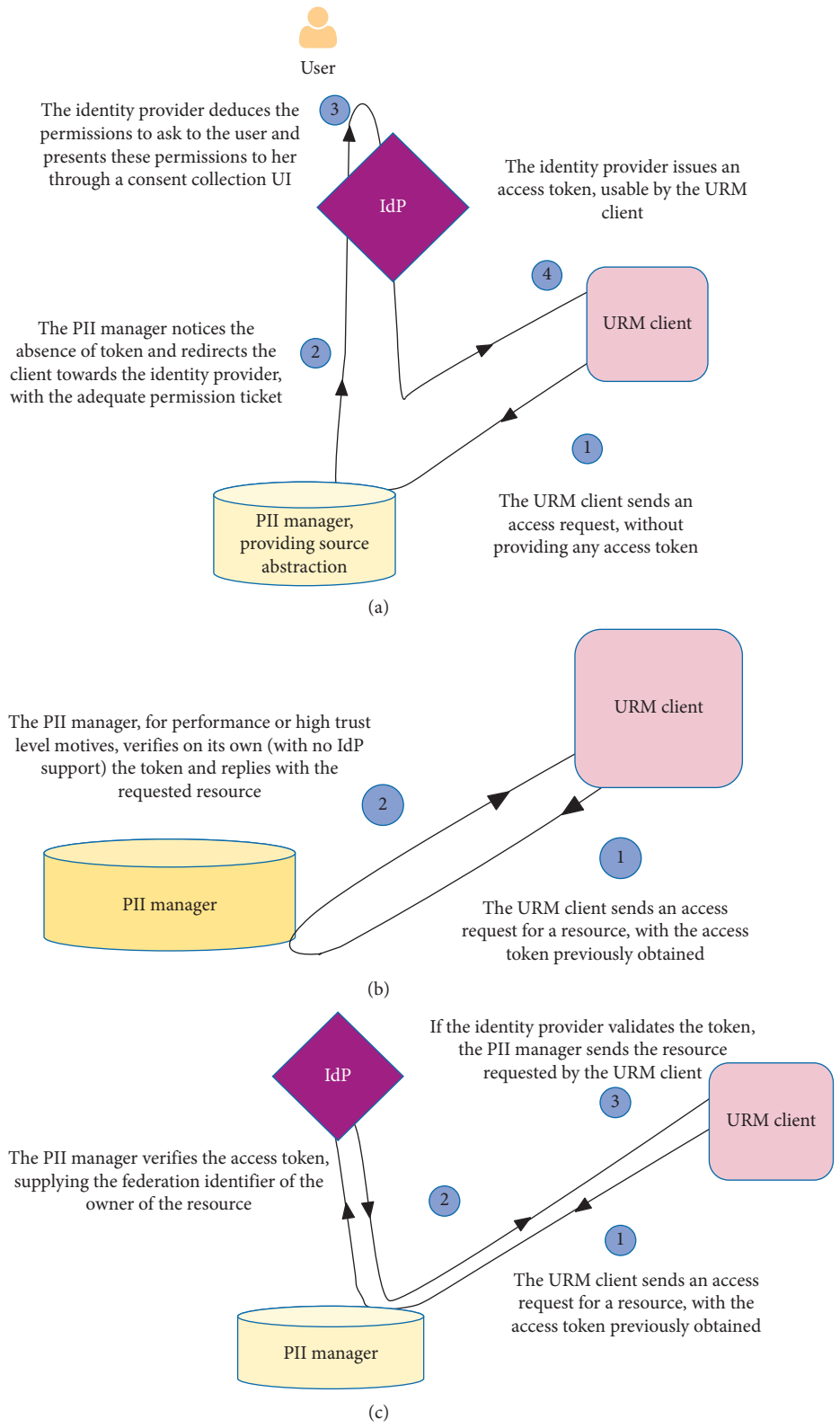


Figure 6: PII collection by the URM client. (a) Access token issuance by the user. (b) Direct user token consumption. (c) User token consumption after verification by the IDP.

keywords that constitute the scopes set. The set of scopes is used in the PQI retrieval endpoints to specify the PII requested.

Finally, this section also discusses consent management as part of the source backend when interfacing with remote sources. For each type of sources, a study of the

authorization information structure is given. Strategies to map authorization information to our consent model are defined.

7.2. Supported Protocols. The list of supported protocol is as follows:

(i) An **OAuth 2.0 (including OIDC) provider** does not require any translation, as it is directly usable by the UMA authorization process used within the URM platform. The scopes used by the OIDC identification protocol are directly compatible with UMA.

(ii) A **SAML provider** translation mainly relies on the use of the OAuth 2.0 assertion framework [31] and its use with the translation of SAML 2.0 assertions [32]. These specifications provide "out-of-the-box" processes for translating SAML assertions to OAuth 2.0 authorization information. Thus, SAML assertions can be used either as OAuth 2.0 client authentication information or authorization grants.

(iii) The **Plain read-only REST sources** only require a static set of scopes predefined by the URM platform administrator. A direct mapping between HTTP verbs as used by REST sources and standard OAuth scopes used by UMA can be established.

(iv) The resource server operating according to the **Kerberos** authorization protocol needs a valid permission ticket. The ticket scope always concerns access to a resource. Finer scopes of authorizations are not supported by the protocol (Kerberos tickets contain an optional authorization data field that can be used to implement authorization scope support. However, it is not covered by the specification, see Chapter 5.3 in [23]. The way the PII manager manages the Kerberos session key, and the manager's registration in the principals' database maintained by the Kerberos administration server, is out of scope of this paper as it deals with the registration process more than the authorization.

7.3. Consent Management

7.3.1. General Considerations. The main objective of consent management is the respect of the user's choices when it comes to considering the URM platforms' PII queries. The authorization system of the PII manager hence relies on such consent management. The authorization lifecycle is therefore managed by the user.

In particular, consent management on the architecture implies keeping track of the users' previous choices regarding the collection of their PII by service providers. The user's consents have a limited lifetime, in particular, they can be given for a single immediate collection, and they have scope denoting the extent of the granted authorization.

We can still define the basic capabilities of the consent management part of the proposed solution. These capabilities are derived from the consent receipt model defined in [27]. This consent receipt model formalizes user authorizations in the consent management system.

The information contained in the receipts as defined in [27] belongs to three categories: (i) receipt transaction fields, (ii) transaction parties' fields, and (iii) data, collection, and use fields.

All three categories are relevant to enforce consent management as per our consent model. In particular, transaction parties' fields enable the declaration of PII controllers that collect the data and whether this collection happens on behalf of another controller. Additionally, data, collection, and use fields enable the declaration of termination policies that apply for the consent as well as purposes and PII categories that apply for the receipt.

A comparison table of the consent models supported by the PII manager is shown in Table 2.

The following criteria are used to make the comparison:

(i) Revocability means the ability to cancel a previously given consent information.

(ii) Multidomain management is relevant when, as depicted in Figure 1(b), several URM platforms interface with a single PII manager. The management of consent information across domains is therefore a key element.

(iii) Terms-of-usage versioning refers to the ability to record the version number of the terms-of-usage for the PII collection that has obtained the user's consent.

(iv) Authorization scope means being able to specify unitary elements defining the authorization request.

(v) Interuser resource sharing defines whether the consent information specifies the ability for other users to directly access the PII.

(vi) Direct verifiability means whether the verifiability information requires a request to an authorization server or can be directly verified by the resource server.

The remaining of this section is organized as follows.

First, the user identifier mapping as part of the consent management is presented. Second, the translation of the authorization information according to the four protocols supported by the sources is detailed. For each protocol, a brief description of the authorization information structure is given, and a translation procedure is then introduced.

7.3.2. User Identifier Mapping. The PII manager is responsible for implementing the user identifier mapping across the sources and the URM platforms. It is of utmost importance that the collected PII from different sources for one user do actually belong to that user. In order to enforce this user uniqueness, the PII may rely on several possibilities, be it either a unique official identity sources such as France Connect, or automated mapping as presented in [19].

The TCPA deploys an identity provider that respects the identity-federation principles. In particular, it provides sector identifiers according to several service sectors of the

Table 2: A comprehensive comparison between different consent models.

Criterion	Consent model			
	PII manager consent receipt (Kantara)	OAuth 2.0 access token	Kerberos ticket	Standard ACLs
Terms-of-usage versioning	Yes	No	No	No
Direct verifiability	Yes, by definition of locally managed consent	Yes	No	Depends on model
Authorization scope	Yes	Yes	No	Partially supported
Revocability	Yes, by definition of locally managed consent	Yes, through AS	Yes	Yes
Multidomain	Yes	Yes	No	Depends on model
Interuser resource sharing	No	No	Yes	Depends on model

TCPA. As a result, two services belonging to different sectors are provided with two different identifiers corresponding to the same user. The PII manager is thus responsible for managing the user identifier mapping across the services.

First, the authorization information from a first URM platform, say platform α, may present the user information including an identifier u_α. A second platform β only knows the authorization information that contains a user identifier u_β. If the user's consent is applied on platforms α and β, a user-identifier mapping must be performed. The PII manager needs to provide a mapping function m:

$$m: \begin{array}{l} U_\alpha^* \longrightarrow U_\beta^*, \\ u_\alpha \mapsto u_\beta, \end{array} \quad (1)$$

where U_α^* and U_β^* are, respectively, the identifiers' sets of platforms α and β. Since those platforms support the OAuth authorization framework, these identifiers can be (i) pseudonyms, (ii) Universally Unique Identifiers (UUIDs), or (iii) human-friendly attributes such as the user's e-mail addresses. Regardless of the platforms' actual identifier policy, these identifiers are considered to be string characters.

Additionally, the PII manager acting as a resource server interacts with several authorization servers that do not belong to the same federated-identity environment. They therefore do not share the same user identifiers.

This altogether justifies the need for an identifier mapping endpoint.

The identifier is mapped to a pseudonym derived from (a) the source subject identifier and (b) the target sector identifier, in a similar fashion as presented in [1]. The process used to derive these pseudonyms can rely on nonreversible pseudonyms' identifier generation as presented in Chapter 8.1 in [1].

The target service then receives a pseudonym with a set of human-readable information that serves as input for the identity-matching procedure, as presented in Chapter 5 in [19].

7.3.3. Translation to OAuth (including OIDC) Consent Information

(i) Structure of Authorization Information. Access tokens, in their most common JWT [33] form, are structured as follows (the structure may vary when the JSON token is encrypted and when it contains unprotected header):

(i) A header bearing token metadata

(ii) A (cleartext or encrypted) payload, which contains the core authorization information

(iii) Optionally, a signature whose validation information is contained in the header

(ii) Grant Type Translation. Grant type translation happens when using standard-OAuth input authorization information within the (UMA-OAuth) PII manager. The OAuth-supported grant types are the implicit grant and the authorization code grant. The authorization code grant is a two-step grant allowing the URM client to request access tokens, by letting the authorization server know that it was giving the user's authorization. On the contrary, the implicit grant is simpler as no authorization code is involved. The user's consent given to the URM client directly results in the authorization server's response that includes an access token.

(iii) Scope Translation. Translating scopes is performed in three steps:

(1) Identify the scopes of interest for the URM platform

(2) List all the other scopes that can be part of the translated assertion

(3) For each of these other scopes, provide a mapping to the scopes supported by the URM platform

(iv) The OIDC Profile. The only supported grant type is the authorization code. OIDC is a simplified version of OAuth, in which the identity provider is also the resource server (in particular, the identity information is the requested resource).

(v) Direct Mapping. The direct mapping from the PII manager's consent model to OAuth authorization information happens as follows:

(1) Issuer to iss

(2) Start timestamp to iat

(3) Expiry timestamp to exp

(4) Authorization resource to the resource URI of the authorization process

(5) Authorization scope to the scopes

(6) Authorization user identifier to sub

7.3.4. Translation to Kerberos Consent Information

(i). Direct Mapping. The direct mapping from the PII manager's consent model to Kerberos' permission ticket information happens as follows:

(i) Issuer to cname and crealm

(ii) Start timestamp to starttime

(iii) Expiry timestamp to endtime

(iv) Authorization resource to authorization data payload of the ticket

(v) Authorization user identifier to principal

7.3.5. Translation to Plain ACL Consent Information

(i) Structure of Authorization Information. According to our hypotheses, plain ACLs' consent information is made of (i) authorization metadata (e.g., temporal and spatial validity metadata) and (ii) core authorization information (e.g., subject and object of authorization).

(ii) Direct Mapping. The PII manager's consent model maps to the ACL information. For instance, the following mapping applies:

(1) Issuer maps to client

(2) Start timestamp maps to beginson

(3) Expiry timestamp maps to endson

(4) Authorization resource maps to resources-uris

(5) Authorization user identifier maps to subject

(6) Delegation flag maps to delegated

7.3.6. Translation to SAML Assertions

(i) Structure of Authorization Information. The PII fields of a SAML assertion are as follows:

(1) The subject of the authorization information (Subject):

  (i) A technical identifier (NameID)

  (ii) A set of human-friendly PII about the subject

  (iii) Validation metadata

(2) The authorization:

  (i) The assertion statement

  (ii) Additional assertion validation metadata

When the assertion is signed and/or encrypted, the public keys and algorithm declarations are available in the server's and service provider's respective metadata [34].

(ii) Direct Mapping. The SAML assertion translation specification of [32], as part of the OAuth 2.0 assertion framework presented in [31], is used for the SAML driver. Campbell et al. [32] specified the way SAML assertions can be used as authorization grants or as client authentication information.

Using this framework means that mappings for the elements of a SAML assertion are supported. For instance, the following elements need mapping:

(i) The user identifier needs to be mapped to the UUID within the URM platform. This mapping is handled by the identity provider of the URM platform, which offers a user-identifier resolution service to the PII hub.

(ii) Scopes of authorization need to be translated to the scopes that actually are enforced by the URM platform. There is no possible comprehensive list for these scopes, as the OAuth-based protocols can extend the standard scope model.

In terms of mapping the content of the SAML assertion to the consent model, the following elements need to be considered:

(i) User identifier maps to the NameID. The NameID format must be one of UUID or e-mail address.

(ii) Start timestamp maps to NotBefore.

(iii) Expiry timestamp maps to NotOnOrAfter.

(iv) Names, formats, and values of standard attributes also need mapping. This mapping depends on the type and format of attribute and is not covered in this document.

(v) Our consent model also supports extended, admin-definable, attributes that can be mapped in a similar fashion, depending on their respective types and formats.

7.4. Considerations Regarding Token Exchange. As specified in [35], plain OAuth access tokens can also be exchanged for delegation or impersonation purposes. The delegation and impersonation (impersonation is not used in a negative way in [27]. It means that the target service does not need to be aware of the delegation that happens between a subject entity and an actor entity) features require that the PII manager be able to perform an additional round of redirection, that enables the retrieval of a security token.

In particular, it requires the ability to

(i) Receive an access token and to choose the adequate backend

(ii) Retrieve the newly-issued security token

(iii) Submit the token for consumption to the proper backend

However, the Token Exchange IETF Request for Comments (RFC) is quite recent, and the use of such protocol in the industry is not widespread yet.

8. PII Management User Interface (PMUI)

8.1. Overview. The PII management interface's main purpose is to provide users with an interface for managing authorizations of URM Clients on their PII. The configuration capabilities of the component enforce such a management even when the user is offline.

User-Managed Access in Section 3.3.4 in [16] proposes a procedure to ensure these offline properties. The procedure relies on the OAuth scopes on resources requested by the URM clients.

Additionally, the PMUI reflects the ability of the PII manager to abstract the PII location. As a reminder, we note that the PII abstraction property requires that the PII manager acts first (i) as a requesting party for the registered PII sources and then conversely (ii) as a resource server for the URM platform.

This management user interface offers the visualization of PII transfers with service providers. When granting PII access to the SPs, the PII manager provides logs' information to the associated users and to the data owners. These logs can be visualized at the convenience of the user, i.e., whenever it is suitable for the user. Information obtained at client registration time is also presented to the user, such as the category of service providers and the purpose of collection.

Eventually, the user must be able to revoke a previously granted access. The PII Management User Interface thus includes management pages for each previously created access rule.

8.2. User-Definable Parameters. We rely on an access policy definition at resource registration time, enabling the user to define the parameters below:

(i) The required scopes for the resource: these scopes describe usual operations such as reading, deleting, modifying a resource, and accessing a subresource. Alternatively, they can also be specific third-party application scopes.

(ii) The time window of access authorization: when issuing access tokens within the URM system, the PII manager uses these user-defined parameters to adjust the validity time window of the token.

(iii) The service or the category of service for this authorization rule: the services accessing to the user's PII are sorted according to user-defined categories. Any authorization rule defined by the user is applicable to a category of services only.

When user data are provided by sources acting as SAML or OIDC providers or as OAuth resource servers, this information may already be provided along with the PII payload. Yet, it may be overriden by the user. For plain REST sources, however, this information needs to be provided at registration time.

As a result, the user interface adopts this approach, letting the user define the aforementioned parameters when the metadata provided by the source does not provide this information.

The constraint of unicity regarding the resource, the scope of action, the time window, and the category of services altogether is enforced. At a particular moment in time, for a given resource, a category of service, and a scope of action, at most one authorization rule can apply. The access control system denies all by default.

The algorithm applying, based on the consent model given in [27], can be formulated as the verification:

(i) Of the issuance date

(ii) Of the expiry date

(iii) That the terms-of-use version applies

(iv) That the consent geographical location applies

(v) Of the scopes according to previously-granted scopes for the category of service as follows:

  (1) Retrieve the set of previously granted scopes as defined in Section 3.3.4 in [16]

  (2) Translation of authorization information into OAuth scopes known to the PII manager

  (3) If the authorization server chooses to reduce the set of granted scopes, in comparison with the requested scopes, then a reverse translation is necessary: the OAuth scopes known to the PII manager are reverse-translated to the OAuth authorization information model as dealt by the requesting party.

For instance, while accessing a REST source, the delegated authorization flow is the following one, as summarized in Figure 7:

(1) URM client request: the URM (OAuth) client asks for access to resources/picture1.jpg with the read write print caption scopes

(2) First way translation: the client requested scopes translate to the ability to perform GET, POST, or PATCH on the URI

(3) Reduction: the PII manager gets from its own internal authorization information that only the GET verb is authorized for that resource and this URM client

(4) Reverse translation: the PII manager reverse-translates to the read OAuth scope

(5) PII manager response: the PII manager sends an access token with the reduced read scope

9. Functional Analysis of the Architecture

9.1. Overview. This section provides an informal analysis of the compliance of our PII manager approach with the targeted functional requirements described in Section 3.3. It starts first by listing below useful elements of our solution for fulfilling that compliance and then it provides a full description for some, in the sections that follow.

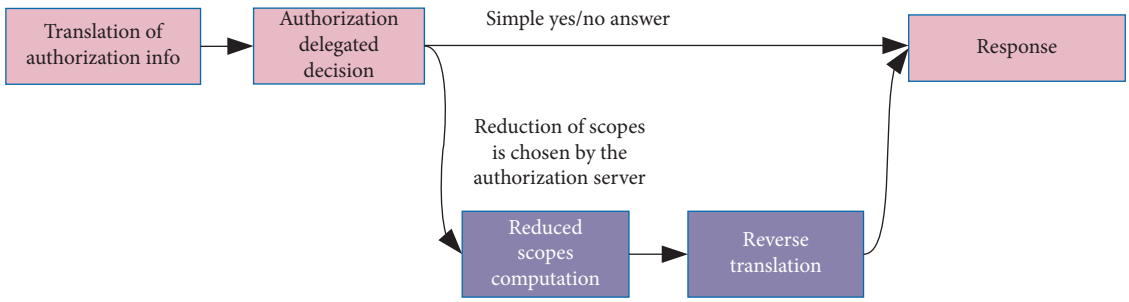


Figure 7: PII collection by the URM client: extension of the UMA decision process by the PII manager.

(1) **The PII management capabilities** map to requirement "usage definition" (requirement #1)

(2) **The PQI and source backend of the PII manager** map to requirements "consent monitoring" and "usage monitoring" (#2 and #3)

(3) **The delegation capabilities** map to requirement "delegation capabilities" (#4)

(4) **The PQI and the source backend**, again, map to requirement "PII location abstraction" (#5)

(5) **The unified authorization scheme** maps to requirements "protocol standardization" and "access uniformization" (#6 and #7)

(6) **The support of several types of sources** maps to requirement "authorization protocol interoperability" (#8)

9.2. Usage Definition (Requirement #1). The consent receipt model adopted in our contribution covers usage definition. Indeed, the data fields and the transaction fields that are part of this model make it possible to specify the purpose of PII collection as part of user consent information.

9.3. Consent Management and Usage Monitoring (Requirements #2 and #3). Consent management is achieved thanks to the use of consent receipts and the mapping of authorization information. Usage monitoring is ensured by PII manager, acting as a single resource server for the URM platform(s).

9.4. Delegation Capabilities (Requirement #4). Section 8.2 specifies the necessary delegation capabilities that our PII manager supports in order to comply with the use case. In particular, that section provides a pseudoalgorithm for the reduction of the authorization scopes set, compatible with the UMA-delegated authorization process.

9.5. PII Location Abstraction (Requirement #5). The translation of authorization information defined in Section 6 enables the PII manager to provide the TCPA URM platforms with the user's PII regardless of the PII actual location.

9.6. Protocol Standardization and Access Uniformization (Requirements #6 and #7). The access control rules describe whether the user authorization information for accessing PII can be granted to a URM client, either directly or through inference based on contextual information.

The direct grant is performed through the definition of preferences by the user. Moreover, these preferences are directly linked to a service provider's client.

This model helps ensuring the minimization of PII transfers: the PII manager, when deciding whether to authorize PII access to a given URM client regarding several categories of PII, can quickly verify if one of the categories of PII is not accessible by that URM client.

A simple PII verification algorithm is as follows:

(i) Claims and scopes are checked against the rules defined for this URM client. The two main elements involved in this process are as follows:

  (1) The translation of the source's authorization information into scopes that are known to the authorization server. This translation step depends on the type of source, as explained in Section 7.

  (2) The comparison of the required scopes with the user-defined preferences.

(ii) When user-defined preferences are insufficient in order to take action, the required scopes are also compared to the scopes previously granted to the URM client. Based on the UMA OAuth 2.0 grant access control process provided in [16], the server can decide to reduce the required scopes to a set of scopes that are appropriate regarding the URM client and the requested resource. Alternatively, the resource server may reject the URM client's request.

9.7. Authorization Protocol Interoperability (Requirement #8). As described in Section 5, respecting our use case involves a strong correlation between the PII management entity and the TCPA URM platform. We now discuss the (a) interoperability property and (b) more specifically the possibility for the PII management entity to interface with other TCPA's URM platforms.

In order to ensure this interoperability property, four necessary subproperties are identified: (i) interface standardization, (ii) dynamic registration (or not configuration

at all), (iii) authorization protocol(s) standardization, and (iv) data exchange format(s) standardization.

(i) It means that the PII manager can be used for several URM platforms at a time. This subproperty is ensured by offering a standard REST API. Such an API offers unambiguous data location format, standardized data operation syntax using HTTP verbs, and use of common Web technologies.

(ii) It is necessary if that interoperability property is expected to be seamless, i.e., with no configuration whatsoever by any human agent involved. This is achieved by OAuth 2.0 Dynamic Client Registration [30] and its associated management protocol [36]. In order for the PII manager to perform dynamic registration of the URM platforms, the following information is necessary:

(a) Endpoints information (support grant types and token authentication methods)
(b) Redirection URIs
(c) Keysets' locations

From the user's point of view, when registering a new PII manager to their URM platform, it is sufficient to simply provide the PII manager URL.

In delegated authorization mechanisms such as the UMA grant for the OAuth 2.0 authorization protocol, a URM tool acting as a Requesting Party needs to identify itself (with a prior registration on the authorization server) before obtaining the requested authorization data.

(iii) It is provided when the PII manager acts as an OAuth 2.0 Resource Server. This PII manager is therefore able to verify the validity of an access token for a given Requesting Party. This validation is performed according to the authorization server's token introspection endpoint [37].

(iv) It implies that the PII exchanged is presented in a way that is recognised by both the sender and the receiver. This standardization is enforced by the common use of OAuth-based protocols and the assertion framework that makes it possible to interface with other federated-identity management protocols such as SAML.

10. Implementation Considerations and Proof of Concept

This section describes the implementation considerations resulting from the prototype design of the PII manager in a TCPA URM software system.

10.1. Proof-of-Concept Implementation. A proof-of-concept implementation is visible at the following public git repository: (https://git.entrouvert.org/pii-manager-poc.git/). It is distributed under the Affero General Public License, in its third version (AGPLv3) (for more information about the AGPL and its differences with its more famous sibling the General Public License (GPL), see (https://www.gnu.org/licenses/agpl-3.0.html). It provides a proof of concept for the support of OAuth 2.0 sources and REST sources.

It uses the Django web framework, in its second version. The noteworthy parts of the implementation are as follows:

(1) The data model implements, amongst other models, the consent receipt model specified in [27].

(2) The view logic performs OAuth authorization with the scope reduction logic presented in Section 8.2. The view logic implementation conceals the complexity of gathering PII from several sources from the user's point of view.

(3) The source backend implements support for OAuth 2.0 sources and REST sources and puts the base layout for a future support of SAML and Kerberos sources.

Of course, some parts need improvement, such as the PMUI, presented in Section 8, which for now only relies on the Django administration user interface ("/admin/") for the management of Django data models.

10.2. Considerations and Guidelines Regarding the Implementation of the PII Manager

10.2.1. Client Type. The OAuth client type depends on the ability of the client to store the secret information that was delivered by the authorization server. For instance, public clients are unable to safely keep a client secret and are therefore excluded from some authorization grant types. It is likely that the PII manager as an OAuth 2.0 client will be able to store its client secret and to operate according to any of the four main authorization grant types defined by the OAuth protocol.

10.2.2. Access Token Lifetime and Refresh Token Persistence. Providing PII abstraction can result in longer PII retrieval time by the PII manager. The lifetime of access token delivered by the PII manager for usage within the URM platform should take this extra delay into consideration, at the implementation level.

The choice to persist refresh tokens must be evaluated according to its privacy-usability tradeoff: persistent refresh tokens are more convenient for the PII Manager. On the contrary, for obvious reasons they make it more difficult to enforce client revocation.

More generally, adequately choosing token lifetime as well as authorization code expiration timestamp can help enforce privacy-compliant properties such as forward secrecy: a malicious user obtaining a token will be limited by its lifetime (that is, in the case of a plain bearer token [39], when cryptographic tokens such as JSON Web Tokens (JWT) are used, man-in-middle attacks such as the theft of a token are preventable).

10.2.3. Introspection Endpoint and Token Validation. The choice of performing token introspection by the

authorization server from which the token originates [37] must also be examined carefully while performing the implementation of the PII manager. In some cases, token introspection by the origin authorization server may not even be possible, in which case a partial validation would be the only possible option.

11. Conclusion

Our PII manager, presented into practical implementation level of details, provides an abstraction solving issues due to multiple sources being considered. These issues include variety of protocols being implemented by the sources, and the resulting variations in the authorization information and in the user consent enforcement. Our approach relies on three main specified components allowing the support of functional requirements identified in our use case, including the support of several sources.

The PII Query Interface (PQI) specifies the way the PII manager interacts with URM platforms, possibly involving an identifier mapping service.

The source backend (SB) specifies the interface with sources obeying to different authorization and PII retrieval protocols. Operating a SB requires a unified consent model, involving an authorization information translation across protocols. This consent model unification step, as performed by the SB, also needs a reverse translation step when the authorization scopes need reduction.

The PII Management User Interface (PMUI) specifies the user-definable parameters that take part in the support of multiple sources and the enforcement of user consent on the PII retrieved across these sources.

Our functional analysis of the architecture demonstrates that the requirements identified in the use case have been correctly addressed.

However, an inherent limitation due to the use case and its requirements can be identified. Indeed, this use case and requirements are based on the industrial hypotheses that have their own bottlenecks and limitations. First, the implementations' variations of effective sources from the theoretical specifications bring complexity to the solution and make it not entirely generalizable. For instance, the inner authorization logic within vanilla OAuth authorization servers (i.e., non-UMA servers) is out of scope of [2]. Having a clear inner authorization logic known to the PII manager when interfacing with OAuth sources would be beneficial, yet not possible at the moment. More generally, we acknowledge that, in our architecture, the PII manager, albeit necessary to enforce our use case while maintaining its functional requirements, results in adding an extra indirection layer that complexifies the overall architecture, induces extra costs, and requires more computing resources. Eventually, the industrial reality that new standards and protocol does not, for practical reasons, always mean the actual depreciation of the previous ones, brings interoperability concerns, as it complexifies the implementation of the PII manager by increasing the number of required drivers.

Finally, solutions such as the PII format proposed by the System for Cross-domain Identity Management (SCIM) [38], although not supported by effective production official sources yet, would be a solution to that issue.

Abbreviations

API: Application programming interface
AS: Authorization server
Authn: Authentication
GDPR: General Data Protection Regulation
IDP: Identity provider
KDC: Key distribution center
PII: Personally Identifiable Information
PQI: PII Query Interface
PMUI: PII Management User Interface
SB: Source backend
TCPA: Territorial Collectivities and Public Administration
TGS: Ticket-granting server
TGT: Ticket-granting ticket
UI: User interface
UMA: User-Managed Access
URM: User-Relationship Management
UUID: Universally Unique Identifier.

Data Availability

The results rely on open specifications (mostly from the Internet Engineering Task Force—IETF) and available publications. All the hyperlinks appearing in the references or in the text have been checked prior to manuscript submission. Additionally, the academic and industrial software solutions studied in this article are free software, whose source code can be obtained freely on each solution's website as appearing in the references.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] N. Sakimura, J. Bradley, M. Jones, B. De Medeiros, and C. Mortimore, "Openid connect core 1.0 incorporating errata set 1," https://openid.net/specs/openid-connect-core-1_0.html Technical Report, OpenID, San Ramon, CA, USA, 2014, https://openid.net/specs/openid-connect-core-1_0.html Technical Report.

[2] D. Hardt, "The OAuth 2.0 authorization framework," https://rfc-editor.org/rfc/rfc6749.txt Technical Report 6749, Internet Engineering Task Force, Fremont, CA, USA, 2012, https://rfc-editor.org/rfc/rfc6749.txt Technical Report 6749.

[3] N. Kaaniche, M. Laurent, and S. Belguith, "Privacy enhancing technologies for solving the privacy-personalization paradox: taxonomy and survey," *Journal of Network and Computer Applications*, vol. 171, Article ID 102807, 2020.

[4] M. Ates, S. Ravet, A. Mohamat Ahmat, and J. Fayolle, "An identity-centric Internet: Identity in the cloud, identity as a service and other delights," https://ieeexplore.ieee.org/document/6045976 Technical Report, IEEE, Vienna, Austria, 2011, https://ieeexplore.ieee.org/document/6045976 Technical Report.

[5] Y.-A. de Montjoye, E. Shmueli, S. S. Wang, and A. S. Pentland, "OpenPDS: protecting the privacy of metadata through safe answers," *PLoS One*, vol. 9, no. 7, Article ID e98790, 2014.

[6] E. Papadopoulou, A. Stobart, N. K. Taylor, and M. H. Williams, *Enabling Data Subjects to Remain Data Owners*, Springer International Pulishing, Cham, Switzerland, 2015.

[7] H. Haddadi, H. Howard, A. Chaudhry, J. Crowcroft, A. Madhavapeddy, and R. Mortier, "Personal data: thinking inside the box," 2015, http://arxiv.org/abs/1501.04737.

[8] Entr'ouvert, "Fargo document manager presentation page," 2021, https://dev.entrouvert.org/projects/fargo.

[9] D. Nuñez and I. Agudo, "Blindidm: a privacy-preserving approach for identity management as a service," *International Journal of Information Security*, vol. 13, no. 2, pp. 199–215, 2014.

[10] Entr'ouvert, "The authentic 2 identity manager presentation page," 2021, https://dev.entrouvert.org/projects/authentic.

[11] ForgeRock, "Openidm (documentation)," https://backstage.forgerock.com/docs/openidm Technical Report, ForgeRock, San Francisco, CA, USA, 2021, https://backstage.forgerock.com/docs/openidm Technical Report.

[12] The OpenStack Foundation, "Keystone, the openstack identity service (documentation)," https://docs.openstack.org/keystone/pike/ Technical Report, OpenStack, Austin, TX, USA, 2021, https://docs.openstack.org/keystone/pike/ Technical Report.

[13] R.H.. Keycloak, "Iam Documentation," Technical Report, RedHat, Raleigh, NC, USA, 2021.

[14] C. Jan and B. Pfitzmann, *Federated Identity Management*, Springer, Berlin, Heidelberg, 2007.

[15] C. Paquin, "U-prove technology overview v1.1 (revision 2)," 2013, https://www.microsoft.com/en-us/research/publication/u-prove-technology-overview-v1-1-revision-2/.

[16] E. Maler, M. Machulak, J. Richer, and T. Hardjono, "User-Managed Access (UMA) 2.0 Grant for OAuth 2.0 Authorization," Internet Engineering Task Force, Fremont, CA, USA, 2019, https://datatracker.ietf.org/doc/html/draft-maler-oauth-umagrant-00 Internet-Draft draft-maler-oauth-umagrant-00.

[17] A. Ceccanti, M. Hardt, B. Wegh, and A. Paul Millar, "The indigo-datacloud authentication and authorization infrastructure," *Journal of Physics: Conference Series*, vol. 898, no. 10, Article ID 102016, 2017.

[18] European Parliament, "Regulation (eu) 2016/679 of the european parliament and of the council of 27 4 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation)," 2016.

[19] P. Marillonnet, M. Ates, M. Laurent, and N. Kaaniche, "An identity-matching process to strengthen trust in federated-identity architectures," in *Proceedings of the 17th International Joint Conference on e-Business and Telecommunications*, vol. 3, pp. 142–154, SciTePress, Paris, France, July 2020.

[20] Organization for the Advancement of Structured Information Standards, "Security assertion markup language (Saml) v2.0," http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html Technical Report, OASIS, Burlington, MA, USA, 2005, http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html Technical Report.

[21] Roy Thomas Fielding, "REST: architectural styles and the design of network-based software architectures," http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm Doctoral dissertation, University of California, Berkeley, CA, USA, 2000, http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm Doctoral dissertation.

[22] J. Reschke, "The "basic" HTTP authentication scheme," https://rfc-editor.org/rfc/rfc7617.txt Technical Report 7617, Internet Engineering Task Force, Fremont, CA, USA, 2015, https://rfc-editor.org/rfc/rfc7617.txt Technical Report 7617.

[23] C. Neuman, S. Hartman, K. Raeburn, and Y Taylor, "The kerberos network authentication service (V5)," https://rfc-editor.org/rfc/rfc4120.txt Technical Report 4120, Internet Engineering Task Force, Fremont, CA, USA, 2005, https://rfc-editor.org/rfc/rfc4120.txt Technical Report 4120.

[24] M. Wolf and C. Wicksteed, "Date and time formats," https://www.w3.org/TR/NOTE-datetime Technical Report, W3C, Cambridge, MA, USA, 1997, https://www.w3.org/TR/NOTE-datetime Technical Report.

[25] C. Newman and K. Graham, "Date and time on the internet: timestamps," Technical Report 3339 URL https://rfc-editor.org/rfc/rfc3339.txt, Internet Engineering Task Force, Fremont, CA, USA, 2002.

[26] H. Schulzrinne, "The tel URI for telephone numbers," https://rfc-editor.org/rfc/rfc3966.txt Technical Report 3966, Internet Engineering Task Force, Fremont, CA, USA, 2004, https://rfc-editor.org/rfc/rfc3966.txt Technical Report 3966.

[27] Kantara Consent & Information Sharing Work Group, "Consent receipt specification," https://kantarainitiative.org/file-downloads/consent-receipt-specification-v1-1-0/ Technical Report, Kantara Initiative, Lonavla, India, 2018, https://kantarainitiative.org/file-downloads/consent-receipt-specification-v1-1-0/ Technical Report.

[28] M. Hodder, M. Lizar, M. Sabadello, and J. Wunderlich, "Minimum viable consent receipt (Mvcr) specification v.05," https://kantarainitiative.org/confluence/display/archive/Minimum+Viable+Consent+Receipt+%28MVCR%29+Specification+v.05 Technical Report, Kantara Initiative, Lonavla, India, 2014, https://kantarainitiative.org/confluence/display/archive/Minimum+Viable+Consent+Receipt+%28MVCR%29+Specification+v.05 Technical Report.

[29] R. Shekh-Yusef, D. Ahrens, and S. Bremer, "HTTP digest access authentication," https://rfc-editor.org/rfc/rfc7616.txt Technical Report 7616, Internet Engineering Task Force, Fremont, CA, USA, 2015, https://rfc-editor.org/rfc/rfc7616.txt Technical Report 7616.

[30] J. Richer, M. Jones, J. Bradley, M. Machulak, and P. Hunt, "OAuth 2.0 dynamic client registration protocol," https://rfc-editor.org/rfc/rfc7591.txt Technical Report 7591, Internet Engineering Task Force, Fremont, CA, USA, 2015, https://rfc-editor.org/rfc/rfc7591.txt Technical Report 7591.

[31] B. Campbell, C. Mortimore, M. Jones, and Y. Yaron, "Goland. assertion framework for OAuth 2.0 client authentication and authorization grants," https://rfc-editor.org/rfc/rfc7521.txt Technical Report 7521, Internet Engineering Task Force, Fremont, CA, USA, 2015, https://rfc-editor.org/rfc/rfc7521.txt Technical Report 7521.

[32] B. Campbell, C. Mortimore, and M. Jones, "Security assertion markup language (SAML) 2.0 profile for OAuth 2.0 client authentication and authorization grants," https://rfc-editor.org/rfc/rfc7522.txt Technical Report 7522, Internet Engineering Task Force, Fremont, CA, USA, 2015, https://rfc-editor.org/rfc/rfc7522.txt Technical Report 7522.

[33] M. Jones, J. Bradley, and N. Sakimura, "JSON web token (JWT). RFC 7519," 2015, https://rfc-editor.org/rfc/rfc7519.txt.

[34] C. Scott, J. Moreh, R. Philpott, and E. Maler, "Metadata for the oasis security assertion markup language (Saml) V2. 0," https://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf Technical Report, OASIS, Burlington, MA, USA, 2005, https://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf Technical Report.

[35] M. Jones, N. Anthony, B. Campbell, J. Bradley, and C. Mortimore, "OAuth 2.0 token exchange," https://rfc-editor.org/rfc/rfc8693.txt Technical Report 8693, Internet Engineering Task Force, Fremont, CA, USA, 2020, https://rfc-editor.org/rfc/rfc8693.txt Technical Report 8693.

[36] J. Richer, M. Jones, J. Bradley, and M. Machulak, "OAuth 2.0 dynamic client registration management protocol," https://rfc-editor.org/rfc/rfc7592.txt Technical Report 7592, Internet Engineering Task Force, Fremont, CA, USA, 2015, https://rfc-editor.org/rfc/rfc7592.txt Technical Report 7592.

[37] J. Richer, "OAuth 2.0 token introspection," https://rfc-editor.org/rfc/rfc7662.txt Technical Report 7662, Internet Engineering Task Force, Fremont, CA, USA, 2015, https://rfc-editor.org/rfc/rfc7662.txt Technical Report 7662.

[38] P. Hunt, G. Kelly, E. Wahlstroem, and C. Mortimore, "System for cross-domain identity management: core schema," https://rfc-editor.org/rfc/rfc7643.txt Technical Report 7643, Internet Engineering Task Force, Fremont, CA, USA, 2015, https://rfc-editor.org/rfc/rfc7643.txt Technical Report 7643.

[39] M. Jones and D. Hardt, "The OAuth 2.0 authorization framework: bearer token usage," Internet Engineering Task Force, 2012, https://rfc-editor.org/rfc/rfc6750.txt Technical Report 6750.