Sara

Slide 1:
Good morning, my name is Sara, and this are my colleagues Rafael and Ricardo. Today we will present our work.

Slide 2:
Our system is a Hospital like system that allow to gather and store sensitive information from patients. This information is compartmentalized in different records which are stored in the Hospital Server. Hospital Employees can access and manipulate this information based on their Role. There are also Partner labs that communicate with the Hospital Server to provide test results, these tests results are stored in a way that allow to check their authenticity whenever necessary. To guarantee that the information is only accessed by certain roles an externalized entity called Policy Authoring ensures that the request made to the Hospital server is valid, giving or denying permission for it to be executed. The system also has two modes, Regular and Pandemic, these can affect the information the roles are allowed to access.

Slide 3:
As shown, our system consists of 5 entities, 2 of them, the Employees and the Partner Labs constitute the users. The users authenticate themselves with the Hospital Server to interact with the system, however, the Employees do not need their own certificate for this interaction. On the other hand, the Partner Labs do because of the way the test results they send are kept.
Regarding the Hospital Server's database, implemented in Postgres, it is used to store these user's credentials, along with patient information and records.
As it will be elaborated later, the Policy Authority is reached by the Hospital Server in order to validate users' requests based on the defined policies.
We assume that none of the channels between entities, implemented in gRPC, are secure and therefore try to ensure Confidentiality, Integrity, Authenticity, Freshness and Forward Secrecy while also allowing for Employees and Partner Labs to authenticate themselves with the Hospital Server.
Also, all entities except the CA, which is our physical machine, run on a different VM as they would in a real-life scenario.

Rafael

Slide 4:
In order for the hospital server to decide if allows or denies access to records, every request related with records are passed though the Policy Authoring.
To verify if the request can be made, the Policy Authoring will check its Policies.
The policies are written in XACML and are using role based accessing control in order to have better efficiency.
This diagram shows how our policies are organized, so we have a policy set MTR with references to 2 policies sets, one for the regular mode and one for the pandemic mode, and each of this sets have references to policies that are created by role, when a role access permissions changes between modes the references in each mode are different for example the porter, when they don't change, both modes point to the same reference.
This is an example of a policy, it has a target and rules, the rules are only checked if the target is valid, so on the target we have the role of the user, in this case clinical assistant.

Slide 5:
Each node that needs a pair of keys and certificate signing requests, manually generates them.
The certificates signing requests are signed by the CA, which is a trusted entity, and this is also made manually.
Between the Hospital Server and The Employee and between the Hospital Server and the Policy Authoring it is used TLS.
For the communication between the partner lab and the Hospital Server we use our custom security protocol, where the partner lab sends its certificate, and the hospital server sends its access token and its certificate along with a timestamp, nonce and Signature.
After this, they use Diffie-Helman to establish a secret key that they both use.
The messages of the Diffie-Helman also send a timestamp a nonce and a signature in order to prevent man in the middle attacks and guarantee freshness and integrity during the handshake.

Ricardo

Slide 6:
The Communications between the Employee and the Hospital Server, and the Hospital Server and the Policy Authoring use TLS in order to guarantee its messages Confidentiality, Integrity and Freshness. On the other hand, the Communication between the Partner Lab and the Hospital Server is made Secure by using our Custom Security Protocol, in which:

- All messages are encrypted using the Secret Key established in the Key Exchange in order to ensure its Confidentiality, Forward Secrecy is also ensured.
- Regarding the message's Freshness, a Timestamp and a Nonce are also sent along. Since Timestamps older than 30 seconds are discarded, all Nonces older than that, can be erased in order to save memory and improve performance.
- Moreover, an HMAC is also made, in order to guarantee the message's Integrity.

Authentication of the Hospital Server Users is ensured by enforcing them to Login before any other Request could be performed. To do so, each of them has a Username and a Password which Hash, generated by a Key derivation Function, and Salt are stored in the Hospital Server Database along with its associated Role, which allows the Policy Authoring to decide whether the User has permissions or not, to access a record. Our system has also a *"Throttler"* detector, which prevents a Hacker of making a Brute-Force attack, in order to guess a User's Password. This is achieved by kindly declining all Login attempts, following three failed ones, in a 5-minute period. After the User is Authenticated, its Access Token is activated, making the User able of performing other requests.

It is also important to note, that all Test Results are sent along with its Signature, which can be verified later by the Hospital Server, since it already has the Partner Lab Certificate, which was exchanged between them in the Handshake.