Criptografía Aplicada

Plan

- Introducción
- Métodos criptográficos
  - Algoritmos simétricos
  - Algoritmos asimétricos
- Hashes y Macs
- Firmas digitales
- Gestión de claves y autenticación
- Infraestructuras de clave pública (PKI)

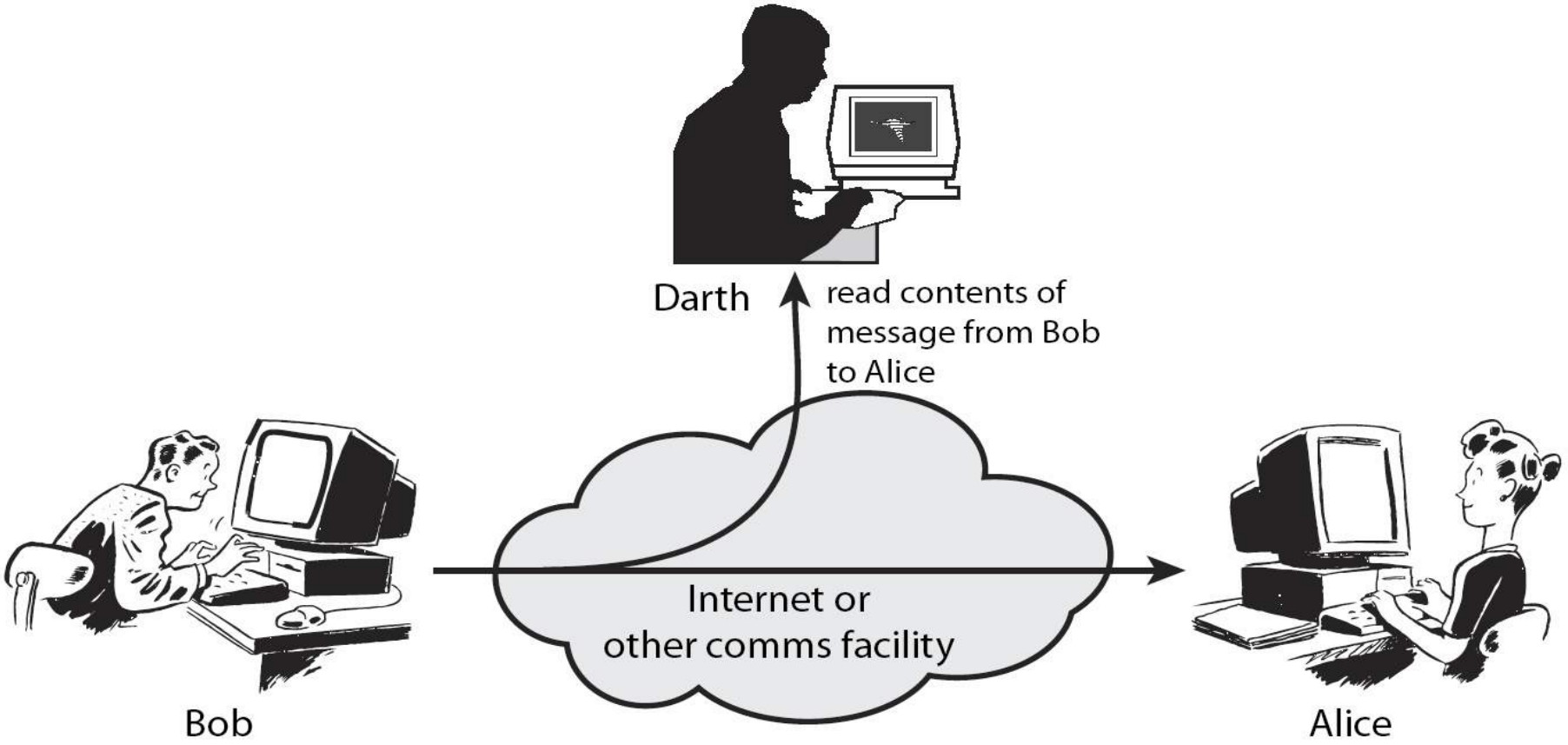
Algunas definiciones

- Criptografía: Estudio de principios o métodos de cifrado
- Encriptar o cifrar: informalmente, proceso de convertir la información a una forma “disfrazada” para ser transmitida a través de un canal inseguro
- Desencriptar o descifrar: proceso aplicado a la información “disfrazada” para obtener el mensaje original

Criptografía

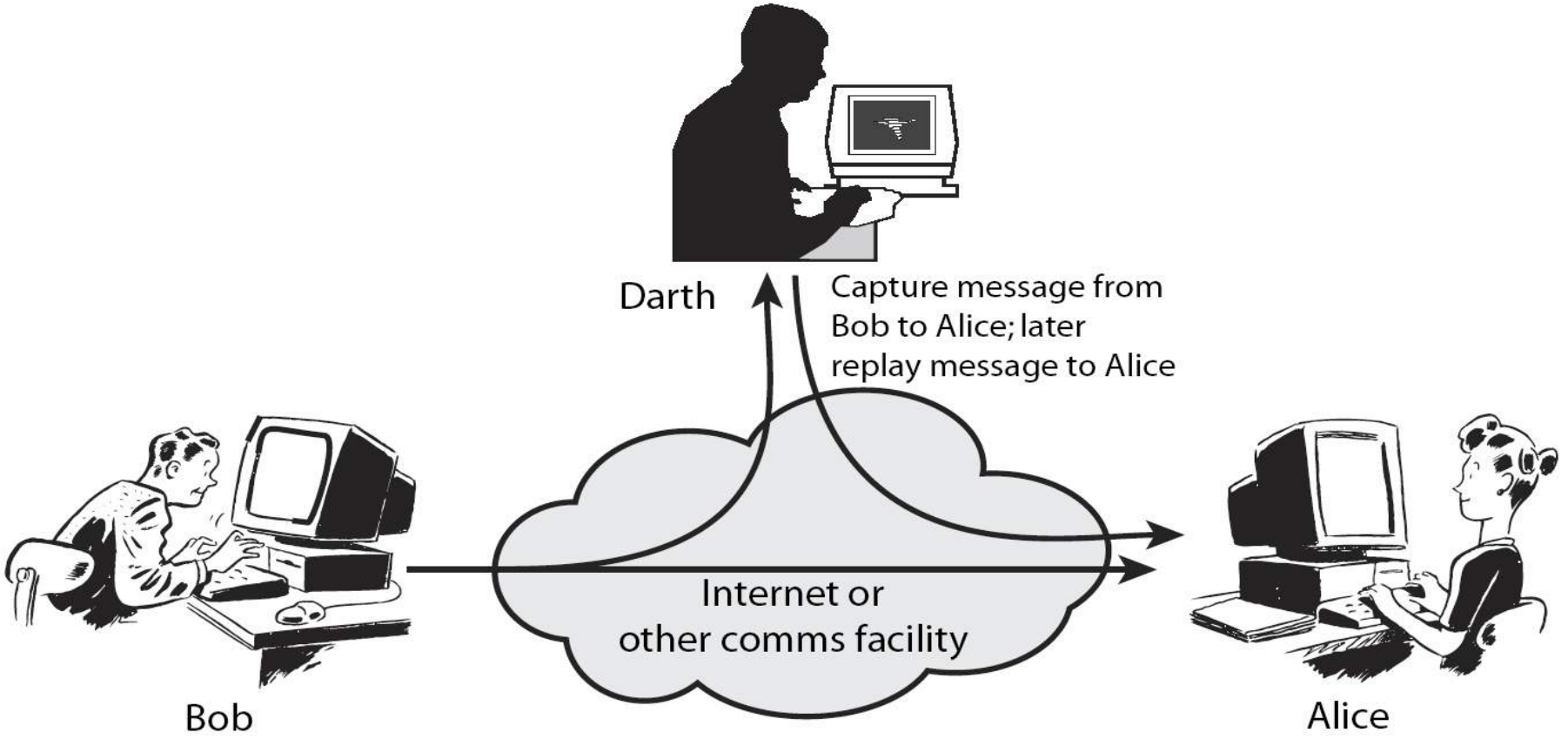
- La criptografía utiliza la matemática para obtener seguridad sobre la información transmitida o almacenada. Algunos objetivos:
  - Privacidad
  - Integridad de los datos
  - Autenticación
  - No repudio
- Permite mejorar la seguridad al almacenar o transmitir información por canales inseguros

Ataques pasivos



William Stallings. Cryptography and Network Security, 4ta. Ed. Figura 1.3a

Ataques activos



William Stallings. Cryptography and Network Security, 4ta. Ed. Figura 1.3b

Más definiciones

- Texto plano: información en su forma natural
- Texto cifrado o encriptado: texto al cual se le aplicó algún método de cifrado o encriptación
- Oponente o adversario: entidad maliciosa que “ataca” la conexión entre Bob y Alicia

Criptosistema clásico

- Se llama criptosistema al conjunto de:
  - Un conjunto finito de posibles textos planos
  - Un conjunto finito de posibles textos cifrados
  - Un conjunto finito de posibles claves
  - Un conjunto de reglas de cifrado y otro de reglas de descifrado, que cumplen que:

    $d_k(e_k(x)) = x$ para todos los posibles textos planos x

Criptoanálisis

- Es el estudio de métodos para obtener el significado de la información que está cifrada
- Es la ciencia de descifrar códigos, decodificar secretos, violar esquemas de autenticación y romper protocolos criptográficos
- El ideal del criptoanálisis es descubrir la clave secreta y no solo decodificar algunos mensajes
- Dos tipos de ataques:
  - Ataques criptoanalíticos
  - Ataques de fuerza bruta

Criptoanálisis: Clasificación de ataques

- Texto cifrado solamente (*Ciphertext only*): El oponente conoce solo una porción del texto cifrado
- Texto plano conocido (*Known plaintext*): El oponente conoce una porción del texto plano y su correspondiente texto cifrado
- Texto plano elegido (*Chosen plaintext*): El oponente tiene acceso a la maquinaria de cifrado. Elige texto y ve la salida
- Texto cifrado elegido (*Chosen ciphertext*): El oponente tiene acceso a la maquinaria de descifrado. Elige texto cifrado y ve el texto plano

Criptoanálisis: Tipos de ataque

- Ataques por fuerza bruta: pruebo todas las claves, hasta que encuentro la correcta
  - No prácticos si el espacio de claves es muy grande
- Ataques criptoanalíticos: atacan la estructura del algoritmo o de los protocolos que lo utilizan, buscando un “atajo” para realizar menos trabajo que un ataque por fuerza bruta
  - Se dice que son “exitosos” si el esfuerzo esperado es menor al ataque por fuerza bruta

Seguridad incondicional y seguridad computacional

- Seguridad incondicional:

  No importa que tiempo o poder de cálculo se disponga, el cifrado no podrá ser quebrado ya que no hay suficiente información para determinar de forma única el correspondiente texto plano

- Seguridad computacional:

  Dado un poder de recursos computacionales, el tiempo necesario de cálculo esperado para quebrar un código es más grande que el tiempo de vida del mensaje

Cifrados incondicionalmente seguros

- Existen cifrados incondicionalmente seguros
- Ejemplo: One time pad:
  - Transmisor y receptor deben compartir una “clave” de al menos el mismo largo del texto plano a cifrar
  - Los bits de la clave deben ser elegidos aleatoriamente de forma independiente
  - Si $p_1,p_2,p_3...$ texto plano, $c_1,c_2,c_3...$ texto cifrado y $k_1,k_2,k_3...$ la clave de cifrado

$$c_i = p_i \oplus k_i \quad \text{(cifrado de Vernam)}$$

$\oplus$: operación OR exclusivo (XOR)

Métodos de cifrado

- La seguridad de algunos métodos de cifrado se basa en el secreto de su algoritmo
- En el mundo actual, estos métodos usualmente no son útiles debido al costo de desarrollar nuevos sistemas si los viejos son comprometidos
- Todos los métodos (o algoritmos) actuales basan su seguridad en el uso de claves
- En una primera clasificación, existen dos clases de métodos de cifrado basados en claves:
  - Métodos simétricos
  - Métodos asimétricos

Algoritmos simétricos

- Es la forma tradicional de la criptografía
  - Su origen conocido se remonta al menos a la época de los romanos
- Los algoritmos simétricos utilizan la misma clave para cifrar y descifrar la información
  - $D_k(E_k(x)) = x$
- A la clave se le denomina “clave secreta”, “Llave secreta”, “secreto compartido” (Secret-Key o shared key en inglés)

Algoritmos simétricos

- Los cifrados simétricos se pueden dividir en:
  - Cifrados en bloque
  - Cifrados stream
- El cifrado en bloque procesa el mensaje para cifrar o descifrar en bloques de tamaño fijo (cantidad fija de bits, p. ej. 64 o 128 bits por bloque)
- El cifrado en stream lo procesa de a bit o byte

Cifrado en bloque

- Dado un bloque de texto plano de tamaño *n* bits, y una clave de tamaño *L* bits, la función *E* de cifrado genera un texto cifrado de *n* bits
- Típicamente se utilizan bloques de 64 o 128 bits
- Para cada clave *k*, E_K es una permutación sobre el conjunto de los bloques de entrada
  - De las posibles, la clave k selecciona entre 2^L permutaciones distintas (L largo de clave)
- D_K, la función de desencriptado, realiza la transformación inversa para obtener el texto plano

Ejemplos de algoritmos de Cifrado en bloque

- AES (Advanced Encryption Standard). 2001. Bloque de 128 bits, clave de 128, 192 o 256 bits
- DES (Data Encription Standard). Adoptado como standard en 1977. Clave de 56 bits, bloque de 64 bits. Considerado débil hoy en día
- DES triple (aplicar DES 3 veces con al menos 2 claves distintas)
- IDEA, Blowfish, Twofish, y muchos más

Propiedades de los algoritmos

- Para los usos comunes de AES el único ataque efectivo conocido es por fuerza bruta
  - Y 2^{128} está fuera del alcance de la tecnología actual
- DES: Ataque lineal y diferencial
  - Igualmente la forma más eficiente de atacarlo en la mayoría de los casos es la fuerza bruta
- DES triple
  - No se conocen ataques criptoanalíticos. Desventaja: más lento que AES

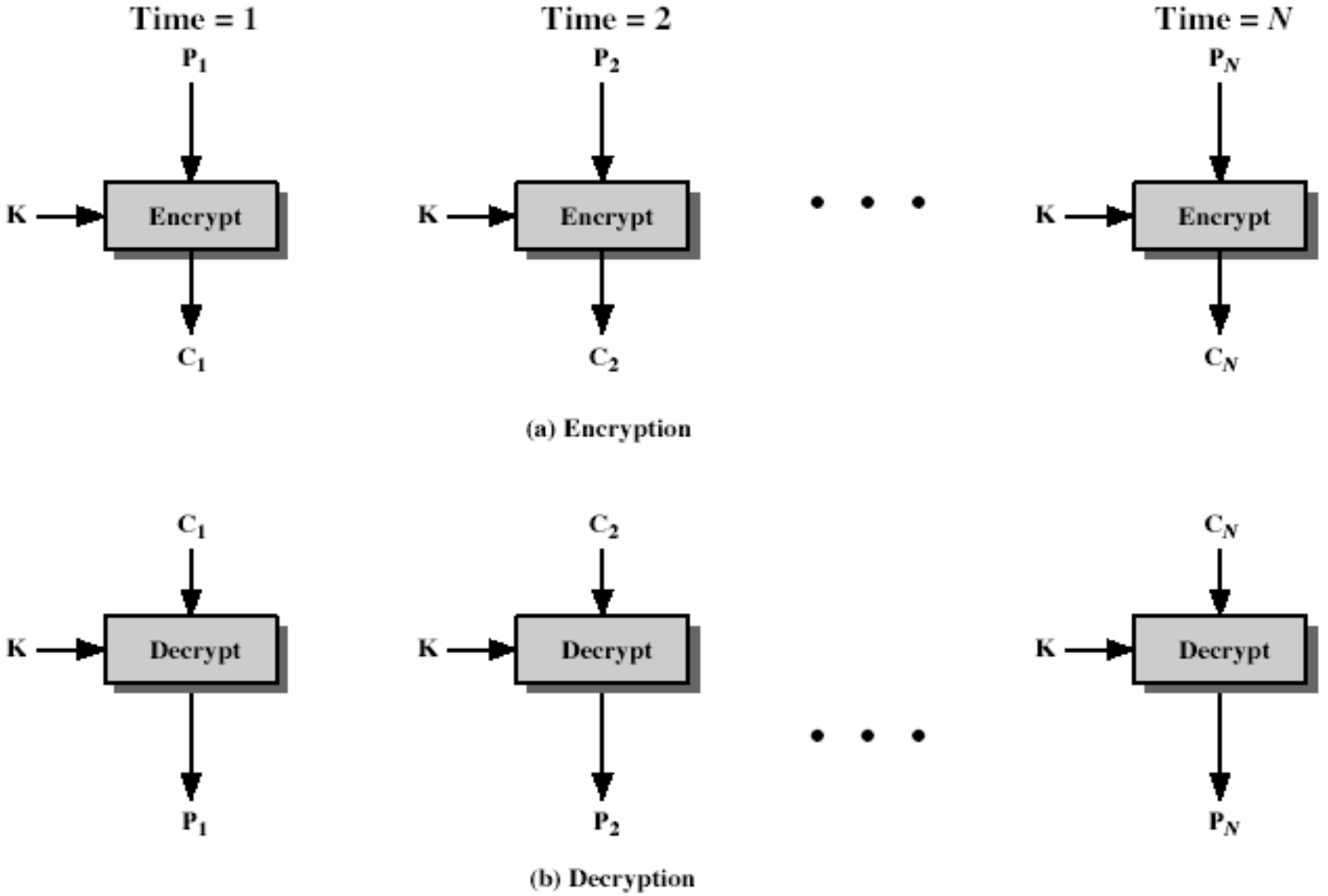
Modos de operación

- En general querremos cifrar datos que no son del tamaño de un bloque. Esto nos lleva a definir los llamados “modos de operación”, que nos dicen cómo cifrar un conjunto de bloques que forman un mensaje
- Hay muchos modos de operación propuestos. Algunos de ellos:
  - ECB: *Electronic Code Book*
  - CBC: *Cipher Block Chaining*
  - CFB: *Cipher FeedBack*
  - OFB: *Output FeedBack*

ECB (Electronic Code Book)

- Se completa el texto plano para llevarlo a un múltiplo del tamaño de bloque
- El texto plano se divide en bloques de tamaño idéntico
- Cada bloque es cifrado independientemente de los demás
- Los bloques cifrados son del mismo tamaño y generados con la misma clave
- Es como si tuviéramos una tabla o libro donde para cada texto plano se da su correspondiente texto cifrado

ECB


Time = 1
Time = 2
Time = N
P1
P2
PN
K
Encrypt
C1
C2
CN
(a) Encryption
C1
C2
CN
K
Decrypt
P1
P2
PN
(b) Decryption

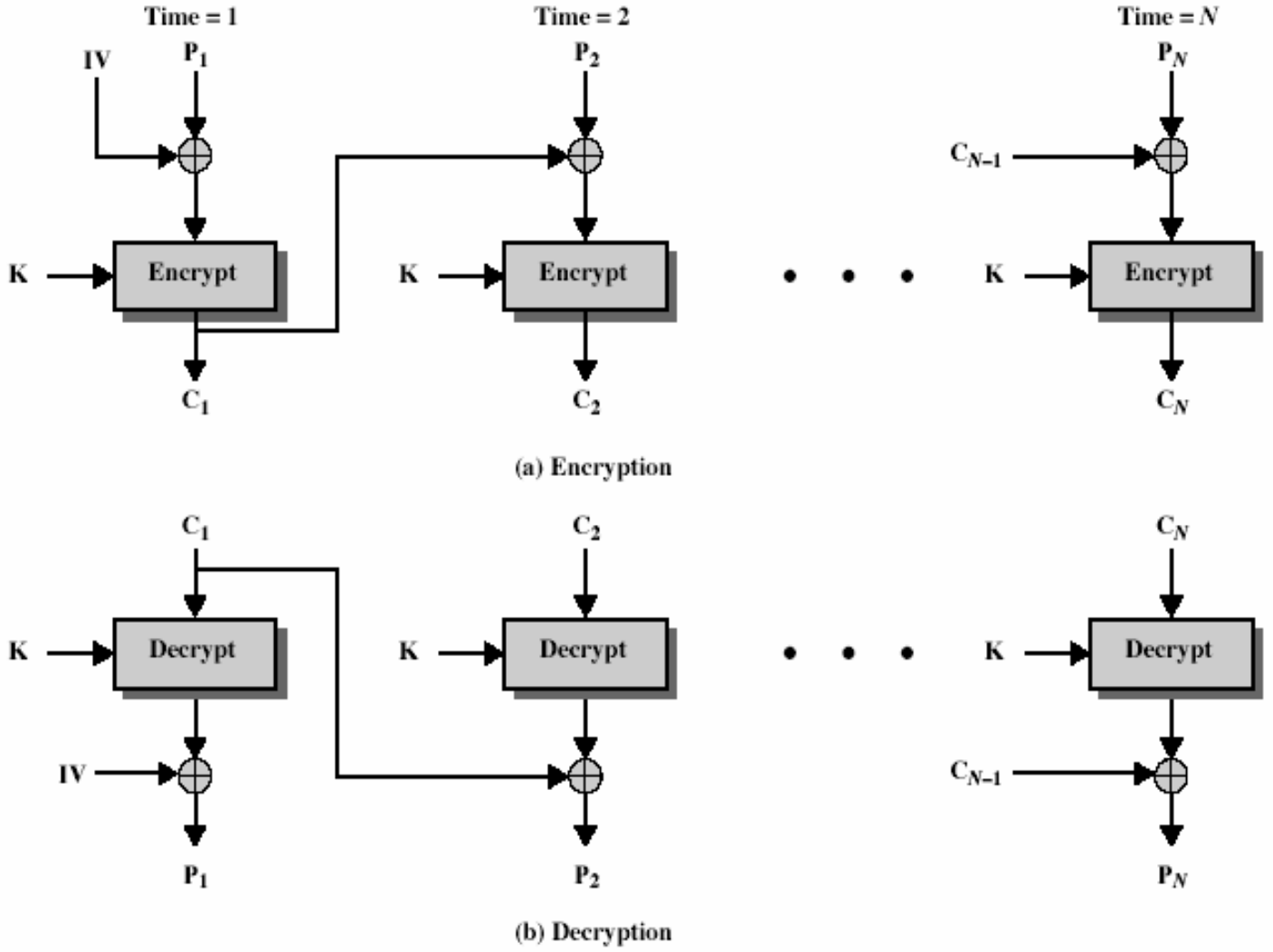
Problemas de ECB

- Al mismo texto plano le corresponde siempre el mismo bloque cifrado
  - Podemos obtener información acerca del mensaje original sin conocer la clave
  - Podemos sustituir bloques
- Ejemplo:
  - Se que hay 2 mensajes posibles: “disparar misil” y “no disparar misil”
  - Veo pasar el primer mensaje, y el enemigo no dispara el misil: para futuros mensajes, ya sabré lo que el mensaje quiere decir

CBC (Cipher Block Chaining)

- El texto plano se divide en bloques de tamaño idéntico (*n bits*)
- Se utiliza un vector de inicialización de *n bits* (IV: initialization vector)
- Cada texto cifrado se hace depender de los anteriores
- Se diferencia del ECB en que el mismo bloque de texto plano genera distinto texto cifrado de acuerdo a los bloques previos y al vector de inicialización
- Cualquier cambio en un bloque afecta a los siguientes
  - $C_i = E_K(P_i \text{ XOR } (C_{i-1}))$ $\quad C_0 = IV$

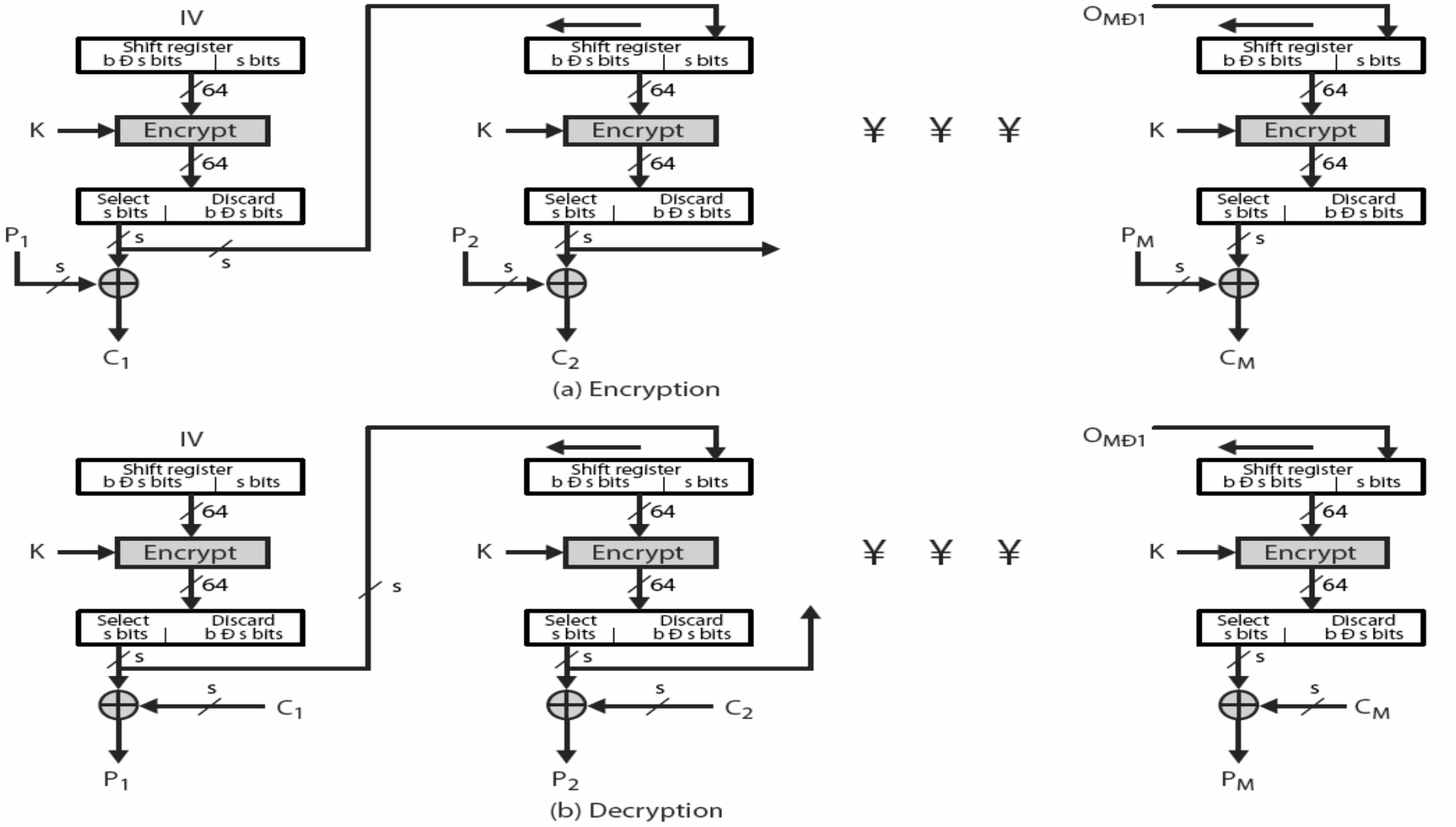
CBC


Time = 1
Time = 2
Time = N
IV
P_1
P_2
P_N
C_{N-1}
K
Encrypt
C_1
C_2
C_N
(a) Encryption
Decrypt
(b) Decryption

OFB (Output Feedback)

- El mensaje es tratado como un flujo de bits
- La salida del cifrado es reutilizada para la siguiente etapa
- La reutilización es independiente del mensaje
  - $O_i = E_K(O_{i-1})$
  - $O_{-1} = IV$
  - $C_i = P_i \text{ XOR } O_i$
- Los bits de error no son propagados (si tengo un error de un bit en la transmisión, a la salida solo ese bit será erroneo)

OFB


IV
Shift register
b Đ s bits
s bits
64
K
Encrypt
Select
s bits
Discard
b Đ s bits
P1
C1
P2
C2
OMĐ1
PM
CM
(a) Encryption
(b) Decryption

Cifrado por flujo (stream)

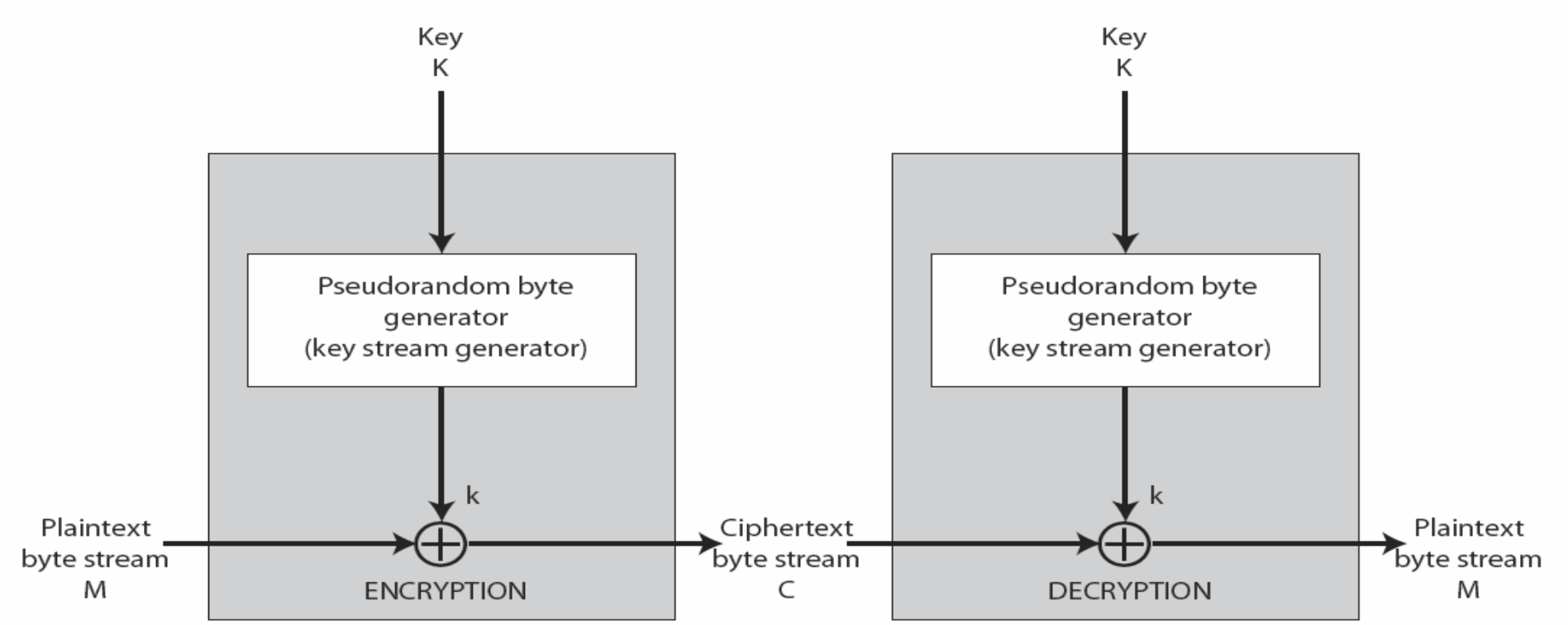
- Intentan aproximar un cifrado incondicionalmente seguro
- El problema del cifrado incondicionalmente seguro es que la clave debe ser tan grande como el texto plano
- Esto motiva el diseño de cifrados por flujo en donde la clave es generada de forma pseudoaleatoria a partir de una clave más chica
  - NO es incondicionalmente seguro
- El cifrado de un texto plano se hace usualmente de a caracter (byte) por vez, aunque podría ser bit a bit

Cifrado por stream

- Los generadores de números pseudoaleatorios usan una función que produce un stream de bits determinista y que, en algún momento, se repetirán
- Cuanto más grande sea el período de repetición, más difícil será realizar un criptoanálisis
- Se busca que, estadísticamente, sean indistinguibles de una secuencia de números aleatorios
- Se suelen usar en aplicaciones de telecomunicaciones, o en aplicaciones con muy bajos recursos
- Tiene poco o ningún error de propagación

Cifrado por stream

- Estos tipos de cifrados no son incondicionalmente seguros, tratan de ser computacionalmente seguros



Ejemplo: RC4

- RC4 (Rivest Cipher 4) es un cifrado por stream diseñado por RSA security (1987)
- Está basado en el uso de permutaciones aleatorias
- Es usado en los estándares SSL/TLS (Secure Sockets Layer/Transport Layer Security) definidos para la comunicación entre clientes web y servidores
- También es utilizado en protocolos WEP (*Wired Equivalent Privacy*) y WPA (*WiFi Protected Access*) que son parte del estándar IEEE 802.11 de redes inalámbricas
- Algunos problemas dependiendo de cómo se use. Su uso va disminuyendo

Algoritmos asimétricos

- Los algoritmos simétricos, conocidos como de clave secreta, requieren una comunicación previa de la clave entre las entidades participantes de forma segura (secreta)
- Los métodos asimétricos, conocidos como de clave pública, se basan en encontrar un criptosistema donde se tienen dos claves distintas para cifrar y descifrar, y es computacionalmente imposible obtener la clave de descifrado (d_k) a partir de la clave de cifrado(e_k) (o viceversa)
- De esa forma, una de las claves puede ser hecha pública

Algoritmos asimétricos

- La idea del sistema de clave pública (*public-key*) fue propuesta por Diffie y Hellman en 1976
- La primer realización práctica de un sistema criptográfico de clave pública fue realizada por Rivest, Shamir y Adleman (1978, algoritmo RSA)
- Los criptosistemas de clave pública nunca pueden brindar seguridad incondicional
  - El adversario puede cifrar cualquier texto plano (ya que dispone de la clave de cifrado) y verificar si alguno coincide con el texto cifrado

Algoritmos asimétricos

- La seguridad de los criptosistemas asimétricos se basa en que la clave privada solo puede ser computada a partir de la pública resolviendo un problema “difícil”
  - Mientras que generarlas juntas es “fácil”
- Se utilizan problemas matemáticos para los cuales no se conocen soluciones eficientes
  - Estudio de estos problemas: Ver complejidad computacional

Algoritmos asimétricos

- Algunos problemas utilizados en criptosistemas asimétricos son:
  - Factorización
  - Logaritmos discretos
  - Logaritmos discretos sobre curvas elípticas

Factorización

- Todo número entero puede ser representado de forma única como el producto de números primos (ej.: 2277 = 3^2 * 11 * 23)
- Un posible algoritmo de factorización es ir dividiendo el número (con números primos) hasta que el resto sea un número primo
- En los algoritmos asimétricos basados en factorización se buscan números del orden de 10^{300}, lo que requiere tratar con todos los primos hasta 10^{150} (unos 10^{147} primos)

Factorización

- Métodos modernos para factorizar números “grandes”:
  - Algoritmo de Pollard
  - Curvas elípticas
  - NFS (*Number Field Sieve*) (1990)
    - Es el mejor propuesto hasta el momento
- Todos ellos son extremadamente lentos para números muy grandes

Encripción RSA

- Se basa en la seguridad que brinda la factorización
- Todo número entero puede ser expresado como producto de factores primos
- Utiliza la propiedad que la factorización de un número es dificil de obtener (para números grandes)
- Si se encontrara un método sencillo de factorización, el criptosistema sería fácil de quebrar

RSA

- Sea $n = pq$ (p y q números primos)
- Sea $\Phi(n) = (p - 1)(q - 1)$
- Sea $e \in Z_{\Phi(n)}$, primo con $\Phi(n)$
- Sea d el inverso multiplicativo de e ($d \in Z_{\Phi(n)}$ tal que $ed = 1 \pmod{\Phi(n)}$)
- Cifrado: $c = x^e \bmod n$ Clave pública (n,e)
- Descifrado: $x = c^d \bmod n$ Clave privada (n,d)
- Se prueba que $(x^d)^e \bmod n = (x^e)^d \bmod n = x$ si $x<n$

Utilizando RSA (generación de claves)

- Para generar sus claves, Alice realiza:
- Obtener p y q, primos, de largo en bits similar
- Calcular n y $\Phi(n)$
- Elegir e (es común tomar e=65537 o e=17)
- Calcular d (utilizando el algoritmo de euclides extendido)
- Distribuir n y e

Utilizando RSA (envío de mensajes)

- Bob obtiene la clave pública de Alice
- Representa el mensaje M como un entero
  - $0<M<n$
  - Usualmente se le da un formato previamente (padding scheme), agregándole bytes aleatorios y otros (ver por ejemplo PKCS#1)
- Realiza el cálculo de $C=M^e \bmod n$
- Envía C

Utilizando RSA (recepción)

- Calculamos $M' = C^d \bmod n$
- Si el formato incluye redundancia, verificamos que la misma sea correcta

- La seguridad está dada porque sin conocer d no podemos calcular M'

Utilizando RSA (firma digital)

- Para realizar una firma digital, podemos (simplificadamente) cifrar el mensaje con nuestra clave privada
- Dado M, Alice realiza $C=M^d \bmod n$
  - Se envían M y C
- Quien quiera verificar la firma, debe verificar que $M= C^e \bmod n$
- Solo quien tiene la clave privada puede generar C
- Luego veremos más sobre firmas digitales

Uso práctico de algoritmos asimétricos

- Los algoritmos de cifrado asimétricos son mucho más lentos que los algoritmos simétricos
- Usualmente se utilizan solamente para autenticación, para firmas, y para el intercambio de claves de sesión para un algoritmo simétrico

Secure Hashes y MACs

- Secure Hash: un hash que cumple ciertas propiedades básicas de seguridad
- MAC: Message Authentication Code (incluye clave)
- Ambos tipos de funciones comparten ciertas características
  - A partir de un mensaje de largo arbitrario se genera un valor de largo fijo, que de alguna manera representa al mensaje original (Message Digest, “resumen” del mensaje)
  - No debe poder derivarse un mensaje del resumen

Funciones de hash

- Mapean un mensaje de largo arbitrario, en un valor de largo fijo relativamente pequeño (propiedad de Compresión)
- Dado M, debe ser relativamente sencillo calcular h(M)
- Difusión: el resumen h(M) debe ser una función compleja de todos los bits del mensaje M: si se modifica un solo bit del mensaje M, se espera que el hash h(M) cambie aproximadamente la mitad de sus bits

Propiedades de los hashes para seguridad

- Unidireccionalidad: conocido un resumen R, debe ser computacionalmente imposible encontrar un mensaje M tal que R=h(M)
- Resistencia débil a las colisiones: Conocido M, es computacionalmente imposible encontrar M' tal que h(M) = h(M')
- Resistencia fuerte a las colisiones: Es computacionalmente imposible encontrar X e Y tales que h(X) = h(Y)

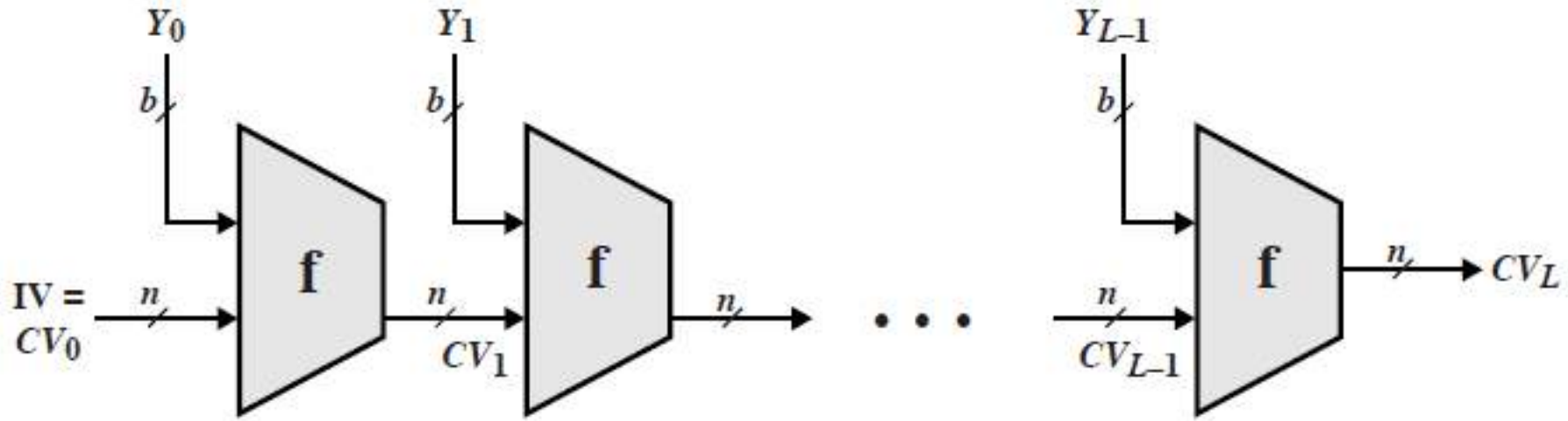
“Paradoja” del cumpleaños

- Dado un grupo de X personas, ¿qué tan grande tiene que ser X para que con probabilidad >0,5 2 de ellos cumplan el mismo día?
- Respuesta: 23
- A cada edad de cada persona la comparo con la de todas las demás
- En criptografía, nos dice que si tengo un hash de x bits, puedo encontrar (con p~50%) una colisión con un esfuerzo del orden de $2^{x/2}$
  - Entonces la longitud del hash no puede ser chica

Operación típica de un algoritmo de hash

- Se completa el mensaje para que sea múltiplo de un determinado tamaño de bloque
- Se procesa el mensaje por bloques
- Se toma un valor inicial conocido para el hash
- Para cada bloque, se realizan diversas operaciones incluyendo el valor del hash en el paso anterior
- El hash del mensaje es el valor al finalizar el procesamiento del último bloque

Estructura general de muchos algoritmos de hash



IV = Initial value
CV_i = chaining variable
Y_i = ith input block
f = compression algorithm

L = number of input blocks
n = length of hash code
b = length of input block

- Stallings. Figura 11.9

Uso típico de un algoritmo de hash

- Los principales usos son la verificación de integridad y función auxiliar para firmas digitales
- Se calcula el resumen de un “mensaje” M
- Se protege la integridad del valor del resumen, no así del mensaje
- Al requerir verificar la integridad del mensaje, se calcula nuevamente el resumen y se compara con el que teníamos protegido

Algoritmos populares de hash

- **MD5:** (refinamiento de MD2, MD4), Ron Rivest 1992. Hash de 128 bits.
- **SHA-1**: NIST 1994. 160 bits
- **SHA-2**: Algoritmos similares a SHA-1 pero con versiones de 224, 256, 384 y 512 bits de hash
- SHA-3: “reciente” (Octubre 2012)
- Otros
  - RIPEMD, Whirlpool, SHA-0, N-Hash, Snefru, Tiger, Panama, Haval, otros

Seguridad actual de MD5

- Wang et al, 2004, y múltiples trabajos posteriores: Se puede encontrar fácilmente colisiones en MD5 (X e Y tales que MD(X)=MD(Y))
- Especialmente importante para quienes firman mensajes generados por otros
  - Ataque “demo” al proceso de una autoridad certificadora en 2008
- No debe utilizarse
- “Los ataques siempre mejoran, nunca empeoran”

Ataques a SHA-1

- Febrero 2005.“Broken”. El esfuerzo (teórico) para encontrar una colisión bajó de 2^{80} a 2^{69}
- Agosto 2005 -> 2^{63}
- Febrero 2017: 1er. Colisión encontrada
- https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html

“This attack required over 9,223,372,036,854,775,808 SHA1 computations. This took the equivalent processing power as 6,500 years of single-CPU computations and 110 years of single-GPU computations”

Debe dejar de usarse

Ataques a algoritmos de hash (cont.)

- Las buenas noticias: no todos los usos de SHA-1 (y MD5) se ven afectados
  - Solo aquellos en que el atacante puede manipular ambos mensajes
- En cualquier caso, es recomendable abandonar SHA-1 y MD5
- El sustituto propuesto es SHA-3
- Para aplicaciones donde aún no se soporte SHA-3, se recomienda utilizar SHA-256 o SHA-512

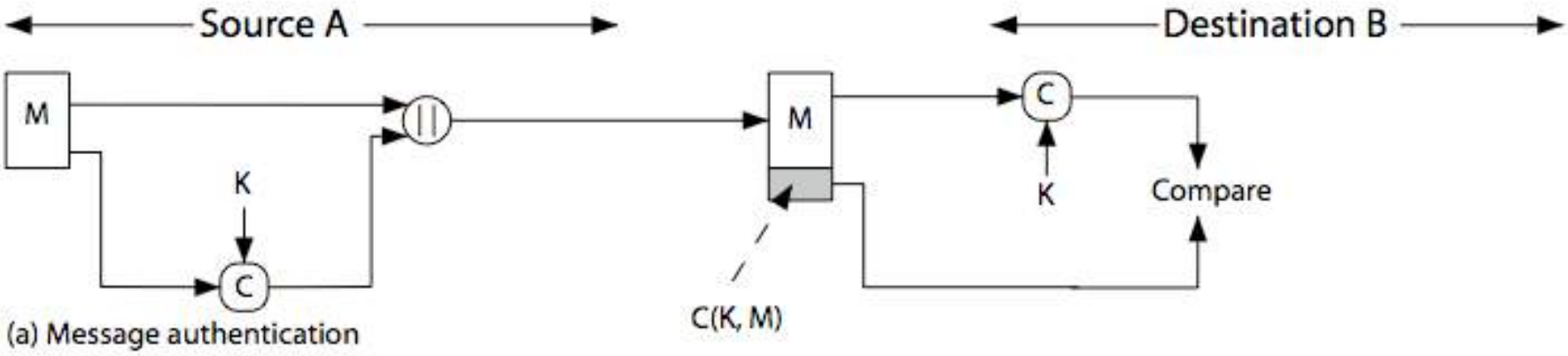
MAC (Message Authentication Code)

- Un algoritmo que, dado un mensaje (de largo arbitrario) y una clave (de largo fijo), produce un bloque de bits de largo fijo
  - Parecido a hashes, salvo por la clave
- Se comparten requerimientos de seguridad
  - Resistencia a colisiones, unidireccionalidad
- No debe poderse recuperar la clave, ni generar MAC sin conocer la clave

Propiedades necesarias de MAC

- Conociendo un mensaje y su MAC, es computacionalmente imposible encontrar otro mensaje con el mismo MAC
- Conociendo un mensaje y su MAC, es computacionalmente imposible encontrar la clave

Uso de MAC para autenticación de mensaje



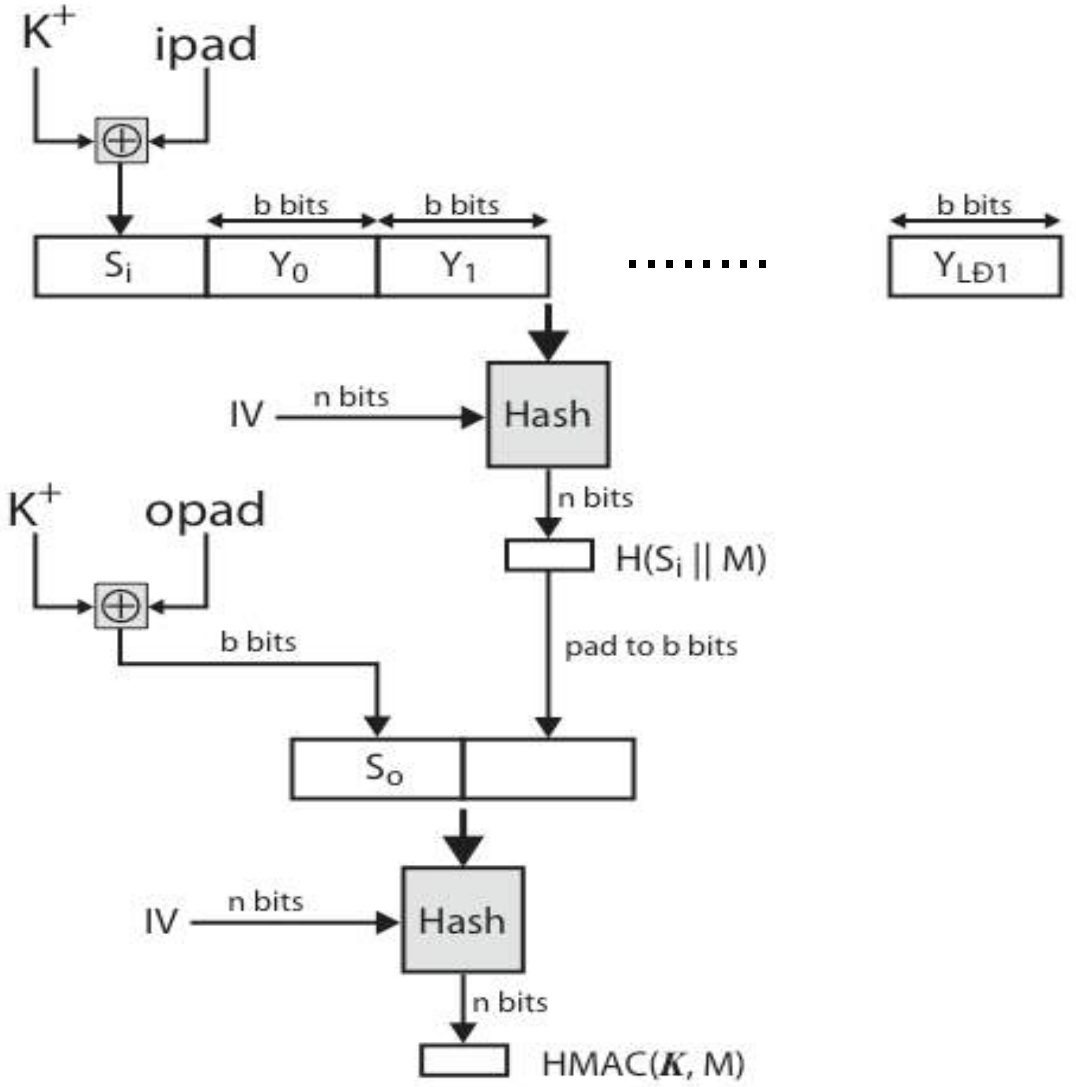
- El documento M se transmite sin cifrar (por ejemplo, puede ser público)
- B puede verificar su integridad

Funciones de MAC utilizando hashes

- Comparten propiedades
- Las funciones de hash son normalmente rápidas
- Propuesta original:
  - MAC(mensaje)=hash (key||mensaje)
  - Se encontraron debilidades
- Actualmente, HMAC y otros
- $HMAC_K$ = Hash[(K^+ XOR opad) || Hash[(K^+ XOR ipad) || M)]]
- Por ejemplo, IPSec utiliza HMAC

Nota: || indica concatenación

HMAC. RFC 2104



Seguridad de HMAC

- Se probó relación entre seguridad de HMAC y del hash utilizado
- Ataques requieren:
  - Fuerza bruta en la clave
  - Ataque del cumpleaños sobre el algoritmo de hash (pero se requiere ver un número muy grande de mensajes con la misma clave)
- Se suele utilizar MD5 o SHA-1 como algoritmo de hash
  - Se está migrando a otros (SHA-3, SHA-256...)

Funciones de MAC basadas en cifrado simétrico

- Originalmente FIPS PUB 113
  - Restricción en el largo del mensaje a procesar
- CMAC: Cypher based message authentication code
- NIST Special Publication 800-38B (para usarse con 3DES y AES)

Aplicaciones de Hashes

- En firmas digitales
- Verificación de integridad
- Identificadores “únicos” para archivos
  - Redes p2p
  - Comparación con un archivo que no puedo ver
- Sistemas de password
- MACs
- Blockchain

Aplicaciones de MAC

- Integridad del mensaje
- Origen del mensaje (quien lo generó conoce la clave compartida)
    - NO es una firma digital

Firmas digitales

Motivación

- Se pretende tener un equivalente a la firma manual de un documento
- Debe verificar el autor y usualmente la fecha y hora de la firma
- Debe autenticar el contenido cuando fue firmado
- Debe ser verificable por terceros
- Incluye la función de autenticación

Diferencias con la firma manuscrita

- Se piden más cosas (timestamp, etc)
  - Pero las copias de un documento digital no son distinguibles del original
- Recordemos, que en la firma digital, entre el individuo que firma y la firma, hay un hardware, un sistema operativo, uno o varios programas...
- Una firma digital nos dice que un programa, que tuvo acceso a la información privada del individuo que “firma”, realizó ciertas operaciones

Problemas a resolver

- Autenticidad del emisor
- Integridad del mensaje
- Actualidad (no replay)
- No repudio (del emisor y receptor)
- Detección de usurpación de identidad
- Deben ser verificables por terceros

Firma digital

- En muchos sistemas prácticos, se utiliza criptografía asimétrica
- Si ciframos mensaje con la clave privada, alcanza con verificar que es descifrado correctamente con la clave pública
- Los algoritmos asimétricos son lentos y se generarían firmas tan largas como el mensaje

  -> Se cifra un hash del mensaje con la clave privada

Ejemplo de Procedimiento Usando RSA

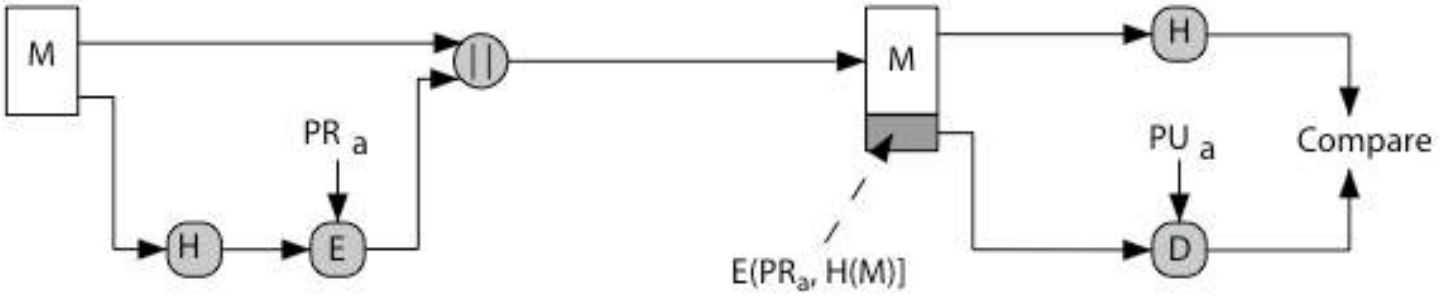
- Sea d_a y e_a las claves pública y privada de A
- A envía a B el mensaje M (cifrado o no), y $E_{e_a}(H(M))$
- El receptor calcula $H' = D_{d_a}(E_{e_a}(H(M)))$, calcula H(M'), y los compara
- Si coinciden, tiene la “seguridad” de que solo A pudo generar el mensaje
  - No se puede generar otro mensaje con el mismo hash
  - Solo quien conoce e_a pudo generar $E_{e_a}(H(M))$

DSA

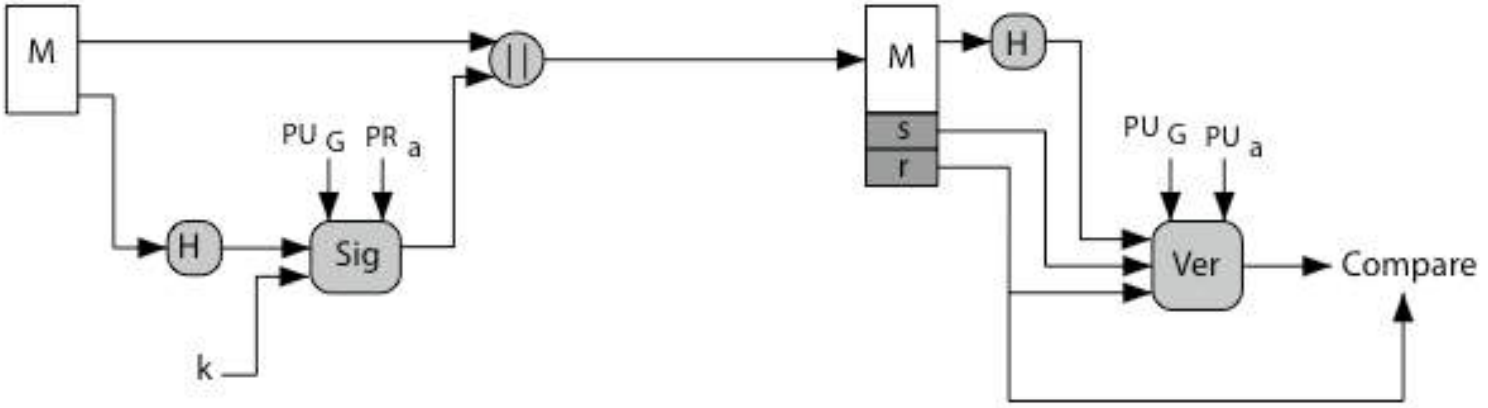
- Digital Signature Algorithm
- Seguridad: dificultad de calcular logaritmos discretos
- Definido en “Digital Signature Standard (DSS). Standard del NIST (FIPS 186, última revisión FIPS 186-3)
- La última revisión incluye firmas con DSA, RSA y con curvas elípticas (ECDSA)

DSA y RSA

(Stallings. Cryptography and network Security. Fig. 13.1)



(a) RSA Approach



(b) DSS Approach

Problemas

- ¿Cómo distribuir la clave pública?
- ¿Cómo estoy seguro que la clave pública que tengo es realmente la de A, y que es actual?
- Repudio....¿Qué pasa si A alega que el no firmó? ¿y si alega que le robaron la clave?

Contra el repudio y el robo de claves

- Exigir que cada mensaje lleve un timestamp, y exigir reporte “inmediato” de claves comprometidas a una autoridad central

o

- Utilizar un “árbitro”, una entidad confiable que certifique el origen y contenido del mensaje
  - Se puede hacer con algoritmos simétricos o asimétricos
  - Dependen de encontrar la entidad confiable

PKI (Public Key Infrastructure)
Infraestructura de clave pública

Certificados digitales: distribución escalable de claves públicas

Definición de PKI (RFC 2828)

- Un sistema de CAs (certification authorities), opcionalmente Ras (autoridades de registro) y otros servidores y agentes, que realizan algún subconjunto de las funciones de: administración (management) de certificados, archivo de certificados expirados, administración de claves, administración de tokens, para una comunidad de usuarios, en una aplicación de criptografía asimétrica
- (O) PKIX usage: The set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates based on asymmetric cryptography

Funciones de una PKI

- Las funciones principales son:
  - Registrar entidades (usuarios) y emitir sus certificados digitales
  - Revocar certificados cuando se requiera
  - Archivar los datos necesarios para validar los certificados en el futuro
- Puede además, por ejemplo, generar pares de claves pública/privada (se recomienda que lo haga el cliente), establecer CPSs (políticas), etc.

¿Qué es un certificado digital?

- Documento firmado por una CA (autoridad de certificación) que contiene diversos datos, al menos:
  - Identificación de una entidad (usuario)
  - su clave pública
- Emisor y receptor deben confiar en esa CA
  - Solo podré validar certificados emitidos por CAs en las cuales yo confío
- Estandar más usado: X.509 (ITU-T)

Utilidad del certificado

- Me permite asociar la clave pública con la identidad presente en el certificado
  - Si puedo verificar la firma y confío en la CA, me dice que la CA atesta que esa clave pública pertenece al dueño del certificado
- O sea, permite la distribución de claves públicas de forma confiable

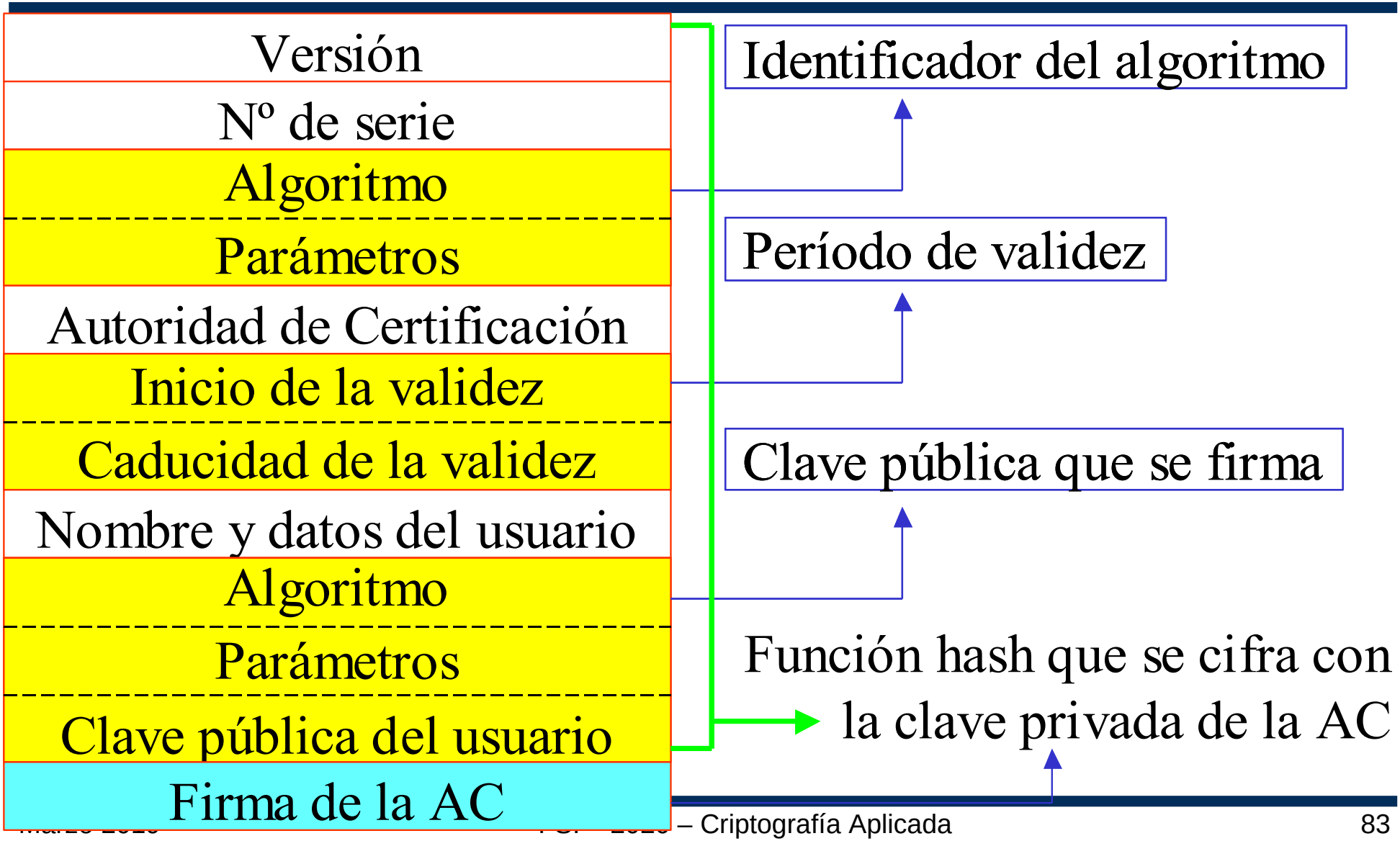
Validación del certificado

- Si tengo la clave pública de la CA: puedo verificar la firma de la CA
  - Me asegura que la información en el certificado es la misma que la CA firmó
  - Si confío en la CA, puedo asumir que la CA verificó la información contenida en el certificado

X.509

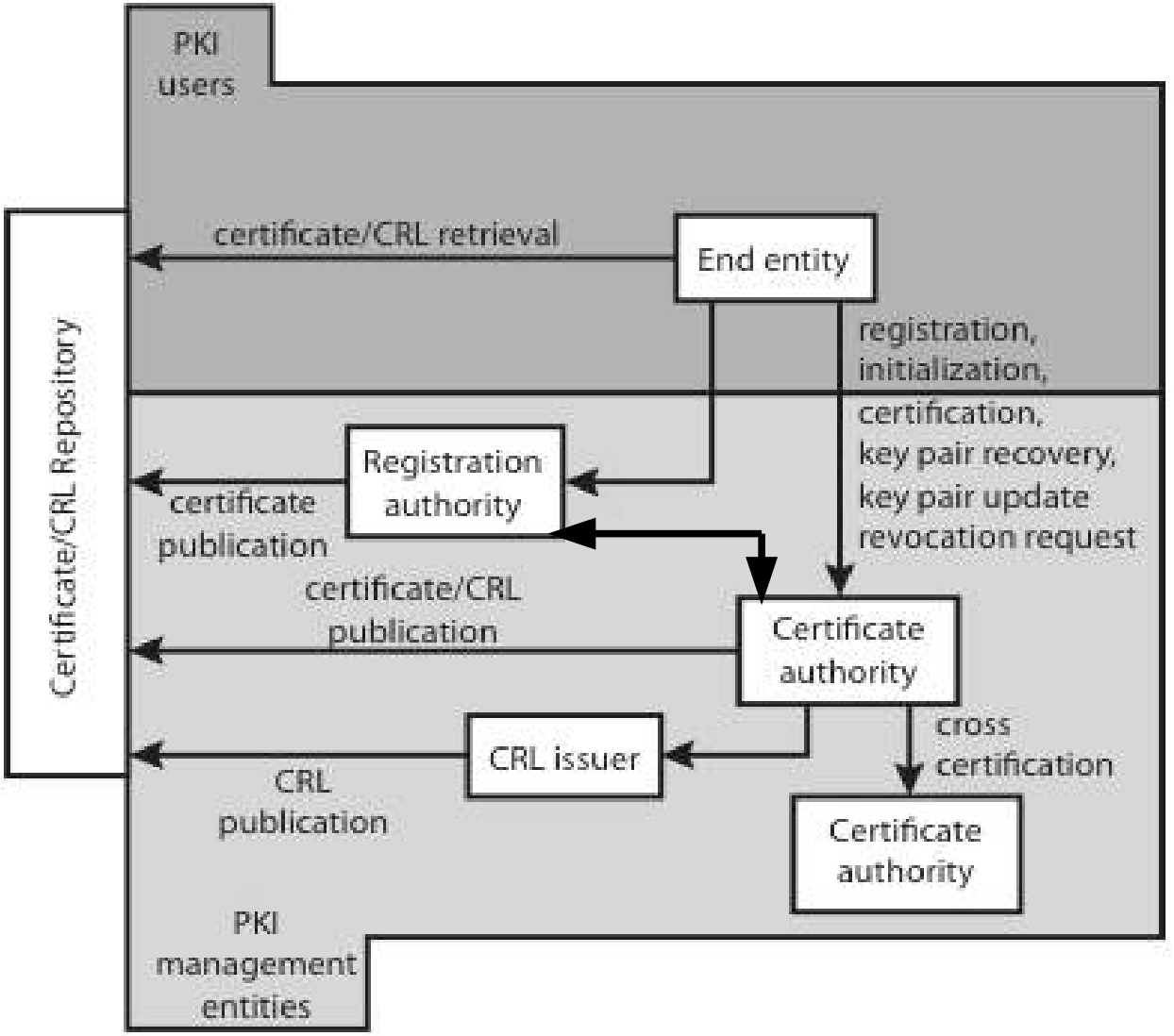
- Framework para brindar servicios de autenticación a los usuarios del directorio X.500
- Actualmente, X.509v3
- RFCs que instancian el framework para Internet

Esquema (simplificado) de un certificado x.509



Estructura de una PKI

(Stallings Figura 14.7)



Entidades

- CA: Autoridad de Certificación. Es quien firma los certificados
- RA: Autoridad de Registro. Es quien debería verificar la identidad del solicitante. Es la interfaz con la CA
- CRL: Certificate Revocation List. Lista de certificados revocados
  - Hay formatos para la información, y protocolos para acceder a ese repositorio

Algunas características

- El sistema de autenticación debe tener:
  - Una política de certificación
  - Un certificado de la CA
  - Los certificados de los usuarios (X.509)
  - Los protocolos de autenticación, gestión y obtención de certificados:
    - Se obtienen de bases de datos (directorio X.500)
    - O bien directamente del usuario en tiempo de conexión (Ej. WWW con SSL)

Obteniendo un certificado

- Cualquier usuario de la CA puede leer cualquier certificado
- Solo la CA (o quien conozca su clave privada) puede modificar un certificado
  - Los certificados pueden ponerse en un directorio público
  - Puedo aceptarlo de la propia entidad que quiere autenticarse!

Jerarquía de CAs

- Si los participantes comparten una CA:
  - Se asume que conocen su clave pública
- De lo contrario:
  - CAs deben formar una jerarquía
  - Se usan certificados para validar a otras CA
  - Cada CA tiene certificados de sus clientes, y de sus superiores
  - Permite a un usuario de un CA verificar los certificados de otras CAs en la jerarquía

Revocación de certificados

- Certificados tienen un período de validez
- Puede ser necesario forzar su expiración antes
  - Compromiso de la clave privada
  - Compromiso de la clave de la CA
  - La CA no certifica más a ese usuario
- La CA mantiene una lista de certificados revocados (CRL, Certificate Revocation List), y/o un servicio de consulta (OCSP)
- Usuarios deberían chequear la validez de cada certificado
- Ver

  http://news.netcraft.com/archives/2013/05/13/how-certificate-revocation-doesnt-work-in-practice.html

Quién decide en quién confiamos

- Queda claro en una CA interna a una empresa
- En el caso de los navegadores, otro decidió por nosotros...

Problemas de la vida real

- 2001. Verisign emite 2 certificados a nombre de Microsoft a “desconocidos”
  - Los certificados de Verisign no incluían link a una CRL
  - Internet Explorer no chequeaba CRLs
- 2006. Phishing contra “Mountain America Credit Union”
  - Sitio y certificado de www.mountain-america.net
  - ¿Quién puede registrar un determinado nombre?

Problemas de la vida real (2)

- ¿Marzo? 2011: Atacantes hackean una cuenta de un partner de una autoridad de certificación, Comodo, y generan 8 certificados (mail.google.com, www.google.com, etc.)
- Julio/Agosto 2011: “Hackers” logran penetrar la red de DigiNotar (una CA) y generar más de 200 certificados para distintos sitios

  ……..

Autenticación mediante claves compartidas

Autenticación mediante claves compartidas

- Clase particular de protocolos de autenticación
- Uso como soporte de otras operaciones
  - Comunicación entre bancos y cajeros automáticos
  - Pay-TV
  - Tarjetas Inteligentes: Global Platform
  - Red celular
- Ejemplo de gestión general de claves simétricas en sistemas distribuídos: Kerberos

Posibilidades para distribuir claves simétricas

- A genera la clave y se la entrega físicamente a B
- C selecciona la clave y la entrega a A y B
- A y B usaron recientemente una clave, y envían la nueva clave cifrada con la vieja clave
- A y B tienen una conexión cifrada con C -> C puede entregar una clave a A y B por los enlaces cifrados.
  - Key Distribution Center

Esquema Básico

- Principio: Dos entidades (principales) que quieren comunicarse utilizan un “tercero confiable”
- Alice y Bob (“communicating principals”)
- Sam (“trusted third-party”)
- Cada entidad precisa una clave compartida con Sam

$$
\begin{aligned}
&A \rightarrow S : A, B \\
&S \rightarrow A : \{A,B,K_{AB},T\}K_{AS}, \ \{A,B,K_{AB},T\}K_{BS} \\
&A \rightarrow B : \{A,B,K_{AB},T\}K_{BS}, \ \{M\}K_{AB}
\end{aligned}
$$

Protocolo de Needham-Schroeder

- La mayoría de los protocolos de distribución de claves han sido derivados a partir de este protocolo inventado en 1978
- Sigue el esquema básico discutido en la slide anterior pero usa nonces en lugar de timestamps
  - Nonce: número “use once”
- En su forma original tiene vulnerabilidades

Needham-Schroeder

(NS1) A -> S : A, B, N_A
(NS2) S -> A : {N_A,B,K_{AB},{K_{AB}, A}K_{BS}}K_{AS}
(NS3) A -> B : {K_{AB}, A}K_{BS}
(NS4) B -> A : {N_B}K_{AB}
(NS5) A -> B : {N_B - 1}K_{AB}

- Problema: Bob tiene que asumir que la clave K_{AB} que le envía Sam (vía Alice) es "fresca"

Kerberos (Generalidades)

- Importante derivado práctico de Needham-Schroeder
- Sistema de control de acceso distribuído originario del MIT y opción por defecto de autenticación en Windows 2000 y sucesores
- En lugar de una única trusted third-party:
  - Servidor de autenticación en el que los usuarios se autentican
  - Servidor dispensador de tickets que permiten acceder a recursos como archivos

Kerberos (Procedimiento simplificado)

- Alice obtiene del servidor de autenticación una clave de sesión K_{AS} (cifrada con su password), con un esquema similar al anterior
- Luego, para poder acceder a un recurso B, se aplica el siguiente protocolo (con el servidor de tickets S):

(K1) A -> S : A, B
(K2) S -> A : $\{T_S, L, K_{AB}, B, \{T_S, L, K_{AB}, A\}K_{BS}\}K_{AS}$
(K3) A -> B : $\{T_S, L, K_{AB}, A\}K_{BS}$, $\{A, T_A\}K_{AB}$
(K4) B -> A : $\{T_A + 1\}K_{AB}$

TLS/SSL

TLS (Transport Layer Security) SSL (secure sockets layer)

- SSL desarrollado originalmente por Netscape Communications Corporation
- Es independiente de la aplicación
- Última versión SSL v3 (obsoleto)
- Sucesor: TLS (Transport Layer Security)
  - Inicialmente muy similar a SSL v3
- Provee autenticación, integridad y privacidad
- Actualmente TLS 1.2 (RFC 5246, Agosto 2008)
- TLS 1.3 Draft. Algunos problemas de compatibilidad

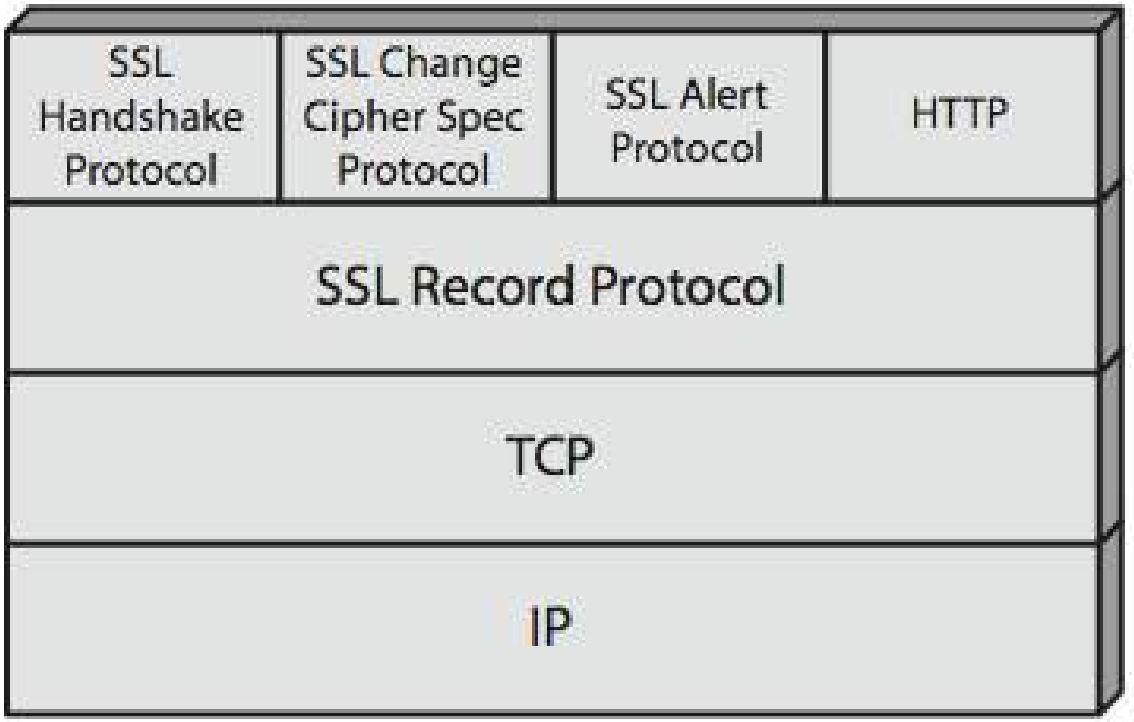
SSL/TLS

- Dividido en 2 capas, sobre una conexión stream confiable (TCP)
  - Funciona “entre” TCP y la aplicación
- SSL Record layer
  - Confidencialidad, autenticación, y protección contra replay
  - Define el formato para intercambiar los datos
- SSL Handshake Protocol
  - Intercambio de claves

Visión desde 1 km. de altura

- Cliente envía versión, protocolos soportados (key exchange, métodos de cifrado y algoritmos de Hash), y un número aleatorio
- Servidor envía su certificado, algoritmos elegidos y un número aleatorio
- Cliente chequea certificado
- Cliente genera clave secreta (premaster key), la encripta con clave pública del servidor y la envía
- Servidor la desencripta, y ambos generan una clave de sesión, que será usada a partir de ahí

Arquitectura SSL



Stallings Figura 17.2

Conexión y Sesión

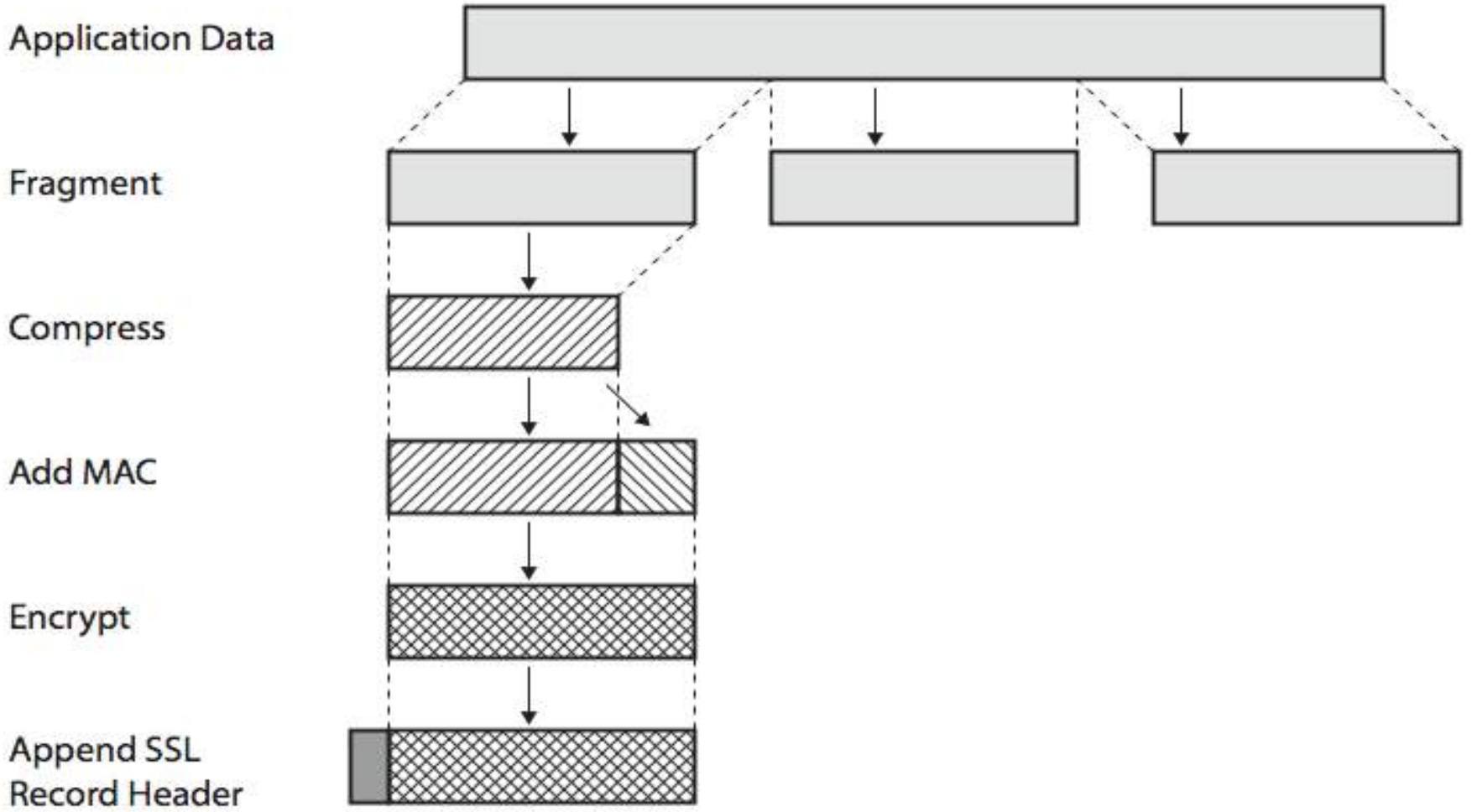
- Conexión SSL
  - Flujo transitorio entre equipos
  - Asociado con una sesión SSL
- Sesión SSL
  - Asociación entre cliente y servidor
  - Creada por el protocolo de Handshake
  - Define un conjunto de parámetros criptográficos
  - Puede ser compartida por múltiples conexiones
    - Evita negociar parámetros y verificar claves en cada conexión

Servicios del "Record Protocol"

- Integridad de mensajes
  - Usando una MAC con clave secreta compartida
  - TLS usa HMAC (SSL utilizaba un algoritmo parecido)
- Confidencialidad
  - Usando cifrado simétrico con una clave secreta definida por el protocolo de handshake
  - AES, IDEA, ~~RC2-40~~, ~~DES-40~~, ~~DES~~, 3DES, Fortezza, ~~RC4-40~~, RC4-128, etc., etc.
- Mensaje se comprime antes de cifrar (opcional)

Operación

(Stallings Figura 17.3)



Protocolo de Handshake

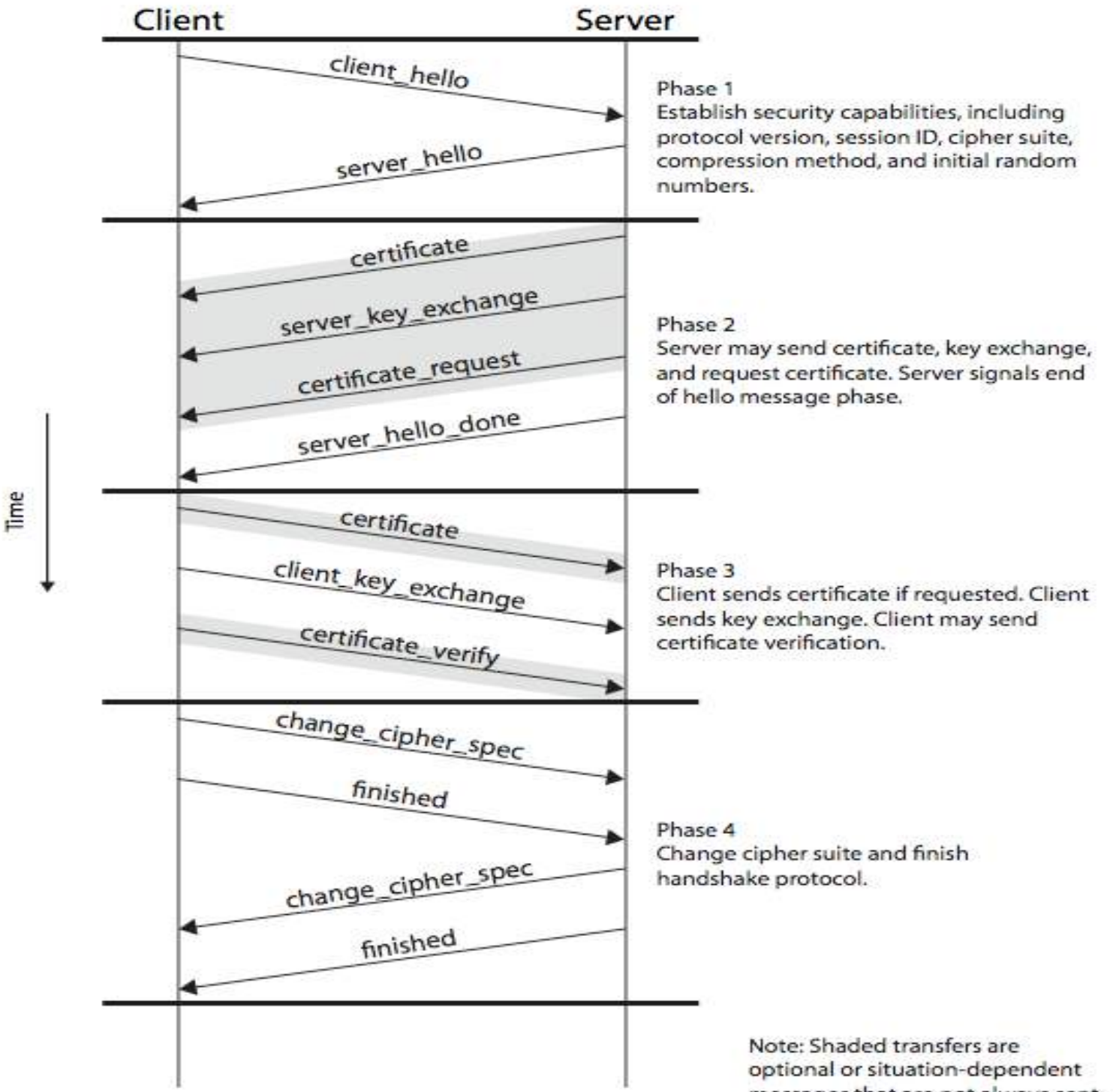
- Permite a cliente y servidor:
  - Autenticarse mutuamente
  - Negociar algoritmos de cifrado y MAC
  - Negociar claves criptográficas
- Varios mensajes en 4 fases

Handshake

1. Establecimiento de “capabilities”
   - Envío de versión, algoritmos soportados, etc.
   - Iniciada por el cliente
2. Autenticación del servidor e intercambio de clave
   - Servidor envía certificado
   - Servidor inicia intercambio de claves
   - Puede pedir certificado al cliente

Handshake

3. Autenticación del cliente e intercambio de clave
   - Verificar certificado del servidor
   - Enviar certificado si es requerido por el servidor
4. Fin. Se envía un “change cypher spec” y se termina el intercambio, comenzando a utilizarse los nuevos parámetros


Client
Server
client_hello
server_hello
Phase 1
Establish security capabilities, including protocol version, session ID, cipher suite, compression method, and initial random numbers.
certificate
server_key_exchange
certificate_request
server_hello_done
Phase 2
Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase.
Time
certificate
client_key_exchange
certificate_verify
Phase 3
Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.
change_cipher_spec
finished
change_cipher_spec
finished
Phase 4
Change cipher suite and finish handshake protocol.
Note: Shaded transfers are optional or situation-dependent messages that are not always sent.

Algoritmos y protocolos criptográficos - implementación

- La implementación de los algorimtos y protocolos criptográficos requiere no solamente la correctitud de la función a implementar, sino también tomar en cuenta otras amenazas
- En especial, todo lo que tiene que ver con ataques “Side channel”
- No los veremos aquí
  - Pero tener mucho cuidado si alguna vez intentan implementar algo
  - Vean por ejemplo la historia de OpenSSL

Bibliografía y Referencias

- **W. Stallings**, *Cryptography and Network Security,* Prentice Hall, 2006.
- **R. Rivest, A. Shamir, L. Adleman**, *A method for obtaining digital signatures and public key criptosystems,* Communications of the ACM, 1978.
- **C. Shannon**, *Communication Theory of Secrecy Systems,* Bell Systems Technical Journal, 1949.

Bibliografía y Referencias

- **A. Menezes, P. Oorschot, S. Vanstone**, *Handbook of Applied Cryptography,* CRC Press, 1996.
- **D. Stinson**, *Cryptography – Theory and Practice,* CRC Press, 1995.
- **G. Vernam**, *Cipher Printing Telegraph Systems for Secret Wire and Radio Telegraphic Communications,* Journal IEEE Vol. 55, 1926.

Bibliografía y Referencias

- Material online de apoyo a curso de Stallings
  - http://williamstallings.com/Crypto/Crypto4e-inst.html
- Handbook of Applied Cryptography. A. Menezes, P. van Oorschot, and S. Vanstone, CRC Press, 1996.
  - Online http://www.cacr.math.uwaterloo.ca/hac/
- Cryptored. Material docente de libre distribución en Internet. http://www.criptored.upm.es/guiateoria/gt_m001a.htm