Bisoños Usuarios de GNU/Linux de Mallorca y Alrededores | Bergantells Usuaris de GNU/Linux de Mallorca i Afegitons

Criptografía - Firmas digitales ilustradas (38428 lectures)

Per **Eduard Llull**, ***Daneel*** ()

Creado el 21/09/2001 20:25 modificado el 21/09/2001 20:25

Las firmas digitales son una de estas cosas de las que todo el mundo ha oido hablar pero muchos no saben exactamente que son. En este artículo explico el funcionamiento y la generación de firmas digitales y es la traducción de un artículo(1) escrito por J. Orlin Grabbe.
Por cierto, recomiendo a todo el mundo que quiera iniciarse en el mundo de la criptorafía, se pase por la página web de J. Orlin Grabbe(2), donde encontrará multitud de artículos introductorios sobre distintos aspectos de la criptgrafía.

Firmas Digitales Ilustradas

por J. Orlin Grabbe
traducción de Eduard Llull

Leí la declaración jurada en la asesoría jurídica. Se trataba de una disputa sobre options-trading entre un comerciante y un banco, y yo era un experto contratado por los abogados del demandante. Al acabar de leer cada una de las páginas, la firmaba.

"¿Que está haciendo?, me pregunto uno de los abogados.

"Estoy firmando cada página."

"Nosotros normalmente solo firmamos la última", dijo.

Meneé mi cabeza ante su ingenuidad, y continué firmando cada página. Un banco de New York al que conozco muy bien ha alterado --en más de una ocasión-- el texto de un contrato firmado, substituyendo alguna de las páginas. Entonces, cuando aparecía algún problema con el contrato, el banco no entendía a que podían deberse las diferencias entre la versión que tenía el banco y la que tenía el cliente.

Cuando surgen estas discrepancias, inevitablemente se sugiere que "la muchacha que clasifica las copias" ha mezclado páginas de una versión anterior del contrato con las de la final. Y cada una de las partes asegurará que su versión es la final.

En cuanto a mi, elegí simplemente firmar cada pagina para evitar problemas en cualquier cosa que firmé, incluso si tales accidentes no habrían podido responder a ningún propósito excepto el de perder tiempo.

Las ***firmas digitales*** tienen la agradable característica de permitir verificar la autenticidad del documento o contrato firmado ("mensaje"), además de permitir verificar la identidad del firmante. Si se cambia siquiera una letra del mensaje original, ya no podrá verificar la firma del mensaje.

¿Que Son Las Firmas Digitales?

Las firmas digitales son análogas a las firmas manuscritas. Estas últimas están basadas en la forma física en que la persona firma su nombre. Pero pueden ser fácilmente falsificadas.

Una firma digital es una precisa forma matemática de adjuntar la identidad de una persona a un mensaje. Son mucho más difíciles de falsificar que las firmas escritas, y el mensaje firmado no puede ser modificado sin invalidar la firma.

Las firmas digitales se basan en la criptografía de clave pública. Este tipo de sistemas criptográficos utiliza dos "claves". Coja un mensaje y aplíquele una de las claves en el proceso de cifrado, y al final obtendrá un mensaje distorsionado o "cifrado" (o, en el contexto actual, "firmado"). Aplíquele la otra clave al mensaje distorsionado en un proceso de descifrado u obtendrá el mensaje original.

Una de esas claves es "***publica***". Todo el mundo conoce esa clave --o puede obtenerla, como si fuera un número de teléfono. La otra clave es "***privada***". Solo tu conoces tu clave privada. Al firmar (cifrar) algo con tu clave privada, le estas poniendo tu sello personal. Nadie más puede hacerlo, ya que ellos no conocen tu clave privada.

Tratando el Mensaje

Por razones prácticas, cuando se utilizan las firmas digitales la gente no firma (cifra) realmente el mensaje original con su clave privada. Primero se produce una versión condensada del mensaje. Esta versión condensada a menudo se llama un "resumen del mensaje" (*digest*). Otros términos equivalentes son "*hash* del mensaje"o "huella digital".

Lo que se firma, entones, es el resumen del mensaje. El usuario firma el resumen del mensaje con su clave privada. Cualquiera puede descifrar el resumen del mensaje utilizando la clave pública del Firmante.

Existen diferentes algoritmos para resumir mensajes. Más adelante veremos ejemplos de cuatro de ellos: MD5, RIPEM128, SHA-1 y RIPEM168.

Así es como funciona el proceso de firma. Supongamos que tenemos un emisor y un receptor de un mensaje. El receptor desea verificar que el mensaje no haya sido alterado, así como la identidad del remitente:

Emisor

S1. El emisor envía un mensaje.

S2. El emisor calcula un resumen (*hash*) del mensaje original.

S3. El emisor firma este resumen del mensaje con su clave privada, y envía el resumen firmado junto con el mensaje original

Receptor

R1. El receptor recibe el mensaje, así como el resumen firmado.

R2. El receptor también calcula por separado un resumen del mensaje (*hash*) para el mensaje recibido.

R3. El receptor utiliza la clave pública del remitente para descifrar el resumen firmado del mensaje que recibió, y compara el resultado con el resumen calculado en el paso de progresión R2. Si el resumen del mensaje calculado es igual que el valor descifrado del resumen firmado, entonces el receptor ha verificado que el mensaje no se ha alterado. Además, puesto que la clave pública del remitente fue utilizada para el descifrado, el receptor puede estar seguro de la identidad del remitente.

Así es como funciona. Bueno, excepto por los números. Cifrar un mensaje implica procesarlo numéricamente de alguna forma.

Numerando

Para poder realizar una firma digital, es necesario primero convertir el mensaje en un **Número**. Este **Número** es entregado a la función de *Hash*, que produce el resumen del mensaje. Esta función convierte un **número grande** (el mensaje) en un **número pequeño** (el resumen).

Para que esto funcione, no debería ser sencillo encontrar dos **mensajes** que produjeran el mismo **resumen**. Si se pudiera hacer, podrías cambiar el mensaje correspondiente a una firma, como aquel banco que cambió páginas internas del contrato.

El **número pequeño** del resumen suele tener una longitud de 128 bits (MD5, RIPEM128), o de 160 bits (SHA-1, RIPEM160). Cada bit puede ser tanto un "0" como un "1". Por lo tanto existen 2 elevado a 128 posibles resúmenes de 128 bits de largo, o 2 elevado a 160 resúmenes de 160 bits.

Ahora bien, el proceso de obtención del resumen a partir del mensaje debe ser determinístico. Debe ser repetible. El mismo mensaje siempre debe dar el mismo resumen. Sino, el proceso de verificación no funcionaría.

Pero, al mismo tiempo, la salida de la función de *hash* debe parecer aleatoria. Debería resultar imposible obtener el mensaje a partir del resumen. De otra manera, alguien podría obtener varios mensajes que tendrían el mismo resumen.

Para que una función de *hash* sea buena, debe ser una función unidireccional. Debe funcionar en un sentido, pero no en el contrario. Además debe ser muy difícil encontrar dos mensajes diferentes que produzcan el mismo resumen.

Cuando ya dispone del resumen del mensaje, el **número pequeño**, debe firmarlo (cifrarlo). Esto también involucra una transformación matemática. Una forma muy común de cifrar un número (por ejemplo el algoritmo RSA) consiste en elevar ese número a una potencia, y después tomar el resultado de la exponenciación modulo **n**, donde **n** es un entero grande. (Módulo **n** significa dividir el número por **n**, y coger el resto de la división. Por ejemplo, 11 modulo 5 es igual a 1, porque el cociente es 2 y el resto es 1.)

Muy bien. Con esto es suficiente. Veamos un ejemplo.

Un Ejemplo de Mensaje

Supongamos que el mensaje que debe ser firmado es la siguiente carta del Gran Banco del Caribe al Senador Hand N. Till:

```
**********

Senator Hand N. Till
Washington, DC

Dear Senator:

This letter is to inform you we have
opened account no. 338907021 on your
behalf, and deposted therein a
legislative incentive payment of
$1 million.  For withdrawl, you will
need to use your password BJRGUD7693.

Yours very truly,

Arnold C. Creole
Vice President
Greater Caribbean Bank
**********
```

El mensaje empieza por la "S" de "Senator" y termina pon la "k" de "Bank". Cada línea excepto la última acaba con un retorno de carro (CR) y un salto de línea (LF), que indica el fin de la actual línea y el principio de la siguiente.

Destacar que hay una falta de ortografía en la carta: "withdrawl" en lugar de "withdrawal". Pero el mensaje ha sido firmado tan pronto se ha acabado de escribir, si se corrige la falta de ortografía, la firma no se correspondería con la carta.

Veamos ahora los pasos seguidos para firmar esta carta ("mensaje"). Primero, queremos convertir el mensaje en un número. Existen distintos metodos. Pero la forma más simple consiste en utilizar la codificación ASCII de los caracteres que forman el mensaje. Cada código ASCII es un número binario. Después de esta conversión, podemos concatenar todos los número binarios para obtener un gran número.

Empecemos por la primera frase. La traducción es la siguiente:

Carácter	Valor Binario	Valor Hex	Valor Decimal
S	1010011	53	83
e	1100101	65	101
n	1101110	6E	110
a	1100001	61	97
t	1110100	74	116
o	1101111	6F	111
r	1110010	72	114
	0100000	20	32
H	1001000	48	72
a	1100001	61	97
n	1101110	6E	110
d	1100100	64	100
	0100000	20	32
N	1001110	4E	78
.	0101110	2E	46
	0100000	20	32
T	1010100	54	84
i	1101001	69	105
l	1101100	6C	108
l	1101100	6C	108
CR	0001101	0D	13
LF	0001010	0A	10

La representación binaria de todo el mensaje para el Senador Till es:

```
1010011  1100101  1101110  1100001  1110100  1101111 1110010
0100000  1001000  1100001  1101110  1100100  0100000 1001110
0101110  0100000  1010100  1101001  1101100  1101100 0001101
0001010  1010111  1100001  1110011  1101000  1101001 1101110
1100111  1110100  1101111  1101110  0101100  0100000 1000100
1000011  0001101  0001010  0001101  0001010  1000100 1100101
1100001  1110010  0100000  1010011  1100101  1101110 1100001
1110100  1101111  1110010  0111010  0001101  0001010 0001101
0001010  1010100  1101000  1101001  1110011  0100000 1101100
1100101  1110100  1110100  1100101  1110010  0100000 1101001
1110011  0100000  1110100  1101111  0100000  1101001 1101110
1100110  1101111  1110010  1101101  0100000  1111001 1101111
1110101  0100000  1110111  1100101  0100000  1101000 1100001
1110110  1100101  0001101  0001010  1101111  1110000 1100101
1101110  1100101  1100100  0100000  1100001  1100011 1100011
1101111  1110101  1101110  1110100  0100000  1101110 1101111
0101110  0100000  0110011  0110011  0111000  0111001 0110000
0110111  0110000  0110010  0110001  0100000  1101111 1101110
0100000  1111001  1101111  1110101  1110010  0001101 0001010
1100010  1100101  1101000  1100001  1101100  1100110 0101100
0100000  1100001  1101110  1100100  0100000  1100100 1100101
1110000  1101111  1110011  1110100  1100101  1100100 0100000
1110100  1101000  1100101  1110010  1100101  1101001 1101110
0100000  1100001  0001101  0001010  1101100  1100101 1100111
1101001  1110011  1101100  1100001  1110100  1101001 1110110
1100101  0100000  1101001  1101110  1100011  1100101 1101110
1110100  1101001  1110110  1100101  0100000  1110000 1100001
1111001  1101101  1100101  1101110  1110100  0100000 1101111
1100110  0001101  0001010  0100100  0110001  0100000 1101101
1101001  1101100  1101100  1101001  1101111  1101110 0101110
```

```
0100000  0100000  1000110  1101111  1110010  0100000 1110111
1101001  1110100  1101000  1100100  1110010  1100001 1110111
1101100  0101100  0100000  1111001  1101111  1110101 0100000
1110111  1101001  1101100  1101100  0001101  0001010 1101110
1100101  1100101  1100100  0100000  1110100  1101111 0100000
1110101  1110011  1100101  0100000  1111001  1101111 1110101
1110010  0100000  1110000  1100001  1110011  1110011 1110111
1101111  1110010  1100100  0100000  1000010  1001010 1010010
1000111  1010101  1000100  0110111  0110110  0111001 0110011
0101110  0001101  0001010  0001101  0001010  1011001 1101111
1110101  1110010  1110011  0100000  1110110  1100101 1110010
1111001  0100000  1110100  1110010  1110101  1101100 1111001
0101100  0001101  0001010  0001101  0001010  1000001 1110010
1101110  1101111  1101100  1100100  0100000  1000011 0101110
0100000  1000011  1110010  1100101  1101111  1101100 1100101
0001101  0001010  1010110  1101001  1100011  1100101 0100000
1010000  1110010  1100101  1110011  1101001  1100100 1100101
1101110  1110100  0001101  0001010  1000111  1110010 1100101
1100001  1110100  1100101  1110010  0100000  1000011 1100001
1110010  1101001  1100010  1100010  1100101  1100001 1101110
 0100000  1000010  1100001  1101110  1101011
```

Notar que los espacios que hay entre los números binarios no tienen ningún tipo de significado. Sólo son para que resulta más fácil su lectura. El número realmente es una larga tira de ceros y unos, todos seguidos, sin espacios.

Las cadenas de ceros y unos ocupan mucho espacio, por eso normalmente la gente no utiliza representación binaria. Los número hexadecimales (base 16) necesitan una cuarta parte de los carácteres que precisa la representación en binario. Por este motivo escribiremos el número correspondiente al mensaje en representación hexadecimal en lugar de en binario. Remarcar que la notación hexadecimal utiliza los numerales del 0 al 9 y las letras de la "a" a la "f", donde "a" equivale al decimal 10,"b" al decimal 11, y así hasta llegar a la "f", que equivale al decimal 15. (Es muy común que los números hexadecimales se escriban utilizando las letras en mayúsculas de la "A" a la "F". Tienen el mismo significado que las letras en minúsculas, esto es, los número decimales del 10 al 15.)

Así, el mismo mensaje al Senador Till, escrito en notación hexadecimal, es:

```
53 65 6e 61 74 6f 72 20 48 61 6e 64 20 4e 2e 20 54 69 6c 6c 0d 0a
57 61 73 68 69 6e 67 74 6f 6e 2c 20 44 43 0d 0a 0d 0a 44 65 61 72
20 53 65 6e 61 74 6f 72 3a 0d 0a 0d 0a 54 68 69 73 20 6c 65 74 74
65 72 20 69 73 20 74 6f 20 69 6e 66 6f 72 6d 20 79 6f 75 20 77 65
20 68 61 76 65 0d 0a 6f 70 65 6e 65 64 20 61 63 63 6f 75 6e 74 20
6e 6f 2e 20 33 33 38 39 30 37 30 32 31 20 6f 6e 20 79 6f 75 72 0d
0a 62 65 68 61 6c 66 2c 20 61 6e 64 20 64 65 70 6f 73 74 65 64 20
74 68 65 72 65 69 6e 20 61 0d 0a 6c 65 67 69 73 6c 61 74 69 76 65
20 69 6e 63 65 6e 74 69 76 65 20 70 61 79 6d 65 6e 74 20 6f 66 0d
0a 24 31 20 6d 69 6c 6c 69 6f 6e 2e 20 20 46 6f 72 20 77 69 74 68
64 72 61 77 6c 2c 20 79 6f 75 20 77 69 6c 6c 0d 0a 6e 65 65 64 20
74 6f 20 75 73 65 20 79 6f 75 72 20 70 61 73 73 77 6f 72 64 20 42
4a 52 47 55 44 37 36 39 33 2e 0d 0a 0d 0a 59 6f 75 72 73 20 76 65
72 79 20 74 72 75 6c 79 2c 0d 0a 0d 0a 41 72 6e 6f 6c 64 20 43 2e
20 43 72 65 6f 6c 65 0d 0a 56 69 63 65 20 50 72 65 73 69 64 65 6e
74 0d 0a 47 72 65 61 74 65 72 20 43 61 72 69 62 62 65 61 6e 20 42
61 6e 6b
```

¿A que resulta mucho más claro?

El siguiente paso consiste en firmar este mensaje, de tal manera que el Senador Till sabrá que vino del Gran Banco del Caribe, y no de, digamos, *Louis Freeh's Henchcreatures*. Esto significa que debemos usar una función de *hash*.

Resumiendo El Mensaje

Aquí tenemos cuatro resúmenes producidos por cuatro funciones de *hash* distintas.

Función de *hash*	Resumen del mensaje
MD5	5670E64BF6CEBB46 31A25CF6990F82C0
RIPEM128	B4BB17FD0E09091A 2DF095F0B9647B41
SHA-1	1A01B56EB33FA84A 39EEDDD927977726 38331E94
RIPEM160	ABA54F46348F56D1 E492AE09A472D143 9D64E0F1

Sólo por curiosidad, veamos que pasa con los resúmenes si realizamos pequeñas alteraciones al mensaje original. Primero, corregiremos el fallo de ortografía de "withdrawal". Con esta corrección, ahora el mensaje es:

```
**********

Senator Hand N. Till
Washington, DC

Dear Senator:

This letter is to inform you we have
opened account no. 338907021 on your
behalf, and deposted therein a
legislative incentive payment of
$1 million.  For withdrawal, you will
need to use your password BJRGUD7693.

Yours very truly,

Arnold C. Creole
Vice President
Greater Caribbean Bank
**********
```

Con este cambio, los resúmenes del mensaje son totalmente diferentes, como se puede ver en la siguiente tabla:

Función de *hash*	Resumen del mensaje
MD5	8BDF43C9BC320AE8 874E9EED73DDCF55
RIPEM128	5BF83DAE28ACBF49 BD9EDB6D26DE1EE9
SHA-1	22257C5870B7CB00 E6B5AABA45913C24 DEAB6F7D
RIPEM160	713FE23D55F95ACD B5D744C0B616774B 1DDAA4E7

Como segundo ejemplo, dejemos "withdrawl" tal y como estaba, pero cambiaremos en número de cuenta, de tal manera que el último dígito sea un "2" en lugar de un "1". De esta manera el mensaje quedaría como:

```
**********

Senator Hand N. Till
Washington, DC

Dear Senator:

This letter is to inform you we have
opened account no. 338907022 on your
behalf, and deposted therein a
legislative incentive payment of
$1 million.  For withdrawl, you will
need to use your password BJRGUD7693.

Yours very truly,

Arnold C. Creole
Vice President
Greater Caribbean Bank
**********
```

A continuación podemos ver los resúmenes del nuevo mensaje con el dígito alterado:

Función de *hash*	Resumen del mensaje
MD5	0742CD5D4EA8B857 E57352C6B21CE7FB
RIPEM128	9BE546E061E64BD3 3659E49569774A25
SHA-1	B55F080E45CF15D9 3542A9F4C7CED8B3 48EF1E17
RIPEM160	C97428911B8C925E 990839FE2BB5B9E9 BD0EC4EF

Con estos ejemplos hemos demostrado que si se altera el mensaje original, el valor de la función *hash* también cambiará. Por tanto, el resumen del mensaje puede utilizarse para comprobar la integridad del mensaje original.

El Mensaje Firmado

Para nuestra firma digital, utilizaremos la función de *hash* SHA-1:

```
hash = 1A01B56EB33FA84A 39EEDDD927977726 38331E94
```

El siguiente paso consiste en cifrar el resumen del mensaje con la clave privada del firmante. Esto significa que necesitamos algún método de cifrado, con un par de claves (pública/privada). La clave privada se usará para firmar, mientras que la clave pública se utilizará para la verificación.

Usaremos el sistema RSA. Este sistema firma valores elevandolos al exponente de la clave privada **d**, tomando el resultado módulo **n**. Supongamos que **n** y **d** tienen los siguientes valores:

```
n =

AF8207E25BF6A83E 17F5980E23AB498F 5B77E84821905716
EEDBD4F621EDE141 D38117147AD98E8B 549BC35B0C689475
D710013D83AE1945 FF405E2D4EC54A56 CB88BBB220C31F12
63978A307A36F0C2 316CA3CB921271B3 7B550728D560A884
4E1740C31DFCA3BC 0FB80D34AAF986D7 02FD633BD26F16AA
8A3C329E2D1E970D

d =

0F2591B89F67322D E9B3706408000861 2EEBB248475D45A6
DD066BE2B21AED8D D8CB134AD92F5D75 F8DF5884CB155B7A
B00CD98E8D86C0F7 A187D498E46B7276 D6881D1502BA90C5
5EE073BB1565A969 681C3CE6660A8744 B6D54C6659299E1A
B8424F41DE1B277C D22DC1D81DEA69B2 8F99A4C9C852A82C
6123EDAAB94D0CA1
```

Donde **n** es un número de 1024 bits, que es lo mismo que 128 bytes o 256 números hexadecimales. Igualmente con **d**, excepto que **d** tiene cuatro ceros al principio, por lo que realmente solo tiene 1020 bits de longitud.

Lo que queremos hacer es elevar el ***hash*** al exponente **d**, y despues tomar el resultado módulo **n**.

Esta es la idea. Sin embargo, el procedimiento de firma del RSA especifica que el valor *hash* 1A01B56EB33FA84A 39EEDDD927977726 38331E94 debe ser rellenado (*padded*) para que tenga la misma longitud que **n**. Entonces, como el *hash* tiene 160 bits, los bits de relleno --exactamente, 864 bits-- se añaden delante para que tenga 1024 bits de longitud.

Todo esto esta detallado en el RSA's Public Key Cryptography Standard No. 1 (PKCS#1). Explicar los motivos de este procedimiento se salen de los objetivos de este artículo. Pero diremos que añadir bits de relleno, en principio, no supone ningún riesgo. (dependiendo de como lo hagamos). Siempre podemos cifrar el *hash* con el relleno, y más tarde eliminar ese relleno una vez hayamos descifrado el criptograma.

Según el PKCS#1 debemos añadir 00 delante del *hash*, quedando:

> 00 1A01B56EB33FA84A 39EEDDD927977726 38331E94

Esta es la cola del número de relleno. El principio de ese número es 00 02, seguido de tantos números hexadecimales obtenidos aleatoriamente como sean necesarios para obtener un total de 256 dígitos hexadecimales (1024 bits).

De esta manera, el número debería ser algo como:

00 02 . . . números aleatorios . . 00 1A01B56EB33FA84A 39EEDDD927977726 38331E94

Este es uno de los conjuntos de reglas disponibles. Por supuesto, no todo el mundo sigue las mismas reglas. Yo estoy haciendo los cálculos con el lenguaje de programación Java, usando las bibliotecas de clases de Cryptix. Cryptix rellena el *hash* usando una variante del PKCS#1, obteniendo el siguiente entero:

```
entero con relleno =

01FFFFFFFFFFFFFF FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF FFFFFF0030213009
06052B0E03021A05 0004141A01B56EB3 3FA84A39EEDDD927
97772638331E94
```

Podemos ver que el relleno empieza por 01, al que le siguen un montón de FFFFFF's, después 00, un número misterioso "30213009 06052B0E03021A05 000414", y finalmene el *hash* 1A01B56EB33FA84A 39EEDDD927977726 38331E94.

Muy bien, sigamos. Tomemos el **entero con relleno**, lo elevamos a la potencia **d**, y reducimos el resultado módulo **n**. Esto nos da la firma digital **firm**. Si hacemos los cálculos, obtenemos:

```
firm =

149522ECA960A6B4 A46F1546B6D5F74B C3570CD7DD981EA1
0B506B346FB159BE 7F7BAA26F6A8A143 090B4D0A944AE4D7
96C17A4587267B05 A991D76EDE989583 9E47C19054CDB818
5BD21EE36BAC9803 CE005383A1083AB5 79411AB26BE28631
1BF17D021002B6D5 2EE82CEB2FC554A8 BDE5874D82B20B9F
21EBD65F5FD93102
```

Esta es la firma digital que se enviará junto a la carta al Senador Till para que pueda verificar su autenticidad, así como la identidad del firmante.

Verificando La Firma

Cuando el Senador Till recibe el mensaje, calcula su propio *hash* (llamémosle **hash***) a partir del mensaje recibido. Debería obtener el valor:

```
hash* = 1A01B56EB33FA84A 39EEDDD927977726 38331E94
```

si el mensaje no ha sido alterado. Seguidamente compara hash* con el valor descifrado de **firm** para ver si son iguales.

Para descifrar **firm**, obtiene la clave pública del Gran Banco del Caribe (**e**, **n**), y ve que **e** tiene un valor (en hexadecimal):

```
e = 010001
```

Pequeño y dulce. Pero no muy inteligente, ya que muchas organizaciones a parte del Gran Banco del Caribe pueden estar usando el mismo valor de **e**. (Sinembargo, lo que la hace única es la combinación de **e** con el módulo **n**. Siempre que **n** sea único, no tendremos porque preocuparnos de **e**.)

Así que el Senador Till toma el valor **firm**, lo eleva a **e** y se queda con el módulo **n** del resultado.Esto produce el número (confíen en mi, he hecho los cálculos):

```
01FFFFFFFFFFFFFF FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF FFFFFF0030213009
06052B0E03021A05 0004141A01B56EB3 3FA84A39EEDDD927
97772638331E94
```

Una vez que el Senador Till quita el relleno, él (o, más bien, su software) ve que el final es igual que el valor **hash***, lo nos asegura la validez de la firma.

Y así es como funcionan las firmas digitales.

J. Orlin Grabbe
Artículo original(1)

Lista de enlaces de este artículo:

1. http://www.aci.net/kalliste/digsig.htm
2. http://www.aci.net/kalliste/

E-mail del autor: daneel _ARROBA_ bulma.net
Podrás encontrar este artículo e información adicional en: http://bulma.net/body.phtml?nIdNoticia=868