COPIAS DE SEGURIDAD REMOTAS

iSecurity

Alejandro Castro Valero
Gabriel Martínez Antón
Sergio Sebastián Aracil
Héctor Esteve Yagüe

DESCRIPCIÓN

- El proyecto tiene como objetivo alojar archivos multimedia en un servidor en la nube.
- Los usuarios administrarán sus archivos subidos mediante una aplicación móvil.
- Junto a los archivos subidos a la nube se guardarán los datos de dicha aplicación.
- Los archivos serán accesibles únicamente a usuarios que posean permisos mediante contraseñas.
- Para ello se recurren a algoritmos de cifrado para garantizar dicha seguridad.

ALGORITMOS DE SEGURIDAD

- **AES**: Algoritmo de cifrado en bloque por clave secreta. Algoritmo simétrico. Velocidad de cifrado bastante alta. Tamaño de clave de 128 bits. Se usará para encriptar los archivos antes de subirlos a la nube.
- **RSA**: Algoritmo de cifrado por clave pública. Algoritmo asimétrico. Cifrado robusto, distribución de claves fácil y segura. Tamaño de clave 1024 bits. Se usará para encriptar las claves simétricas para cada archivo.
- **SHA-3**: Algoritmo de hash criptográfico. Proporciona ciertas propiedades de seguridad. 512 bits de espacio. Se usará para proteger las contraseñas a introducir por los usuarios

FUNCIONAMIENTO

- Un usuario **A** desea subir el archivo $\boldsymbol{archive_1}$ a la nube. Generamos un par de claves AES que llamaremos $\boldsymbol{k_1}$ y encriptamos el mensaje con dicha clave:

$$E_{K_1}^{AES}(archive_1) = c_1$$

- Subimos el archivo $\boldsymbol{c_1}$ ya cifrado a la nube concatenando en su cabecera la otra clave generada $\boldsymbol{k_1}$. Una vez en la nube, se generan un par de claves RSA $\boldsymbol{K_{A\,publ}}$ y $\boldsymbol{K_{A\,priv}}$

- Para que el criptosistema $\boldsymbol{c_1}$ no sea accesible a otro usuario o al mismo servidor, ciframos la cabecera $\boldsymbol{k_1}$ con la clave pública $\boldsymbol{K_{A\,publ}}$

$$E_{K_{A\,publ}}^{RSA}(k_1) = k_A$$

FUNCIONAMIENTO

- La clave $K_{A\,priv}$ la guardaremos en un archivo de texto archivo de texto **clave.txt** para su posterior uso.
- Generamos otro par de claves k_2, con AES. El archivo **clave.txt** se volverá a cifrar

$$E_{K_2}^{AES}(clave.txt) = c_2$$

- k_2 será **password** de la que tendremos que acordarnos para poder desencriptar los dos criptosistemas generados, de esta forma la expresión anterior sería

$$E_{pass}^{AES}(clave.txt) = c_2$$

FUNCIONAMIENTO

- Cuando un usuario quiera recuperar una copia del documento subido a la nube debe introducir la contraseña adecuada. En ese momento se usará el algoritmo SHA-3 con la **password** para mayor seguridad.

$$SHA\text{-}3(password) = 512\ bits\ (Cogemos\ 128\ bits)$$

- Una vez introducida la **password**, desciframos el criptosistema c_2

$$D_{pass}^{AES}(c_2) = clave.txt$$

- Del archivo de texto obtenemos $K_{A\,priv}$, que usaremos para desencriptar k_A obtener por último nuestro archivo

$$D_{K_{A\,priv}}^{RSA}(k_A) = k_1$$

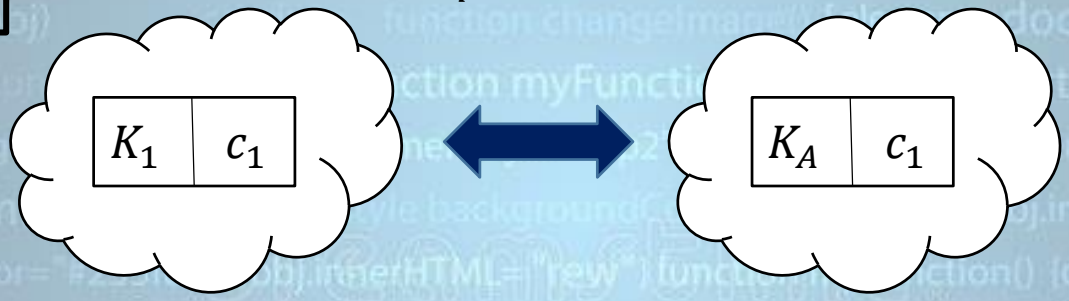
$$D_{K_1}^{AES}(c_1) = archive_1$$

ESQUEMA

$K_{A\,publ}$
$K_{A\,priv}$

$$E^{RSA}_{K_{A\,publ}}(k_1) = k_A$$
$$D^{RSA}_{K_{A\,priv}}(k_A) = k_1$$

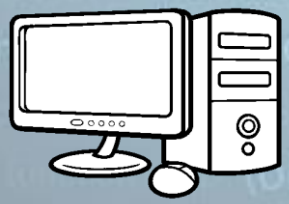
clave.txt[$K_{A\,priv}$]
K_2 ($password$)



$$E^{AES}_{K_1}(archive_1) = c_1$$
$$D^{AES}_{K_1}(c_1) = archive_1$$

$$E^{AES}_{pass}(clave.txt) = c_2$$
$$D^{AES}_{pass}(c_2) = clave.txt$$

$archive_1$
k_1



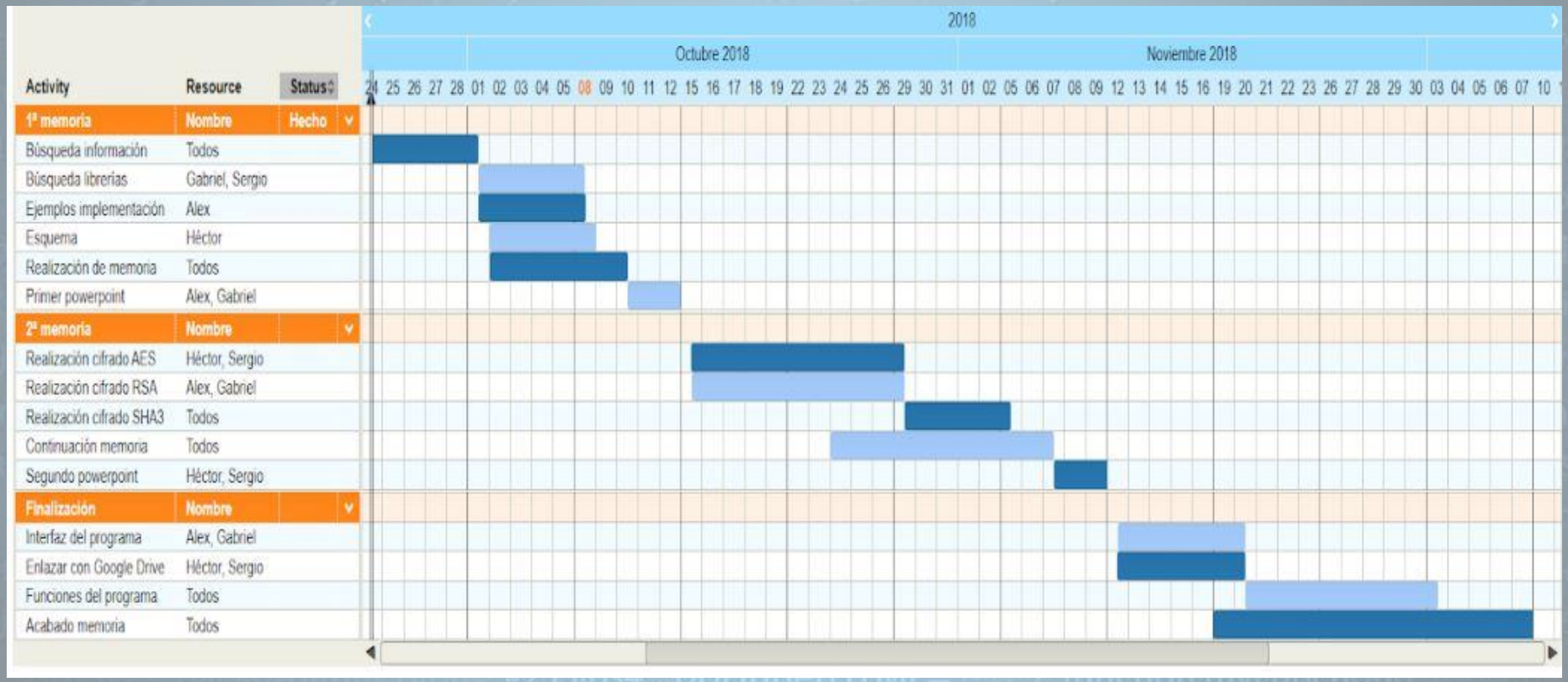
SHA-3(password) = 512 bits
(Cogemos 128 bits)

LIBRERÍAS

Las principales que utilizaremos son:

- Java.io: lectura y escritura de archivos.
- Java.security: seguridad (general).
- Javax.crypto: algoritmo AES.

PLANIFICACIÓN



SOFTWARE

- Eclipse Java
- Librerías de java
- Google Drive
- GitHub

FUNCIONES DE SEGURIDAD

Java.io

- FileInputStream: Convertir ficheros a bytes.
- FileOutputStream: Utilizada para crear ficheros a partir de bytes.

Java.security

- Key: Genera y maneja claves.
- InvalidKeyException: Clave no valida.
- KeyFactory: Utilizada para claves que no pueden ser creadas con el generador de claves.
- KeyPair: Contiene pares de claves privadas y públicas.
- KeyPairGenerator: Genera pares de claves privadas y públicas
- PrivateKey: Claves privadas.
- PublicKey: Claves públicas.

FUNCIONES DE SEGURIDAD

Javax.crypto

- Cipher: Crea una instancia del cifrador/descifrador AES.
- KeyGenerator: Genera un par de claves secretas para un sistema simétrico.
- SecretKeySpec: Genera una clave secreta a partir de un array de bytes.

COPIAS DE SEGURIDAD REMOTAS

iSecurity

Alejandro Castro Valero
Gabriel Martínez Antón
Sergio Sebastián Aracil
Héctor Esteve Yagüe