Universidad de Castilla-La Mancha

Álgebra y Matemática Discreta

Practica Incremental 2020-21 (Grupo Español)

Firmas y Certificados Digitales

Profesorado:
José Luis Espinosa Aranda (josel.espinosa@uclm.es)
Ricardo García Ródenas (ricardo.garcia@uclm.es)
José Ángel Martín Baos (joseangel.martin@uclm.es)
Jesús Javier Ortega Triguero (jesusjavier.ortega@uclm.es)

A tener en cuenta:

1. La realización de la práctica es individual.
2. Esta práctica se evalúa sobre 2 puntos del global de la asignatura y vale aproximadamente un $57\,\%$ de la parte de prácticas de la asignatura.
3. No es necesario realizar todos los hitos para que la práctica pueda ser entregada.
4. Para la evaluación de cada uno de los apartados se tendrán en cuenta tanto la claridad del código como los comentarios incluidos en éste.
5. Para la corrección de la práctica se utilizará un programa de detección de copia. En caso de que la similitud entre dos o más prácticas se encuentren fuera de lo permisible, todas ellas serán calificadas con un 0 y aparecerá suspenso en la asignatura tanto en la convocatoria ordinaria como extraordinaria.
6. Fecha límite de entrega: Hasta el 23 de Mayo. Se habilitará una tarea en campus virtual para poder subirla.

Objetivos de la práctica:

Las firmas y certificados digitales son una aplicación de la criptografía en clave pública. La firma digital tiene la misma finalidad que la tradicional firma "en papel", pero con documentos digitales. Y a diferencia de aquella, se pretende que sea ***auténtica e infalsificable***, de modo que sólo quién manifiesta haber firmado un documento digital haya podido hacerlo; y también ***inalterable***, para que después que un documento haya sido firmado, no sea posible alterarlo sin que la firma siga siendo válida. Un certificado digital proporciona a un persona (o entidad) una identidad en internet. También debe ser auténtico e infalsificable y, además, servir para que una persona se identifique remotamente ante otra de manera inequívoca.

En este trabajo vamos vamos a crear firmas y certificados digitales con MATLAB a partir del criptosistema RSA, estudiado en la sesión 3 de las prácticas de la asignatura.

1. Firma digital

La obtención de una firma digital a partir de un criptosistema de clave pública es bastante simple: se obtiene primero un ***resumen*** del documento digital que se quiere firmar y se cifra con la clave ***privada***. Este cifrado es precisamente la firma digital del documento. Es infalsificable porque sólo el conocedor de la clave privada ha podido generar la firma. Para comprobar la autenticidad de la misma, se descifra con la clave pública y se comprueba si coincide con el resumen del documento. La inalterabilidad de la firma nos la proporciona el resumen del documento, ya que se obtiene con una función hash.

1.1. Funciones hash

Una función hash comprime una secuencia de bits, de cualquier longitud, en otra de unos pocos bits de modo que, si cambiamos un sólo bit en la secuencia inicial, el resultado es

totalmente diferente. Con cierta formalidad: una función h, que a cada secuencia finita de bits s le hace corresponder otra $h(s)$ de un número fijo de bits, se dice una función hash si es prácticamente imposible encontrar dos secuencias distintas s y t tales que $h(s) = h(t)$. La imagen $h(s)$ es el resumen de la secuencia s.

Aquí vamos a emplear la función hash MD5 que viene implementada en MATLAB con el nombre `Simulink.getFileChecksum()`. Produce resumenes de 128 bits de un archivo cuyo nombre se pasa como parámetro. Por ejemplo, `Simulink.getFileChecksum('Calendario.pdf')` da como resultado:

```
'46552B7B0EEE4F034849DCB5BF1F69A8'
```

una cadena de 32 caracteres hexadecimales (de 4 bits): los 128 bits que decíamos.

Nota: Hoy en día, la función MD5 ha quedado obsoleta y en su lugar se utilizan las funciones de la familia SHA (Secure Hash Algorithm), que producen resúmenes con más bits.

1.2. Firmas RSA

Consideramos una clave RSA de por lo menos 200 bits: (*n, e, d*), siendo *n* el módulo, *e* el exponente y *d* la clave privada. Los resúmenes producidos por la anterior función MD5 podemos contemplarlos como enteros representados en base 16 (hexadecimal). Sea *m* el resumen generado al aplicar la función MD5 a un fichero concreto. Antes de cifrar *m* con la clave privada, lo vamos a "enmascarar" de la siguiente manera: Generamos un entero aleatorio **impar** *g de* 64 bits. Como es impar, tiene inverso módulo 2^{128}. Sea g^{-1} este inverso, calculado `powermod(g, -1, sym(2)^128)`. Calculamos $h = g^{-1} \cdot m \bmod 2^{128}$ y construimos $M = g \cdot 2^{128} + h$. (Notamos que *M* tiene 192 bits y, por tanto, es menor que el módulo *n*). Calculamos

$$f = M^d \bmod n$$

y lo representamos en base 16. Finalmente, la cadena hexadecimal obtenida es la firma digital.

Nota: La función hash MD5 empleada devuelve una cadena de caracteres hexadecimales, no obstante, para operar en MATLAB se deben utilizar enteros simbólicos. Para convertir esta cadena hexadecimal de caracteres en enteros simbólicos se puede emplear la función de MATLAB `sym()`, que fue vista en la sesión 3 de prácticas, poniéndo el string '0x' delante de la cadena a convertir. Recíprocamente, puede obtenerse la representacion hexadecimal de cualquier entero con `dec2hex()`.

Observaciones:

1. La razón de enmascarar *m* es que si ciframos *m* directamente entonces de la ecuación $f = m^d \bmod n$ podría recuperarse la clave privada *d* mediante el cálculo de logaritmos discretos, ya que *m*, f y *n* son públicos.

2. Es importante notar que la inclusión del entero aleatorio *g* hace que la obtención de la firma digital no sea un proceso determinista: cada vez que se firme un mismo documento se obtendrán firmas diferentes. Esto no importa: lo importante es que sólo el conocedor de la clave privada *d* puede generarlas.

3. En la actualidad, se utilizan claves RSA de 2048 bits. Y el proceso de enmascaramiento del resumen es más complejo.

1.3. Hito 1 (0.5 puntos)

a) Implementar en MATLAB una función, `Firmadigital()`, que admita como parámetros el nombre de un archivo, el módulo de una clave RSA y su clave privada, y devuelva la firma digital del archivo conforme a la construcción anterior. Se proporciona como ayuda la función `randimpar()` que genera un número simbólico aleatorio impar de 64 bits y que puede ser usada para generar *g*.

b) En el archivo `datos.mat` figura la clave RSA `keyAuto`. Es una estructura con tres **enteros simbólicos**: `modulo`, `exponente` y `privada` que contienen, respectivamente, el módulo, la clave pública y la clave privada. Con esta clave `keyAuto`, firmar el archivo `Calendario.pdf` que se proporciona. Generar un script, llamarlo `Hito1.m`, que presente dicha firma del modo habitual en este contexto: *como cadena de caracteres hexadecimales*.

Nota: Para cargar el archivo de datos `datos.mat`, se puede utilizar el comando `load('datos.mat')`.

2. Verificación de una firma digital

Elevando f al exponente *e* (clave pública) módulo n recuperamos *M*. Y con *M*, obtenemos *m* de la siguiente manera:

Recordad que $M = g \cdot 2^{128} + h$, por lo tanto, primero dividimos *M* entre 2^{128}. De esta manera, el cociente de la división será *g* y el resto *h*. En MATLAB se puede utilizar la función `quorem(A, B)`, la cual devuelve el cociente y el resto de dividir A entre B.

Recordad también que $h = g^{-1} \cdot m \bmod 2^{128}$, por lo tanto, múltiplicando ambos lados de la expresión por *g*, tenemos:

$$g \cdot h = g \cdot g^{-1} \cdot m \bmod 2^{128} \implies g \cdot h = m \bmod 2^{128}.$$

De esta manera, queda que $m = (g \cdot h) \bmod 2^{128}$. Sólo resta comprobar si *m* es el entero que se corresponde con el resumen del documento firmado. En tal caso, la firma es válida. En otro caso, no lo es.

2.1. Hito 2 (0.5 puntos)

a) Implementar en MATLAB una función, `Verificafirma()`, que admita como parámetros el nombre de un archivo, una cadena hexadecimal (la fima), el modulo de una clave RSA y su exponente; y devuelva tres datos: el primero, una variable booleana que indique si la firma es válida; el segundo, el valor *m* oculto en la firma, en hexadecimal; y el tercero el hash del archivo.

b) El archivo `datos.mat` contiene el array: `FirmasHito2` con 5 firmas digitales del archivo `Calendario.pdf`. ¿Cuáles de esas 5 firmas han sido obtenidas con `keyAuto`? Presentar la respuesta en forma de tabla donde se vea si `FirmasHito2{i}` es auténtica, el correspondiente valor *m*, en hexadecimal, y el hash del archivo. Escribir un script, llamarlo `Hito2.m`, que lo muestre.

3. Certificados digitales

Como decíamos, un certificado digital proporciona a un persona (o entidad) una identidad en internet. Los certificados los expide una Autoridad como, por ejemplo, el Estado o una empresa. Un certificado es un contenedor con los siguientes datos:

- Datos del usuario: nombre, DNI, dirección, e-mail...
- Datos del certificado: Autoridad que lo emite, fechas de expedición y caducidad, nº de serie...
- Las claves pública y privada del criptosistema que se utilice.
- La firma de la Autoridad, dando autenticidad al certificado.

Todos estos datos son **públicos**, excepto, claro está, **la clave privada**. (También es pública la clave pública de la Autoridad).

En este apartado, vamos a construir certificados muy simples: van a ser estructuras (struct en MATLAB) con los siguientes campos, todos ellos **tipo strings**:

- *iden*: el nombre de cada alumno.
- *mod*: la representación hexadecimal del módulo de una clave RSA.
- *exp*: la representación hexadecimal del exponente.
- *priv*: la representación hexadecimal de la clave privada.
- *firma*: la firma digital de la Autoridad, obtenida con la clave `keyAuto` mencionada en el Hito 1 (en hexadecimal).

Para obtener la firma de la Autoridad (campo *firma*), unimos las cadenas *iden, mod* y *exp* (en este orden) con `strcat()`. A continuación, creamos un archivo temporal en binario con el comando `fopen('ficherotemporal.bin', 'w')`, escribimos en él la cadena con la unión de *iden, mod* y *exp* usando la función `fwrite()` y cerramos el archivo con `fclose()`. Para más ayuda con el manejo de archivos de MATLAB, recomendamos leer la documentación oficial. El campo *firma* del certificado que vamos a crear es entonces la firma digital de dicho archivo temporal que hemos creado usando la clave privada `keyAuto`. Finalmente, podemos borrar el archivo temporal con `delete 'ficherotemporal.bin'`.

3.1. Hito 3 (0.5 puntos)

a) Escribir un script denominado `GenerarCertificadoPersonal.m` el cual genere una clave RSA de entre 200 y 256 bits (tal y como se estudió en la sesión 3 de prácticas). Puede utilizar la función `randprimo()` de la sesión 3 de prácticas para generar dos primos distintos p y q que son necesarios para generar la clave RSA. Tenga en cuenta que $n = pq$. Por lo tanto, la longitud de ambos primos debe ser la mitad del módulo. Con esta clave, crear un certificado digital conforme a la construcción anterior. Llamarlo `micertificado` y guardarlo en el archivo `micertificado.mat`.

b) Seguidamente, escribir un nuevo script que se denomine `Hito3.m` y que lea el certificado `micertificado.mat`, lo muestre en pantalla y compruebe que es auténtico verificando la igualdad del valor *m* que esconde la firma con el hash del archivo temporal que almacena las cadenas *iden, mod* y *exp*. Hacer uso de la función `Verificafirma()` creada en el Hito 2.

4. Esquemas de identificación

El modo habitual de indentificarnos en internet es mediante un nombre de usuario y una clave, "la password". Cuando se requiere nuestra identificación, enviamos ambos datos. El problema está en que si alguien intercepta esta comunicación, puede suplantarnos (no es fácil, puesto que la clave se enmascara de maneras muy enrevesadas, pero puede hacerse). Y recíprocamente, también es deseable estar seguro que nos estamos identificando en el sitio correcto; no vaya a ser que dicho sitio sea fraudulento, como ocurre en el "phishing". Con el uso de certificados digitales se evitan estos problemas. Veamos un modo de hacerlo.

Imaginemos que un usuario *U*, poseedor de un certificado, solicita identificarse telemáticamente ante una organización *R*, poseedora también de otro certificado. Tras la solicitud, el usuario genera un entero aleatorio *u* y lo envía a la organización *R*. Ésta genera también un número aleatorio *r* y con los datos *R, u* y *r* compone un mensaje que firma con su certificado, obteniendo una firma $f_R(R,u,r)$. Entonces transmite al usuario *U* los siguientes datos:

- Los datos **públicos** de su certificado *cert(R)*, es decir, el módulo y el exponente. Notar que nunca se envía la clave privada.
- Los enteros aleatorios *u* y *r*.
- La firma $f_R(R,u,r)$.

El usuario verifica la validez del certifcado *cert(R)* y de la firma $f_R(R,u,r)$, aceptando así la identidad de *R*. Entonces, con los datos *U, u* y *r* compone un mensaje que firma con su certificado, obteniendo una firma $f_U(U,u,r)$, y envía a la organización *R*:

- Los datos **públicos** de su certificado *cert(U)*, es decir, el módulo y el exponente.
- Los enteros aleatorios *u* y *r*.
- La firma $f_U(U,u,r)$.

Finalmente, la organización comprueba la validez del certificado *cert(U)* y de la firma $f_U(U,u,r)$ y acepta la identidad de *U*.

De este modo, mediante las firmas digitales, tanto *U* como *R* tienen certeza de que el otro es quién dice ser. Notamos que en lugar de las claves privadas se transmiten enteros aleatorios. Estos deben ser distintos en cada identificación.

4.1. Hito 4 (0.5 puntos)

Supongamos que deseamos identificarnos con el certificado creado en el Hito 3 ante cierta organización. Le hemos transmitido un entero aleatorio de 128 bits: `int_u` y nos ha respondido con una estructura, `deRaU`, que figura en el archivo `datos.mat` y tiene los siguientes campos, todos ellos de **tipo strings**:

- *iden*: el nombre de la organización.
- *mod:* modulo de su clave RSA, en hexadecimal.
- *exp*: el exponente, en hexadecimal.
- *firma*: la firma digital de su certificado (firmado por la autoridad usando la clave `keyAuto`).
- *int_u*: el entero aleatorio de 128 bits, en hexadecimal.
- *int_r*: otro entero aleatorio de 128 bits, en hexadecimal.
- *f_mensaje*: la firma $f_R(R,u,r)$ señalada anteriormente.

La firma $f_R(R, u, r)$ se ha obtenido como en el Hito 3: se unen las cadenas *iden, int_u* e *int_r* (en este orden), se escribe la cadena resultante en un archivo binario temporal y se firma dicho archivo.

Escribir un script, llamarlo `Hito4.m`, que:

a) Lea la estructura `deRaU`, la muestre en pantalla y compruebe que la identidad de la organización es auténtica verificando las dos firmas que figuran en el mensaje. Incluir también el valor *m* que ocultan las dos firmas y los correspondientes hash.

b) Componer después la estructura respuesta `deUaR`, mostrarla en pantalla y comprobar la autenticidad de la identidad del usuario del mismo modo que antes.