Firmas digitales DSA y ElGamal

Pedro Palacios Almendros

Álgebra Computacional
Universidad Complutense de Madrid

11 de noviembre de 2022

Firmas digitales: Motivación

Muchas aplicaciones requieren poder verificar la autenticidad de un mensaje:

- Trámites oficiales con la administración pública
- Firma de contratos electrónicos
- Verificación de la integridad de archivos y descargas

Firmas digitales: Propiedades

Definición

Una firma digital es un mecanismo criptográfico que permite al receptor de un mensaje firmado digitalmente identificar al emisor de dicho mensaje y cumple las siguientes propiedades:

- **Infalsificable**. Las firmas deben poder ser generadas solamente por el firmante.
- **Inalterable**. La firma digital confirma que el mensaje no ha sido alterado. Si se altera el mensaje, la firma deja de ser válida.
- **No repudiable**. Una vez realizada la firma, el firmante no puede *echarse atrás* e invalidarla.
- **Verificable**. Las firmas deben ser fácilmente verificables por los receptores de las mismas.

Funciones hash

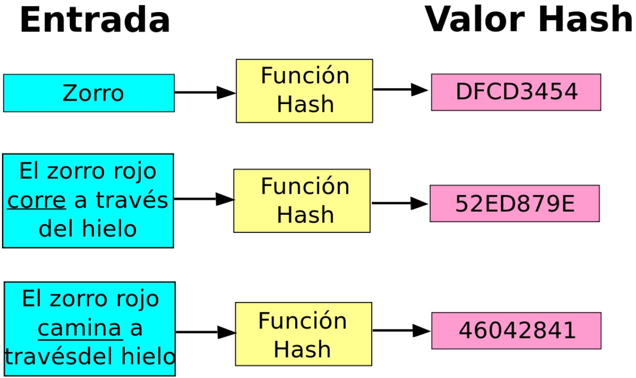
Definición

Una función hash es una función de un conjunto de tamaño de entrada (cadenas) a un conjunto de tamaño fijo de salida con las siguientes propiedades:

- **Uniformidad**. Todos los elementos del conjunto de salida deberían tener relativamente la misma probabilidad.
- **Eficiencia**. Calcular la función hash de una cadena de entrada debe ser rápido.
- **Irreversibilidad**. No debe ser factible obtener (una o varias) cadenas cuya imagen sea cierto elemento dado.
- **Determinismo**. La función hash de una cadena debe ser siempre la misma.

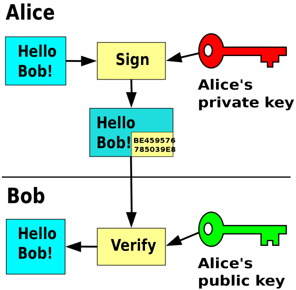
Funciones hash

Las funciones hash resumen una cadena de ancho variable en un valor de un conjunto finito de tamaño fijo (por ejemplo un elemento de $\mathbb{Z}/n\mathbb{Z}$).



Las cadenas hash pueden usarse para verificar la integridad de un mensaje. Si el mensaje cambia un poco, su hash cambiará sustancialmente.

Claves pública y privada



El emisor tiene dos claves (una pública y una privada) que se utilizan de forma dual para firmar y verificar:

- La **clave privada** solo la conoce el emisor, y la utiliza para firmar el mensaje digitalmente.
- La **clave pública** se publica a todo el mundo, y cualquiera la puede utilizar para verificar el mensaje.

ElGamal: Protocolo de firma y verificación

El protocolo para firmar y verificar un mensaje consiste en los siguientes pasos:

1. Generación de parámetros compartidos por todos los usuarios del sistema
2. Generación de claves diferentes para cada usuario del sistema
3. Distribución de las claves públicas entre los usuarios
4. Firma por parte del emisor del mensaje
5. Verificación por parte del receptor

ElGamal: Generación de parámetros

Se eligen parámetros que serán compartidos por todos los usuarios del sistema.

1. Se elige un tamaño de clave N
2. Se elige un número primo p de N bits
3. Se elige una función hash H cuyo conjunto de salida son los enteros de L bits, con $L \leq N$
4. Se elige un generador $g < p$ del grupo multiplicativo de los enteros módulo p, $(\mathbb{Z}/p\mathbb{Z})^*$.

Los parámetros del protocolo son (p, H, g).

ElGamal: Generación y distribución de claves

Dados los parámetros del sistema, se elige una clave privada y una pública distintas para cada usuario.

1. Se escoge aleatoriamente $x \in \{1, ..., p-2\}$.
2. Calculamos $y := g^x \mod p$ usando exponenciación modular.

La **clave privada** es x, la **clave pública** es y.

Nótese que para obtener la clave privada desde la clave pública habría que resolver el problema del logaritmo discreto, para el que no se conoce ninguna implementación efectiva para el caso general.

Únicamente la **clave pública** se distribuye a todos los usuarios, la clave privada se mantiene secreta.

ElGamal: Firma

Supongamos que tenemos un mensaje m que queremos firmar:

1. Elegimos un entero $k \in \{2, ..., p-2\}$ tal que k sea coprimo con $p-1$
2. Calculamos $r := g^k \mod p$ usando exponenciación modular
3. Computamos $k^{-1} \mod (p-1)$ utilizando el algoritmo extendido de Euclides, que existe pues $\gcd(k, p-1) = 1$
4. Computamos
$$s := (H(m) - x \cdot r) \cdot k^{-1} \mod (p-1)$$
5. En el caso improbable de que $s = 0$, empezamos de nuevo con otro k aleatorio

El resultado del proceso es la firma digital, que será la tupla (r, s).

ElGamal: Verificación

Para verificar que una firma (r, s) es válida para un mensaje m hacemos la siguiente comprobación:

1. Comprobamos que $0 < r < p$ y que $0 < s < p - 1$
2. La firma será válida si y sólo si

$$g^{H(m)} \equiv y^r \cdot r^s \pmod{p}$$

ElGamal: Corrección

Demostremos que el algoritmo es correcto, es decir, que una firma generada por el algoritmo de verificación será aceptada por el verificador.

- Recordamos la definición de s en la firma

$$s := (H(m) - x \cdot r) \cdot k^{-1} \mod (p-1)$$

- Multiplicando por k y despejando, se tiene que

$$H(m) \equiv xr + sk \mod (p-1)$$

- Como g es generador de $(\mathbb{Z}/p\mathbb{Z})^*$, tiene orden $\phi(p) = p-1$, por lo que $g^{\alpha} \equiv g^{\alpha \mod (p-1)} \mod p$. Aplicando esto:

$$\begin{aligned} g^{H(m)} &\equiv g^{xr+sk} \mod p \\ &\equiv (g^x)^r \left(g^k\right)^s \mod p \\ &\equiv (y)^r (r)^s \mod p \end{aligned}$$

ElGamal: Ejemplo

- Supongamos que los parámetros del sistema son $p = 13$, $H(m) = m$, $g = 2$.
- Supongamos que el usuario elige la clave privada $x = 3$, por lo que su clave pública será $y := g^x \mod p \equiv 2^3 \mod 13 \equiv 8 \mod 13$.
- Supongamos que queremos firmar el mensaje $m = 11$ y elegimos $k = 5$ ($\gcd(k, p-1) = \gcd(5, 12) = 1$):
  - $r := g^k \mod p \equiv 2^5 \mod 13 \equiv 32 \mod 13 \equiv 6 \mod 13$
  - Con el algoritmo extendido de Euclides hallamos que $5 \cdot 5 - 2 \cdot 12 = 1$, por lo que $k^{-1} \equiv 5 \mod 12$
  - $s := (H(m) - x \cdot r) \cdot k^{-1} \mod (p-1) \equiv (11 - 3 \cdot 6) \cdot 5 \mod 12 \equiv 1 \mod 12$
- La firma es por tanto $(r, s) = (6, 1)$. Verifiquemos que es correcta:
  - $g^{H(m)} \mod p \equiv 2^{11} \mod 13 \equiv 2048 \mod 13 \equiv 7 \mod 13$
  - $(y)^r (r)^s \mod p = 8^6 \cdot 6^1 \mod 13 \equiv 1572864 \mod 13 \equiv 7 \mod 13$

ElGamal: Seguridad

La seguridad de ElGamal radica en la dificultad de resolver ciertos problemas que se consideran difíciles:

- Hallar la clave privada usando la clave pública. Requiere resolver el problema del logaritmo discreto: $y := g^x \mod p$.
- Encontrar colisiones en la función hash: $H(m) = H(M) \mod (p-1)$
- Falsificar firmas: dados (g, m, y, p), encontrar (r, s) tal que $g^{H(M)} \equiv y^r r^s \mod p$
    - Si se fija r y se intenta despejar s, el problema es equivalente a resolver el logaritmo discreto.
    - Si se fija s y se intenta despejar r, no se ha demostrado que el problema sea NP, pero tampoco se ha encontrado una solución eficiente.
    - Tampoco se ha encontrado un algoritmo eficiente para despejar (r, s) a la vez.

ElGamal: Ataques

- Si se usa la misma k para dos mensajes distintos, podríamos obtener la clave privada x resolviendo el sistema de ecuaciones lineales:

$$\begin{cases} H(m_1) \equiv x \cdot r_1 + s_1 \cdot k \mod (p-1) \\ H(m_2) \equiv x \cdot r_2 + s_2 \cdot k \mod (p-1) \end{cases}$$

  El sistema tiene dos ecuaciones y dos incógnitas (x y k), el resto son datos conocidos.
- Generar firmas para ciertos mensajes derivados a partir de una firma valida (si $H(M) = M$):
  - Si un emisor legítimo firma un mensaje M, seleccionando arbitrariamente A, B, C arbitrariamente tal que $\gcd(Ar–Cs, p-1) = 1$, entonces se puede ver que (r', s') firma m' donde

$$r' \equiv r^A g^B y^C \mod p$$

$$s' \equiv \frac{sr'}{Ar - Cs} \mod (p-1)$$

$$m' \equiv \frac{r'(Am + Bs)}{Ar - Cs} \mod (p-1)$$

DSA

DSA: Generación de parámetros

Se eligen parámetros que serán compartidos por todos los usuarios del sistema.

1. Se elige una función hash H con tamaño de salida $|H|$.
2. Se elige un tamaño de clave L. Originalmente L era un múltiplo de 64 entre 512 y 1024.
3. Se elige un tamaño del módulo $N < L$. Si $N < |H|$, tomamos únicamente los últimos N bits de la función hash. Originalmente $(L, N) \in \{(1024, 160), (2048, 224), (2048, 256), (3072, 256)\}$
4. Se elige un primo q de N bits
5. Se elige un primo p de L bits tal que $q | (p - 1)$
6. Se elige $h \in \{2, ..., p - 2\}$. Usualmente $h = 2$
7. Se elige $g := h^{\frac{p-1}{q}} \mod p$. En el caso improbable de que $g = 1$, se elige otro h distinto.

Los parámetros del protocolo son (p, q, H, g).

DSA: Generación y distribución de claves

Dados los parámetros del sistema, se elige una clave privada y una pública distintas para cada usuario.

1. Se escoge aleatoriamente $x \in \{1, ..., q-1\}$.
2. Calculamos $y := g^x \mod p$ usando exponenciación modular.

La **clave privada** es x, la **clave pública** es y.

Nótese que al igual que en ElGamal, para obtener la clave privada desde la clave pública habría que resolver el problema del logaritmo discreto, para el que no se conoce ninguna implementación efectiva para el caso general.

De nuevo, únicamente la **clave pública** se distribuye a todos los usuarios, la clave privada se mantiene secreta.

DSA: Firma

Supongamos que tenemos un mensaje m que queremos firmar:

1. Elegimos un entero $k \in \{1, ..., q-1\}$
2. Calculamos $r := \left(g^k \mod p\right) \mod q$. En el caso improbable de que $r = 0$, empezar de nuevo con otro k.
3. Computamos $k^{-1} \mod q$ utilizando el algoritmo extendido de Euclides, que existe pues q es primo.
4. Computamos
$$s := \left(k^{-1}\left(H(m) + xr\right)\right) \mod q$$
5. En el caso improbable de que $s = 0$, empezamos de nuevo con otro k aleatorio

El resultado del proceso es la firma digital, que será la tupla (r, s).

DSA: Verificación

Para verificar que una firma (r, s) es válida para un mensaje m hacemos la siguiente comprobación:

1. Comprobamos que $0 < r < q$ y que $0 < s < q$
2. Calculamos $w = s^{-1} \mod q$
3. Calculamos $u_1 = H(m)w \mod q$
4. Calculamos $u_2 = r \cdot w \mod q$
5. Calculamos $v = (g^{u_1} y^{u_2} \mod p) \mod q$
6. La firma será válida si y sólo si $v \equiv r \mod q$

DSA: Corrección (1)

Demostremos que el algoritmo es correcto, es decir, que una firma generada por el algoritmo de verificación será aceptada por el verificador.

- Primero notamos que como $g = h^{\frac{p-1}{q}} \mod p$, aplicando el pequeño teorema de Fermat, $g^q \equiv h^{p-1} \equiv 1 \mod p$. Por tanto, g tiene orden $\leq q$. Como $g > 1$ y q es primo y el orden de g divide a q, entonces g tiene orden q.
- El firmante calculó

$$s := \left(k^{-1} \left(H(m) + xr\right)\right) \mod q$$

Como $s \not\equiv 0 \mod q$, $\exists s^{-1}$, podemos despejar k:

$$\begin{aligned} k &\equiv H(m)s^{-1} + xrs^{-1} \mod q \\ &\equiv H(m)w + xrw \mod q \end{aligned}$$

- Como g tiene orden q

$$\begin{aligned} g^k &\equiv g^{H(m)w} g^{xrw} \mod p \\ &\equiv g^{H(m)w} y^{rw} \mod p \end{aligned}$$

DSA: Corrección (2)

- Hemos visto que

$$
\begin{aligned}
g^k &\equiv g^{H(m)w} y^{rw} \mod p \\
&\equiv g^{u_1} y^{u_2} \mod p
\end{aligned}
$$

- Por último, vemos que

$$
\begin{aligned}
r &\equiv \left(g^k \mod p\right) \mod q \\
&\equiv \left(g^{u_1} y^{u_2}\right) \mod p \\
&\equiv v \mod q
\end{aligned}
$$

DSA: Ejemplo (1)

- Supongamos que los parámetros del sistema son $p = 283$, $q = 47$, $H(m) = m$, $g = 60$. Se tiene que $p - 1 = 282 = 6 \cdot 47 = 6q$.
- Supongamos que el usuario elige la clave privada $x = 24$, por lo que su clave pública será $y := g^x \mod p \equiv 60^{24} \mod 283 \equiv 158 \mod 283$.
- Supongamos que queremos firmar el mensaje $m = 41$ y elegimos $k = 15$:
  - $t = g^k \mod p \equiv 60^{15} \mod 283 \equiv 207 \mod 283$
  - $r := t \mod q \equiv 207 \mod 47 \equiv 19 \mod 47$
  - Con el algoritmo extendido de Euclides hallamos que $22 \cdot 15 - 7 \cdot 47 = 1$, por lo que $k^{-1} \equiv 22 \mod 47$
  - $s := k^{-1}(H(m) + x \cdot r) \mod q \equiv 22 \cdot (41 + 24 \cdot 19) \mod 47 \equiv 30 \mod 47$
- La firma es por tanto $(r, s) = (19, 30)$.

DSA: Ejemplo (2)

- Los parámetros del sistema son $p = 283$, $q = 47$, $H(m) = m$, $g = 60$.
- La clave privada $x = 24$, y la clave pública es $y := 158 \mod 283$.
- Verifiquemos ahora que la firma $(r, s) = (19, 30)$ del mensaje $m = 41$ es correcta:
  - Con el algoritmo extendido de Euclides hallamos que $11 \cdot 30 - 7 \cdot 47 = 1$, por lo que $w := s^{-1} \mod q \equiv 11 \mod 47$
  - $u_1 := H(m)w \mod q \equiv 41 \cdot 11 \mod 47 \equiv 28 \mod 47$
  - $u_2 := rw \mod q \equiv 19 \cdot 11 \mod 47 \equiv 21 \mod 47$
  - Calculamos v:

$$
\begin{aligned}
v &:= (g^{u_1} y^{u_2} \mod p) \mod q \\
&\equiv \left(60^{28} \cdot 158^{21} \mod 283\right) \mod 47 \\
&\equiv 207 \mod 47 \\
&\equiv 19 \mod 47
\end{aligned}
$$

  - Comprobamos que efectivamente, $v \equiv 19 \equiv r \mod 47$

DSA: Seguridad

Al igual que en ElGamal, la seguridad de DSA radica en la dificultad de resolver ciertos problemas que se consideran difíciles:

- Hallar la clave privada usando la clave pública. Requiere resolver el problema del logaritmo discreto: $y := g^x \mod p$.
- Encontrar colisiones en la función hash: $H(m) = H(M) \mod (p-1)$
- Falsificar firmas: dados (g, m, y, p), encontrar (r, s) tal que

$$r = \left(g^{\left(H(m)\cdot s^{-1} \bmod q\right)} y^{\left(r\cdot s^{-1} \bmod q\right)} \mod p\right) \bmod q$$

  - No se conoce ningún algoritmo eficiente para poder despejar (r, s).

DSA: Ataques

En DSA es muy importante que el valor de k sea totalmente aleatorio y no sea predecible.

- Se descubrió la clave privada (ECDSA) que Sony usaba para firmar los programas de la Play Station 3 debido a que se reusaba siempre el mismo k.
- Si se usa el mismo k en dos firmas distintas, entonces:

$$\begin{cases} s_1 & \equiv & \left(k^{-1}\left(H(m_1) + xr_1\right)\right) \mod q \\ s_2 & \equiv & \left(k^{-1}\left(H(m_2) + xr_2\right)\right) \mod q \end{cases}$$

  Que se puede reescribir en el siguiente sistema lineal:

$$\begin{cases} ks_1 - xr_1 & \equiv & H(m_1) \mod q \\ ks_2 - xr_2 & \equiv & H(m_2) \mod q \end{cases}$$

  Como $\mathbb{Z}/q\mathbb{Z}$ es un cuerpo, podemos despejar (k, x) utilizando el método de Gauss.

Comparación entre ElGamal y DSA

- Hoy en día se utiliza sobre todo DSA en vez de ElGama, por motivos de eficiencia.
- En DSA trabajamos con dos primos p y q. La clave pública tiene la resistencia al logaritmo discreto de p (que suele ser grande). Sin embargo, la mayor parte de las operaciones se realizan módulo q (en ElGamal debemos realizar exponenciaciones modulares con exponentes de tamaño similar a p), y $q << p$.

$\|\mathbf{p}\|$	$\|\mathbf{q}\|$	**Hash**
1024	160	SHA-1
2048	224	SHA-224
2048 / 3072	256	SHA-256

- Además, las firmas (r, s) de DSA son mucho más pequeñas que las de ElGamal, pues r, s son enteros módulo q, en vez de serlo módulo p.

Generación de parámetros

Generación de números primos

Para generar un número primo seguimos los siguientes pasos:

1. Fijamos N, el tamaño en bits del número primo.
2. Generamos un número aleatorio n entre $2^{n-1}+1$ y 2^n-1, ambos inclusives.
3. Forzamos que sea impar haciendo el bit menos significativo 1.
4. Comprobamos si algunos primos pequeños (<300) precomputados con una criba de eratóstenes dividen a n. Si lo hacen, volvemos a elegir n.
5. Ejecutamos el test de primalidad de Miller-Rabin (con 20 iteraciones). Si el test indica que n un número compuesto, elegimos un nuevo n y probamos de nuevo.

Generación de parámetros ElGamal

Para generar los parámetros de ElGamal, necesitamos generar un primo p de N bits y un generador g de $(\mathbb{Z}/p\mathbb{Z})^*$. Para facilitar la tarea de encontrar una raíz primitiva, vamos a dotar de cierta estructura a los p generados:

1. Generamos un primo q de tamaño $N - 1$ bits.
2. Computamos $p = 2q + 1$ y comprobamos si es primo. Si no lo es, volvemos a elegir q aleatoriamente.
3. Por el teorema de Lagrange, como $p - 1 = 2q$ con q primo, los posibles órdenes de los elementos de $(\mathbb{Z}/p\mathbb{Z})^*$ son 1, 2, q, $p - 1$. Nótese que el único elemento de orden 2 es $p - 1 \equiv -1 \mod p$ y el único elemento de orden 1 es la identidad. Habrá $\phi(q) = q - 1$ elementos de orden q, por lo que aproximadamente el 50 % de los elementos serán generadores.
4. Elegimos aleatoriamente $g \in \{2, ..., p - 2\}$. Si $g^q \not\equiv 1 \mod p$, hemos acabado y g es un generador. En caso contrario, elegimos un nuevo g aleatoriamente.

Generación de parámetros DSA

Para generar los parámetros de ElGamal, necesitamos generar un primo q de L bits y un primo p de N bits tal que $q|(p-1)$.

- Generamos un primo q de tamaño L bits.
- Generamos un número m de $N-L$ bits.
- Comprobamos si $p = mq + 1$ es primo. Si no lo es, escogemos otro múltiplo m. Si tras muchos intentos con el mismo q no encontramos ningún múltiplo, elegimos otro q.

¿Alguna pregunta?