

Firmas digitales

10.1.

los sistemas de cifrado hasta ahora tienen dos metas principales

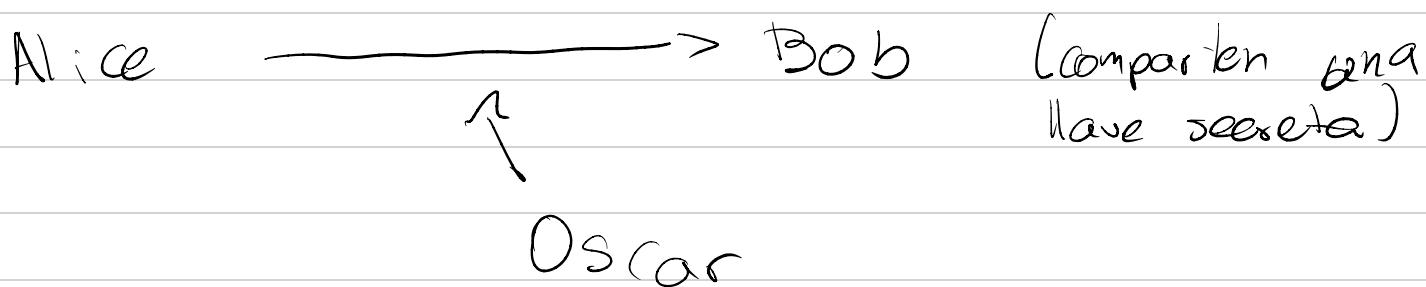
1) Cifrar datos

2) Intercambiar llaves

Podríamos pensar que solo debemos prestar atención a los mecanismos de seguridad

Vamos a ver un caso en el cual los códigos simétricos fallan al proveer la función de seguridad deseada.

Caso típico:



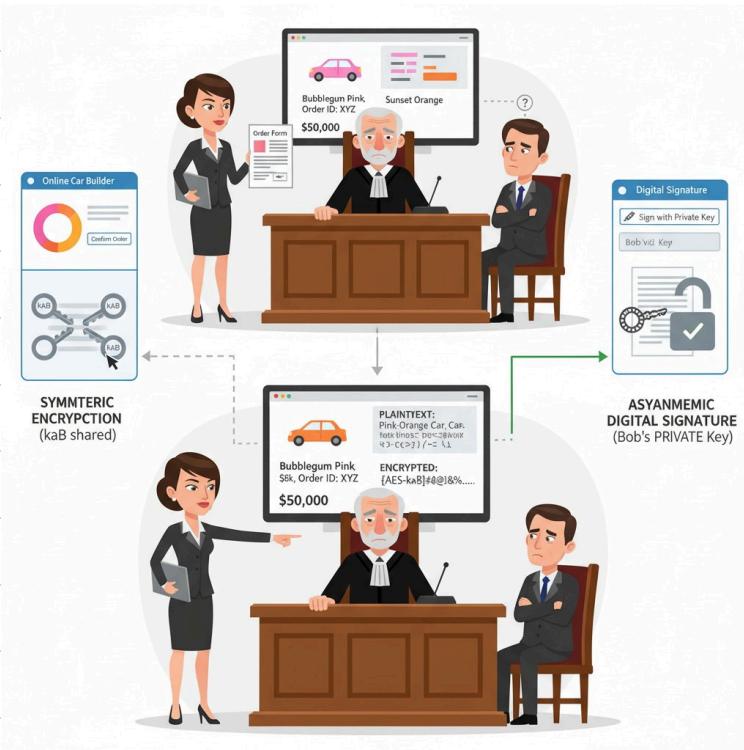
Si Alice y Bob usan un cifrado simétrico (AES) pueden estar seguros de que Oscar no va a aprender el texto plano.

Podrían incluso emplear un código de autenticación de mensajes (cap 13), y Bob estaría seguro de que el mensaje viene de Alice y no de Oscar.

Los códigos de autenticación de mensajes usan cifrados simétricos.

"Todo parece super bien" pero hemos assumido siempre que el **chico malo** es alguien **externo!!!**

Es usual que Alice y Bob se quieran comunicar seguramente y al mismo tiempo engañarse uno al otro!!!



¿Porqué no podemos usar algún esquema de llaves simétrico (diffie-hellman) para verificar o probar que envio Bob?

Alice y Bob tienen la misma información, cualquier persona puede hacer lo mismo.

La solución es usar cripto de llave pública.

El setup asimétrico de los esquemas de llave pública le da al juez la forma de distinguir quién envió.

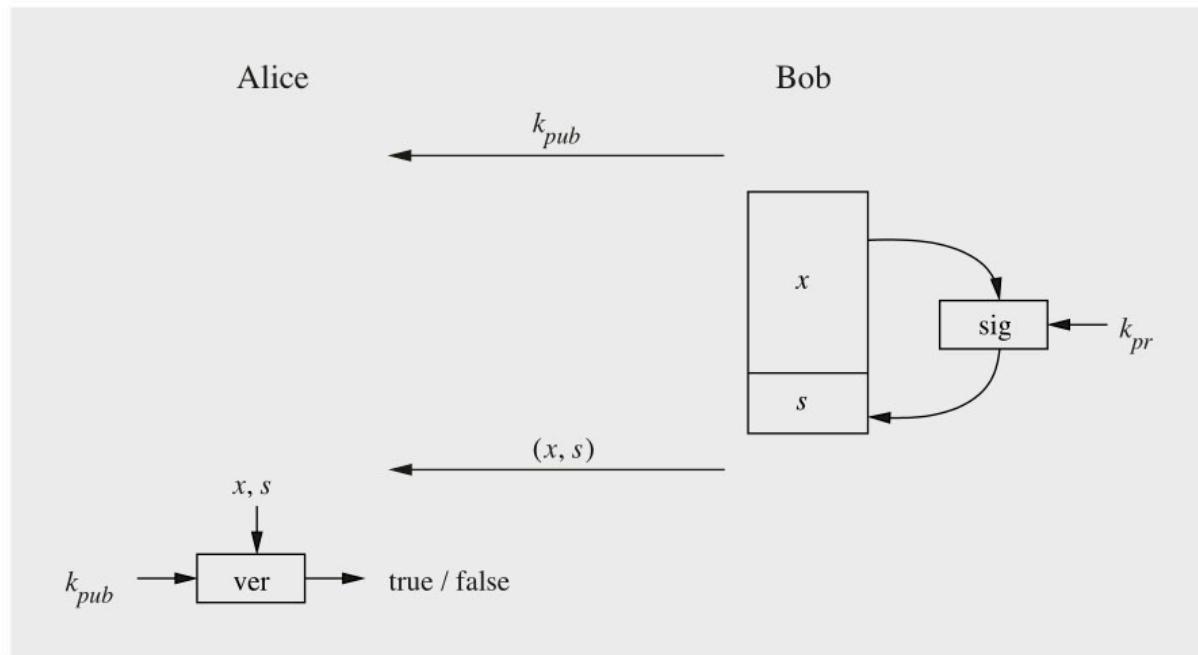
Las firmas digitales son los equivalentes de llaves públicas que resuelve las situaciones de "enganarse" uno a otro.

Principios de firmas digitales

Firmas digitales es el análogo a firmar en papel un documento.

Solo la persona que crea el mensaje digital puede crear una firma válida.

La idea básica: la persona que firma usa la llave privada y la que recibe usa la llave pública que hace match con la llave privada.



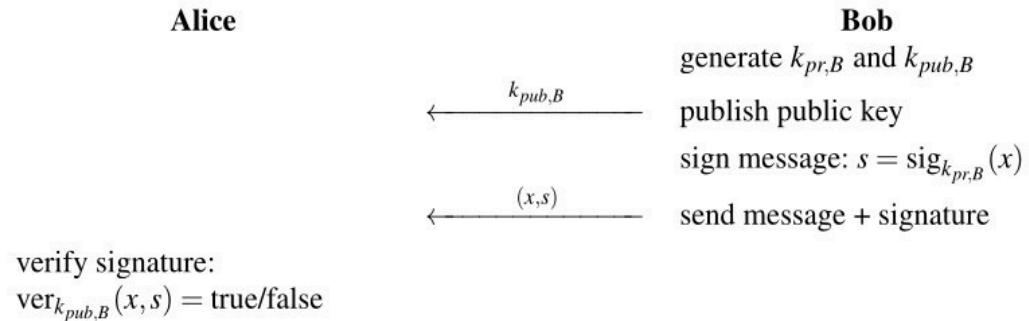
la firma dig.
es una función
de la llave priv.
de Bob

la firma dig
es un entero
grande, una
cadena de 2048
bits

Fig. 10.1 Principle of digital signatures, which involves signing and verifying a message

La única salida es verdadero o falso.

Basic Digital Signature Protocol



Este esquema no "cifra" el mensaje, para eso se debe usar algún otro cifrado simétrico o otro.

Calcular de las 3 familias de algoritmos de clave pública sirven para construir firmas digitales:

- factorización de enteros
- logaritmos discretos
- curvas elípticas

Servicios de seguridad

¿Cuáles son los posibles objetivos de seguridad que un sistema de seguridad debe poseer?

- * Confidencialidad:
- * Integridad:
- * Autenticación del mensaje o del origen de los datos
- * No-repudio:

Confidencialidad, integridad y disponibilidad
se conocen como el CIA triad ya
que se consideran importantes para asegurar
el sistema.

Ejemplos:

* Emails: si es privado los 3 primeros si es
corporativo también no repudio

* Actualización de software en un cel: integridad
y autenticación de message; el manufacturer quiere
asegurarse de que solo soft. original se le
pone al equipo. Confidencialidad no se
necesita porque el soft. está disponible
de cualquier forma.

Otros servicios de seguridad:

* Identificación o autenticación de entidad:
establecer y verificar la identidad de una
entidad, ej persona, tarjeta, etc.

* Control de acceso: acceso restringido a recursos
de entidades con privilegios.

* Disponibilidad: asegurar que el syst. elec.
nico esté disponible fielmente.

* Auditoria: proporcionar evidencia sobre act. relevantes para la seguridad, como guardar registro de eventos.

* Seguridad física:

* Anonimato: proporciona protección contra el descubrimiento y el uso indebido de la identidad.

APLICACIONES DE FIRMAS DIGITALES

* Certificados: sirven para garantizar que una clave pública realmente pertenece a la identidad que dice representar, evitando ataques con claves falsas.

* Secure Boot and Secure firmware update: buscan garantizar integridad y autenticidad del software.

Solo puede ejecutarse soft. confiable del proveedor original

Se usan firmas digitales para verificar el software en cada inicio del sistema (arranque autenticado)

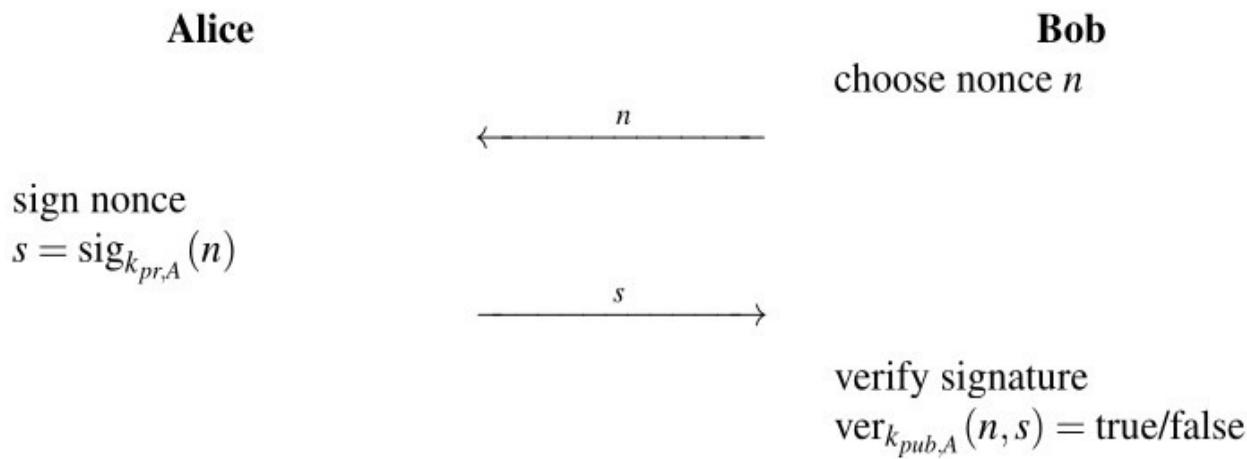
Ventaja principal: los usuarios solo requieren la clave pública del proveedor o el único que puede generar actualizaciones válidas con su clave privada.

Existen estándares de arranque seguro (UEFI) ampliamente adoptados en industria.

* Proof-of-Knowledge protocols: permiten que una persona demuestre su identidad

sin revelar su decreto.

Ej. Alice necesita convencer a Bob de que posee un secreto sin mostrárselo. Se diferencian de los sistemas basados en contraseñas donde el secreto si se comparte



Bob comprueba la validez de la firma que Alice conoce el secreto "la llave privada"

Algunas aplicaciones son tarjetas inteligentes para autenticación en ATM o acceso a edificios.

10.2 (1) esquema de firma RSA

Su seguridad recae en el problema de factorización de enteros.

10.2.1 Schoolbook of RSA Dig. Sig.

El padding/belleno debe de usarse para prevenir ataques.

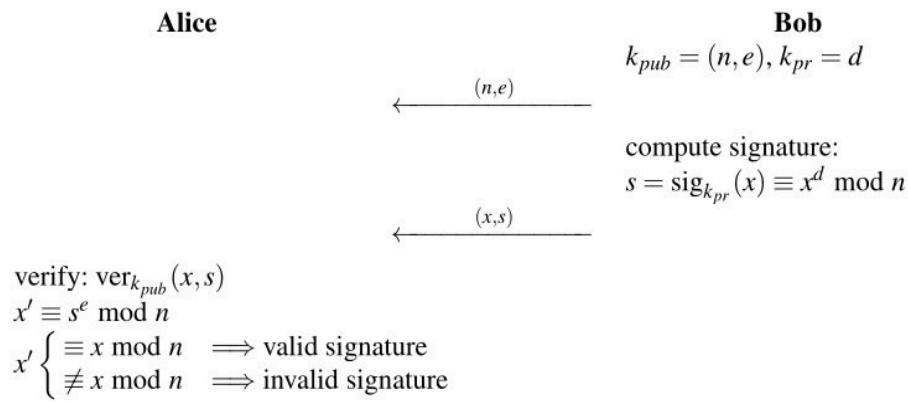
Sup. que Bob le quiere enviar un mensaje a Alice. De sigue el procedimiento de RSA y se tiene

RSA Keys

- Bob's private key: $k_{pr} = (d)$
- Bob's public key: $k_{pub} = (n, e)$

El protocolo de firma es el siguiente. El mensaje x que se va a firmar esta en el rango $(1, 2, \dots, n-1)$.

Basic RSA Digital Signature Protocol



Bob es el único que puede usar k_{priv} y por lo tanto la actan verifica como el dueño del mensaje firmado

Alice recibe el mensaje firmado y usando la k_{pub} de Bob le regresa x' .

S: $x' = x$ ent. Alice sabe 2 cosas:

- 1) El autor del mensaje tiene la k_{pr} de Bob y si solo Bob tiene la k_{pr} ent. es el autor del mensaje. (**Autenticación del mensaje**)
- 2) El mensaje no fue cambiado en el camino (**Integridad**)

Prueba: la verificación regresa "true" si el mensaje y la firma no fueron modificados en el camino.

Verificamos la operación $s^e \bmod n$

$$s^e \equiv (x^d)^e = x^{de} \equiv x \bmod n$$

Ya que por la relación entre la K_{pr} y K_p

$$de \equiv 1 \bmod \phi(n)$$

entonces calcular $x \in \mathbb{Z}_n$ a la (de) -potencia nos da el mismo entero.



En los esquemas de firmas digitales con RSA, el rol de las llaves se invierte

* En RSA:

- La llave pub. se aplica al mensaje
- El receptor usa la llave privada para descifrar

* En Firmas digitales

- La K_{pr} se aplica para generar la firma.
- La K_{pub} se usa para verificar la firma

Ejemplo: Bob quiere enviar un mensaje firmado ($x=4$) a Alice.

Alice

$$\xleftarrow{(n,e)=(33,3)}$$

Bob

1. choose $p = 3$ and $q = 11$
2. $n = p \cdot q = 33$
3. $\Phi(n) = (3-1)(11-1) = 20$
4. choose $e = 3$
5. $d \equiv e^{-1} \equiv 7 \pmod{20}$

compute signature for message
 $x = 4$:
 $s = x^d \equiv 4^7 \equiv 16 \pmod{33}$

$$\xleftarrow{(x,s)=(4,16)}$$

verify:

$$x' = s^e \equiv 16^3 \equiv 4 \pmod{33}$$

$x' \equiv x \pmod{33} \Rightarrow$ valid signature

Alice concluye que Bob generó el mensaje y que no fue alterado (integridad y autenticidad)

10.2.2 Aspectos Computacionales

La firma digital tiene la misma longitud que el módulo n ($\log_2 n$ bits)

* Se recomienda que $n \geq 2048$ bits

* La generación de llaves es similar al cifrado RSA

* Para calcular y verificar firmas se usa el algoritmo de square-and-multiply

* Las técnicas de aceleración de RSA también se aplican a firmas digitales

- * Uso de llaves públicas con los (e^{2^k}, f) resulta en una verificación muy rápida.
- * En la práctica, los mensajes se firman una sola vez pero se verifican muchas veces

10.2, 3 Seguridad

Lo mismo de RSA:

→ verificar que RSA no debe romperse matemáticamente = dificultad de la factorización

Se requieren primos p y q de al menos 1024 bits y ent el módulo $n \geq 2048$ bits.

Es esencial garantizar que los llaves pub sean auténticas

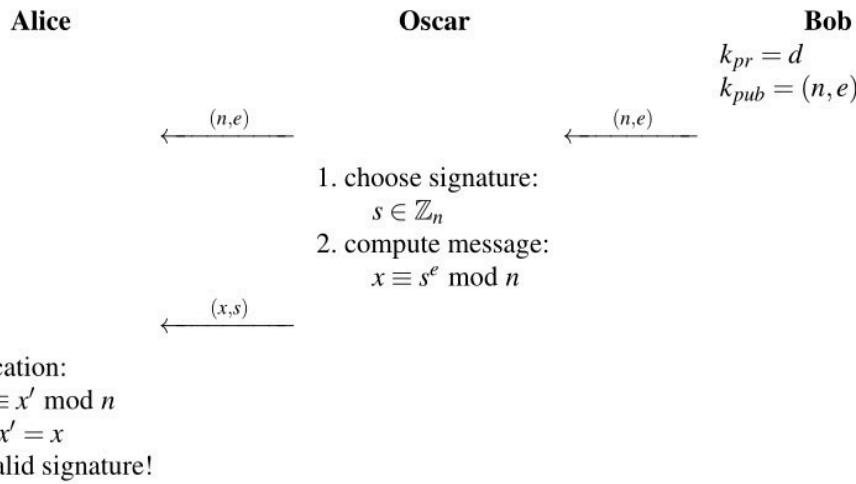
Riesgo: si un atacante entrega una llave pública falsa, puede firmar mensajes en nombre del verdadero emisor

Solución: uso de certificados digitales para asegurar la autenticidad de los llaves.

Existental Forgery

Es un ataque posible en los esq. de firmas digitales

Existential Forgery Attack Against RSA Digital Signature



En el RSA

básico una
versión puede
generar una fir-
ma válida para
un mensaje de este
tipo.

Modo de ataque:

- El atacante se hace pasar por Bob
- Alice verifica la firma como correcta porque realiza los mismos cálculos.

Limitación: el atacante elige primero la firma y luego el mensaje por lo que no controla el contenido (no puede ordenar una transferencia)

Problema: los sistemas automáticos no detectan la falsificación.

Consecuencia: El RSA básico casi no se usa en la práctica

Solución: uso de espacios de padding.

RSA Padding: The probabilistic Signature Standard (PSS)

El ataque anterior se puede prevenir imponiendo formatos específicos (padding) en los mensajes.

* Ej: exigir que todos los mensajes tengan 100 bits de ceros al final. La probabilidad de coincidencia por azar es 2^{-100} , prácticamente imposible.

RSA-PSS (Probabilistic Signature Scheme): es una extensión del RSA básico que combina firma + padding (encoding).

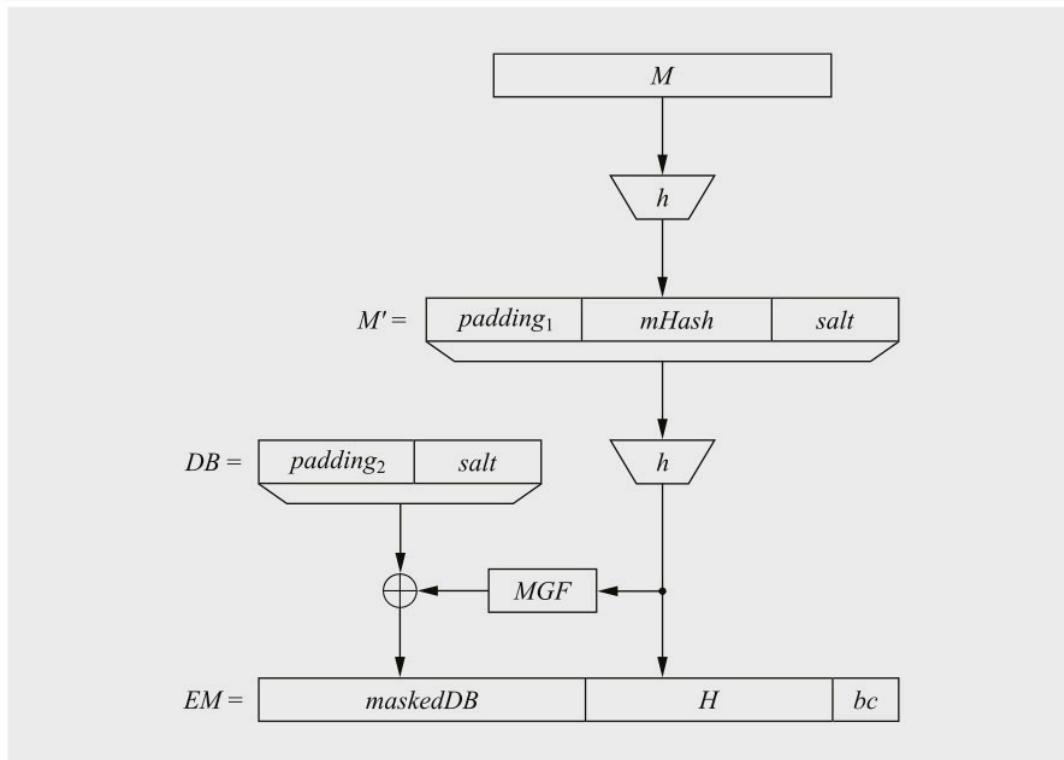


Fig. 10.2 Principle of EMSA-PSS encoding

Procedimiento
de encoding
conocido como
Encoding Method
for Signature
with Appendix
(EMSA) Probabi-
listic Signature
Scheme (PSS).

Encoding for the EMSA Probabilistic Signature Scheme

Let $|n|$ be the size of the RSA modulus in bits. The encoded message EM has a length $\lceil(|n|-1)/8\rceil$ bytes such that the bit length of EM is at most $|n|-1$ bits.

1. Generate a random value $salt$.
2. Form a string M' by concatenating a fixed padding $padding_1$, the hash value $mHash = h(M)$ and $salt$.
3. Compute a hash value H of the string M' .
4. Concatenate a fixed padding $padding_2$ and the value $salt$ to form a data block DB .
5. Apply a mask generation function MGF to the hash of the string M' to compute the mask value $dbMask$. In practice, a hash function such as SHA-2 is often used as MGF .
6. XOR the mask value $dbMask$ and the data block DB to compute $maskedDB$.
7. The encoded message EM is obtained by concatenating $maskedDB$, the hash value H and the fixed padding bc .

En la práctica, no se firma el mensaje directamente, sino su huella digital (hash) de longitud fija (620 bits)

El esquema EMSA-PSS introduce:

- Salt (valor aleatorio) antes del segundo hash → hace que las firmas sean probabilísticas
- Misma entrada firmada dos veces → firmas distintas, lo cual es deseable

Proceso de firma:

mensaje → hash → encoding → Firma RSA
(con padding + salt)

Proceso de verificación: el receptor reconstruye el encoding (con salt y padding) y valida la firma con la clave pública.