Aadhaar-enabled Data Sharing

CONTENTS

LIST OF FIGURES

LIST OF TABLES

DOCUMENT HISTORY

Version & Date	Changes	Change-makers
0.1, Dec 17	Initial version	Venkata Pingali

1 ABSTRACT

TBD

2 OVERVIEW

Government of India, private industry, and residents gather and maintain vast amount of personal data such as personal contact information, addresses, school records, and health records. It is distributed and duplicated, and used mainly by the institution with access to a copy of data such as specific government departments. Residents usually do not have access or control over this data. The Government of India is taking major steps to enhance transparency by sharing very significant amount of information such as Job cards in the NREGA program. What is missing, however, is a standardized framework for authorized and controlled sharing that is applicable across institutions and that can be collaboratively developed within the e-governance community of stakeholders.

In this document we describe a standardized *approach* to Aadhaar-enabled data sharing. We show how Aadhaar can be combined with Internet standard OAuth2.0 protocol and REST architectural style to enabling secure and controlled sharing of various kinds of data such as health records. The building blocks are well known. The purpose of this document is simply to bring them together in the Aadhaar context without any strong claims of originality. An additional step of standardizing protocol-level details is required in order to build real applications. That discussion is beyond the scope of this document.

We summarize the framework first and follow it with detailed discussion of individual components and flows in the framework. The intended audience for this document is the technical software design community within the government and in industry, and assumes familiarity with UIDAI authentication protocol.

2.1 Why Data Sharing?

There are three basic reasons for deploying data-sharing frameworks extensively. First, it is the resident's right to access his/her data in a relatively easy, low-cost, and time and location-independent manner. Internet-based online data sharing has all those attributes. Second, it is cumbersome for the resident to authenticate themselves each and every time they request a service. A trusted mechanism that stores and shares the results of the authentication saves time and effort for everyone. Third, experience in the advanced countries suggests that as more data is shared, services that use the data as a substrate emerge such as health monitors and pension reminders. This is beneficial to the residents and economy. Last, this advances the transparency objectives of the Government of India.

Trust is crucial to the proposed data sharing mechanism. Aadhaar, as we point out later, is the source of both the data and trust.

2.2 Technology Framework

The proposed data-sharing framework has three basic components: OAuth2.0 protocol implementation, Aadhaar authentication, and REST API. They address the security, trust, and operational efficiency aspects of the mechanism.

The table below summarizes the components. We discuss them in more detail in the next section.

Table 1 Framework components and their roles

Component	Function
IETF OAuth2.0	Secure controlled access to data
Aadhaar	Source of data and/or trust in the data
REST API	Efficient cross-platform application development

The workflow is mainly dictated by the OAuth2.0 protocol. The organization hosting data shares the data with a trusted third party when authorized by the resident. The nature and quantity of data shared, the duration for which the sharing occurs and the degree of anonymization or obfuscation are specified by the resident.

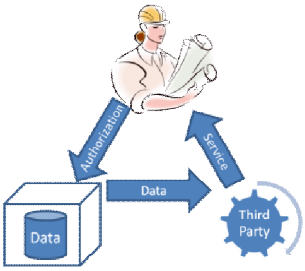

Figure 1 Datasharing workflow

A key challenge here is the cognitive complexity experienced by the resident. OAuth2.0 goes a long way to reducing the complexity. However, it must be complemented with standardization and best practices for interfaces to reduce the amount of learning for the resident.

3 COMPONENTS

In this section we discuss each of the building blocks, viz., the OAuth 2.0 protocol, Aadhaar authentication, and REST API, in the context of the Aadhaar-enabled data sharing framework.

3.1 OAuth2.0 and its Application

OAuth is an IETF-approved data-sharing protocol [5]. It is used by established player such as Google[2], Facebook[1], and Twitter[3] to enable easy development of third party applications without requiring username and password.

3.1.1 OAuth2.0 Protocol

There are two versions of the protocol. OAuth1.0 was the original OAuth protocol. It was found to be complex and difficult to implement. OAuth2.0 is a more recent and simpler version of the protocol that is in the middle of the standardization process [XX]. However, OAuth2.0 needs more careful implementation in order to ensure high security.

There are four basic principals in the OAuth 2.0 protocol – the resource owner, the client, the authorization server and the resource server. The resource owner is the entity to which the data ultimately belongs. The client is the entity that seeks access to the data in order to provide some service that the resource owner wishes to receive. The authorization server gives controlled access to data using appropriate security and authorization protocol. The resource server hosts the data and provides access to it to the client when the client presents proof of authorization.

The resource owner, or some related entity, approaches the client for service that triggers the flow. The client first acquires authorization from the resource owner. It then presents the authorization received to the authorization server to acquire appropriate credentials to present to the resources. The client receives access to the data upon presenting credentials.

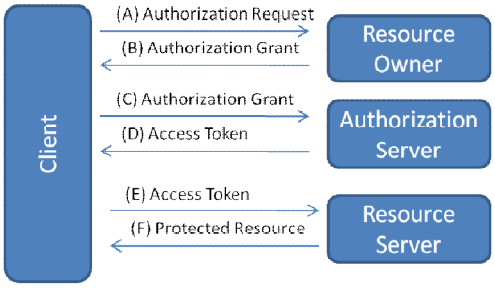

Figure 2 OAuth 2.0 protocol flow

3.1.2 Workflows

In the Aadhaar-enabled data-sharing context, the resource owner is the resident. The resource server and authorization is typically one entity because the implementations of the two are closely tied. This entity is likely to be an existing institution that provides a significant public service such as a Government of India department or a hospital. It must have a reason to store the data to begin with. The client is typically a mobile or web application that provides some service that the resident wishes to receive. The table below shows some use cases and corresponding principals.

Table 2 Example principals and usecases

Service	Class of Residents	Application	Authorizer
Health reimbursements	Employee of Company X	Company X's automated medical reimbursement service	Hospital
Pension reminder service	Pensioner	Collection reminder service	Government of India
Automatic form filling	Mobile service consumer	Fill PoI and PoA forms	Aadhaar-enabled KYC service

The workflows for the resident and applications are shown in the two figures below. The application first creates a client at the authorizer's site. The authorizer generates a client key or id and client secret along with information on server configuration such as the URL for the authorization. At this point the resource server is ready to receive an authorization request from the client. A good practice is to let the application developer download a client configuration file so that there are no copy-paste errors in setting up the client. An example is shown below.


```
{
  "resource_name": "Resource Site",
  "resource_server": "http://localhost:8000",
  "client_secret": "57edb78728e1e61c25cca5e104f9c5",
  "client_key": "d53d90894c157ab94d693a24d7a0cc"
  "authorize_url": "http://localhost:8000/oauth2/authorize",
  "access_token_url": http://localhost:8000/oauth2/token
}
```


Figure 3 Example client configuration

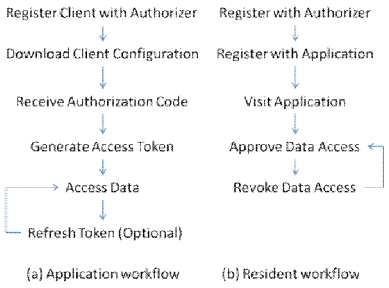

Figure 4 Workflows for the application and the resident

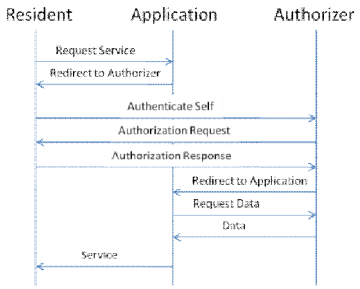

Figure 5 OAuth2.0 flow in Aadhaar context

The resident signs up with both the authorizer as well the application. At the authorizer, the resident authenticates himself/herself using Aadhaar. Depending on the nature of data being shared and required level of trust, the authentication requirements may range from none to full

demographic and biometric attributes. Once Aadhaar-authenticated, the resident can specify preferences for context-dependent sharing – the duration, the scope, and degree of anonymization for the data for various context of use. As mentioned later, a key challenge is to design the scopes to be few in nature and easy to understand.

When the resident visits the application, the application redirects the resident to the authorizer site. The resident is presented the scope and authorization request. When the resident authorizes, the application receives a code with limited lifetime, say 5mins. The application in the background uses the code to first acquire an access token, and later access data using the token. The access token can have longer lifetime such as 1 hour.

A key challenge is the design of the API so that the application can relatively easily acquire the data. We discuss the API design later in the document.

3.2 Aadhaar Authentication

Aadhaar serves two purposes in the context of data-sharing. It is a direct or indirect source of data, and an implicit or explicit source of trust. The table below summarizes various classes of data. The trust in direct data (e.g., authentication records) is derived from UIDAI's authentication service [6] and the belief in the correctness of the implementation. The trust in indirect data and linked data is implicit, and is based on the perceived increased ability to detect faults and correct them using Aadhaar. Within Aadhaar system, there is no way to check the authenticity of any demographic and biometric data except by invoking authentication again. If we have to do it frequently enough, then sharing such data using whatever mechanism does not add any value. For linked data and optional data such as BPL numbers, UIDAI does not provide any mechanism to verify the data.

Table 3 Various classes of data associated with Aadhaar

Data	Aadhaar's Role	Access to Data
Aadhaar Authentication Records	Direct and indirect source	Multiple ways – UIDAI's own data sharing mechanism, RTI, AUA
Demographic and biometric data	Indirect source	Capture devices and mechanisms, e.g., PoS terminals, Internet sites
Linked data (e.g., health records, NREGA payments)	Association	Associations created explicitly or implicitly by the resident or other third parties

3.3 REST API

Machine accessible interfaces or APIs are crucial for application development. REST[4] is an emerging and popular architectural style for building interfaces between clients and servers. Some of key principles of REST include:

1. Full specification of the resource being manipulated by the client
2. Each manipulation (access, modification, and deletion) be complete
3. The interface be self-describing and platform-independent

There are number of surveys on the Internet on REST and a detailed discussion of the REST is beyond the scope of this document. REST-style was used in the design of HTTP protocol but more recently it has been applied to APIs. The authentication mechanism for the REST API is provided by the OAuth2.0 protocol. The access token is used as HTTP authorization mechanism. The HTTP binding is specified by the OAuth2.0 protocol.

REST API is recommended for data sharing. The table below shows some sample scopes that can be served by the API, and in the following figure we show sample JSON output of an API request. It is recommended that the authorizer support standardized templates for various verticals and usecases, and publish them to enable quick application development. In case the resident requests anonymized data sharing, then the authorizer must use mathematically sound algorithms for masking and anonymizing.

Table 4 Example scopes and semantics

Scope	Example information
lpg	LPG consumer information
address	Full address
pii	Name, phone number and email address

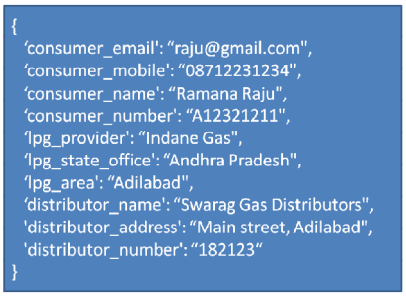

Figure 6 Example JSON response from the server

4 REFERENCES

[1] Facebook Inc. "Authentication," Available at http://developers.facebook.com/docs/authentication/

[2] Google Inc. "Authentication and Authorization to Google APIs," Available at http://code.google.com/apis/accounts/docs/OAuth2.html

[3] Twitter Inc., "Authentication & Authorization," Developer documentation. Available at https://dev.twitter.com/docs/auth

[4] Wikipedia, "Representational State Transfer," Available at http://en.wikipedia.org/wiki/Representational_state_transfer

[5] Hammer-Lahev et al "The OAuth 2.0 Authorization Protocol," Draft available at http://tools.ietf.org/html/draft-ietf-oauth-v2-22

[6] UIDAI, "Aadhaar Authentication Version 1.5 (rev 1)", Available at http://uidai.gov.in