Seguridad y Protección de Sistemas Informáticos

Fco. Javier Lobillo Borrero

Departamento de Álgebra, Universidad de Granada

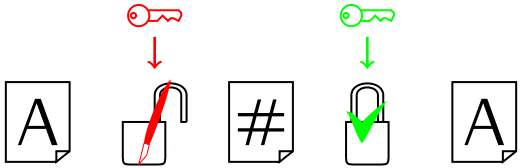
Curso 2017/2018

Índice

Autenticidad

Los criptosistemas de clave pública no garantizan la *autenticidad* del mensaje, no hay forma de saber si el mensaje o el remitente son auténticos. Tampoco puede garantizarse el *no repudio*, no hay forma de certificar que alguien ha enviado un mensaje.

La confidencialidad viene garantizada por el conocimiento de la clave privada. ¿Podemos utilizar este conocimiento para garantizar autenticidad? Si en un sistema RSA "ciframos" con la clave privada y "desciframos" con la clave pública, perdemos la confidencialidad pero obtenemos autenticidad y no repudio.



Índice

Concepto

Un sistema de firma es una quíntupla $(\mathcal{M}, \mathcal{F}, \mathcal{K}, \text{sgn}, \text{vfy})$ donde

- $\mathcal{M}$ es el espacio de los mensajes.
- $\mathcal{F}$ el conjunto de las firmas.
- $\mathcal{K}$ es el espacio de claves.
- sgn es la función firma $\text{sgn} : \mathcal{K} \times \mathcal{M} \to \mathcal{F}$.
- vfy es la función de verificación $\text{vfy} : \mathcal{K} \times \mathcal{M} \times \mathcal{F} \to \mathbb{Z}_2$
- Si $y = \text{sgn}_k(x)$ entonces $\text{vfy}_k(x, y) = 1$.
- Dados $k \in \mathcal{K}$ e $y \in \mathcal{F}$, es computacionalmente difícil calcular $x \in \mathcal{M}$ tal que $\text{vfy}_k(x, y) = 1$.

Para un sistema RSA,

$$\text{sgn}_{(n,d)}(m) = m^d \bmod n, \quad \text{vfy}_{(n,e)}(m, y) = \begin{cases} 1 & \text{si } y^e \equiv m \mod n, \\ 0 & \text{en otro caso.} \end{cases}$$

Índice

Necesidad

Con el esquema anterior la firma incrementa considerablemente el tamaño de la información. Dependiendo del criptosistema empleado, podemos duplicar el tamaño del mensaje, caso RSA, incluso triplicar, usando un esquema derivado de ElGamal. Necesitamos una herramienta que resuma un mensaje de longitud arbitraria a una cadena de tamaño fijo.

Las funciones hash criptográficas son la herramienta que necesitamos. Estas funcione se emplean en los siguientes escenarios:

- Protocolos de firma digital.
- Comprobación de la integridad de claves públicas.
- Generadores pseudoaleatorios.
- Usados con claves secretas, se convierten en códigos de autenticación de mensajes (MAC message authetication code).

Concepto

Una función hash es una función

$$h : \mathbb{B}^* \to \mathbb{B}^N$$

con las siguientes propiedades:

- Computacionalmente fácil de calcular, tanto en hardware como en software.
- Son funciones unidireccionales (*one-way*), es decir, dado $y \in \mathbb{B}^N$ es computacionalmente imposible encontrar $m \in \mathbb{B}^*$ tal que $h(m) = y$.
- Son funciones *resistentes a segundas preimágenes*, es decir, dado $m \in \mathbb{B}^*$ es computacionalmente imposible encontrar $m' \in \mathbb{B}^*$, $m' \neq m$ tal que $h(m) = h(m')$.
- Son *resistentes a colisiones*, es decir, es computacionalmente imposible encontrar $m, m' \in \mathbb{B}^*$ talen que $h(m) = h(m')$.

One-way y colisiones.

Proposición

Una función hash resistente a colisiones es unidireccional

Demostración.

Sea $n > N$ y consideremos $h : \mathbb{B}^n \to \mathbb{B}^N$. Supongamos que h no es unidireccional. Tenemos un algoritmo A que dado $y \in \mathbb{B}^N$ calcula $m \in \mathbb{B}^n$ con $h(m) = y$. Cojamos un $m \in \mathbb{B}^n$ aleatorio, y apliquemos A a $h(m)$ para calcular $m' \in \mathbb{B}^n$ tal que $h(m) = h(m')$. Calculemos la probabilidad de que $m \neq m'$. Sea $[m] = \{x \in \mathbb{B}^n \mid h(m) = h(x)\}$. La probabilidad de que $m' \neq m$ es $\frac{|[m]|-1}{|[m]|}$. La media de todas estas probabilidades será la probabilidad de encontrar una colisión. Esta media es:

$$\begin{aligned}\frac{1}{|\mathbb{B}^n|}\sum_{m\in\mathbb{B}^n}\frac{|[m]|-1}{|[m]|} &= \frac{1}{|\mathbb{B}^n|}\sum_{y\in\mathbb{B}^N}\sum_{m\in h^{-1}(y)}\frac{|[m]|-1}{|[m]|} = \frac{1}{|\mathbb{B}^n|}\sum_{y\in\mathbb{B}^N}\sum_{m\in h^{-1}(y)}\frac{|h^{-1}(y)|-1}{|h^{-1}(y)|}\\ &= \frac{1}{|\mathbb{B}^n|}\sum_{y\in\mathbb{B}^N}(|h^{-1}(y)|-1) = \frac{1}{|\mathbb{B}^n|}\sum_{y\in\mathbb{B}^N}|h^{-1}(y)| - \frac{1}{|\mathbb{B}^n|}\sum_{y\in\mathbb{B}^N}1\\ &= 1 - \frac{|\mathbb{B}^N|}{|\mathbb{B}^n|} = 1 - \frac{2^N}{2^n} = \frac{2^{n-N}-1}{2^{n-N}} \geq \frac{1}{2}.\end{aligned}$$

□

Paradoja del cumpleaños I

Sea $f : X \to Y$ una aplicación, supongamos $|X| = m$, $|Y| = M$. ¿Cuál es la probabilidad de, elegidos t elementos en X, al menos dos de ellos tengan la misma imagen?

Calculamos el suceso contrario. La probabilidad de que las imágenes de t elementos de X sean distintas es:

$$\left(1 - \tfrac{1}{M}\right)\left(1 - \tfrac{2}{M}\right)\cdots\left(1 - \tfrac{t-1}{M}\right) = \prod_{i=1}^{t-1}\left(1 - \tfrac{i}{M}\right) \approx \prod_{i=1}^{t-1} e^{-\frac{i}{N}} = e^{-\frac{t(t-1)}{2M}}.$$

La probabilidad buscada es

$$1 - e^{-\frac{t(t-1)}{2M}} = \varepsilon.$$

Despejando de esta expresión el valor de t se obtiene:

$$t(t-1) = 2M \ln(\tfrac{1}{1-\varepsilon}),$$

de donde:

$$t \approx \sqrt{2M \ln(\tfrac{1}{1-\varepsilon})}.$$

Paradoja del cumpleaños II

Obtenemos que el valor de t es directamente proporcional a la raíz cuadrada de M.
Para $\varepsilon = \frac{1}{2}$ y $M = 365$,

$$t \approx \sqrt{2 \cdot 365 \ln(2)} = \sqrt{\ln(4)}\sqrt{365} = 1,17 \cdot 19,105 = 22,4944.$$

Para $\varepsilon = \frac{1}{2}$ y $M = 2^N$,

$$t \approx \sqrt{2 \cdot 2^N \ln(2)} \approx 1,177 \cdot 2^{\frac{N}{2}},$$

por tanto

valor de 2^N	valor de t
2^{20}	$2^{10} \approx 10^3$
2^{40}	$2^{20} \approx 10^6$
2^{80}	$2^{40} \approx 10^{12}$
2^{160}	$2^{80} \approx 10^{24}$

Ejemplo basado en el logaritmo discreto I

- q primo tal que $p = 2q + 1$ también en primo.
- $g_1, g_2 \in \mathbb{Z}_p$ elementos primitivos.
- Definimos

$$h : \mathbb{Z}_q \times \mathbb{Z}_q \longrightarrow \mathbb{Z}_p \setminus \{0\}, \quad h(x_1, x_2) = g_1^{x_1} g_2^{x_2} \text{ mód } p.$$

- Si $h(x_1, x_2) = h(y_1, y_2)$,

$$g_1^{x_1} g_2^{x_2} \equiv g_1^{y_1} g_2^{y_2} \quad \text{mód } p,$$

de donde

$$g_1^{x_1 - y_1} \equiv g_2^{y_2 - x_2} \quad \text{mód } p.$$

Llamemos $\log_{g_1}(g_2) = l$, es decir, $g_1^l \equiv g_2 \ \text{mód } p$. Encontrar l es computacionalmente difícil. Como

$$g_1^{x_1 - y_1} \equiv g_1^{l(y_2 - x_2)} \quad \text{mód } p,$$

Ejemplo basado en el logaritmo discreto II

l es una solución de una de las ecuaciones

$$x_1 - y_1 \equiv l(y_2 - x_2) \mod 2q$$
$$x_1 - y_1 \equiv l(y_2 - x_2 + q) \mod 2q.$$

Encontrar una colisión es equivalente a resolver el logaritmo discreto.

Ejemplo

La construcción anterior para $q = 6173$, $p = 12347$, $g_1 = 2$ y $g_2 = 8461$ presenta la siguiente colisión:

$$h(5692, 144) = h(212, 4214),$$

con lo que es posible calcular $\log_{g_1}(g_2)$.

Construcción de Merkle-Damgård

Esta construcción permite extender una función (llamada *función de compresión*) $f : \mathbb{B}^{N+r} \to \mathbb{B}^N$ resistente a colisiones a una función $h : \mathbb{B}^* \to \mathbb{B}^N$ resistente a colisiones. Sea $m \in \mathbb{B}^*$.

- Añadimos a m un 1. Añadimos después el menor número de 0 necesario para obtener $\tilde{m} = m||10\cdots0 \in \mathbb{B}^{kr}$.
- Descomponemos $\tilde{m} = m_1||m_2||\ldots||m_k$ con $m_i \in \mathbb{B}^r$ para cada $1 \leq i \leq k$.
- Añadimos un nuevo bloque $m_{k+1} \in \mathbb{B}^r$ que contiene el tamaño inicial de m, luego $\tilde{m} = m_1||\ldots||m_k||m_{k+1}$.[3]
- Fijamos un valor inicial $v_0 \in \mathbb{B}^N$, que puede ser fijo para todos los mensajes.
- Para cada $1 \leq i \leq k+1$, $v_i = f(v_{i-1}||m_i)$.
- Definimos $h(m) = v_{k+1}$.

Proposición

Si f es resistente a colisiones, h es resistente a colisiones.

Demostración.

Ver [DK15, Proposition 2.14]. □

[3] El último bloque evita encontrar colisiones en caso de que $v_0 = v_i$ para algún i

Construcción esponja

En este caso construimos $h : \mathbb{B}^* \to \mathbb{B}^N$ a partir de una función biyectiva $f : \mathbb{B}^r \times \mathbb{B}^c \to \mathbb{B}^r \times \mathbb{B}^c$. Llamemos $l = \left\lceil \frac{N}{r} \right\rceil$. Sea $m \in \mathbb{B}^*$.

- Transformamos m en $\tilde{m} = m_1||m_2||\dots||m_k$ como en la construcción de Merkle-Damgård.
- Reservamos un registro estado $s = (s_1, s_2) \in \mathbb{B}^r \times \mathbb{B}^c$, que inicialmente asignamos $s = (0 \cdots 0, 0 \cdots 0)$.
- Fase absorción. Para cada $1 \leq i \leq k$,

$$s \leftarrow f(s_1 \oplus m_i, s_2)$$

- Fase exprimido. Para cada $1 \leq i \leq l$,

$$g_i \leftarrow s_1, s \leftarrow f(s).$$

- La salida es $g_1||\dots||g_l$ truncada a N bits.

Funciones Hash reales I

En desuso

Basadas en la construcción de Merkle-Damgård, vienen siendo utilizadas desde los años 90. Destacamos las siguientes.

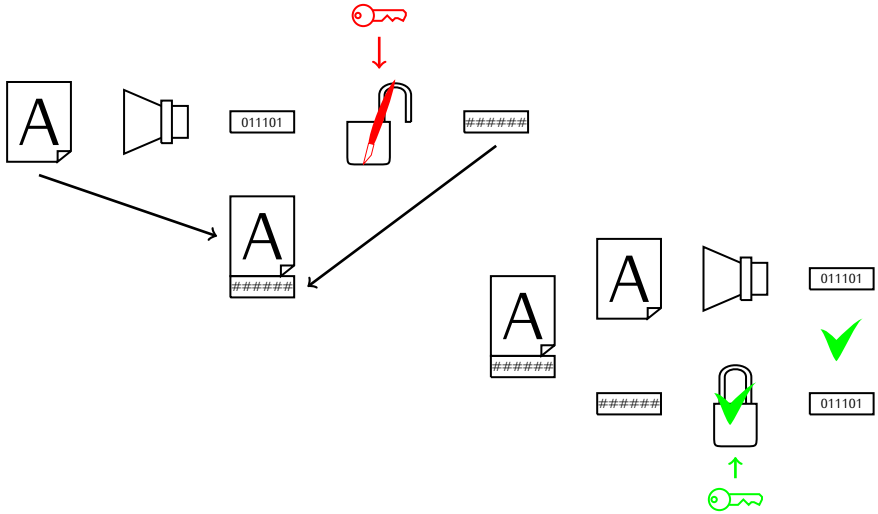
- MD4, diseñado por Ron Rivest en los 90, aunque totalmente desaconsejado por el pequeño tamaño de la salida. Sus principios de diseño se encuentran en algunas de las más populares como MD5, SHA-1 y RIPEMD-160.
- MD5 tiene una salida de 128 bits, SHA-1 y RIPEMD-160 tienen una salida de 160 bits. La tasa de compresión es $r = 512$ bits.
- Hasta Octubre de 2012 el estándar era la familia SHA publicada en [Nat02]. Esta familia incluye SHA-1, SHA-256, SHA-384 y SHA-512, los tres últimos agrupados en el nombre genérico SHA-2. Sus salidas son 160, 256, 384 y 512 bits respectivamente.
- Se han encontrado colisiones para MD5, SHA-1 y RIPEMD-160. SHA-2 sigue siendo resistente a colisiones. Hay muchos menos ataques a la unidireccionalidad.

Funciones Hash reales II

Nuevo estándar

- Concurso propuesto por el NIST en Noviembre de 2007 ante la rapidez con la que se iban produciendo avances en la búsqueda de colisiones en SHA.
- Los candidatos finales fueron BLAKE, Grøstl, JH, Keccak and Skein.
- El ganador ha sido Keccak, diseñado por Bertoni, Daemen, Peeters y Van Assche.
- Keccak, renombrado como SHA-3 (*Secure Hash Algorithm-3*), está basado en una construcción esponja.
- La permutación f actúa sobre un tamaño $b = r + c = 1600$ bits.
- Las posibles salidas son 224, 256, 384 y 512 bits.
- Las tasas de bits correspondientes son $r = 1152,\ 1088,\ 832,\ 576$ respectivamente en función de la salida.

Uso de funciones hash: Firma y verificación digital.



Códigos de detección de modificaciones (MDC)

- Las funciones hash también son conocidas con el nombre de funciones resumen del mensaje, message digest en inglés, y al valor $h(m)$ el resumen, huella dactilar, digest, fingerprint, thumbprint, etc.
- Las funciones hash criptográficas pueden usarse para garantizar la integridad de un mensaje, almacenando el valor $h(m)$ en un lugar seguro. La alteración de m puede detectarse calculando su hash y comparando con el valor almacenado.
- Cuando son usadas así, las funciones hash reciben el nombre de códigos de detección de modificaciones, modification detection codes (MDC) en inglés.
- Por ejemplo, claves públicas, certificados digitales, que veremos más adelante, o paquetes de software suelen llevar asociado una huella dactilar para verificar su integridad, huella publicada en una web razonablemente segura. Incluso puede solicitarse por correo ordinario en ocasiones.

Códigos de Autenticación de Mensajes (MAC)

MAC

La autenticación de un mensaje tiene como objetivo autentificar el origen del mismo y garantizar, al mismo tiempo, la integridad. Se emplean en protocolos como SSL/TLS e IPSec, que veremos más adelante. Un MAC depende de una clave secreta compartida por las dos entidades participantes en una comunicación por lo que las dos entidades pueden generar un MAC válido.

Hay dos técnicas estándar para fabricar MAC

- **HMAC** Basado en funciones hash criptográficas.
- **CBC-MAC** Basado en criptosistemas simétricos.

HMAC

Publicado en [Dan08], es un método estándar para producir un HMAC a partir de una función hash h, derivada de una función de compresión f mediante la construcción de Merkle-Damgård.

- La longitud de los hash y la tasa de compresión son múltiplos de 8, por lo que los medimos en bytes.
- La clave secreta k tiene longitud no mayor de r bytes, extendida a r bytes añadiendo ceros.
- Definimos dos constantes $ipad = 0\text{x}\underbrace{36\cdots36}_{r}$ y $opad = 0\text{x}\underbrace{5\text{C}\cdots5\text{C}}_{r}$
- El HMAC de un mensaje m es

$$\text{HMAC}(k, m) = h((k \oplus opad)||h((k \oplus ipad)||m))$$

- Aplicar la función hash dos veces previene contra un ataque conocido como ataque de extensión de longitud: Si $\text{HMAC}(k, m) = h((k \oplus ipad)||m))$, sea $v_0 = h((k \oplus ipad)||m))$, aplicamos la construcción Merkle-Damgård a un mensaje m', pero añadiendo el tamaño de $\tilde{m}||m'$ en lugar del tamaño de m'. La salida de esta construcción es $\text{HMAC}(k, m')$.

CBC-MAC

- El MAC de un mensaje m es el último criptograma obtenido al aplicar un criptosistema simétrico $e_k : \mathbb{B}^N \to \mathbb{B}^N$ en modo CBC y clave secreta k.
- El vector de inicialización es típicamente $IV = 0\text{x}0 \cdots 0$.
- Partimos m en l bloques de longitud N $m = m_1||m_2|| \cdots ||m_l$.
- Calculamos

$$\begin{aligned} c_0 &= IV \\ c_i &= e_k(m_i \oplus c_{i-1}),\ i = 1, \ldots, l \\ \text{CBC-MAC}(k, m) &= c_l \end{aligned}$$

Índice

Estándar

- El DSA (Digital Signature Algorithm) fue propuesto por el NIST en 1991, y adoptado como DSS (Digital Signature Standard) en 1993. Desde entonces se han propuesto cuatro revisiones del mismo, la última en 2013 [Nat13].
- Es muy similar al esquema de firma digital desarrollado a partir del criptosistema de ElGamal.

Generación de claves

- Se elige un primo q de N bits.
- Se elige un primo p de L bits tal que $p = 2kq + 1$. Las combinaciones N, L permitidas son

L	N
1024	160
2048	224
2048	256
3072	256

- Elegimos $h \in \mathbb{Z}_p^*$ tal que $h^{(p-1)/q} \not\equiv 1 \mod p$. Llamamos $g = h^{(p-1)/q} \mod p$.
- Elegimos $1 \leq x \leq q-1$ aleatoriamente. Calculamos $y = g^x \mod p$.
- La clave privada (usada para firmar) es (p, q, g, x).
- La clave pública (usada para verificar) es (p, q, g, y).

Proceso de firma

- El mensaje es $m \in \mathbb{Z}_q$. Se obtiene como el resultado de aplicar una función hash a un mensaje mayor.
- Seleccionamos aleatoriamente $1 \leq k \leq q$.
- Calculamos $r \leftarrow (g^k \bmod p) \bmod q$ y $s \leftarrow k^{-1}(m + rx) \bmod q$. Si $s = 0$ volvemos a seleccionar k, pero es muy improbable que ocurra.
- El mensaje firmado es (m, r, s).

Verificación

- El receptor conoce el mensaje firmado (m, r, s) y la clave pública (p, q, g, y).
- Calculamos $t \leftarrow s^{-1} \bmod q$ y $v \leftarrow ((g^m y^r)^t \bmod p) \bmod q$.
- La verificación es afirmativa si y sólo si $v = r$.

Proposición

Si la firma es correcta, $v = r$.

Demostración.

$$\begin{aligned} v &= ((g^m y^r)^t \bmod p) \bmod q \\ &= (g^{mt} g^{rxt} \bmod p) \bmod q \\ &= (g^{(m+rx)t \bmod q} \bmod p) \bmod q \\ &= (g^k \bmod p) \bmod q \\ &= r \end{aligned}$$

donde los exponentes se calculan módulo q. □

Índice

Diffie-Hellman

Para este protocolo fijamos un primo p tal que el problema del logaritmo discreto es intratable en $\mathbb{Z}_p^*$, y un elemento primitivo $g \in \mathbb{Z}_p$. Las dos entidades que desean compartir una clave son Alicia y Bernabé.

1. Alicia escoge aleatoriamente $1 \leq a \leq p-2$, calcula $c = g^a \bmod p$ y lo envía a Bernabé.
2. Bernabé escoge aleatoriamente $1 \leq b \leq p-2$, calcula $d = g^b \bmod p$ y lo envía a Alicia.
3. Alicia calcula la clave compartida $k = d^a = g^{ab} \bmod p$.
4. Bernabé calcula la clave compartida $k = c^b = g^{ab} \bmod p$.

La seguridad se basa en la conjetura de Diffie-Hellman. Este protocolo no proporciona autenticación de las partes, y es vulnerable a ataques activos del "hombre en el medio".

Estación a estación

Mejora del protocolo de Diffie-Hellman buscando resolver sus debilidades. Fijamos $g \in \mathbb{Z}_p^*$ como en el protocolo de Diffie-Hellman. Asumimos que tenemos disponible un esquema de firma digital. Cada usuario tiene una pareja de claves (s, v) privada-pública para firma y verificación. Las partes públicas son auténticas y accesibles.

1. Alicia escoge aleatoriamente $1 \leq a \leq p-2$, calcula $c = g^a \bmod p$ y lo envía a Bernabé.
2. Bernabé escoge aleatoriamente $1 \leq b \leq p-2$, calcula $d = g^b \bmod p$ y $k = g^{ab} = (c)^b \bmod p$. Calcula $s = \mathrm{sgn}_{s_B}(c||d)$. Envía $(d, e_k(s))$ a Alicia.
3. Alicia calcula $k = g^{ab} = d^a \bmod p$ y $s = d_k(e_k(s))$, verifica $\mathrm{vfy}_{v_b}(c||d, s)$. Firma $t = \mathrm{sgn}_{s_A}(d||c)$ y envía a Bernabé $e_k(t)$.
4. Bernabé calcula $t = d_k(e_k(t))$ y verifica $\mathrm{vfy}_{v_A}(d||c, t)$.
5. A partir de este momento ambos pueden estar seguros de que la clave secreta k está compartida sólo por ellos dos.

Kerberos (simplificado)

Desarrollado en el MIT, es un ejemplo de protocolo con árbitro, una entidad especial en la que todas las demás confían. En este protocolo el árbitro recibe el nombre de *Kerberos authentication server*, y lo denotamos T. Éste comparte una clave para un criptosistema simétrico e con cada cliente A, k_A, y cada servidor B, k_B.

1. A elige aleatoriamente r_A y envía a T la petición (A, B, r_A).
2. T genera una clave de sesión k y crea un ticket $t = (A, k, l)$, donde l define un periodo de validez para el ticket. Envía $(e_{k_A}(k, r_A, l, B), e_{k_B}(t))$ a A.
3. A recupera $(k, r_A, l, B) = d_{k_A}(e_{k_A}(k, r_A, l, B)))$. Verifica que B y r_A son los que envió. Crea un autentificador $a = (A, t_A)$ donde t_A es un sello de tiempo del reloj local de A. Envía $(e_k(a), e_{k_B}(t))$ a B.
4. B recupera $t = (A, k, l) = d_{k_B}(e_{k_B}(t))$ y $a = (A, t_A) = d_k(e_k(a))$. Comprueba que
   1. el identificador A es el mismo en el ticket t y en el autentificador a,
   2. el sello t_A es actual, en un pequeño entorno del reloj local de B,
   3. el sello t_A está dentro de los límites de l.

   Si todos estos pasos son verificados, B considera A autenticado.
5. B envía $e_k(t_A)$ a A.
6. A compara $d_k(e_k(t_A))$ con el valor de t_A del autentificador. Si coinciden considera B autenticado.

Índice

Comprobación interactiva de conocimiento

- Hay dos participantes en uno de estos protocolos, el sujeto P (*prover* en inglés) y el sujeto V (*verifier* en inglés). P tiene conocimiento de algún dato y quiere convencer a V de que conoce dicho dato. Alternativamente, P y V realizan una serie de *movimientos* al final de los cuales V acepta o rechaza la comprobación de P.
- Hay dos requerimientos para un sistema de comprobación interactiva:

  Completitud Si P conoce el dato, V siempre aceptará la comprobación de P.

  Robustez Si P puede convencer a V con una probabilidad razonable, conoce el dato.
- Los sujetos son llamados *honestos* si siguen el comportamiento diseñado en el protocolo. En caso contrario se llaman *tramposos*. La trampa no está en alterar la sintaxis de la comunicación, sino en modificar los mensajes transmitidos.
- El esquema más básico consiste en que P envíe el dato a V, pero eso obliga a que V también conozca el dato, y cualquiera que vigile la comunicación vería el dato. Este esquema es asimilable a enviar una contraseña.

Conocimiento de una clave privada

P debe probar ante V que conoce la clave privada K asociada a una clave pública conocida k. El protocolo consta de dos movimientos:

1. V selecciona un mensaje aleatorio m, calcula $c = e_k(m)$ y envía c a P.
2. P calcula $m' = d_K(c)$ y lo envía a V.
3. V acepta si y sólo si $m = m'$.

- Completitud y robustez son evidentes.
- Un adversario que observe la comunicación difícilmente podrá suplantar más adelante a P por la aleatoriedad de m.
- Si V es tramposo y envía a P en el primer paso un criptograma obtenido de una tercera parte, obtendrá el texto en claro de dicho criptograma sin necesidad de averiguar la clave.

Protocolo de Fiat-Shamir simplificado I

- P selecciona $n = pq$ con p, q primos aleatorios. La habilidad de calcular raíces cuadradas en $\mathbb{Z}_n$ es equivalente a la habilidad de factorizar n.
- P elige aleatoriamente $y \in \mathbb{Z}_n$ y calcula $x = y^2 \bmod n$, es decir y es la raíz cuadrada de x en $\mathbb{Z}_n$. La pareja (n, x) es pública, mientras que (p, q, y) se mantienen en secreto.
- P debe convencer a V de que conoce y, una raíz cuadrada de x.

Fiat-Shamir simplificado

1. P elige aleatoriamente $r \in \mathbb{Z}_n^*$, calcula $a = r^2 \bmod n$ y lo envía a V.
2. V elige aleatoriamente $e \in \{0, 1\}$ y lo envía a P.
3. P calcula $b = ry^e \bmod n$ y lo envía a V.
4. V acepta si $b^2 \equiv ax^e \mod n$.

Completitud Si P conoce y, ry^e es la raíz cuadrada de ax^e, luego V acepta.

Robustez Un tramposo E puede convencer a V de que conoce x con probabilidad $\frac{1}{2}$:

Protocolo de Fiat-Shamir simplificado II

1. E elige aleatoriamente $r \in \mathbb{Z}_n^*$ y $f \in \{0,1\}$, calcula $a = r^2 x^{-f} \mod n$ y lo envía a V.
2. V elige aleatoriamente $e \in \{0,1\}$ y lo envía a E.
3. E envía r a V.

V acepta si y sólo si $e = f$, lo que ocurre con probabilidad $\frac{1}{2}$.

- Un tramposo E tratando de suplantar a P no puede hacerlo con probabilidad mayor que $\frac{1}{2}$. De ser así, E conocería un valor de a para el cual puede contestar correctamente a ambos retos, es decir, conoce b_1 y b_2 tales que

$$b_1^2 \equiv a \mod n \text{ y } b_2^2 \equiv ax \mod n.$$

  En este caso puede calcular una raíz cuadrada de x, $y = b_2 b_1^{-1} \mod n$, lo que es intratable.
- Ni V ni E tienen algún conocimiento del valor de y.

Índice

Protocolo de conocimiento cero

Un protocolo de conocimiento cero (o mínimo) es aquel en el que cualquier cálculo que un verificador V, honesto o no, puede eficientemente realizar después de interactuar con P, puede ser eficientemente simulado sin interacción.

El protocolo de Fiat-Shamir simplificado es un ejemplo de protocolo de conocimiento cero.

Protocolo de Fiat-Shamir I

- P selecciona $n = pq$ con p, q primos aleatorios. La habilidad de calcular raíces cuadradas en $\mathbb{Z}_n$ es equivalente a la habilidad de factorizar n.
- P elige aleatoriamente $y = (y_1, \ldots, y_t) \in \mathbb{Z}_n^t$ y calcula $x = (y_1^2 \bmod n, \ldots, y_t^2 \bmod n)$. La lista (n, x) es pública, mientras que (p, q, y) se mantiene en secreto.
- P debe convencer a V de que conoce y.

Protocolo de Fiat-Shamir

Repetimos el siguiente proceso k veces:

1. P elige aleatoriamente $r \in \mathbb{Z}_n^*$, calcula $a = r^2 \bmod n$ y lo envía a V.
2. V elige aleatoriamente $e = (e_1, \ldots, e_t) \in \{0, 1\}^t$ y lo envía a P.
3. P calcula $b = r y_1^{e_1} \cdots y_t^{e_t} \bmod n$ y lo envía a V.
4. V rechaza si $b^2 \not\equiv a x_1^{e_1} \cdots x_t^{e_t} \mod n$ y detiene el protocolo.

Completitud Si P conoce y, $r y_1^{e_1} \cdots y_t^{e_t}$ es la raíz cuadrada de $a x_1^{e_1} \cdots x_t^{e_t}$, luego V acepta.

Protocolo de Fiat-Shamir II

Robustez Para que un tramposo realice una suplantación similar a la del protocolo simplificado, debe realizar kt elecciones aleatorias en el conjunto $\{0,1\}$, por lo que la probabilidad debe ser 2^{-kt}.

- Un tramposo E tratando de suplantar a P no puede hacerlo con probabilidad mayor que 2^{-kt}. De ser así, E conocería k valores de $a^1,\dots,a^k$ para el cual puede contestar correctamente a dos retos retos diferentes $(e^1,\dots,e^k)$ y $(f^1,\dots,f^k)$. Hay una iteración j para la cual $e^j \neq f^j$, es decir, puede calcular b_1 y b_2 tales que

$$b_1^2 \equiv a x_1^{e_1^j} \dots x_t^{e_t^j} \mod n \text{ y } b_2^2 \equiv a x_1^{f_1^j} \dots x_t^{f_t^j} \mod n.$$

  En este caso $b_2 b_1^{-1}$ mód n es una raíz cuadrada del elemento aleatorio $x_1^{f_1^j - e_1^j} \cdots x_t^{f_t^j - e_t^j}$, lo que es intratable.
- Se puede demostrar que para $t \in \mathcal{O}(\log_2(|n|))$ y $k \in \mathcal{O}(|n|^l)$, el protocolo sigue siendo de conocimiento cero.

Índice

Esquemas de umbral

Un dato secreto se trocea en n piezas de manera segura y se reparte entre el mismo número de usuarios. Una coalición de algunos de los usuarios debe ser capaz de recuperar el dato secreto.

Esquema (t, n)-umbral.

Un centro de confianza trocea el secreto en n piezas que reparte entre n usuarios. Cualesquiera t usuarios juntos deben ser capaces de recuperar el valor del secreto, mientras que $t-1$ o menos usuarios juntos deben ser incapaces de recuperarlo.

Esquema umbral de Shamir

El centro de confianza T reparte un secreto $s \in \mathbb{Z}$ entre n usuarios $\{P_1, \ldots, P_n\}$.

1. T escoge un primo $p \geq \max\{n, s\}$ y asigna $a_0 \leftarrow s$.
2. T selecciona aleatoriamente $a_1, \ldots, a_{t-1} \in \mathbb{Z}_p$ y obtiene el polinomio $f(x) = \sum_{i=0}^{t-1} a_i x^i \in \mathbb{Z}_p[x]$.
3. T escoge $x_1, \ldots, x_n \in \mathbb{Z}_p$ distintos, para cada i calcula $s_i \leftarrow f(x_i)$ y envía la pareja (x_i, s_i) a P_i de manera segura.

- Como $\deg f(x) = t - 1$, son necesarios t puntos para calcularlo usando, por ejemplo, el polinomio de interpolación de Lagrange. Con menos de t valores no es posible saber el polinomio, y todos los valores $a \in \mathbb{Z}_p$ pueden aparecer con menos de t puntos con la misma probabilidad.
- Se puede extender con facilidad a nuevos usuarios.
- Un usuario puede tener uno o más trozos.

Bibliografía I


Quynh H. Dang.
The Keyed-Hash Message Authentication Code (HMAC).
National Institute of Standards and Technology (NIST), July 2008.


Hans Delfs and Helmut Knebl.
Introduction to Cryptography. Principles and Applications.
Information Security and Cryptography. Springer, third edition, 2015.


National Institute of Standards and Technology (NIST).
DATA ENCRYPTION STANDARD (DES), October 1999.


National Institute of Standards and Technology (NIST).
ADVANCED ENCRYPTION STANDARD (AES), November 2001.


National Institute of Standards and Technology (NIST).
SECURE HASH STANDARD, August 2002.

Bibliografía II


National Institute of Standards and Technology (NIST).
Digital Signature Standard (DSS), July 2013.