TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Sistema de autenticación de doble factor basado en Criptografía de Curva Elíptica

Autor
Sergio García Prados

Director
Prof. Dr. Antonio Francisco Díaz García

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, 20 de junio de 2021

Sistema de autenticación de doble factor basado en Criptografía de Curva Elíptica

Sergio García Prados

Palabras clave: Criptografía, Sistemas operativos, C, Pluggable Authentication Module, Autenticación, Message Queuing Telemetry Transport, Seguridad, Criptografía de Curva Elíptica, Algoritmo de Firma Digital de Curva Elíptica, Computación del Altas Prestaciones, Nube

Resumen

Cada vez está más en boca el término "seguridad". No es de extrañar que desde que comenzó la era digital, los retos tecnológicos son cada vez más complejos. No obstante, dado el grado de conocimiento y herramienta de las que disponemos, estos son bastante factibles y llamativos para cualquiera. Es por eso que el uso de elementos tecnológicos en nuestro día a día se concibe como algo innato en nuestras vidas.

La tecnología nos trae numeros beneficios que vale simplemente con echar la vista atrás y ver como la sociedad ha llegado a conseguir retos tan importantes gracias a la ayuda de esta. Sin embargo, la tecnología tiene su lado adverso y es la seguridad. Por muy llamativa y útil que pueda ser una tecnología, si no se lleva a cabo unos cumplimientos de seguridad bien definidos y estrictos durante su desarrollo, el usuario final puede no llegar a usarlo y por tanto quedar en vano el trabajo realizado.

Gracias a la era del "Big Data" en la que vivmos, tenemos acceso a una ingesta cantidad de information al instante la cual se estudia y se llevan a cabo análisis complejos sobre la misma. Esta característica asociada a la seguridad hace que el conocimiento acerca de las posibles vulnerabilidades sea un campo que nunca parace acabar. Es por eso que todo sistema no es 100 % seguro.

El objetivo de este trabajo es el estudio de la seguridad en entornos definidos (HPC y Cloud) donde el volumen de datos es elevado y por consiguiente el almacenamiento y manejo del mismo se tiene que hacer desarrollar de forma segura y garantizando las bases de todo sistema seguro: confidencialidad, integridad y disponibilidad. Se propone una solución basada en un sistema electrónico que garantiza una autenticación a estos entornos de forma más segura.

Este estudio se centra principalmente en la incorporación de un modulo de seguridad a este sistema de forma que pueda ser usado por cualquier otro dispositivo en entornos variados garantizando una solución portable y eficiente.

Double authentication factor system based on Elliptic Curve Criptography

Sergio García Prados

Keywords: Criptography, Operative System, C, Pluggable Authentication Module, Authentication, Message Queuing Telemetry Transport, Security, Elliptic Curve Criptography, Elliptic Curve Digital Signature Algorithm, High Performance Computing, Cloud

Abstract

The term "security" is becoming more and more popular. It is not surprising that since the digital era began, technological challenges have become increasingly complex. However, given the degree of knowledge and tools at our disposal, they are quite feasible and appealing to anyone. That is why the use of technological elements in our day-to-day life is conceived as something innate in our lives.

Technology brings us so many benefits that it is worth just looking back and seeing how society has achieved such important challenges thanks to its help. However, technology has its downside and that is security. No matter how flashy and useful a technology may be, if strict and well-defined security standards are not followed during its development, the end user may not be able to use it and therefore the work done may be in vain.

Thanks to the "Big Data" era in which we live, we have access to an instantaneous intake of information which is studied and complex analyses are carried out on it. This characteristic associated with security means that knowledge about possible vulnerabilities is a field that never seems to end. That is why every system is not 100 % secure.

The objective of this work is the study of security in defined environments (HPC and Cloud) where the volume of data is high and therefore the storage and management of the same must be developed in a secure way and guaranteeing the bases of any secure system: confidentiality, integrity and availability. We propose a solution based on an electronic system that guarantees a more secure authentication to these environments.

This study is mainly focused on the incorporation of a security module to this system so that it can be used by any other device in different environments, guaranteeing a portable and efficient solution.

Yo, **Sergio García Prados**, alumno de la titulación Grado en Ingeniería Informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 77148519X, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Sergio García Prados

Granada a 20 de junio de 2021

D. **Prof. Dr. Antonio Francisco Díaz García**, Profesor del Departamento de Arquitectura y Tecnoloía de Computadores de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Sistema de autenticación de doble factor basado en Criptografía de Curva Elíptica***, ha sido realizado bajo su supervisión por **Sergio García Prados**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 20 de junio de 2021.

El director:

Prof. Dr. Antonio Francisco Díaz García

Agradecimientos

A mi director de Trabajo Fin de Grado, Prof. Dr. Antonio Francisco Díaz García por su coordinación y conocimiento acerca de la materia que me ha ayudado a desarrollar este trabajo a pesar de los inconvenientes.

Tambien me gustaría agradecer este trabajo al apoyo de mi madre y hermanos que han confiado en mi en todo momento y me han dado soporte en los momentos mas dificiles de mi recorrido académico. Pero sobre todo a mi padre, que ha sido mi apoyo incondicional en estos años de carrera y del que seguro que estaría orgulloso de ver lo lejos que he llegado.

¡Gracias!

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

El concepto de "Seguridad de la Información" se acuño por primera vez mencionado por el NIST en 1997 [1] aunque la seguridad en el campo de la tecnología siempre ha estado en boca de todos y ha sido un apsecto a tener en cuenta.

La seguridad en el campo de la tecnología para un campo que se empezó a tener en cuenta con el "*boom*" de la tecnología. No obstante, habría que remontarse unos siglos atras, en concreto cuando Julio César vivía para darse cuenta que en aquella época ya se aplicaban ciertas metodologías para envíar mensajes sin que otros se enterasen. Exactamente, hace más de 2000 años pusieron en prática la técnica de comunicación entre entidades de forma segura a través de un canal inseguro, la criptografía. La criptografía ha estado latente en muchos hitos de nuestra historia, como por ejemplo con la aparición Alang Turing y su ingenio para descifrar las comunicaciones entre integrantes del ejército alemán en código morse.

La criptografía, según la Real Academia Española (RAE) es el "arte de escribir con clave secreta o de un modo enigmático". Esta herramienta ha sido la base de la seguriadad de la información ya que sustenta las bases de todo sistema serguro (CIA): confidencialidad (*confidenciality*), integridad (*integrity*) y disponibilidad (*availability*).

La seguridad es un campo de elevada atracción hacia los investigadores y es por ello que la demanda de sistemas altamente seguros es un requisito indispensable. A esto, hay que añadirle la posibilidad de tener a un clic acceso a toda información que queramos. Esto supone un mayor esfuerzo a la hora de clasificar que información es más sensible que otra y por tanto que grado de seguridad hay que aplicar.

Dada que toda la información no tiene la misma validez, cualquier sistema que envíe tráfico hacia internet con información sensible debe cumplir unos requisitos de seguridad necesarios para que no sea intercedida ni modificada. La empresas invierten cada año más en seguridad dado el incremento de ataques informáticos que suceden a diario, tal y como se puede ver en esta

página [2] y la aparición constante de nuevas vulnerabilidades, las cuales se pueden consultar [3].

1.1. Motivación

La idea de este tarbajo nace de raiz de la tesis doctoral [4] publicada en 2020 por la Universidad de Granada titulada "*Mecanismos de seguridad para Big Data basados en circuitos criptográficos*" y elaborada por *Ilia Blockin*. Esta tesis sugiere soluciones basada en sistemas electrónicos que permitan mejorar la seguridad en el acceso a sistemas donde deben procesarse un elevado volumen de datos. Los sistemas propuestos ofrecen una solución eficiente y flexible para aumentar la seguridad en el acceso a servicios y sistemas que pueden procesar gran cantidad de información.

El planteamiento se detalla en unos de las mejoras propuestas de dicha tesis: "*continuar mejorando los sistemas propuestos agregando compatibilidad con otros métodos de autenticación como el Módulo de autenticación conectable de Linux (PAM) u otros protocolos de autenticación*"

Personalmente, me he decantado por la elección de este tema ya que tengo especial interés en el campo de la ciberseguridad y lo importante que es a día de hoy en el desarrollo de cualquier sistema y producto software.

Considero que este trabajo es relevante dado el incremento de popularidad que se está dando en el campo del *Big Data* y la importancia de que los Centros de Procesamiento de Datos (CPD) que se usan estén bien protegidos y solo el personal autorizado tenga acceso a ellos.

1.2. Objetivos

Para elaborar la list de objetivos, es necesario conocer los requisitos del sistema donde se pretende implantar. Este entorno viene descrito de forma detallada en [5].

El presente trabajo conlleva el cumplimiento de los siguientes objetivos:

1. Crear un Pluggabe Authentication Module (PAM) para el sistema de autenticación propuesto en [5]

2. Implementar el módulo PAM para el servicio Secure Shell (SSH)

3. Usar el protocolo Message Queueing Transport Telemetry (MQTT)

4. Seguir esquema de autenticación *challenge-response* [6]

5. Cifrar el *challenge* mediante el algoritmo de cifrado unidireccional robusto como por ejemplo SHA512 (Secure Hash Algorithm)

6. Usar Criptografía de Curva Elíptica o Elliptic Curve Criptography (ECC) tanto para la firma del *challenge* como para la verificación del mismo usando el algoritmo Elliptic Curve Digital Signature Algorithm (ECDSA)

7. Encriptar las comunicaciones entre los elementos del sistema propuesto usando Transport Layer Security (TLS) versión 1.2

1.3. Estructura del trabajo

El presente trabajo se divide en las siguientes secciones y subsecciones: ....

1.4. Convenciones

El texto de este trabajo mantiene una serie de reglas de cumplimiento de texto:

- Los acrónimos se describen la primera vez que aparezcan en su idioma original seguido del acrónimo abreviado entre paréntesis. Posteriormente, solo se usará el acrónimo abreviado.

- Las palabras en otro idioma se detallan en cursiva.

Capítulo 2

Estado del arte

Vamos a enfocar los trabajos relacionados con respecto a la seguridad que ofrece la Criptografía de Curva Elíptica en dispositivos electrónicos para proporcionar un método de autenticación de doble factor. Los trabajos citados a continuación son comparados con el propuesto en [5]. *Braeken* propone en [7] un protocolo de autenticación para dispositivos Internet of Things (IoT) basado en una función física no clonable o Physical Unclonable Functions (PUF). Ambos sistemas se caracterizan porque usan un esquema de acuerdo de claves en el que cualquier nodo está registrado bajo la supervisión de un tercer elemento de confianza o Trusted Third Party (TTP) y el uso de certificados. Además, en ambos sistemas se usa el mecanismo de autenticación basdo en reto-respuesta o *challenge-response*. Este es un protocolo en el que una entidad se autentica enviando un valor dependiente de un valor secreto y un valor desafío cambiante [8]. Si la respuesta es correcta, se da por légitimo al cliente y se autentica. El sistema prouesto en [5] a diferencia de [7], usa el protocolo MQTT para la comunicación entre los dispositivos de tal forma que lo hace más escalable.

Gao [9] propone un sistema similar que usa un token de autenticación basado en el formato JSON, JSON Web Token (JWT), junto al servicio SSH y desarrolla, al igual que lo que se propone en el presente trabajo, un módulo PAM de tipo desafio-respuesta. Sin embargo, a diferencia de [9], este trabajo no usa un formato concreto para responder al desafio. De manera similar a [7], ninguno implementa el protocolo MQTT para la comunicación entre los nodos que intervienen.

Existen otros sistemas como por ejemplo el propuesto por *Fayad* en [10] que ratifica la robustez que tiene ECC sobre otros mecanismos de autenticacion como las contraseñas de un solo uso (OTP) basadas en mensajes *Hash* (HMACs) o basadas en tiempo (TOTP) para entornos con dispositivos IoT.

Actualmente, existen múltiples métodos de autenticación bien definidos que cumplen con los parametros requeridos: algo que sabes (contraseña), algo que tienes (token) y algo que eres (biometría). Para que sea multi factor,

al menos dos parametros tienen que ser requeridos. Google creó un módulo PAM [11] para garantizar un factor de serguridad en la autenticación usando su producto *Google Authenticator*. Por otra parte, existen productos como llaves de autenticación que verifican tu identidad usando protocolos de seguridad robustos. Es el caso por ejemplo de Yubikey [12]. A diferencia del sistema propuesto, la verificación por llave de seguridad requiere del dispositivo físico. También hay productos software en el mercado que permiten el acceso seguro a sistemas sin la necesidad de tener que proveer de la metodología clásica par usario-contraseña. El inicio de sesión único o Single Sign-On (SSO) [13] es un método de autenticación que permite a los usuarios autenticarse de forma segura en múltiples aplicaciones y servicios web usando un único conjunto de credenciales.

Capítulo 3

Análsis del problema

3.1. Seguridad en la autenticación

La autenticación del usuario es el punto de entrada a diferentes redes o instalaciones informáticas en las que se presta un conjunto de servicios a los usuarios o se puede realizar un conjunto de tareas.

Una vez autenticado, el usuario puede acceder, por ejemplo a la Intranet de una empresa, a consolas, bases de datos, edificios vehículos, etc. La usabilidad de los mecanismos de autenticación se investiga cada vez con más detenimiento y dado que los estos son concebidos, implementados, puestos en práctica y corrompidos por teceras personas, hay que tener en cuenta los factores humanos en su diseño.

Actualmente existen múltiples métodos para autenticar a un usuario contra un sistema, siendo el más común es el par de claves usuario y contraseña, pero no es el único. El uso de certificados está cada vez más extendido en Infraestructuras de Clave Pública (PKIs).

Para garantizar que un sistema es necesario que se garantizen tres aspectos [14]:

- *Confidencialidad*: prevenir la divulgación no autorizada de la información
- *Integridad*: prevenir modificaciones no autorizadas de la información
- *Disponibilidad*: prevenir interrupciones no autorizadas de los recursos informáticos

Usar un método de autenticación no implica que el sistema sea completamente seguro. De echo, la autenticación se convierte en un proceso más robusto y fiable aplicando múltiples métodos, también llamado Multiple Factor Authentication (MFA). Este mecanismo propone tres tipos de factores que permiten a un usuario vincularlo con las credenciales establecidas [15]:

1. *Factor conocimiento*: algo que el usuario conoce (contraseña)
2. *Factor pertenencia*: algo que el usuario tiene (token)
3. *Factor biometrico*: algo que el usuario es (huella dactilar)

La autenticación basada en múltiples factores provee un nivel de seguridad elevado y facilita una protección continua de dispositivos y otros servicios críticos ante accesos no autorizados usando dos o más factores.

Además de la robustez de la seguridad durante el proceso de autenticación, la usabilidad se convierte a su vez en una cuestión estratégica en el establecimiento de métodos de autenticación de usuarios.

La usabilidad puede definirse como "la medida en que un producto puede ser utilizado por determinados usuarios para alcanzar determinados objetivos con eficacia, eficiencia y satisfacción en un contexto de uso específico". La usabilidad de la seguridad se ocupa del estudio de cómo debe tratarse la información de seguridad en la interfaz de usuario y de cómo deben ser los mecanismos de seguridad y los sistemas de autenticación deben ser fáciles de usar [16].

Este sistema [5] propone un mecanismo de autenticación usable, de múltiples factores y escalable gracias principalmente al uso de dispositivos electrónicos con circuitos integrados como es Arduino.

3.1.1. Seguridad en IoT

El Internet de las cosas o Internet of Things (IoT) es un término relativamente nuevo. Se puede definir como una red abierta y completa de objetos inteligentes que tienen la capacidad de auto-organizarse, compartir información, datos y recursos, reaccionar y actuar ante situaciones y cambios en el entorno [17].

El IoT ha facilitado la interconectividad entre dispositios ayudando a conectar sensores, vehículos, hospitles, industrias y consumidores a través de Internet. Las arquitecturas en IoT son cada vez más complejas, descentralizadas y fluidas dado el incremento de dispositivos que se comunuican entre sí y es por ello que la seguridad juega un papel fundamental en él. Los dispositivos IoT deben ser seguros y no pueden ser manipulados por terceras personas no autorizadas.

3.1.2. Message Queueing Transport Telemetry (MQTT)

Message Queueing Transport Telemetry (MQTT) es un protocolo de comunicación que funciona a nivel de la capa de aplicación. Es usado principalmente en entornos con dispositivos IoT por ser un protocolo con un consumo de bando de ancha y batería mínimo [18].

A diferencia del protoclo de comunicación Hyper Text Transfer Protocol (HTTP) que funciona mediante petición-respuesta, MQTT está basado es publicación-respuesta. Este modelo de protocolo requiere de un broker mensajero. Existen múltiples tipos de brokers como Mosquitto [19], HiveMqtt [20], Mosca [21], cloudMQTT [22], MQTT.Js [23], etc. Para el trabajo propuesto, se usa Mosquitto.

3.1.3. Criptografía de Curva Elíptica

La Criptografía de Curva Elíptica (ECCs) pertenece a las tres familias de algrotimos de clave pública de gran relevancia (factorización de enteros, logaritmos discretos y curva elíptica). Este último nació entorno a mitad de los años 80. ECC provee del mismo nivel de seguridad que ivest-Shamir-Adleman (RSA) o sistemas logarítmicos discretos con operaciones considerablemente pequeñas. ECC está basado en el problema de logaritmo discreto (DLPs) [24]. ECC tiene beneficios en cuanto a su capacidad computacional (menos computaciones) y consumo de ancho de banda (claves y firmas más pequeñas) sobre RSA y DL. No obstante, las operaciones RSA que implican una clave pública más pequeña son más rápidas que ECC.

El objetivo de esta sección es explicar las bases de este algoritmo sin entrar en detalles matemáticos. Para mayor documentación se sugiere leer el libro [25] del cual se ha sacado la información.

Esta sección se distribuye de la siguiente manera: definición del concepto de curva elíptica, problema de logaritmo discreto aplicado a las curvas elípticas, algoritmo ECDSA y comparativa con RSA.

Definición de curva elíptica

De las ecuaciones polinómicas de una circunferencia y una elipse se pueden sacar diferentes tipos de curvas. Las curva elíptica es un tipo especial de ecuación polinómica. En criptografía se trabaja sobre un conjunto de números finitos, más popularmente campos primo, donde todas las operaciones aritméticas se desarrollan sobre módulo p, siendo este un número primo.

La definición de curva elíptica es la siguiente:

Definition 3.1.1 (Curva Elíptica)**.** La curva elíptica sobre $\mathbb{Z}_p$, $p > 3$, es el conjunto de pares de punto $(x,y) \in \mathbb{Z}_p$ que cumplen

$$y^2 = x^3 + ax + b \; mod \; p \tag{3.1}$$

junto a un punto infinito imaginario O, donde

$$a, b \in \mathbb{Z}_p \tag{3.2}$$

y la condición

$$4a^3 + 27b^2 \neq 0 \; mod \; p \tag{3.3}$$

Ejemplo de una curva elíptica:

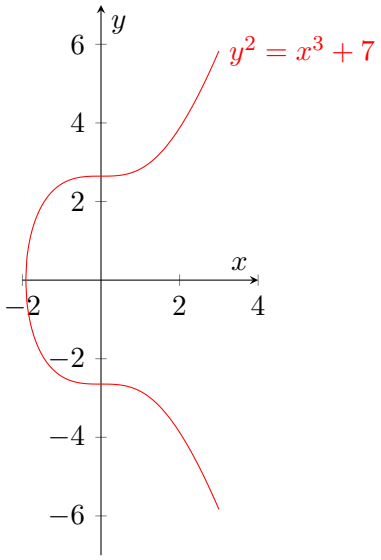

Figura 3.1: Representación gráfica de una curva elíptica

Aspecto fundamentales a tener en cuenta de la curva:

- Es una curva no-singular, es decir que la curva no se intersecta a si mismo siempre y cuando el discriminante de la curva $-16(4a^3+27b^2) \neq 0$
- La curva es simétrica con respecto al eje de abscisas ya que si despejamos la y de la ecuación, ambos valores $\pm\sqrt{x^3 + ax + b}\ mod\ p$
- Hay solo una intersección en el eje de abscisas. Se debe a que si solucionamos la ecuación para $y = 0$, exite una solución real (intersección con el eje de abscisas) y dos soluciones complejas (no mostradas en la gráfica). Existen curvas elípticas con tres soluciones reales.

Elliptic Curve Digital Signature Algorithm (ECDSA)

La curva elíptica tiene ventajas sobre RSA y otros esquemas DL. Los ECC con una clave con un tamaño que oscila entre los 160 y 256 bits proveen de una seguridad similar a otros algoritmos criptográficos de entre 1024 y 3072 bits. Esta propiedad resulta en un tiempo de procesamiento menor así como firmas más pequeñas. Por esa razón, ECDSA fue estandarizado en Estados Unidos por la ANSI en 1998. El estándar ECDSA es definido para curvas elípticas sobre campos de número primo $\mathbb{Z}_p$ y campos de Galois $GF(2^m)$. Este algorimo se compone de las siguientes fases:

1. Generación de claves. Estas deben ser de al menos 160 para un nivel de seguridad elevado

2. Generación de firma. Usa la clave privada para generar una pareja de valores enteros (r, s). Usa

3. Verificación de firma. Usa la clave pública junto al par (r, s) y un valor concreto.

Seguridad sobre ECDSA

Suponiendo que los parámetros de la curva elíptica son escogidos correctamente, el principal ataque analíico sobre ECDSA intentaría resolver el problema de logaritmo discreto de curva elíptica. Si un atacante fuera capaz de llevarlo acabo, podría resolver la clave privada y/o la clave efímera. No obstante, el mejor ataque conocido contra ECDSA tiene una complejidad proporcional a la raiz cuadrada del tamaño del grupo sobre el cual el DL es definido. El parámtero de ECDSA y su correspondiente nivel de seguridad están definidos en la tabla 3.1

q	Tamaño hash	Nivel de seguridad
192	192	96
224	224	112
256	256	128
384	384	192
512	512	256

Cuadro 3.1: Tamaño de bits y nivel de seguridad de ECDSA

3.1.4. Pluggabe Authentication Module (PAM)

Tal y como se comentó en la sección 1.1, la motivación de este trabajo nace de la propuesta de mejora en [4] basada en implementar en el sistema propuesto compatibilidad con otros métodos de autenticación como los módulos PAM.

La empresa tecnológica *SunSoft* propuso en 1996 [26] un mecanismo de autenticación compatible con multiples tipos de sistemas dando capacidad para administrar no solo la autenticación sino también las sesiones y las contraseñas

Los mecanismos y protocolos de autenticación como por ejemplo Secure Shell (SSH), Remote Login (Rlogin) o File Transport Protocol (FTP) tienen como objetivo ser independientes de los mecanismos de autenticación específicos utilizados por las computadoras. No obstante, es importante que se aplique un marco que conectase todos esos mecanismos. Para ello se requiere que las aplicaciones usen una Application Program Interface (API)

estándar que interactue con los servicios de autenticación. Si este mecanismo de autenticación al sistema se mantuviera independiente del usado por el computador, el administrador del sistema podría instalar módulos de autenticación adecuados sin requerir cambios en las aplicaciones.

Lo ideal en todo sistema sería aplicar un mecanismo de autenticación complejo simplemente recordando una contraseña. No obstante, los sistemas son cada vez mas heterogéneos y complejos y por ellos requieren de varios mecanismos de autenticación (problema de inicio de sesión integrado o unificado).

El objetivo reside en la integración modular de las tecnologías de autenticación de red con el inicio de sesión y otros servicios.

Las propiedades que este mecanismo debe seguir son las siguientes:

- El administrador del sistema debe poder elegir el mecanismo de autenticación por defecto
- Debe ser posible configurar la autenticación del usuario para cada aplicacióon
- Debe soportar requisitos de visuallización de las aplicaciones
- Debe ser posible configurar múltiples protocolos de autenticación
- El administrador del sistema debe poder configurar el sistema de tal forma que el usuario pueda autenticarse usando múltiples protocolos de autenticación sin tener que reescribir la contraseña
- No debe ser reconfigurado cuando el mecanismo que funcione por debajo cambie
- La arquitectura debe proveer un modelo modular de autenticación
- Debe soportar los requisitos de autenticación del sistema sobre el que opere
- La API debe ser independiente del Sistema Operativo

Los elementos principales del *framework* PAM son la API (librería de autenticación) considerada el *front-end* y el módulo de autenticación específico, el *back-end*, ambos conectados a través del Service Provider Interface (SPI). El proceso consta de los siguientes pasos:

1. La aplicación escribe a la API de pam
2. Se carga el módulo de autenticación apropiado según especifique el archivo de configuración *pam.conf*

3. La petición es enviada el módulo de autenticación correspondiente) para llevar a cabo la operación concreta

4. PAM devuelve la respuesta desde el módulo de autenticación a la aplicación

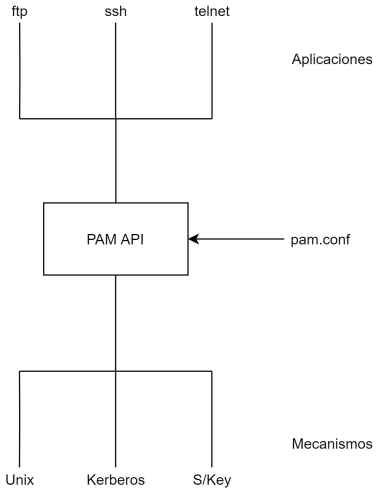

Figura 3.2: Arquitectura básica PAM

PAM unifica autenticación y control de acceso al sistema, y permite añadir módulos de autenticación a través de interfaces bien definidas. Configuración en la autenticación es un componente junto a administración de cuenta, sesión y contraseñas. Para este trabajo, solo se va a usar la parte de autenticación ya que el resto no es necesaria. Cada una de estas áreas funcionales trabajan como módulos separados.

Dado la extensión y temática del trabajo, no se pretende dar detalles acerca del funcionamiento de la API de PAM. Simplemente anotar que para el desarrollo de este proyecto se usó las siguientes funciones de la API:

- *pam_authenticate()* [27]: función para autenticar a un usuario
- *pam_get_user()* [28]: función para obtener el nombre del usuario específico que intenta autenticarse

El archivo de configuración PAM (*pam.conf*) es la base de gestión de los módulos. Tal y como se ha mencionado antes, cuando una aplicación solicita autenticarse usando algún mecanismo que funcione con PAM, la API comprueba los módulos a ejecutar en este archivo así como la política que siguen.

El archivo de configuración A.1 se ha usado en el broker MQTT para establecer los parametros de configuración siguientes [29]:

- *log_dest*: ruta absoluta del archivo de logs
- *log_type*: tipo de mensajes a registrar
- *log_timestamp*: añadir valor de marca de tiempo
- *include_dir*: ruta absoluta del directorio de archivos de configuración externos
- *listener*: puerto de escucha
- *cafile*: ruta absoluta del certificado de la Cerificate Authority (CA)
- *certfile*: ruta absoluta del certificado del broker MQTT
- *keyfile*: ruta absoluta de la clave privada del broker MQTT
- *allow_anonymous*: permitir que clientes se puedan conectar sin proporcionar claves (usuario)

El cliente *mosquitto* escucha por defecto por el puerto 1883 para comunicaciones no seguras y por el 8883 para seguras. El primero solo se usa para comunicaciones internas (*localhost*).

La CA es una entidad que administra certificados digitales, los cuales son usados para vincular una entidad a una clave pública. Para el presente tranajo, a diferencia de [5], se ha usado el broker MQTT como CA. No obstante, la CA debe correr en un servidor independiente debido a su papel fundamental en la seguridad de toda infraestructura.

3.2. Análisis de herramientas

Para el presente trabajo se ha desarrollado el software usando el lenguaje de programación C dada la facilidad y documentación disponible a la hora de desarrollar módulos PAM en dicho lenguaje. Se ha usado *mosquitto* [19] como cliente MQTT dado su extensa documentación, por ser un software de código abierto y estar escrito en C. Con respecto al entorno de pruebas, se ha usado Vagrant [30] como orquestador de máquinas virtuales.

Capítulo 4

Diseño de la solución

La solución propuesta 1.1 se ha desarrollado siguiendo las siguientes fases: una primera etapa de preparación de entorno seguro, una segunda etapa de identificación, y una final de autenticación. Para cada una de las fases se adjunta una diagrama de secuencias [1] [2]

4.1. Fase TLS

En esta fase, el broker MQTT debe crear un certificado firmado por una CA para verificar su identidad. Para comunicarse por TLS con el broker MQTT, se ha llevado a cabo los siguientes pasos:

1. Crear la clave privada de la CAs
2. Crer el certificado CA usando la clave privada del paso 1 para firmarla
3. Crear la clave privada del broker MQTT
4. Crear la solicitud de firma de certificados (CSR) para el broker MQTT usando la clave privada del paso 3
5. Usar la clave y certificado CA para firmar el certificado del broker MQTT
6. Distribuir la clave y certificado en la CA
7. Distribuir el certificado CA a los cliente que se quieran comunicar a través del broker MQTT

[1] Los valores entre < y > indican un valor en concreto, no el valor definido entre ambos símbolos

[2] Los tópicos tienen la estructura emisor/receptor/item

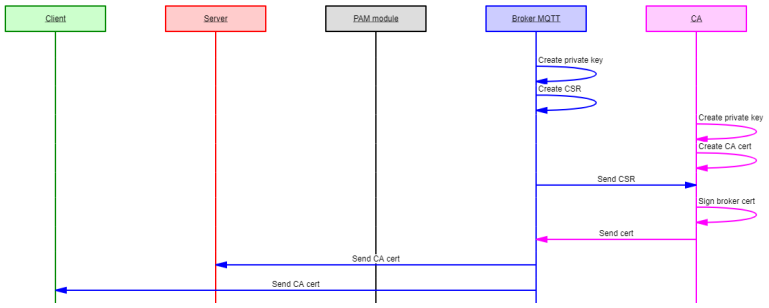

Figura 4.1: Fase TLS

Como último paso, el archivo de configuración del cliente *mosquitto* en el broker MQTT tiene que indicar la clave y certificados necesarios tal y como figura en A.1.

Si un cliente quiere usar el broker MQTT para publicar un mensaje o subscribrse a un tópico, necesita "mostrar" el certificado CA al broker MQTT. No es la única forma de identificación vía TLS. También existe la opción de que cada cliente cree su propio certificado firmados por la CA.

4.2. Fase de identificación

En esta fase, el cliente se tiene que registrar en el servidor para poder usar su servicio. Para ello, el cliente crea un identificador único Universally Unique IDentifier (UUID) y un par de claves pública y privada. Estos se guardan en un directorio en concreto. Se ha escogido *.anubis* en la carpeta *home* del usuario. Las claves se guardan con el UUID como nombre del archivo y finalmente se envía la clave pública al servidor. Se puede usar el protocolo Secure Copy Protocol (SCP). Dado que no se quiere usar un método de autenticación distinto al propuesto por este trabajo, se podría crear una clave pública temporal del servidor y subirla a algún servidor de claves públicas. El cliente por tanto podría usarla para mandar su clave pública a su directorio *.anubis* de forma segura y una vez envíada,

Una vez que el servidor tiene la clave pública del cliente, este primero revocaría la clave pública del servidor de claves y posteriormente lo registra en el archivo de configuración de usuario A.2. Este indica el UUID concreto para un usuario en el sistema. Se usa en caso de que el usuario tenga varias claves públicas y por tanto el servidor sepa que clave usar para la autenticación.

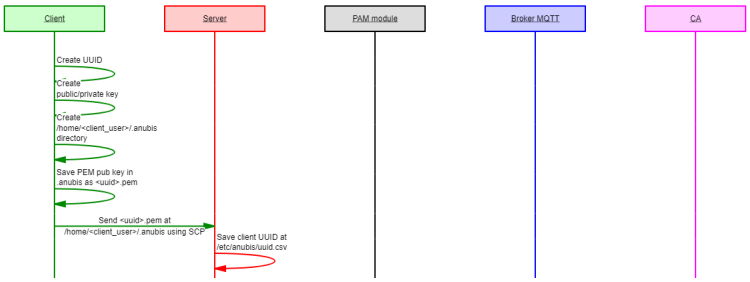

Figura 4.2: Fase de identificación

4.3. Fase de autenticación

En esta fase reside el proces de autenticación del cliente contra el servidor una vez que se ha establecido un canal seguro y registrado el cliente en el servidor.

El cliente se subscribe al tópico *pam/<uuid>/challenge* por el cual recibirá el desafío y envía una petición de para abrir una sesión por SSH.

Al llegar la petición SSH al servidor, este comprueba el UUID del usuario que se quiere autneticar en el archivo de configuración de ususarios A.2. Una vez el servidor conoce el UUID, de plantean dos posibilidades:

1. Que el usuario no necesite autenticarse de la forma propuesta
2. Que el usuario tenga que autenticarse

Cada condición se da según la política definida en el archivo de configuración de anubis A.3. Existen dos tipos de políticas:

1. *relax*: no es necesario aplicar la autenticación
2. *strict*: se aplica la autenticación

La política *relax* es útil en casos en los que por ejemplo el usuario provenga de una red de confianza como puede ser una Universidad. En ese caso, el módulo PAM propuesto devolvería un *PAM_IGNORE* pasando el siguiente módulo. En caso de la política *strict*, es necesario pasar el proceso de autenticación propuesto y que devuelva un *PAM_SUCCESS*.

Para una política *strict*, el servidor se subscribe a dos tópicos por los cuales el cliente publicará el par de valores $[r, s]$ del algoritmo ECDSA definido en 3.1.3 en formato hexadecimal:

- *<uuid>/pam/r*
- *<uuid>/pam/s*

Crea el desafío y lo publica al tópico *pam/<uuid>/challenge*. Al llegar el mensaje al broker MQTT, este lo reenvía a todos los nodos que estén suscritos a dicho tópico. Dado que solo hay un UUID por cliente, el mensaje solo le llega al cliente determinado por el UUID.

El cliente crea el hash del desafío. Concretamente, para este proyecto se ha usado el algoritmo SHA-512 dada su robustez con respecto a otros de menor tamaño como puede ser SHA-256. Una vez que tiene el hash, lo firma usando su clave privada creada en 4.2 y envía ambos valores $[r, s]$ por los tópicos *<uuid>/pam/r* y *<uuid>/pam/s* respectivamente siendo estos retransmitidos por el broker MQTT al servidor ya que es el único que conoce el UUID del cliente.

Finalmente, el servidor general el hash del desafío y junto a la clave pública del cliente verifica que el hash es correcto y está firmado por el cliente verídico. Si es verídico, el servidor devuelve *PAM_SUCCESS*. En caso contrario, *PAM_AUTH_ERR*.

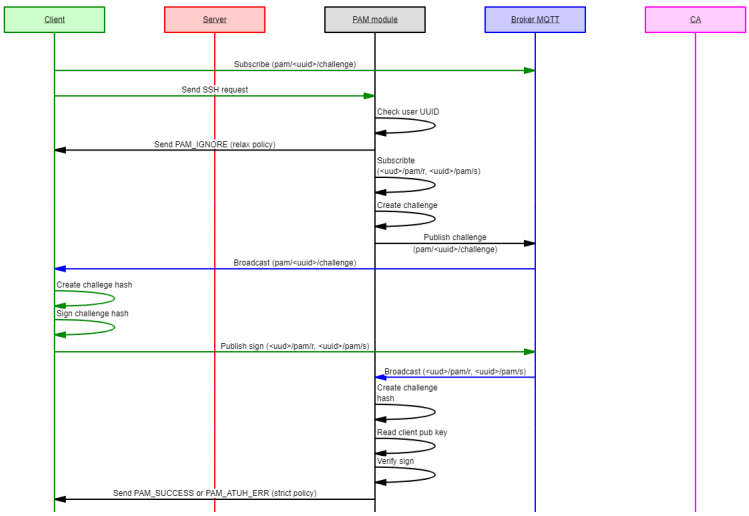

Figura 4.3: Fase de autenticación

En la siguiente imagen se mustra la topología global del sistema propuesto:

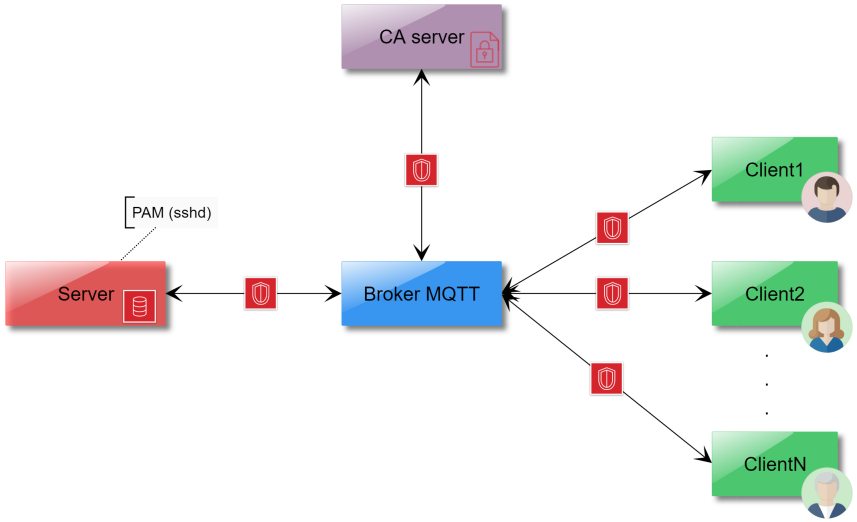

Figura 4.4: Topología del diseño

4.4. Código fuente

El código fuente no se adjunta por simplicidad y limpieza en la memoria pero se encuentra subido a la plataforma de GitHub [31].

Capítulo 5

Análisis de seguridad

Una vez detallado el diseño e implementación del sistema de autenticación propuesto, es necesario conocer la caracterísiticas de seguridad que implementa.

Farooq propone en [32] un sistema similar al propuesto en este trabajo pero centrado en sistemas electrónicos de tipo contador inteligentes. En él, se hace un estudio de los distintas vulnerabilidades a ciberataques que el sistema evita que se sean explotados.

5.1. Man In The Middle (MITM)

El ataque MITM conocido como "Ataque de Hombre en el Medio" es muy común en todo sistema que use una red pública como puede ser Internet para establecer una comunicación entre dos o más integrantes. Para evitar este tipo de ataque, el sistema propuesto usa el algoritmo ECDSA para firmar el desafío y dado que solo el servidor conoce la clave pública del cliente. En caso de una tercera persona, para que esta se pudiera pasar por el cliente necesitaría interceptar el desafío y conocer su clave privada para firmarlo.

5.2. Integridad del desafío

Dado que la respuesta al desafío proviene de la salida de una función hash que ha sido procesada en ambos lados, tanto cliente como servidor, sin que llege a ser envíada por algún canal, implica que si alguna tercera persona modifica el desafío o la respuesta a este, el sistema lo detectará y no pasará.

5.3. Confidencialidad del mensaje

Como ya se ha mencionado en 4.1, la comunicación con el broker MQTT se hace vía TLS. Esto permite que los mensajes vaya cifrados por una clave.

Para usar el broker MQTT y comunicarse con el servidor o viceversa, es necesario presentar un certificado. Esto garantizar que solo usuario legítimos se comumiquen con el servidor.

Capítulo 6

Presupuesto

6.1. Componentes hardware y software

Para la realización de este proyecto no se ha usado dispositvos hardware dedicados como podría ser un Arduino o Raspberry. Simplemente se ha virtualizado el entorno de pruebas mediante software de vitualización (VirtualBox) y por tanto no ha tenido coste económico ninguno. En cuanto al software, tampoco se han usado programas con licencias ya que tanto el cliente MQTT (mosquitto), la API de SSL y el entorno de desarrollo (Visual Studio Code) son de código libre.

6.2. Diagrama de Gantt

		Nombre	Duracion	Inicio	Termin
1		Lectura tesis y PAM	43 days	22/10/20 8:00	21/12/20 17:00
2		Vacaciones Navidad	9 days	22/12/20 8:00	1/01/21 17:00
3		Creación de entorno de pr...	21 days?	4/01/21 8:00	1/02/21 17:00
4		Desarrollo	39 days?	2/02/21 8:00	26/03/21 17:00
5		Vacaciones de Semana Santa	6 days?	29/03/21 8:00	5/04/21 17:00
6		Desarrollo	40 days?	6/04/21 8:00	31/05/21 17:00
7		Documentación	22 days?	1/06/21 8:00	30/06/21 17:00

Figura 6.1: Diagrama de Gantt

Apéndice A

Archivos de configuración

Listing A.1: Archivo de configuración PAM

```
# Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example
pid_file /run/mosquitto/mosquitto.pid

log_dest file /var/log/mosquitto/mosquitto.log
log_type all
log_timestamp true

include_dir /etc/mosquitto/conf.d

listener 1883 localhost
listener 8883

cafile /etc/mosquitto/ca_certificates/ca.crt
certfile /etc/mosquitto/certs/broker-mqtt.crt
keyfile /etc/mosquitto/certs/broker-mqtt.key

allow_anonymous true
```

Listing A.2: Archivo de configuración de usuarios en /etc/anubis/uuid.csv

```
username,uuid
client-1,68263723-e928-4f71-8339-c609478f0a1a
```

Listing A.3: Archivo de configuración anubis en /etc/anubis/anubis.conf

```
# ------------------------#
# /etc/anubis/anubis.conf
# ------------------------#
#
# NOTE
# ----
#
# Configuration file for MQTT-PAM module used to authenticate via
    SSH
# Use only a space between key and value
#
# Permitted values:
# access_type [relax, strict]
# ip_address 150.214.*.*
# ------------------------#
#
# Format:
# key value

access_type relax
```

Bibliografía

[1] AJ Neumann, N Statland, and RD Webb. *Post Processing Audit Tools and Techniques*. 1977.

[2] Digital Attack Map. `https://www.digitalattackmap.com`.

[3] CVE - CVE. `https://cve.mitre.org/`.

[4] Ilia Blokhin. Mecanismos de seguridad para big data basados en circuitos criptográficos, 2020.

[5] Antonio Francisco Díaz García, Ilia Blokhin, Mancia Anguita López, Julio Ortega Lopera, and Juan José Escobar Pérez. Multiprotocol authentication device for hpc and cloud environments based on elliptic curve cryptography, 7 2020.

[6] C Newman, A Menon-Sen, A Melnikov, and N Williams. Salted challenge response authentication mechanism (scram) sasl and gss-api mechanisms. *Internet Requests for Comments, RFC Editor, RFC*, 5802, 2010.

[7] An Braeken. Puf based authentication protocol for iot. *Symmetry*, 10(8):352, 2018.

[8] Henk CA Van Tilborg and Sushil Jajodia. *Encyclopedia of cryptography and security*. Springer Science & Business Media, 2014.

[9] You Alex Gao, Jim Basney, and Alex Withers. Scitokens ssh: Token-based authentication for remote login to scientific computing environments. In *Practice and Experience in Advanced Research Computing*, pages 465–468. 2020.

[10] Achraf Fayad. *Secure authentication protocol for Internet of Things*. PhD thesis, Institut Polytechnique de Paris, 2020.

[11] google/google-authenticator-libpam. `https://github.com/google/google-authenticator-libpam`, June 2021.

[12] YubiKey. `https://www.yubico.com/la-yubikey/?lang=es`.

[13] Single sign-on: What is it & how does it work? https://www.onelogin.com/learn/how-single-sign-on-works.

[14] Bases de la seguridad informática | Seguridad Informática. http://descargas.pntic.mec.es/mentor/visitas/demoSeguridadInformatica/bases_de_la_seguridad_informtica.html.

[15] Aleksandr Ometov, Sergey Bezzateev, Niko Mäkitalo, Sergey Andreev, Tommi Mikkonen, and Yevgeni Koucheryavy. Multi-factor authentication: A survey. *Cryptography*, 2(1):1, 2018.

[16] Christina Braz and Jean-Marc Robert. Security and usability: the case of the user authentication methods. In *Proceedings of the 18th Conference on l'Interaction Homme-Machine*, pages 199–203, 2006.

[17] Somayya Madakam, Vihar Lake, Vihar Lake, Vihar Lake, et al. Internet of things (iot): A literature review. *Journal of Computer and Communications*, 3(05):164, 2015.

[18] KALAIVANAN SUGUMAR. Mqtt-a lightweight communication protocol relative study. *Authorea Preprints*, 2020.

[19] Eclipse Mosquitto. https://mosquitto.org/, January 2018.

[20] HiveMQ - Enterprise ready MQTT to move your IoT data. https://www.hivemq.com/.

[21] moscajs/mosca. https://github.com/moscajs/mosca, June 2021.

[22] CloudMQTT - Hosted message broker for the Internet of Things. https://www.cloudmqtt.com/.

[23] MQTT.js. https://github.com/mqttjs.

[24] Kevin S McCurley. The discrete logarithm problem. In *Proc. of Symp. in Applied Math*, volume 42, pages 49–74. USA, 1990.

[25] Christof Paar and Jan Pelzl. *Understanding cryptography: a textbook for students and practitioners*. Springer Science & Business Media, 2009.

[26] Vipin Samar. Unified login with pluggable authentication modules (pam). In *Proceedings of the 3rd ACM conference on Computer and communications security*, pages 1–10, 1996.

[27] pam_sm_authenticate(3) - Linux manual page. https://man7.org/linux/man-pages/man3/pam_sm_authenticate.3.html.

[28] pam_get_user(3) - Linux manual page. https://man7.org/linux/man-pages/man3/pam_get_user.3.html.

[29] mosquitto.conf man page. `https://mosquitto.org/man/mosquitto-conf-5.html`, June 2021.

[30] Vagrant by HashiCorp. `https://www.vagrantup.com/`.

[31] Sergio García. sergiogp98/mqtt-pam. `https://github.com/sergiogp98/mqtt-pam`, June 2021.

[32] Shaik Mullapathi Farooq, SM Suhail Hussain, and Taha Selim Ustun. Elliptic curve digital signature algorithm (ecdsa) certificate based authentication scheme for advanced metering infrastructure. In *2019 Innovations in Power and Advanced Computing Technologies (i-PACT)*, volume 1, pages 1–6. IEEE, 2019.