**FIRMADOR Y VERIFICADOR DE FIRMAS**

**PROCESO**

* Crear el par de claves con la clase *KeyPairGenerator*, el cual genera las claves pública y privada con un algoritmo particular. En este caso se usó el algoritmo RSA, que sirve para autenticar y cifrar información y requiere menor tiempo de cómputo en comparación al algoritmo DSA.
* Se solicita al usuario a través de la consola los nombres de los archivos en los cuales desea guardar las claves pública y privada y el password para el archivo de la clave privada.
* La clave pública se guarda en su archivo con la clase *FileOutputStream*, que sirve para escribir los datos en un archivo.
* Para encriptar la clave privada con su password se usa el método *encriptarPrivateKey*, el cual sigue el siguiente proceso:
* Se crea el salto (8 bytes)
* Se crea una clave y un cifrador con el algoritmo PBE, el cual se basa en cifrar la contraseña para mejorar la seguridad. Para ello se usan las clases *PBEKeySpec*, la cual se encarga del cifrado de la contraseña, *SecretKeyFactory* y *Cipher,* la cual proporciona la funcionalidad de cifrado criptográfico para cifrar y descifrar.
* Se encripta el salto y los datos de la clave privada con el password.
* Ya teniendo las claves en sus archivos correspondientes, se procede a firmar el archivo, esto se hace para asegurar la autenticidad del documento. Para ello se sigue el siguiente procedimiento:
* Se le solicita al usuario a través de la consola que ingrese la ruta del archivo que desea firmar.
* Se lee el archivo con la clase *BufferedInputStream.*
* Se solicita al usuario que ingrese el password del archivo de la clave privada.
* Se guarda la clave ingresada por el usuario en un arreglo de bytes,
* Se intenta desencriptar el archivo con el password a través del método *desencriptarPrivateKey*: se lee el salto (8 bytes) con la clase *ByteArrayInputStream,* el texto restante (texto – 8 bytes) es el texto cifrado de la clave privada, se crea un descifrador con las clases *PBEKeySpec, SecretKeyFactory* y *Cipher.* Se intenta realizar la desencriptación, si funciona significa que la clave proporcionada por el usuario era correcta, sino, se le solicita nuevamente la clave al usuario.
* Se firma el archivo con la clase *Signature*, que está diseñada para firmar digitalmente los archivos.
* Se guarda el archivo firmado en un nuevo documento.
* Cuando se ha firmado el archivo se procede a verificar que la firma, siguiendo el siguiente procedimiento por medio de la clase *Signature*, la cual permite verificar la última firma añadida a un archivo y compararla con la firma del autor.

**DIFICULTADES**

* Al usar la clase PBEKeySpec tuvimos problemas al momento de elegir el algoritmo que nos ayudaría a cifrar la clave, ya que deseábamos usar AES pero no lo permitió, tampoco permitió usar PKCS5Padding, ya que nos lanzaba problemas de compatibilidad.
* Al momento de ingresar la ruta del archivo que deseábamos firmar tuvimos varias dificultades, ya que la consola no recibe caracteres especiales, por lo tanto los archivos no deben tener la letra “ñ” o tildes en su nombre.

**CONCLUSIONES**

El proyecto nos permitió ver la importancia de la firma de archivos, ya que de esta forma se verifica la fuente del mismo, además, nos permitió conocer nuevas clases y funcionalidades de Java y practicar nuestras habilidades en desarrollo de software.