AUA Design and Implementation

CONTENTS

LIST OF FIGURES

LIST OF TABLES

DOCUMENT HISTORY

Version & Date	Changes	Change-makers
0.1, Dec 17	Initial version	Dr. Venkata Pingali
0.2, Dec 26	Added Sub-AUA/ASA discussion	Dr. Venkata Pingali

ABSTRACT

This document discusses issues in design and implementation of AUAs.

1 OVERVIEW

This document draws upon an existing prototype implementation of an AUA server [2] to discuss design and operation of an AUA.

In Section 2 we discuss configurations and message flows. In Section 3 we discuss design of an hypothetical AUA server. In Section 4 we discuss implementation issues.

This document assumes familiarity with UIDAI system and the authentication architecture.

2 AUTHENTICATION SYSTEM

The UIDAI Authentication Specification v1.5 Rev 1[1] identifies various entities. They include:

1. Terminal: The entity that interacts with residents to collect and package the raw data
2. AUA server: The entity that signs the raw data and forwards to the UIDAI authentication server.
3. Sub-AUA server: The entity that routes the requests from the terminal to the AUA server
4. UIDAI authentication server: The centralized entity that performs the actual verification of the data provided by the resident.
5. ASA server: The entity that routes requests from the AUA server to the UIDAI authentication server.

There are multiple ways of configuring the entire system. In the simplest possible configuration, there are no routing entities. The terminals collect, package and send the authentication request data to the AUA which adds the appropriate headers, and forwards it to the UIDAI authentication server. The response flows back along the same path in the reverse direction.

Figure 1 Simplest possible configuration

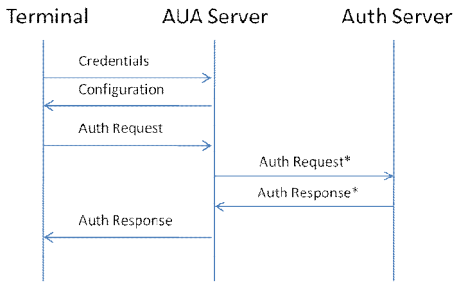

Figure 2 Message flow in a simple configuration

There may be situations where the terminals are not under direct administrative control of the AUA. In such cases a routing entity, called Sub-AUA, is required to mediate route requests back and forth between the AUA and the Terminal. Similarly AUA may not have the resources or requirement to connect directly to UIDAI. In such cases another set of routing entities, called ASA, mediate between AUA and the UIDAI. The relationship between these entities is many-many for reliability, cost, and other reasons. A complex configuration and corresponding message flows are shown in the two figures below.

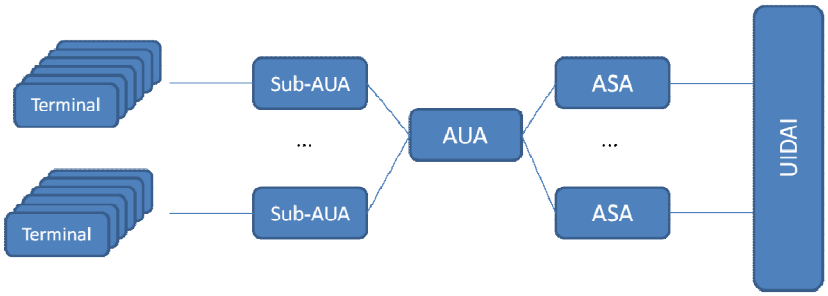

Figure 3 Complex configuration

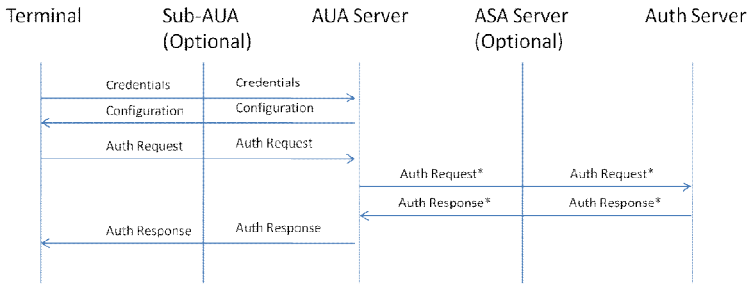

Figure 4 Message flow in a complex configuration

3 AUA SYSTEM ARCHITECTURE

The AUA server in this model has three basic modules – **Forwarder** that is responsible for the data path, **Manager** that is responsible for configuration of the forwarder, and **Accounting & Billing** that handles the business aspects of the requests. We discuss each of them in more detail below.

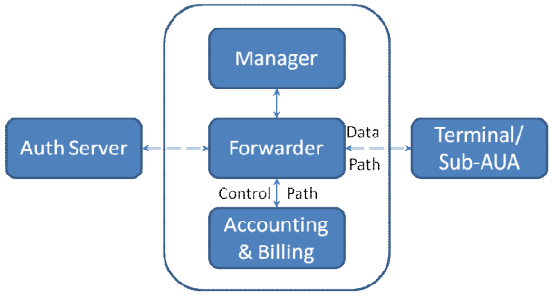

Figure 5 AUA server architecture

3.1.1 Forwarder

This is the core module in an AUA implementation. The forwarder module handles the protocol messages between the terminal and the UIDAI Auth Server. It has several components including:

1. AUA Client Protocol Module: Although the protocol between the AUA server and the UID Auth server is defined, the protocol to transfer data between AUA server and its client (terminal and/or Sub-AUA) is unspecified. It can vary with individual deployment and depends on a variety of the factors including the degree of control over the devices, classes of authentications, threat perceptions, and accounting. An implementation of the forwarder must also define the AUA-client protocol. It could be as simple as a stripped down version of the Auth server.

2. Core Forwarding Module: This module receives requests from the terminals, wraps them with the appropriate headers and forwards them to the UIDAI authentication server. It handles all the protocol messages according to a set of rules specified by the manager. The rules could be as simple as forward any and every incoming message. But in a more complex QoS-driven operation, there may be resource specifications, priorities, and fraud management rules.

3. Async Module: The default forwarding server implements nested forwarding , i.e., each request from the Terminal triggers an inline request to UIDAI authentication server. The AUA may choose to implement a message-passing forwarder, i.e., a forwarder in which interaction with the Terminal is decoupled from the interaction with the server. The coordination of this process is provided by the Async Module. The Terminal could send a request to the AUA server, in the form of, say, an email. The request is stored in persistent memory and processed at some later point in time. The storage, scheduling, and tracking of asynchronous requests is implemented by this Async module.

4. Configuration Module: This module interacts with the manager to receive the configuration, and use that to correctly and securely forward the requests and responses. It also provided business information for accounting purposes to the Accounting & Billing Module.

3.1.2 Manager

This module ensures correct, secure, and reliable operation of the Forwarder. In many ways, this is the most critical module because scaling the forwarder with time and keeping it operational in the presence of inevitable attacks is non-trivial. We elaborate on the operational challenges later. The Manager has several components including:

1. Terminal Management Module: This module interacts with the terminals to provide them appropriate configuration to enable forwarding. Terminals may belong to several classes ranging from PoS devices to web applications. The nature of the configuration information provided and the protocol using which this information is provided must be defined by the individual AUAs. Example configuration includes the forwarder IP address and port number, certificate, and a summary of capabilities of the forwarding server. A simple implementation of the AUA-Terminal protocol is a website at which the terminal operators can register and download the configuration.

2. Forwarder Management Module: At any point in time, there could be multiple, potentially geographically distributed forwarders, multiple authentication server backends, multiple paths with different network characteristics, Terminals of various kinds, and QoS specifications. The rules These must be configured to receive requests from the authorized terminals, redirect requests to specified backup forwarders, and collect information for performance, correctness, and billing reasons. When forwarders fail or under-perform for any reason, this module will ensure that appropriate recovery actions are initiated. They could be as simple as sending an SMS to complex self-healing techniques.

3. Fraud Management Module: The manager must constantly monitor the flow information to identify potential cases of fraud. For this, it may have to interface with external modules such as Weka to implement the required processing and workflows.

4. Administration Interface Module: The manager provides an administrative interface through which the administrator can monitor and modify the forwarder operation. Given the volumes of the authentication are expected to be growing and high, the forwarder may have to support auto-scaling property to ensure smooth handling of the load. Given the security requirements of AUA, the administration module must implement access control and authorization modules, possibly based on Aadhaar authentication itself.

3.1.3 Accounting and Billing

The accounting and billing function must interface with the forwarder to extract information regarding the usage of the AUA server by various terminals. This component will have several modules:

1. Core Billing: This is a standard billing module computes the usage for each Terminal and user.

2. Resource Optimization Model: This module models the resource consumption for various configurations and requests, and continuously optimizes the two to ensure high efficiency in operation.

3. Predictive Analytics Module: This module will model the incoming traffic to determine short and long-term trends that feed into AUA.

4. Audit Module: This module will track the entire lifecycle of ongoing requests, and possibly investigate requests in the past, for technical correctness and other reasons. This is critical to addressing disputes in authentication and billing.

4 IMPLEMENTATION

The figure below shows example implementation of the AUA server. In this section, we discuss implementation choices for the the application and the infrastructural elements of the AUA server.

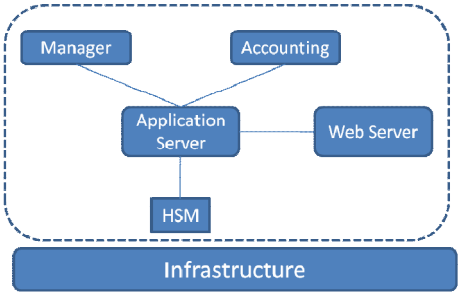

Figure 6 Implementation overview

4.1 Core Application

Forwarder is the core application in the system. The application server provides a simple REST API over SSL Its basic structure is shown in the figure below. We assume a simple nested-call design. See previous section for elaboration of this design choice.

4.1.1 Application Platform

There are a number of considerations for picking the platform for implementing the forwarder. They include:

1. XMLSec support: The platform support easy XML handling and the security extensions for the XML

2. Threading support and distributed implementation: The system requires highly parallel handling of the requests within a single process, across processes, and hosts. The platform must support decentralized implementation

3. Rapid evolution: The system will rapidly evolve as the ecosystem develops and gathers pace. So the application platform must rapidly extensible.

Table 1 Implementation platforms for Forwarder

Language	Example platform

Java	Tomcat + Apache Java XMLSec
Python	Django + pyXMLSec
Ruby	Rails + xmlsec-ruby

4.1.2 Application Configuration

The AUA must download UIDAI authentication server public key from UIDAI website [3] and distribute it to the Terminals using AUA-specific Terminal-AUA protocol. This is used to encrypt the data sent to the UIDAI authentication server.

The AUA must also acquire a valid class 2 or class 3 Certificate from appropriate Certification Authorities (CAs).The set of acceptable CAs is determined by AUA's internal legal team and IT security team together. The selection must ensure compliance with their internal policies, e.g., a bank may have decided only to use class III only, and various IT Acts.

Secure storage of the acquired private keys is critical. The private keys are typically stored in a hardware security module (HSM) that is a tamper-proof system for storing the keys. The HSM does not expose the private key. It provides an API to stream data through the system along with appropriate command such sign and verify. A popular HSM is Safenet Inc.'s Luna-SA [4]. HSM is used to sign the incoming requests before being sent to the backend. The integration of HSM with the application server depends on the API provided by individual HSM. Two key consideration in selecting the keys and HSM are throughput and key length. There is an inverse relationship between the two. The cost of the HSM increases non-linearly with the throughput.

4.1.3 Handling Requests

The application server goes through the following steps for each incoming requests

1. Receive connection
2. Check source and integrity of the Terminal-AUA message
3. Extract and check for PidXML data integrity[1]
4. Wrap PidXML into AuthXML
5. Send content to HSM for signature

6. HTTP POST HSM response to the authentication server
7. Validate and extract the UID authentication server response
8. Generate response to Terminal
9. Close connection

4.2 AUA Infrastructure

4.2.1 System Organization

In the figures below we show increasingly complex configurations of an AUA. The simplest possible cnfiguration has a single machine that implements all the components discussed above. It connects to a Internet using a single, possibly shared, router and link.

Figure 7 Simplest possible AUA

As load increases and more forwarders are deployed, we need a horizontally scalable design with arbitrary number of forwarders with an application-level load balancer to distribute the requests across them. The AUA should implement REST service model or asynchronous communication model in order to reduce/eliminate coupling between requests. Simple relatively-stateless load balancer could then work to distribute the load across the multiple instances of the forwarder.

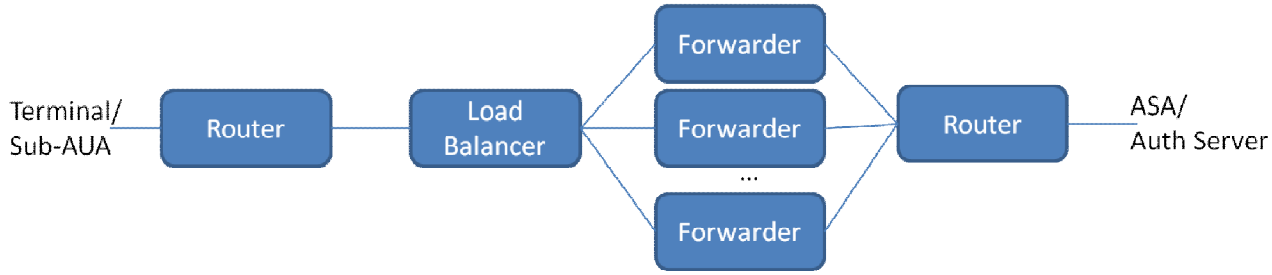

Figure 8 Horizontally scalable AUA

A high availability deployment of the AUA is shown below. It has redundant incoming Internet links, routers and load balancers. This will be required as and when Aadhaar service becomes integrated into various IT systems in the country.

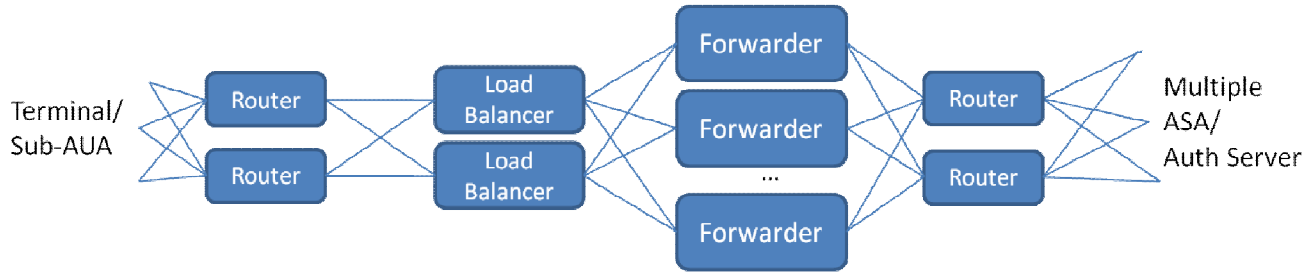

Figure 9 Scalable fully-redundant AUA server

4.2.2 Operation

Aadhaar authentication, if popular, will be on the critical path of national transactions. Although the application is relatively simple, the operational requirements are expected to be stringent. Some of the requirements include:

1. 24x7 availability to support all possible usecases
2. Security
3. Rapid scalability to support bursts of load at short notice
4. Rapid evolvability to keep up with the evolving requirements and specifications
5. Analytics-driven to keep the operation efficient

A variety of tools, services, and approaches will be required to achieve this.

Table 2 Various requirements and approaches to handle them

Requirement	Approach
Availability	Redundant processes, servers, links Robust configuration management using tools such as CA or custom Continuous monitoring platforms – internal and external

	along with rapid response teams
Security	Hardened OS and application platforms
	Secure SDLC processes and implementation of best practices
	Certificate management and use platforms such as HSMs
Rapid scalability	Load-driven auto-scaling support – custom built, load-balancer-driven
Rapid evolvability	Continuous integration processes
Analytics	Extensive instrumentation of the server
	BI platforms to generate reports and feed dashboards

5 ACKNOWLEDGEMENTS

TCS and Prof. Harrick Vin supported the development of this document. Dr. Pramod Varma and Dr. Viral Shah commented on the document.

6 REFERENCES

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

[2] Venkata Pingali, “pyAadhaaAuth,” Available at http://github.com/pingali/pyAadhaarAuth

[3] UIDAI, “UIDAI Production Certificates,” Available online at https://developer.uidai.gov.in/site/auth_cert_details

[4] Safenet Inc., “Luna-SA: Hardware Security Module,” Product sheet available online at http://www.safenet-inc.com/products/data-protection/hardware-security-modules/luna-sa/