Secure Data Management - Public Health Record system

R. Gennaro, B. David, P. Svélecz, R. van de Haterd

Table of Contents

UNIVERSITY OF TWENTE.

01
Project requirements
Functional and Security

02
Solution overview
An “airplane view” of the proposed solution

03
System architecture
System design from a high-level perspective

04
Implementation
Delving into the “nitty-gritty” details

05
Demo
How everything works in practice

06
Limitations
Known boundaries of the system

01

Project Requirements

Functional Requirements

Active Users

Users with write access who constantly interact with the PHR

Passive Users

Users with read-only access who only observe the PHR

Authorities

Entities responsible for issuing keys

UNIVERSITY
OF TWENTE.

Active Users

Patients

Have complete control over their own health record

Doctors

Read patient data and write health data in the PHR

Trainers

Read patient data and write training data in the PHR

Passive Users

Insurance representatives

Employer representatives

Can be granted read access, but not write.

UNIVERSITY OF TWENTE.

Authorities

Hospitals

Health Clubs

Insurance companies

Employers

UNIVERSITY
OF TWENTE.

Security Requirements

Data encryption

Health-related data is highly confidential → has to be stored encrypted

Access Control

Make sure parties can only perform actions that they are entitled to

Authorization

Authorize users before performing actions

UNIVERSITY
OF TWENTE.

02

Solution Overview

Solution Overview

PHR System

- **Access Control** — Cryptographically enforced via MA-CP-ABE
- **Encryption Scheme** — AES Encryption with key encrypted with MA-CP-ABE
- **Authorization** — Cryptographic Challenge-based authorization
- **Client - Server Model** — Central server with aggregated data, and client with cache database

03

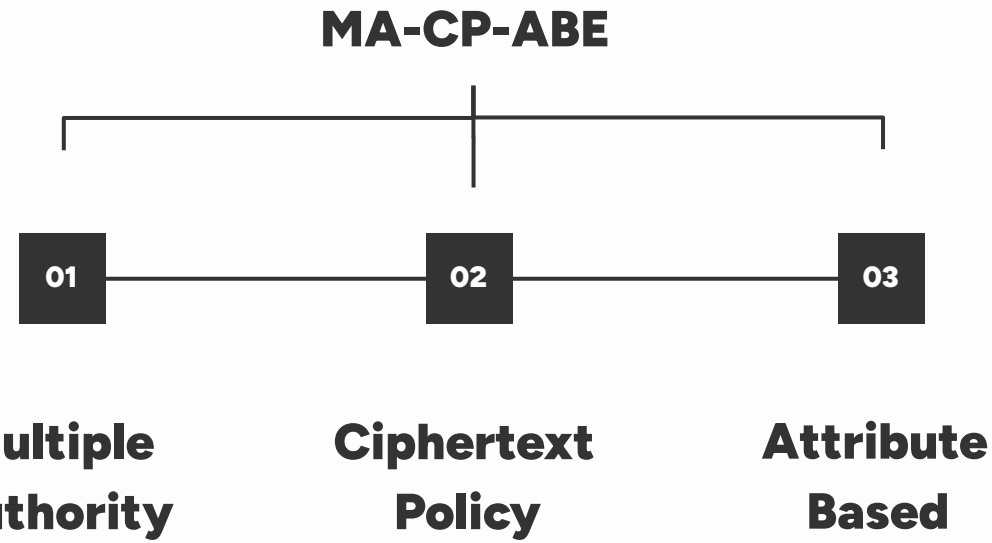
Software Architecture

System Architecture -
Client - Server Model

Client

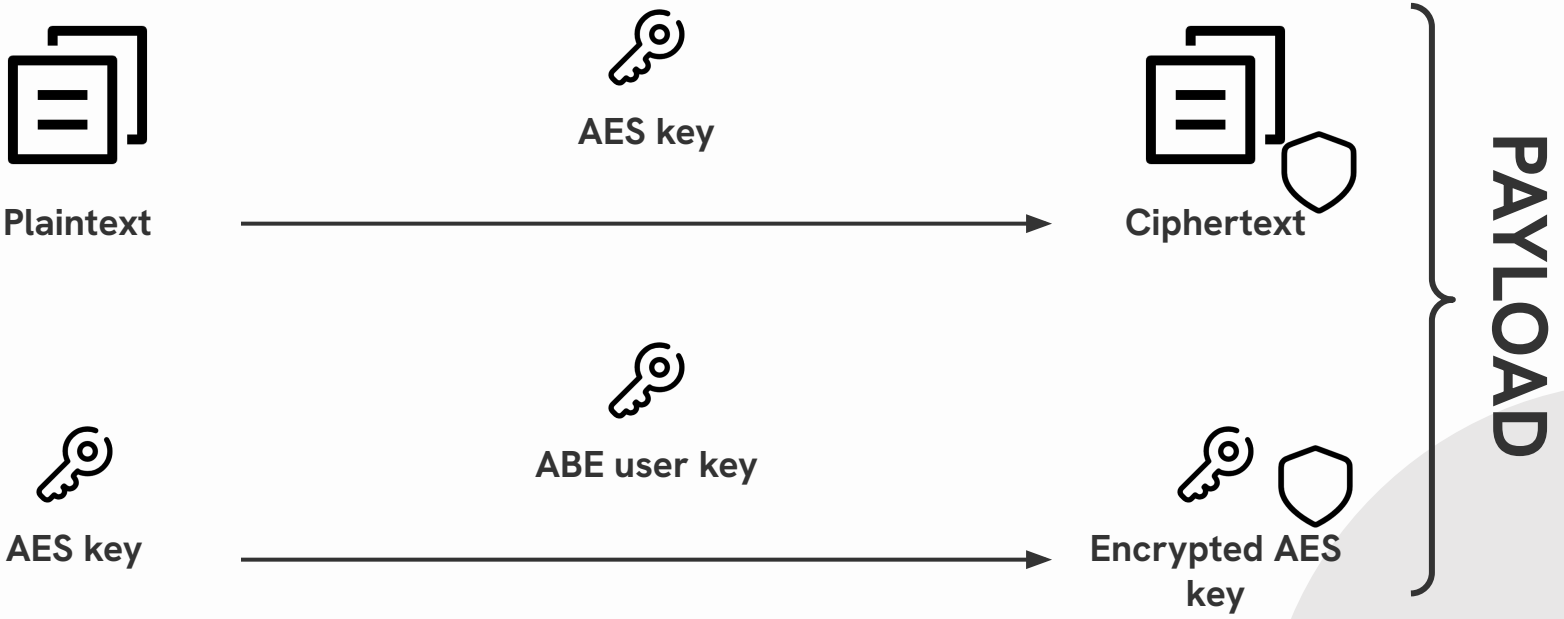
Server

System Architecture -
Access Control



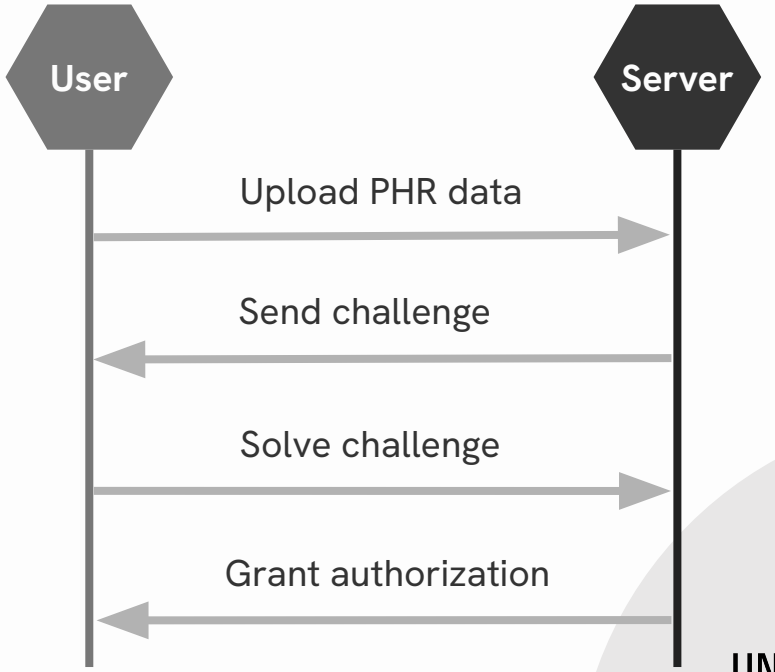
UNIVERSITY OF TWENTE.

System Architecture -
A Hybrid Approach



System architecture -
Authorization

- MA-CP-ABE does not distinguish R/W type of access
- Challenge-Response protocol



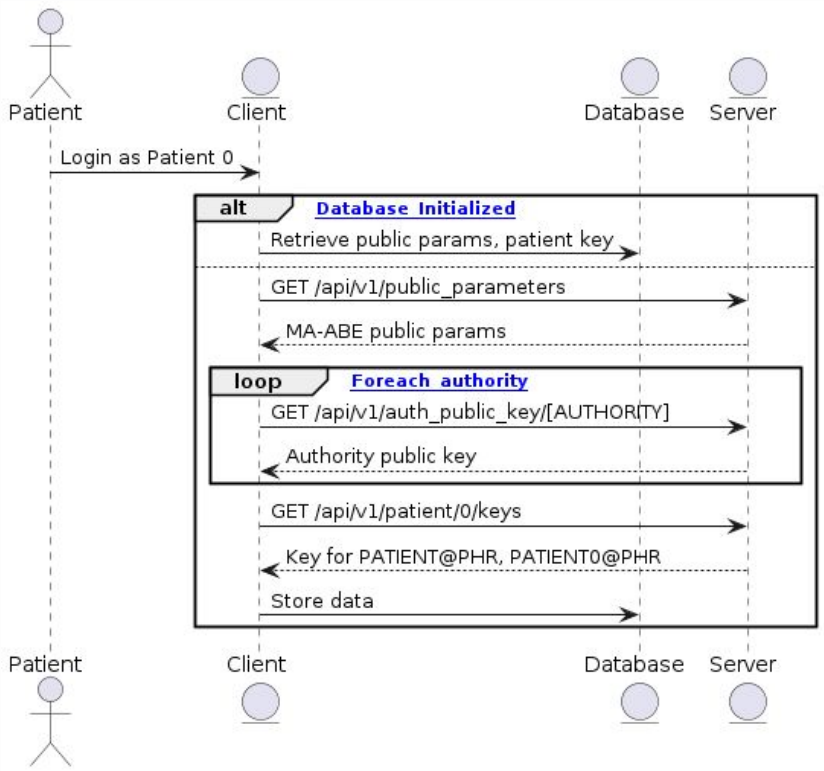
04

Implementation

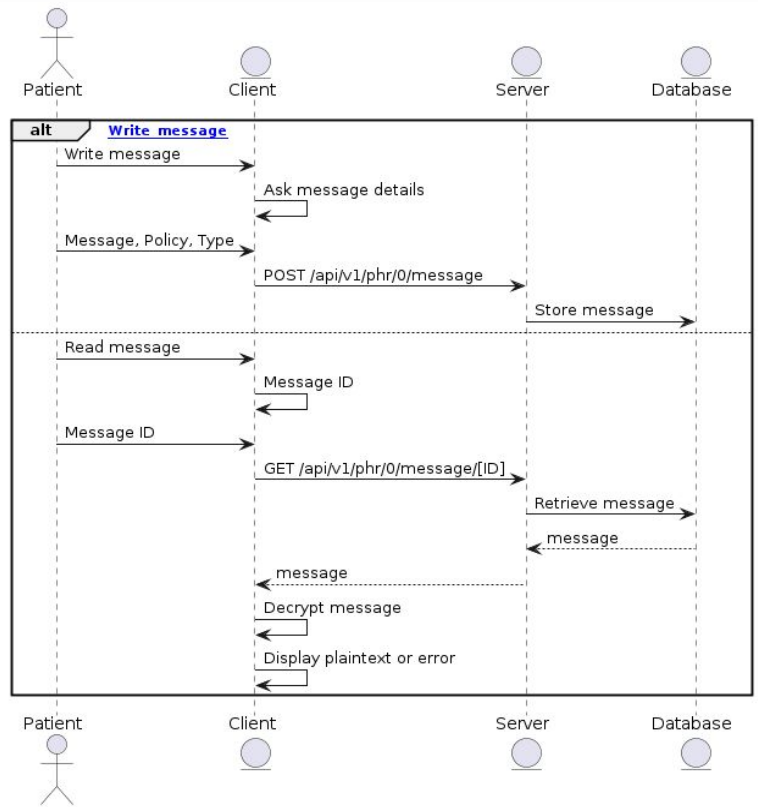
Implementation -
Client and Server model

- Server implemented using Django
- Server provides REST API endpoints
- SQLite DB on both sides

Implementation - Initial Setup



Implementation - Patient Communication



UNIVERSITY OF TWENTE.

Implementation -
Access Control

- Charm framework
- [abenc_maabe_rw15] module

Attribute structure
ATTRIBUTE@AUTHORITY

Policy structure
PATIENT@PHR OR
DOCTOR@HOSTPITAL

UNIVERSITY
OF TWENTE.

Implementation -
Encryption scheme

- Messages, keys stored in DB
- MA-CP-ABE extended with AES

Implementation -
Authorization

Generation
- Server

1. Generate challenge
2. Encrypt with policy
3. Store correct result
4. Send to User

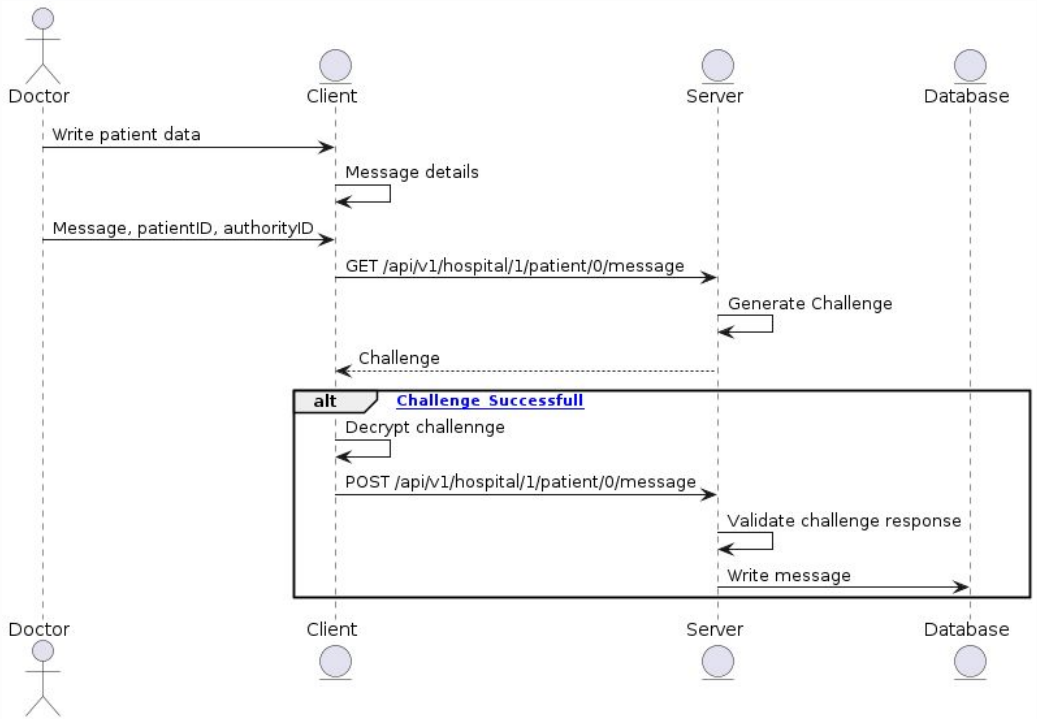
Solving
- User

1. Receive challenge
2. Encrypt with policy
3. Return to Server

Verification
- Server

1. Receive response
2. Check response with saved solution
3. Grant or deny access

Implementation - Authorization



UNIVERSITY OF TWENTE.

Implementation -
Data models

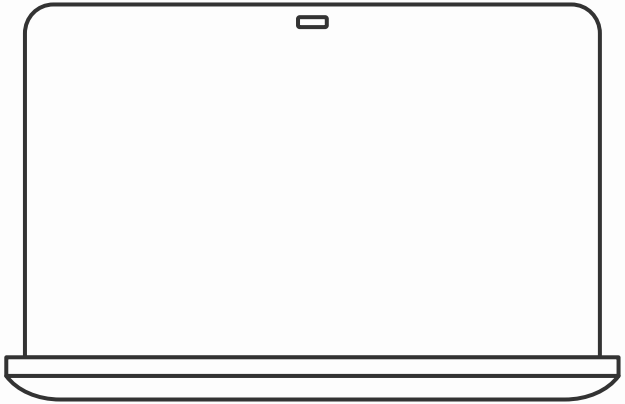
- **Authority** models
- **Key** models: for Authority public & secret keys
- **Authority representative** models: reps. on behalf of Authorities
- **Patient** models
- **Encryption** models: MA-ABE public params. & encrypted AES keys
- **Message** models: health or training related data

- **PatientRep**: relationship between patients and reps.

05

Project demo

Demonstration



UNIVERSITY
OF TWENTE.

06

Limitations

Current Limitations -
Overview

- Possible MitM attack against challenges
- Unique AES keys per message
- Client-side ABE key stored in plaintext
- No key revocation
- Rudimentary UI

Q&A