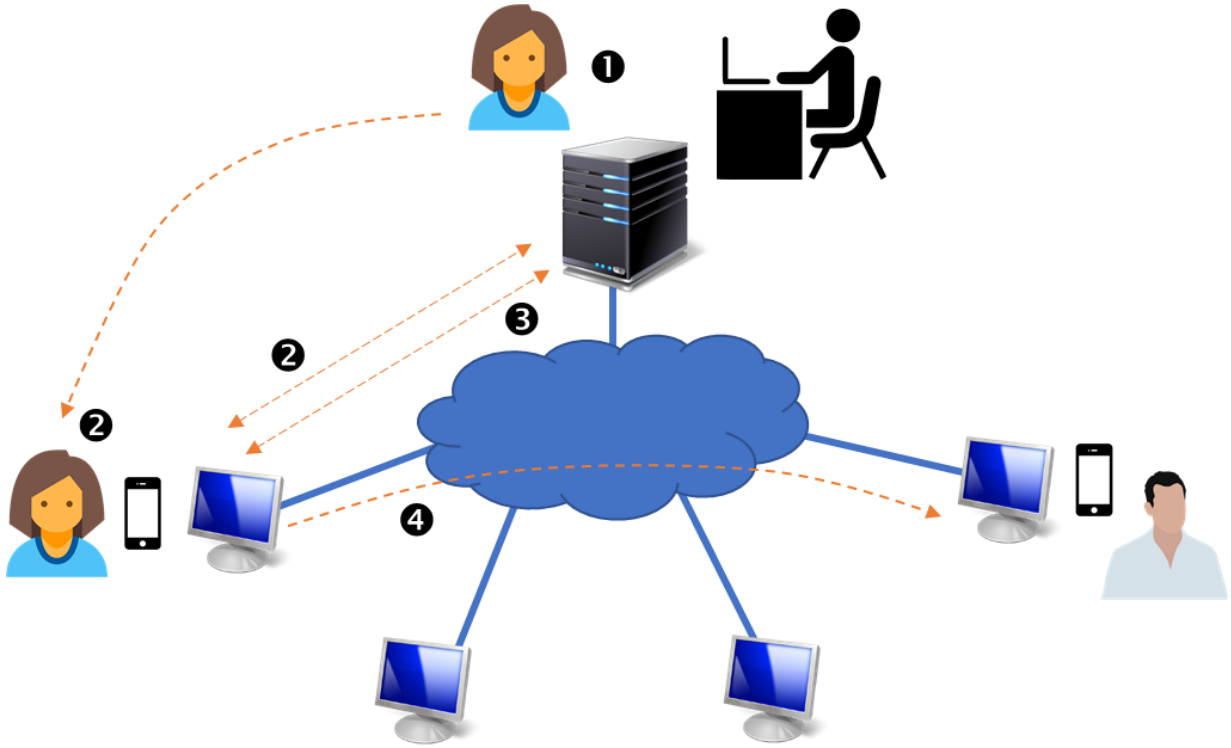
# Practical Security Assignment

## 1. Scenario

An organization intends to design and implement a system with a trusted server supplying services to client applications, used by their collaborators.




The services are REST endpoints that can be called by the client applications. These services have an associated security level (an integer value). Also, collaborators can send messages to each other using the client application. The client applications should be mobile, but, for a proof of concept, they can run on a PC with a CLI (command line interface) or GUI (graphic user interface) interface.

## 2. Requirements

**A.** Collaborators need to perform a **face-to-face pre-registration** (1), where their identity (full name, and other possible and convenient information) is verified and stored in a file or database controlled by the server. At the same time, a username (string with a maximum of 8 characters) and a one-time ID (a random string with 12 characters, comprised of small and capital letters and digits) is generated and communicated to the collaborator. The server also stores the security clearance level (another integer) of the collaborator.

**B.** The collaborator should then install the client application and perform the **registration operation** with the server (using the username and the one-time ID) (2). In this registration, the client application and server should generate/store the necessary cryptographic means to perform an **automatic authentication using asymmetric cryptography** of the client with the server for each session. That instance of the client after registration always represents the same collaborator, from that point on. The client application is **responsible for identifying and authenticating the collaborator locally**, and only authenticate with the server after this.

**C. The client can invoke any service available on the server** (3), but authorization is granted only is the security clearance of the collaborator is greater or equal to the security level of the service (this is a simplified version of a *mandatory access control*). For the purpose of a proof of concept consider three different services (for instance, the calculation of a square root, a cubic root, and a parameterized n-root, where the provided value parameters and the parameter *n* are from a *double float type*). Each of these services has a different security level (1, 2, or 3). Moreover, consider at least three users having 3 different security clearances (also 1, 2, or 3).

**D. A collaborator can send a message to another collaborator** (4) using appropriate services provided by the server at security level 1 and enabling such communication. The message should *travel* directly from the client application of the sender to the client application of the recipient, when he is on-line. Client applications are listening on a port for those messages. From session to session, the listening IP address and port can change (note that this is what typically happens in mobile applications).

**E. All information** that circulates on the network (even the initial registration**) between clients and the server must have guaranties of confidentiality and integrity**. Also, **each endpoint** in a communication **must be identified and authenticated**, and message sending must have a **protection against replay**. Each message received must have the means to **verify authorship and non-repudiation** too.

**F. The messages must be stored** in the recipient computer or device, **in confidential form**, with proper means to recuperate the clear text, **detect changes, and proof of authorship and non-repudiation**. The recipient collaborator should be able to display the message whenever he wants.

**G. Sensitive data** must be **protected wherever is stored**.

## 3. Design and implementation

Your team should design the software components needed for the server and the client application, coupled with the security mechanisms to satisfy the requirements outlined above. For a proof-of-concept all components can execute on the same computer (a different instance for each client application representing a different collaborator).

The **main threat** is the possibility of eavesdropping, or modifying, or fabricate all the network traffic by an intruder. A less probable threat is an intruder gaining physical access to a computer or device where the client application can execute. Both of these threats need to be considered.

You can use any technology and programming language that you are comfortable with, but a recommendation is node.js for the services, and any client technology (using C/C++, C#, Java, Javascript, Python, Ruby, …). For the cryptographic implementations you should be aware that the most used and tested implementation is perhaps the OpenSSL library, that is available through the correct bindings from almost all programming languages and technologies. The library itself is written in C. (There are others, e.g., in Java) that are considered very reliable – external reliable library procurement is also a job of security developers and teams.

## 4. Report

At the end an assignment report must describe your design and implementation and your arguments (including tests) that the requirements are fulfilled, even in the presence of adversaries. All cryptographic operations, their parameters and strength should be indicated, as a description of all protocols used in the communications. It should include illustrations of the use cases (both normal functionalities and responses in case of non-authorized communications). You can also use any testing tools that you consider adequate, presenting their results.