2.1 INTRODUCCIÓN A LA CRIPTOGRAFÍA

2. Comunicaciones seguras

Requisitos de un sistema de comunicación seguro

- Un sistema de comunicación seguro debe proporcionar:
  - Autenticación de cada parte
  - Confidencialidad de datos
  - Integridad de datos
  - No repudio
- Para ello, se usan técnicas criptográficas

Clasificación de los sistemas criptográficos

- Por el tipo de operaciones (reversibles) de transformación:
  - Sustitución: A cada elemento del texto claro se le asigna otro elemento en el texto cifrado
  - Transposición/permutación: Reordenar los elementos del texto claro
  - Sistemas producto: Varias etapas de sustituciones y transposiciones
  - Numérica: Los elementos del texto claro se tratan como números a lo que se aplican algunas propiedades y relaciones matemáticas
- Por el número de claves usadas:
  - Simétrico o de clave secreta: Misma clave para cifrar y descifrar
  - Asimétrico o de clave pública: Una clave para cifrar, otra para descifrar
- Por la forma en la que se procesan los mensajes:
  - De bloque: Procesa un bloque del mensaje cada vez, produciendo un bloque de salida por cada bloque de entrada
  - De flujo: Procesa los elementos de entrada de forma continua, produciendo un elemento de salida cada vez

Criptoanálisis

- Es el proceso de intentar descubrir el texto claro o la clave a partir de la información disponible
- Un esquema de cifrado es **computacionalmente seguro** si el texto cifrado generado por el esquema cumple uno o ambos de los siguientes criterios:
  - El **coste** de romper el cifrado excede el **valor** de la información
  - El **tiempo** requerido para romper el cifrado excede la **vida útil** de la información

Tipos de criptoanálisis

- Los ataques se clasifican en función del tipo de información disponible para el atacante, asumiendo que el algoritmo de cifrado es conocido
  - Solo texto cifrado (*ciphertext-only*): el atacante solo tiene acceso al texto cifrado
  - Texto claro conocido (*known-plaintext*): el atacante tiene acceso a un conjunto de textos cifrados de los cuales se conoce el correspondiente texto claro
  - Texto claro elegido (*chosen-plaintext*): el atacante puede obtener los textos cifrados correspondientes a textos claros arbitrarios
  - Texto cifrado elegido (*chosen-ciphertext*): el atacante puede obtener los textos claros correspondientes a textos cifrados arbitrarios

Técnicas de criptoanálisis

- Análisis de frecuencias: Estudiar la frecuencia de aparición de letras o patrones en el texto cifrado
- Diccionario: Usar muchas contraseñas (≠claves) comunes
- Colisiones de claves: Aprovechar la paradoja del cumpleaños para producir colisiones de claves que revelen información acerca del texto claro
- Criptoanálisis diferencial: Analizar cómo las diferencias en el texto claro se corresponden con diferencias en el texto cifrado
- Criptoanálisis lineal: Usar aproximaciones lineales del comportamiento para derivar bits de la clave o del texto claro
- Claves relacionadas: Usar textos cifrados con dos claves diferentes con alguna relación (ej. compartiendo algunos bits)
- Claves débiles: Aprovechar claves que hacen que el cifrado funcione de alguna forma no deseable
- Combinación de técnicas: Derivar algunos bits de la clave o del texto plano con criptoanálisis y el resto por fuerza bruta

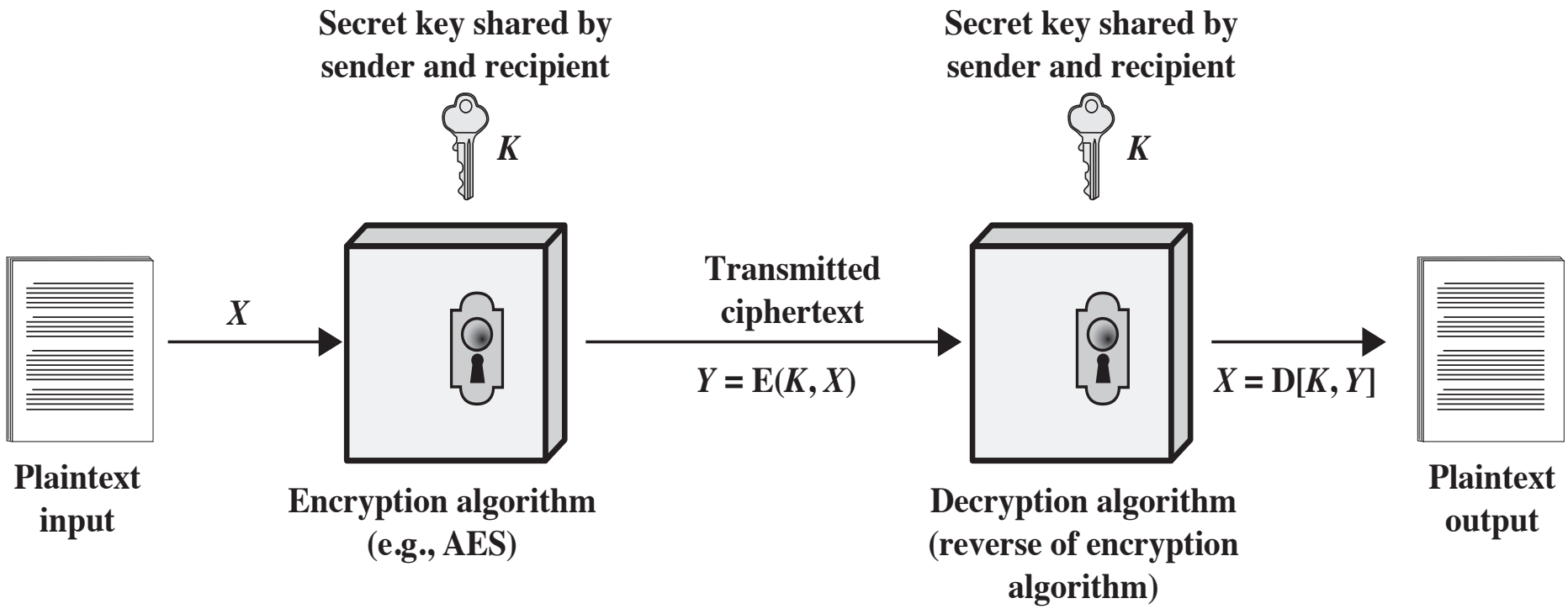
Ataques de fuerza bruta

- Implican intentar todas las posibles claves hasta que se obtenga una traducción inteligible del texto cifrado en texto claro
- En promedio, se debe intentar la mitad de todas las posibles claves para tener éxito
- A no ser que se proporcione el texto claro, el analista debe ser capaz de reconocer el texto claro como tal
- Normalmente, se presupone un cierto grado de conocimiento sobre el texto claro que se espera (lenguaje natural, código fuente, paquete de red…)

2.2 CRIPTOGRAFÍA DE CLAVE SECRETA

2. Comunicaciones seguras

Cifrado simétrico



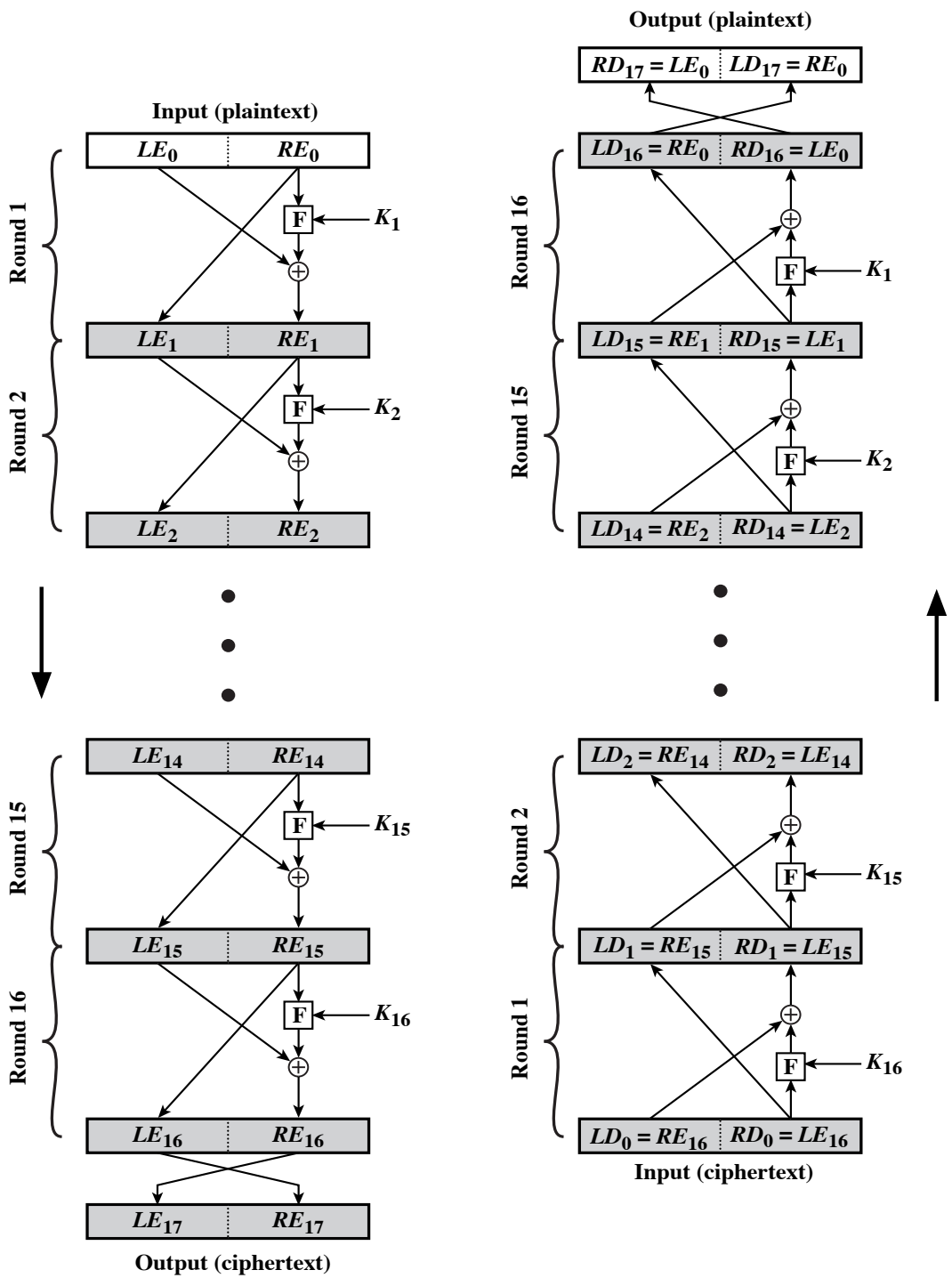
Requisitos

- Hay dos requisitos para el uso seguro del cifrado simétrico
  - Que el algoritmo de cifrado sea seguro
  - Que emisor y receptor hayan obtenido una copia de la clave secreta de forma segura y que permanezca segura
- La seguridad del cifrado simétrico no depende de que el algoritmo sea secreto, sino de que lo sea la clave
  - Esto lo hace factible para su uso generalizado, ya que el algoritmo se puede distribuir en *software* o *hardware*

Cifrado simétrico de bloque

- El cifrado de bloque procesa la entrada de texto claro en bloques de tamaño fijo y produce un bloque de texto cifrado de igual tamaño para cada bloque de entrada
- Los algoritmos de cifrado simétrico de bloque más importantes son:
  - *Data Encryption Standard* (DES)
  - *Triple DES* (3DES)
  - *Advanced Encryption Standard* (AES)

Red de Feistel (16 rondas)



Elementos de diseño de un sistema de cifrado de bloque

- Tamaño de bloque: Mayor tamaño significa mayor seguridad, pero menor velocidad de cifrado/descifrado
- Tamaño de clave: Mayor tamaño significa mayor seguridad, pero puede reducir la velocidad
- Número de rondas: Más rondas ofrecen mayor seguridad
- Algoritmo de generación de subclaves: Mayor complejidad conlleva mayor dificultad de criptoanálisis
- Función de ronda: Mayor complejidad generalmente significa mayor resistencia al criptoanálisis
- Velocidad de ejecución: En *hardware* y en *software*
- Facilidad de análisis: Si el algoritmo es conciso y claro, es más fácil analizar vulnerabilidades y evaluar su fortaleza

Data Encryption Standard (DES)

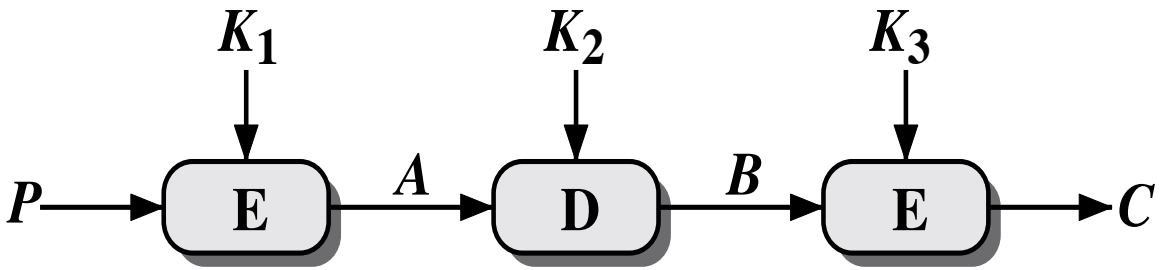
- Propuesto en 1977 por el NIST [FIPS 46]
- Fue el esquema de cifrado más ampliamente usado
- El algoritmo en sí se denomina *Data Encryption Algorithm* (DEA)
- Fue retirado en mayo de 2005

Algoritmo DES

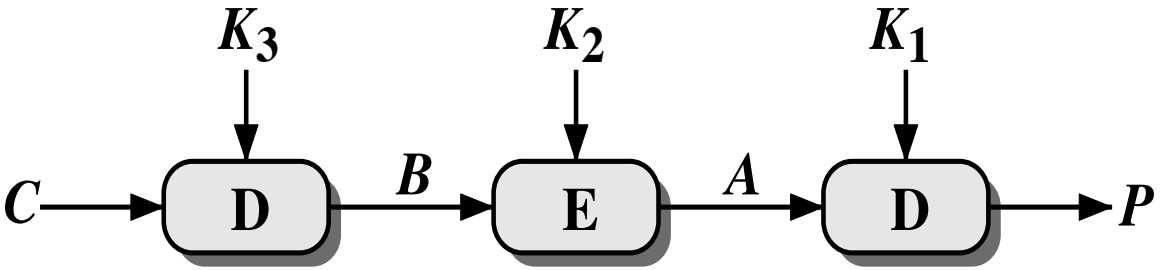
- Descripción del algoritmo:
  - Tamaño de bloque de 64 bits
  - Clave de 56 bits
  - Red de Feistel con pequeñas variaciones
  - 16 rondas de procesamiento
  - El proceso de descifrado es esencialmente el mismo que el de cifrado
- Preocupaciones sobre la seguridad de DES:
  - Sobre el algoritmo en sí: ¿es posible el criptoanálisis?
  - Sobre el uso de una clave de 56 bits: ¿es posible un ataque de fuerza bruta?

Triple DES (3DES o TDEA)

- Propuesto por primera vez en 1985 [X9.17], se añadió al estándar DES en 1999 [FIPS 46-3]
  - El tamaño efectivo de la clave es de 168 bits



(a) Encryption



(b) Decryption

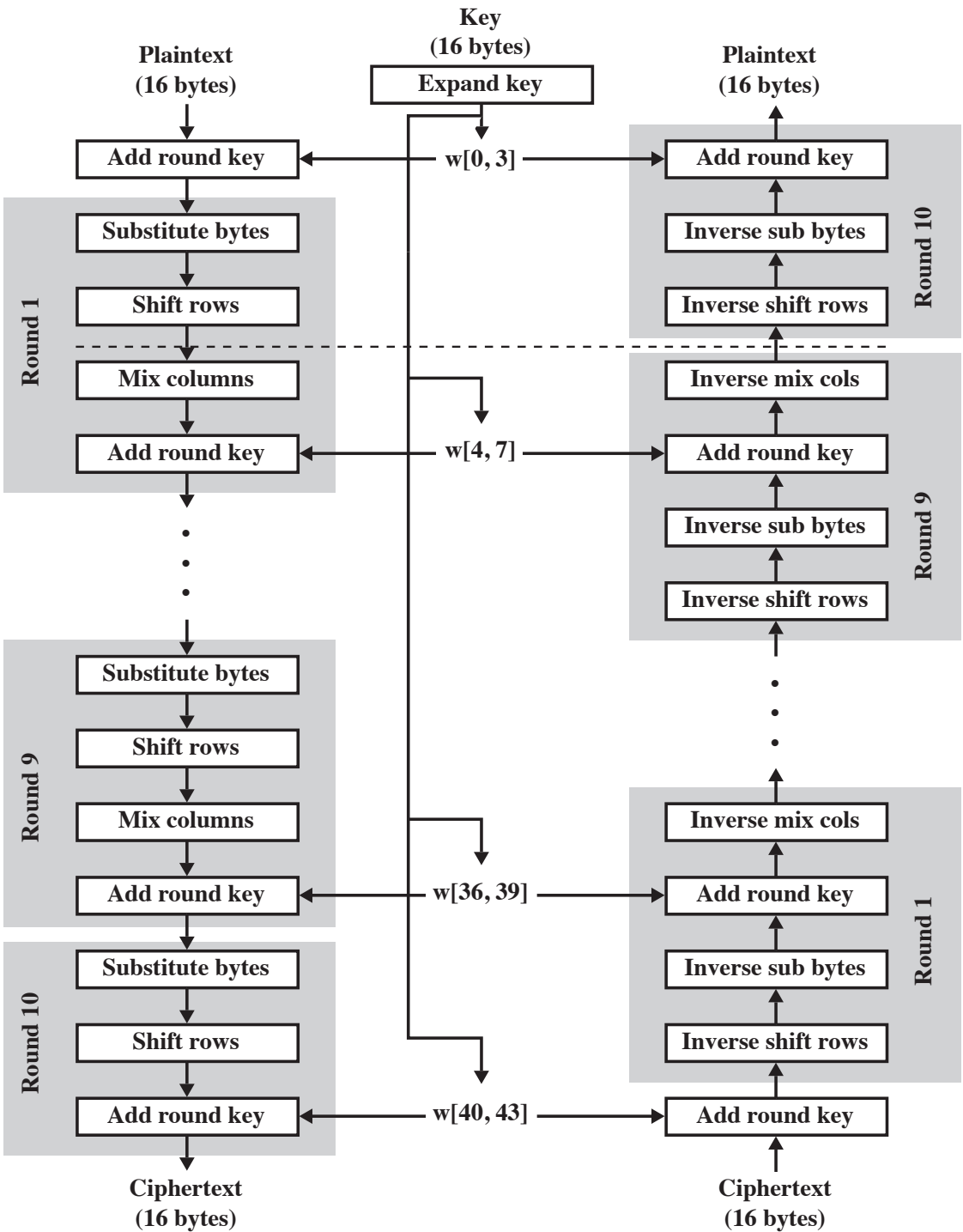
Advanced Encryption Standard (AES)

- DES fue diseñado para una implementación *hardware* y no producía código eficiente
  - Por eficiencia y seguridad, el tamaño de bloque debía ser mayor
- En 1997, el NIST lanzó una convocatoria para AES
  - Debía tener una seguridad igual o mayor que 3DES y mejorar significativamente la eficiencia
  - Debía usar cifrado simétrico de bloque, con un tamaño de bloque de 128 bits, y soportar claves de 128, 192 y 256 bits
  - Los criterios de evaluación incluían seguridad, eficiencia, requisitos de memoria, idoneidad *hardware* y *software*…
- Finalmente, NIST seleccionó el algoritmo Rijndael
  - Se publicó en 2001 [FIPS 197]
  - Los autores son los criptógrafos belgas Dr. Daemen y Dr. Rijmen

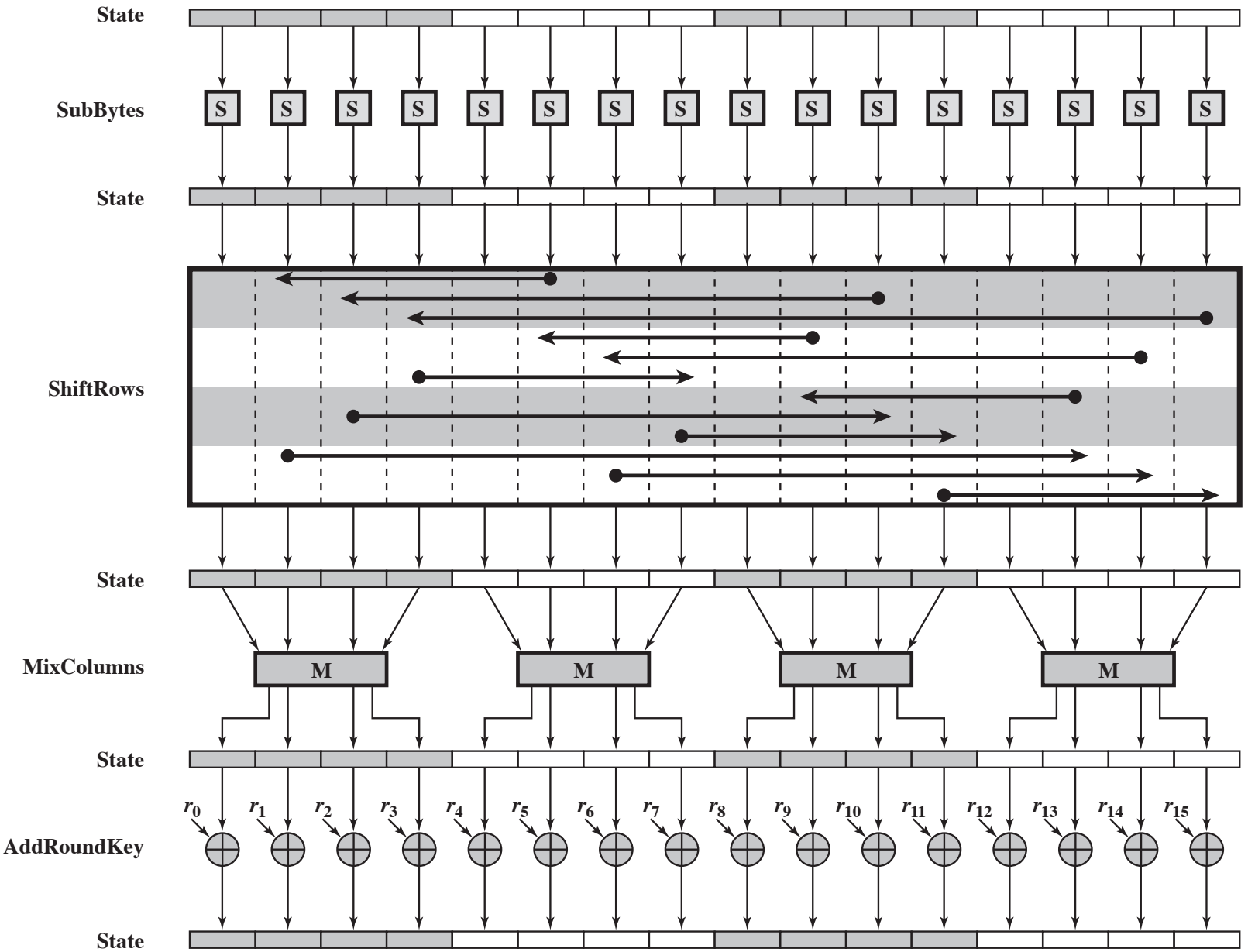
Cifrado y descifrado AES

Red de sustitución-permutación (no es Feistel)

Rápida tanto en software como en hardware



Ronda de cifrado AES



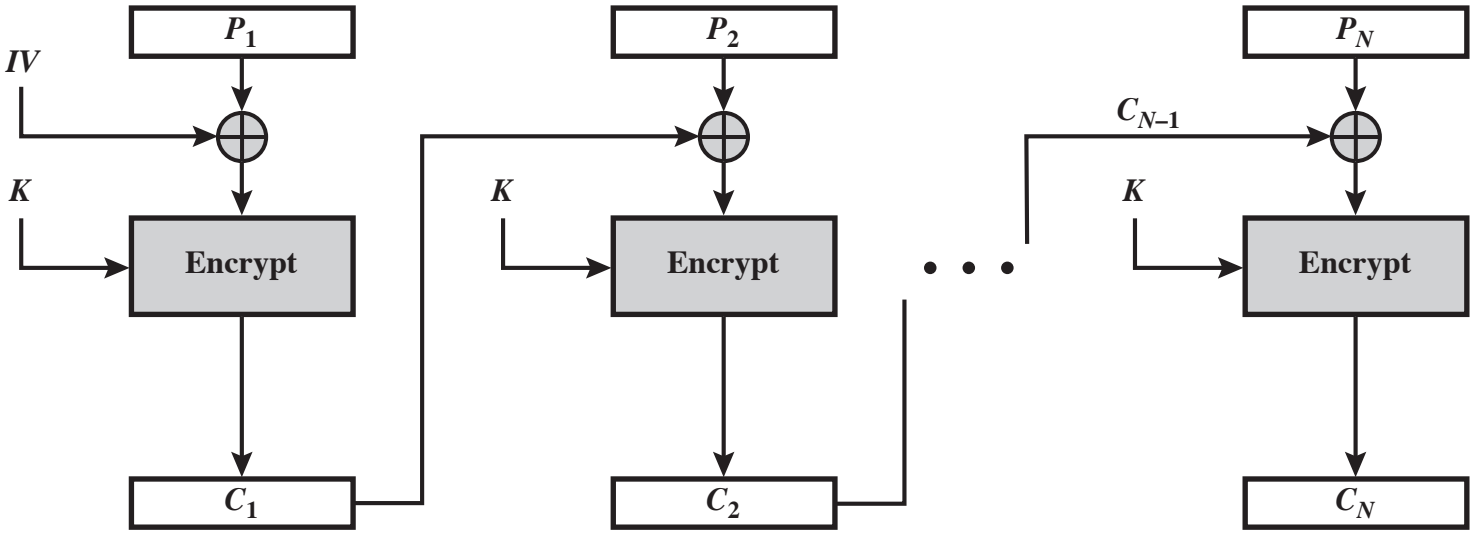
Modos de operación para cifrado de bloque

- El cifrado de bloque procesa un bloque de dato cada vez
  - Para DES y 3DES, el tamaño de bloque es de 64 bits
  - Para AES, el tamaño de bloque es de 128 bits
- Para tamaños mayores de texto claro, es necesario dividirlo en bloques, rellenando el último bloque si fuera necesario
- Se han definido cinco modos básicos [SP 800-38A]
  - Pueden usarse con cualquier cifrado simétrico de bloque
  - Cubren cualquier posible aplicación criptográfica del cifrado de bloque
  - Algunos modos incluso permiten crear sistemas de cifrado de flujo, o bien, proporcionar autenticación o cifrado autenticado

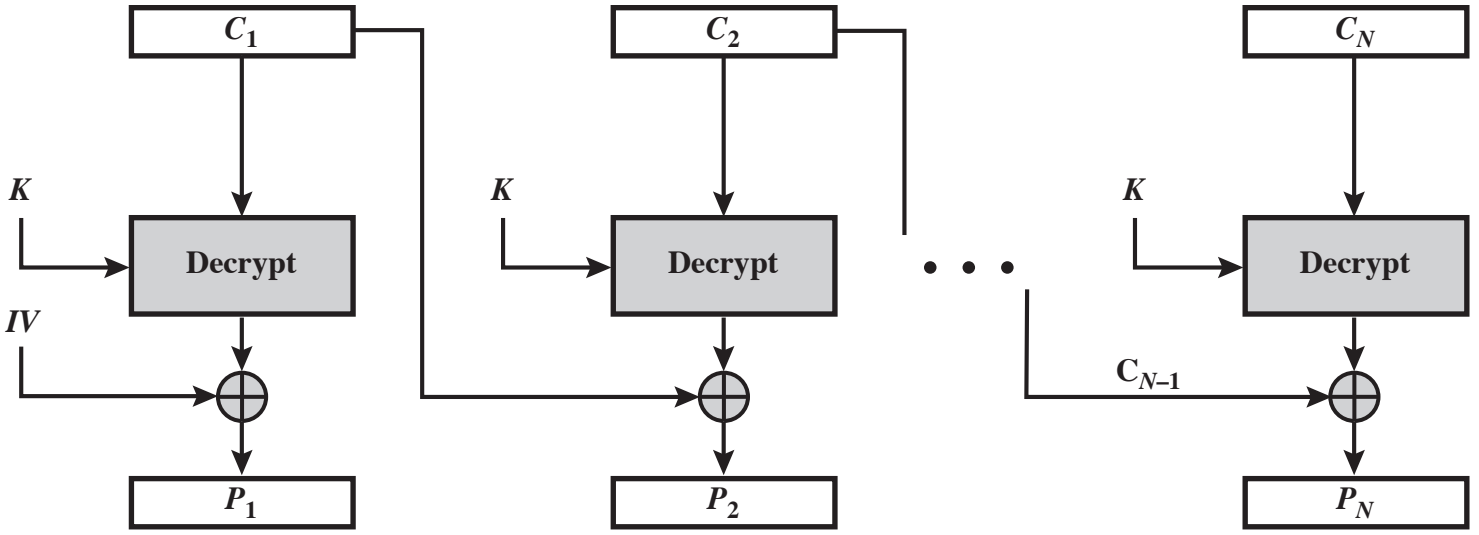
Electronic Codebook (ECB)

- El texto claro se procesa en bloques de bits y cada bloque se cifra usando la misma clave
  - Para una clave dada, hay un único texto cifrado para cada bloque de texto claro
  - De ahí el término "*codebook*"
- Para mensajes largos, puede no ser seguro
- Si el mensaje es muy estructurado, un criptoanalista podría explotar patrones y repeticiones

Cipher Block Chaining (CBC)

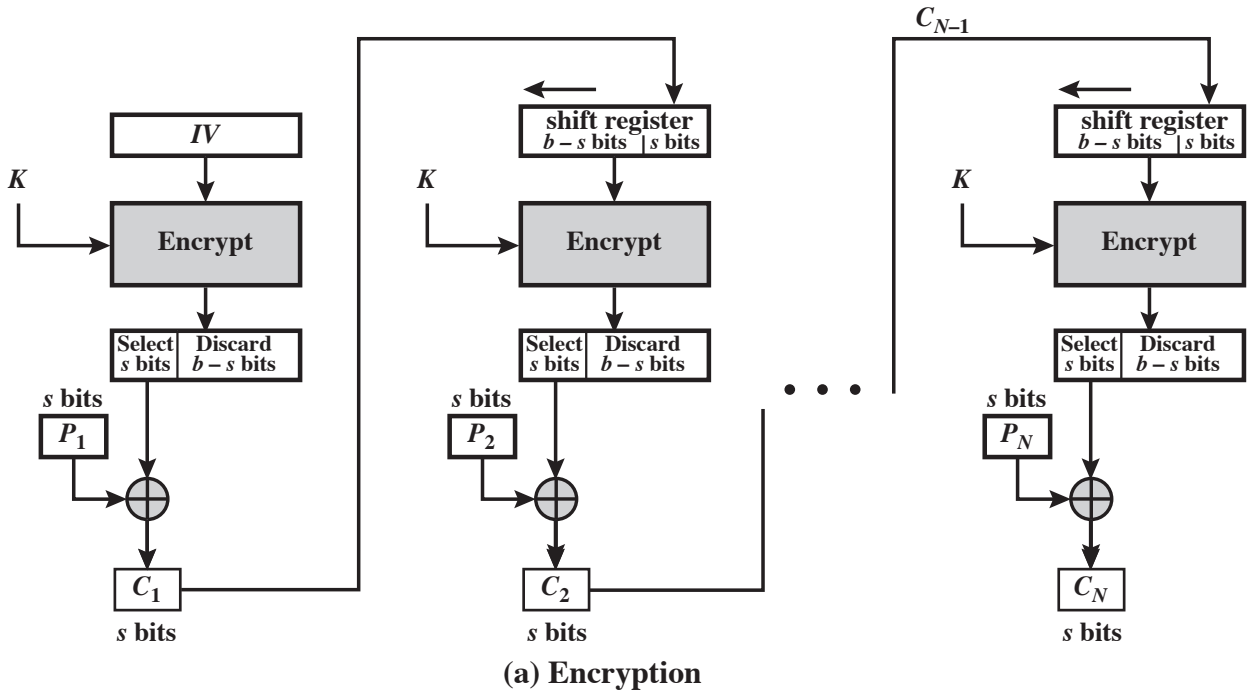


(a) Encryption

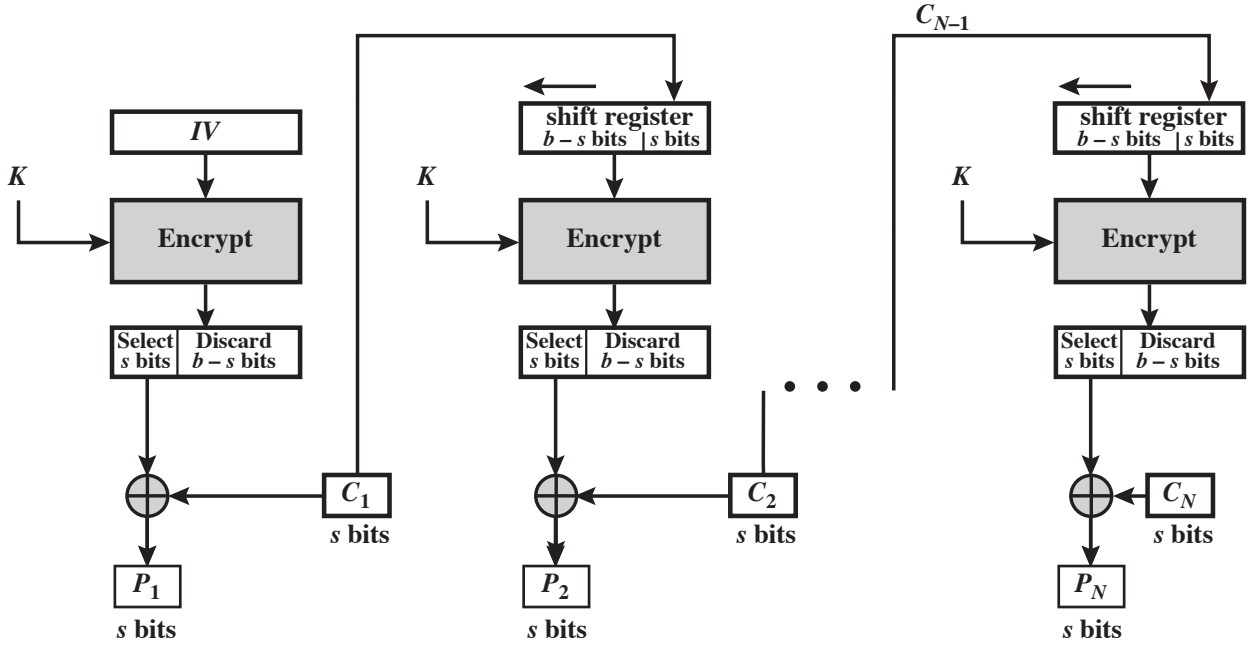


(b) Decryption

Cipher Feedback (CFB)

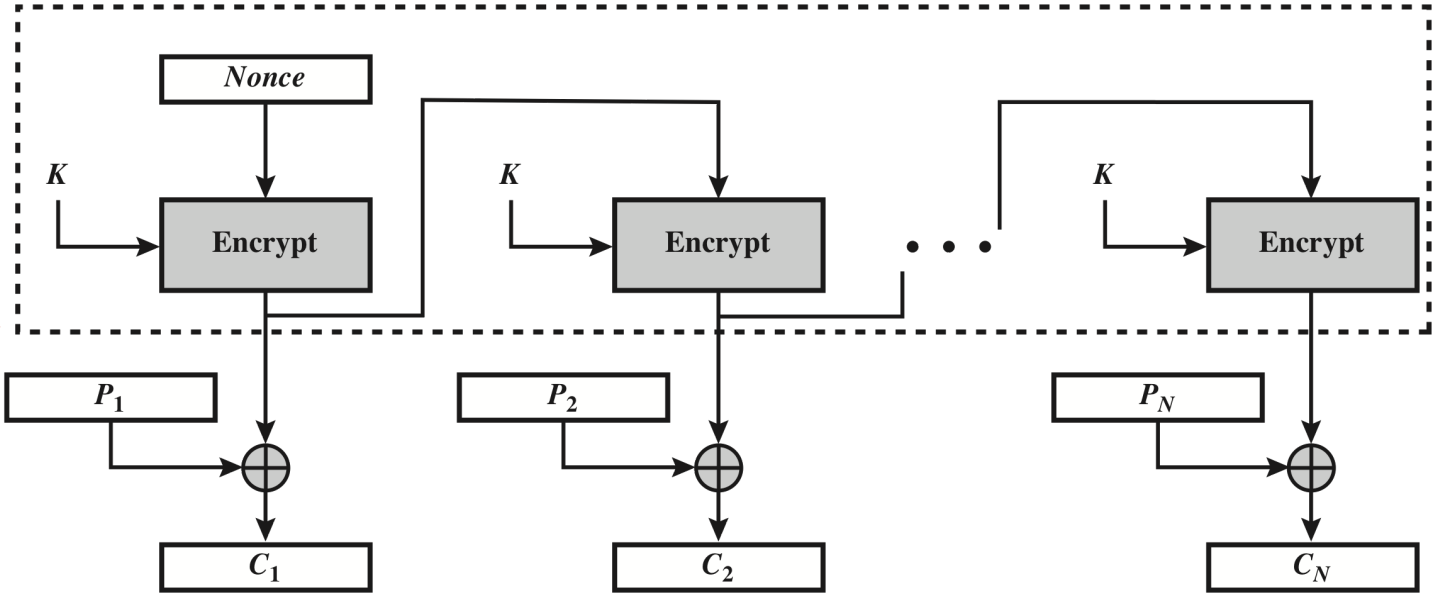


(a) Encryption

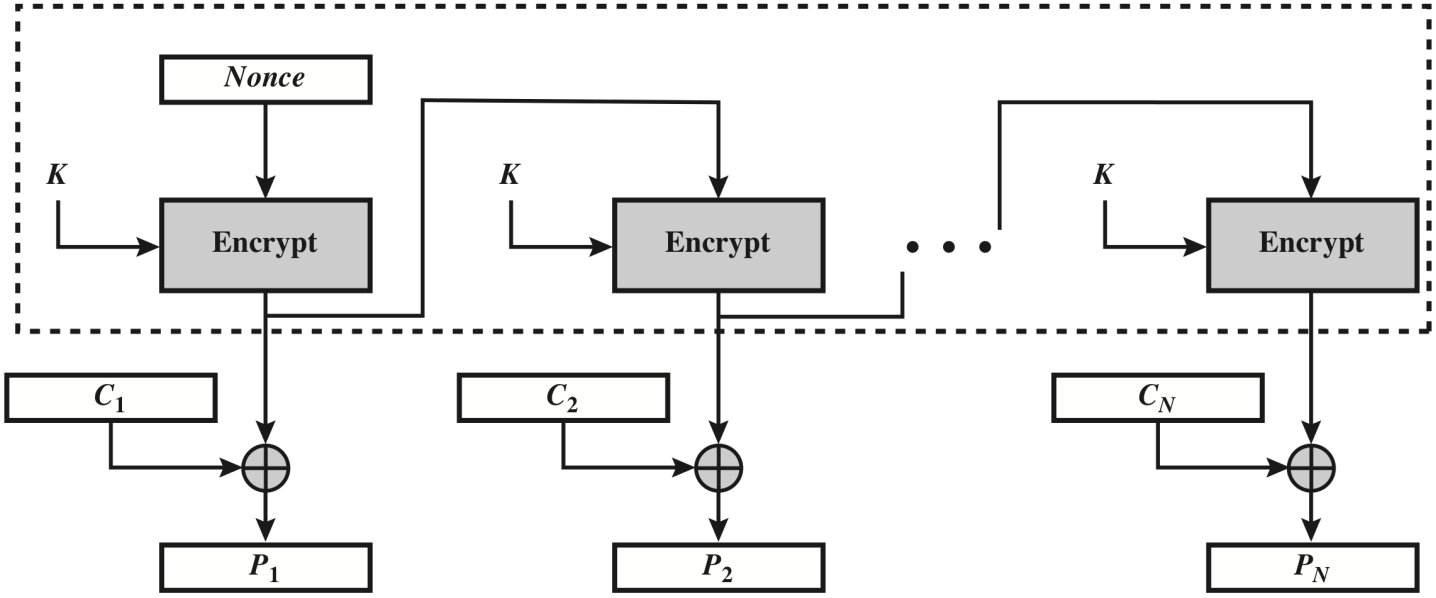


(b) Decryption

Output Feedback (OFB)

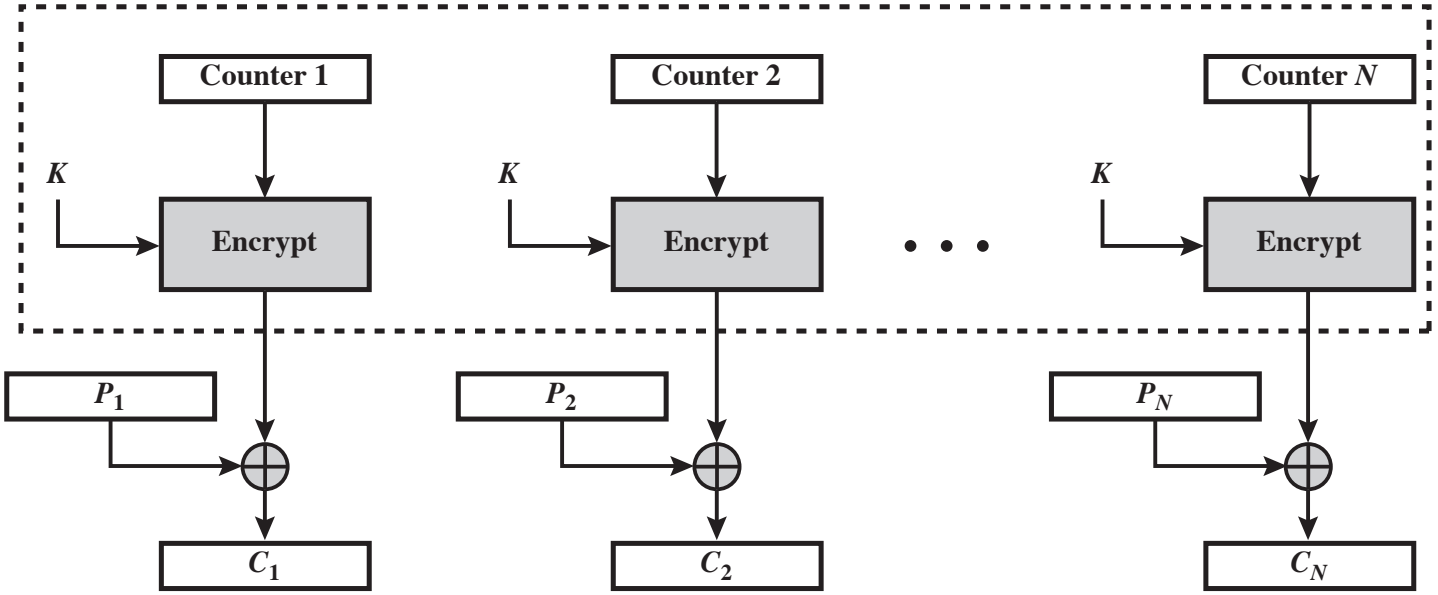


(a) Encryption

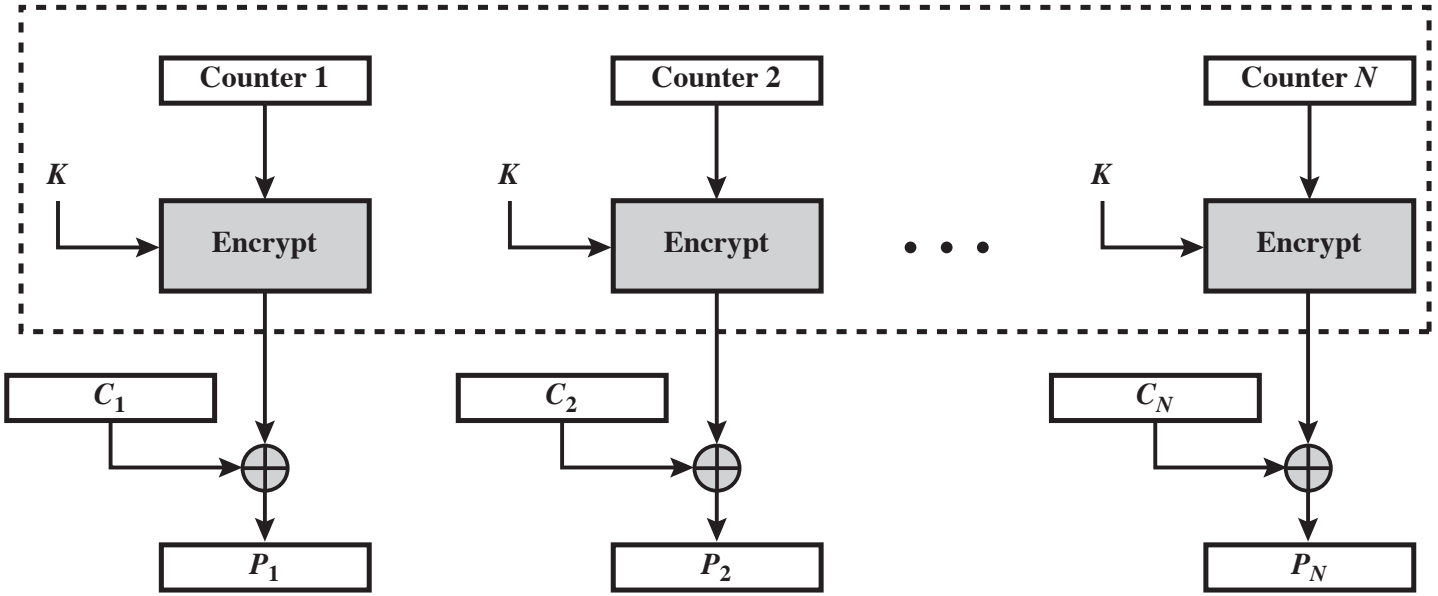


(b) Decryption

Counter (CTR)



(a) Encryption

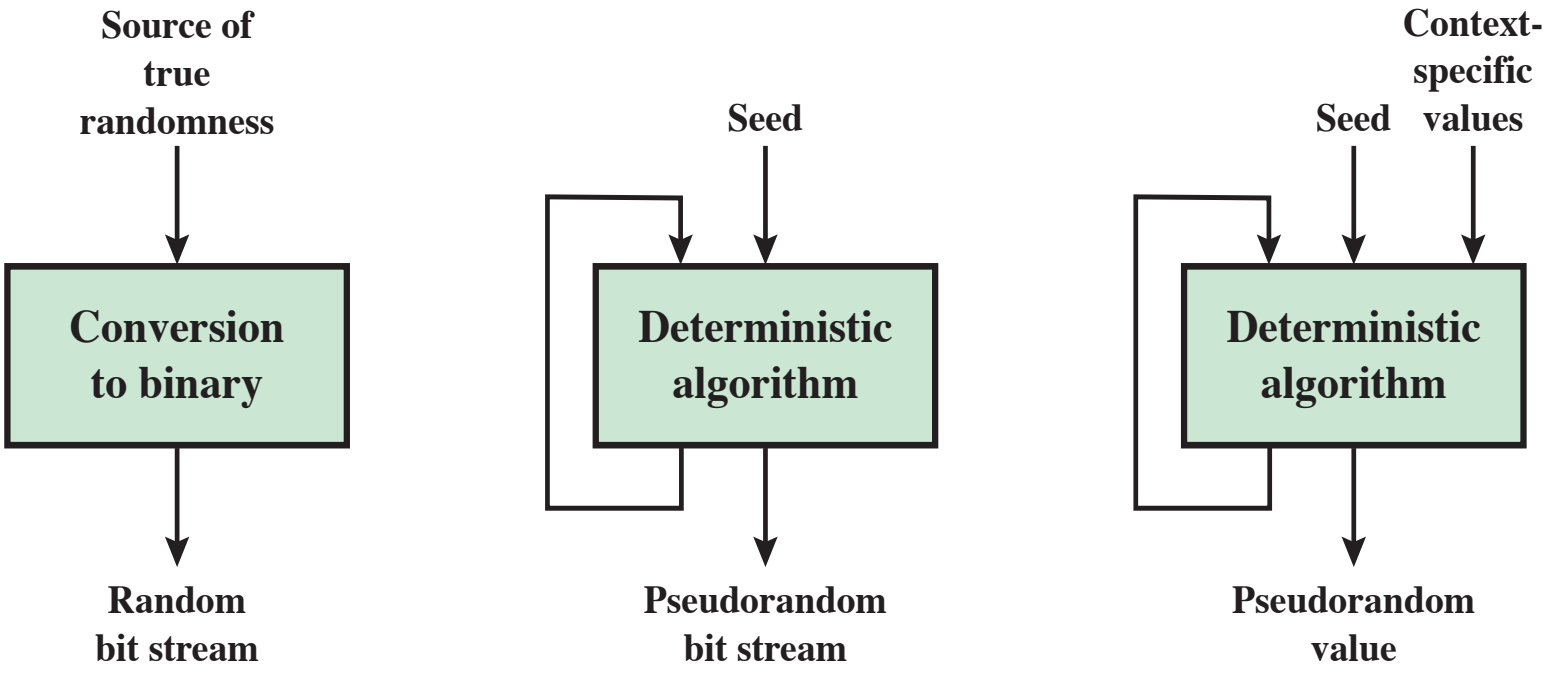


(b) Decryption

Números aleatorios y pseudoaleatorios

- Muchos algoritmos criptográficos utilizan números aleatorios
  - Generación de claves en criptografía de clave pública
  - Generación de claves de sesión temporales en criptografía de clave secreta
  - Evitar ataques de repetición en acuerdos o negociaciones
  - Generación de flujos de claves en en cifrado de flujo
- Existen dos requisitos, no necesariamente compatibles, para una secuencia de números aleatorios:
  - Aleatoriedad (distribución uniforme e independencia)
  - Impredecibilidad (incluso más importante)

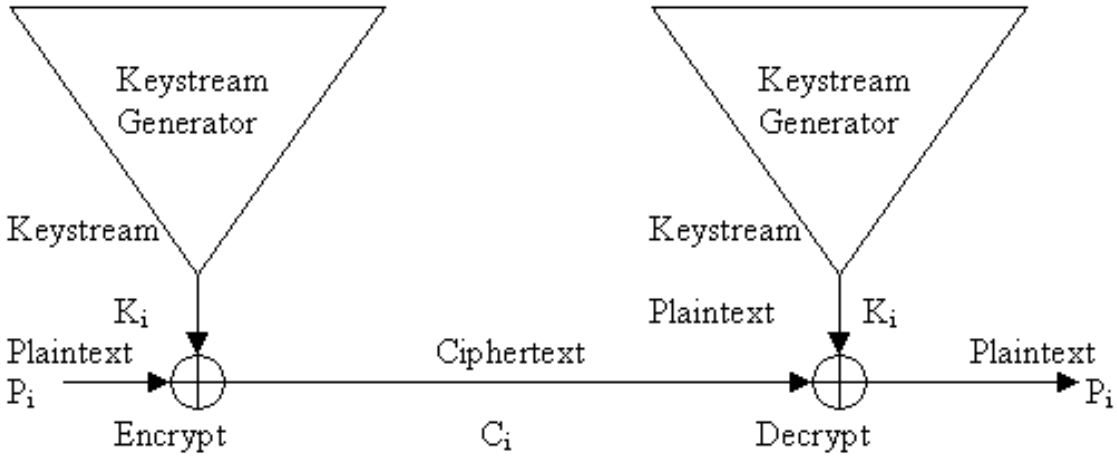
Generadores de números aleatorios



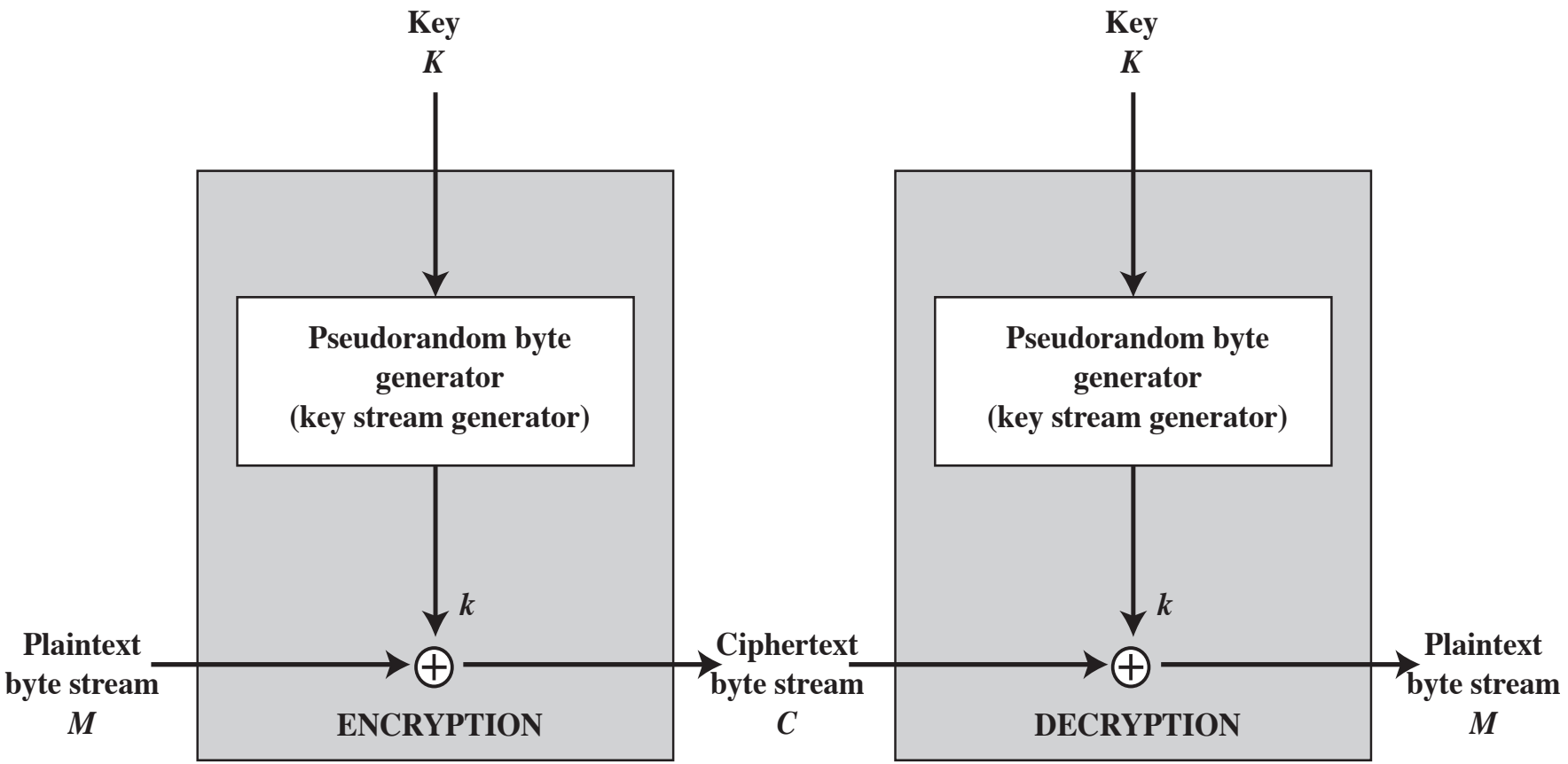
TRNG = true random number generator
PRNG = pseudorandom number generator
PRF = pseudorandom function

Cifrado simétrico de flujo

- Se combinan los dígitos del texto claro con un flujo pseudoaleatorio de dígitos (flujo de claves)
  - Normalmente, usando una XOR (cifrado aditivo binario)
- En el cifrado de flujo **síncrono**, el flujo de claves se genera en función de la clave solamente
- En el cifrado de flujo **asíncrono** (o auto-sincronizable), el flujo de claves se genera en función de la clave y de un número fijo de dígitos previos del texto cifrado



Cifrado de flujo síncrono



Propiedades del cifrado de flujo síncrono

- Emisor y receptor deben estar sincronizados, operando en la misma posición con la misma clave
  - Si se pierde la sincronización, el descifrado solo se puede restaurar con técnicas adicionales de resincronización (reinicio, marcas especiales, prueba de distintas posiciones…)
  - La inserción, borrado o repetición de dígitos del texto cifrado mediante un ataque activo provoca la pérdida inmediata de la sincronización, y por tanto puede ser detectada
- La modificación de un dígito del texto cifrado durante la transmisión no afecta a otros dígitos del texto cifrado
  - Un ataque activo podría hacer cambios en dígitos del texto cifrado seleccionados, sabiendo qué efecto tienen en el texto plano
    - Se necesitan mecanismos adicionales para autenticación de mensajes
- Ejemplos: RC4 y cifrado de bloque en modo CTR

Consideraciones de diseño del cifrado de flujo síncrono

- El flujo de claves debe tener un periodo largo
  - Cuanto más tarde en repetirse, más difícil será el criptoanálisis
- El flujo de claves debe parecerse lo máximo posible a un flujo de números aleatorios reales
  - Cuando más aleatorio parezca, más aleatorio parecerá el texto cifrado, haciendo más difícil el criptoanálisis
- El generador de números aleatorios está condicionado por el valor de la clave de entrada
  - Para protegerse de ataques de fuerza bruta, la clave de entrada debe ser suficientemente grande (al menos 128 bits con la tecnología actual)

Algoritmo RC4

- Diseñado en 1987 por Ron Rivest para RSA Security
  - Se mantuvo como secreto comercial, hasta que fue publicado de manera anónima en septiembre de 1994
- Cifrado de flujo síncrono
- Permutación aleatoria con tamaño de clave variable de 1 a 256 bytes (8 a 2048 bits) y orientado a byte
- Sencillo y rápido
- Con una clave de 128 bits, el período del flujo de claves es mayor que 10^{100}
- Usado en SSL/TLS, WEP y WPA
  - Actualmente obsoleto
  - Desde febrero de 2015, su uso está prohibido en TLS [RFC 7465]

RC4 en C

```c
#include <stdio.h>

char S[256];
int i, j;

void swap(int i, int j) {
  char tmp = S[i]; S[i] = S[j]; S[j] = tmp;
}

/* Key-Scheduling Algorithm */
void ksa(char *key, int key_len) {
  for (i = 0; i < 256; i++)
    S[i] = i;

  for (i = j = 0; i < 256; i++) {
    j = (j + key[i%key_len] + S[i]) & 255;
    swap(i, j);
  }
  i = j = 0;
}
```

```c
/* Pseudo-Random Generation Algorithm */
unsigned char prga() {
  i = (i + 1) & 255;
  j = (j + S[i]) & 255;

  swap(i, j);

  return S[(S[i] + S[j]) & 255];
}

void main(int argc, char *argv[]) {
  char *p = argv[1];

  ksa(argv[2], 5);

  while (*p)
    printf("%02X", *p++ ^ prga());

  printf("\n");
}
```

```
$ ./rc4 "Attack at dawn" "Secre"
14246F1CCFE55713C92202F6D31F
$ echo -ne "Attack at dawn" | openssl enc -rc4-40 -K 5365637265 | xxd -up
14246F1CCFE55713C92202F6D31F
```

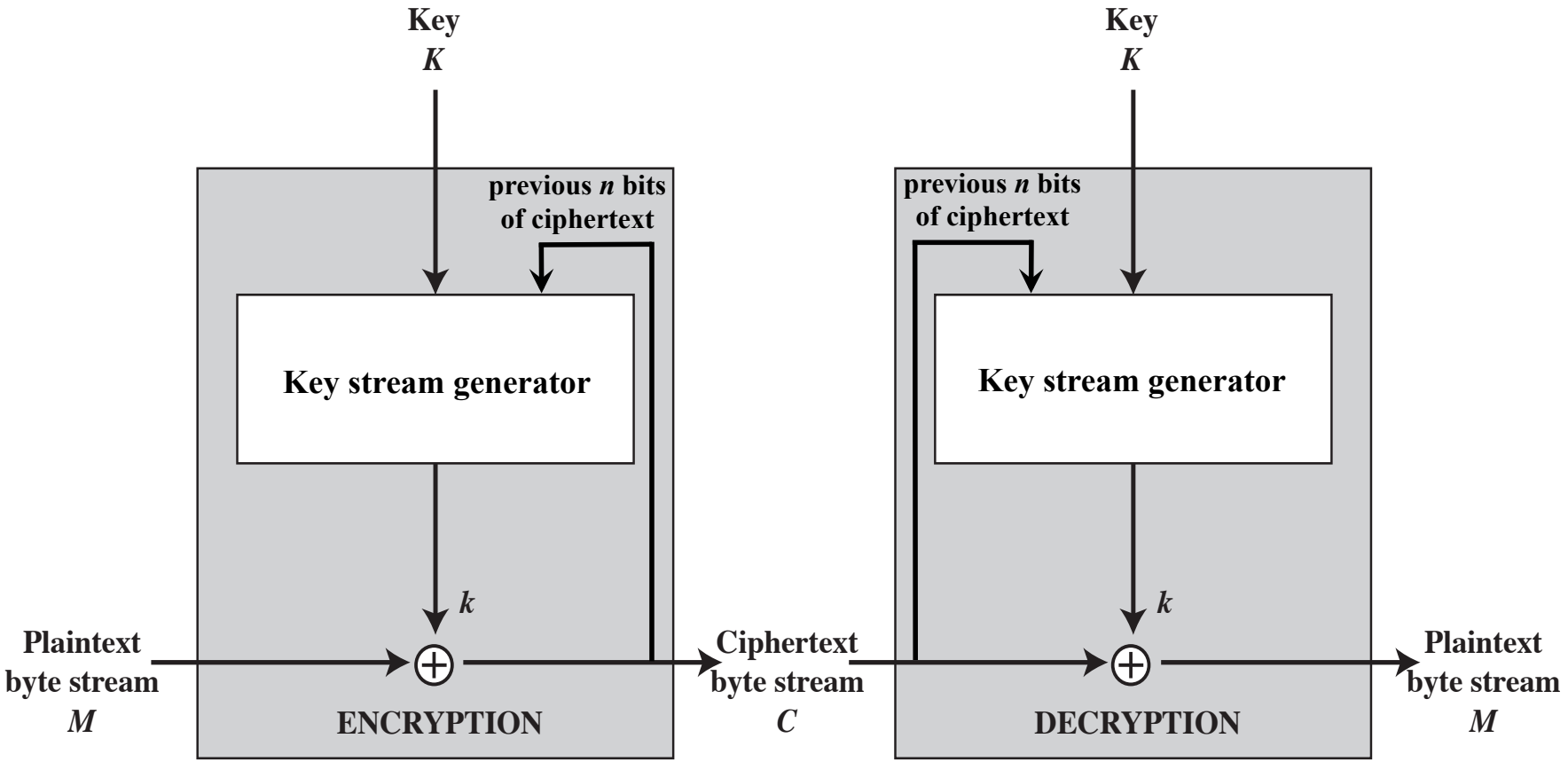
Seguridad de RC4

- Vulnerabilidades
  - El segundo byte del flujo de caves está sesgado hacia cero con alta probabilidad
  - Los primeros bytes son poco aleatorios y revelan información acerca de la clave
- Defensa
  - Descartar los valores iniciales del flujo de claves

Seguridad de sistemas basados en RC4

- WEP es vulnerable a varios ataques
  - Para evitar repeticiones de la clave RC4 en cada paquete
    - Para crear la clave RC4, se **concatena** un vector de inicialización (IV) de 24 bits con la clave WEP de 40 o 104 bits
    - Se envía el IV en claro junto con el paquete cifrado
  - Análisis de paquetes con la misma clave (cancelación XOR)
    - 50% de probabilidad de que se repita uno IV tras 5000 paquetes
    - Algunos paquetes son fáciles de identificar y descodificar (e.g. ARP)
  - Ataque de clave relacionada obtiene la clave [Fluhrer,Mantin,Shamir]
    - Necesita muchos paquetes con IV "débiles" y el primer byte del flujo de claves (el primer byte de la trama 802.11 es siempre 0xAA)
- Otros sistemas que usan RC4 no son vulnerables
  - TKIP (en WPA) y SSL/TLS usan un mecanismo similar, pero hacen un ***hash*** del IV y la clave WEP para crear la clave RC4
    - Las repeticiones de clave son improbables y no hay relación entre claves

Cifrado de flujo asíncrono



Propiedades del cifrado de flujo asíncrono

- La auto-sincronización es posible
  - El descifrado puede restaurarse automáticamente después de una pérdida de sincronización, con solo un número fijo de caracteres del texto claro no recuperables
- Los errores se propagan hasta un número de dígitos del texto cifrado subsecuentes
  - Es más difícil detectar ataques activos de inserción, borrado o repetición de dígitos en el texto cifrado
    - Se necesitan mecanismos adicionales para autenticación de mensajes
  - Es más resistente frente a ataques basados en la redundancia del texto claro
- Ejemplo: Cifrado de bloque en modo CFB

Resumen

- Cifrado simétrico
- Cifrado simétrico de bloque
  - Red de Feistel
  - DES y 3DES
  - AES
  - Modos de bloque: ECB, CBC, CFB, OFB y CTR
- Números aleatorios y pseudoaleatorios
- Cifrado simétrico de flujo
  - Síncrono
    - RC4
  - Asíncrono

2.3 FUNCIONES RESUMEN

2. Comunicaciones seguras

Funciones resumen (*hash*)

- Tiene un mensaje *M* de tamaño variable como entrada y producen un resumen (*digest*) del mensaje *H*(*M*) de tamaño fijo como salida
- No tiene una clave secreta como entrada
- Se usan para la autenticación de mensajes, el almacenamiento de contraseñas y la firma digital
  - Produce una huella (*fingerprint*) de un fichero, mensaje o cualquier otro bloque de datos
- También se usan para la generación de números pseudo-aleatorios

Funciones resumen seguras

- Para ser útil para autenticación de mensajes, una función resumen *H* debe tener las siguientes propiedades:
  1. Tiene entrada de cualquier tamaño y salida de tamaño fijo
  2. Es relativamente fácil calcular *H(x)* para cualquier *x* dado, permitiendo implementaciones *hardware* y *software*
  3. Dado *h*, es computacionalmente inviable encontrar un *x* tal que *H(x)* = *h* (de un sentido o resistencia a pre-imagen)
  4. Dado *x*, es computacionalmente inviable encontrar un *y* con *H(y)* = *H(x)* (resistencia débil a colisiones o a 2ª pre-imagen)
  5. Es computacionalmente inviable encontrar un par (*x*, *y*) tal que *H(x)* = *H(y)* (resistencia fuerte a colisiones)
- La resistencia fuerte protege frente a ataques de colisiones, que aprovechan la paradoja del cumpleaños
  - La resistencia débil protege frente a ataques de pre-imagen

Seguridad de las funciones resumen

- Existen dos formas de atacar una función resumen:
  - Criptoanálisis
    - Consisten en explotar debilidades lógicas del algoritmo
  - Fuerza bruta
    - La fortaleza de la función resumen depende únicamente del tamaño del código *hash* producido por el algoritmo

Secure Hash Algorithm (SHA)

- En 1993, NIST publica *Secure Hash Standard* [FIPS 180]
  - Basado en el algoritmo MD4 de Ron Rivest
- En 1995, publica SHA-1 [FIPS 180-1] con un ligero cambio debido a un fallo significativo no revelado
  - La versión original pasa a conocerse como SHA-0
- En 2001, publica SHA-2 [FIPS 180-2]
  - Revisado en 2008 [FIPS 180-3] y 2010 [FIPS 180-4]
- En 2005, se identifican fallos de seguridad en SHA-1, que comenzó a reemplazarse por SHA-2
- En 2012, una competición del NIST selecciona una nueva función SHA-3 [FIPS 202]
  - No se basa en SHA-2 y no pretende reemplazarlo, sino que NIST percibe la necesidad de una alternativa diferente

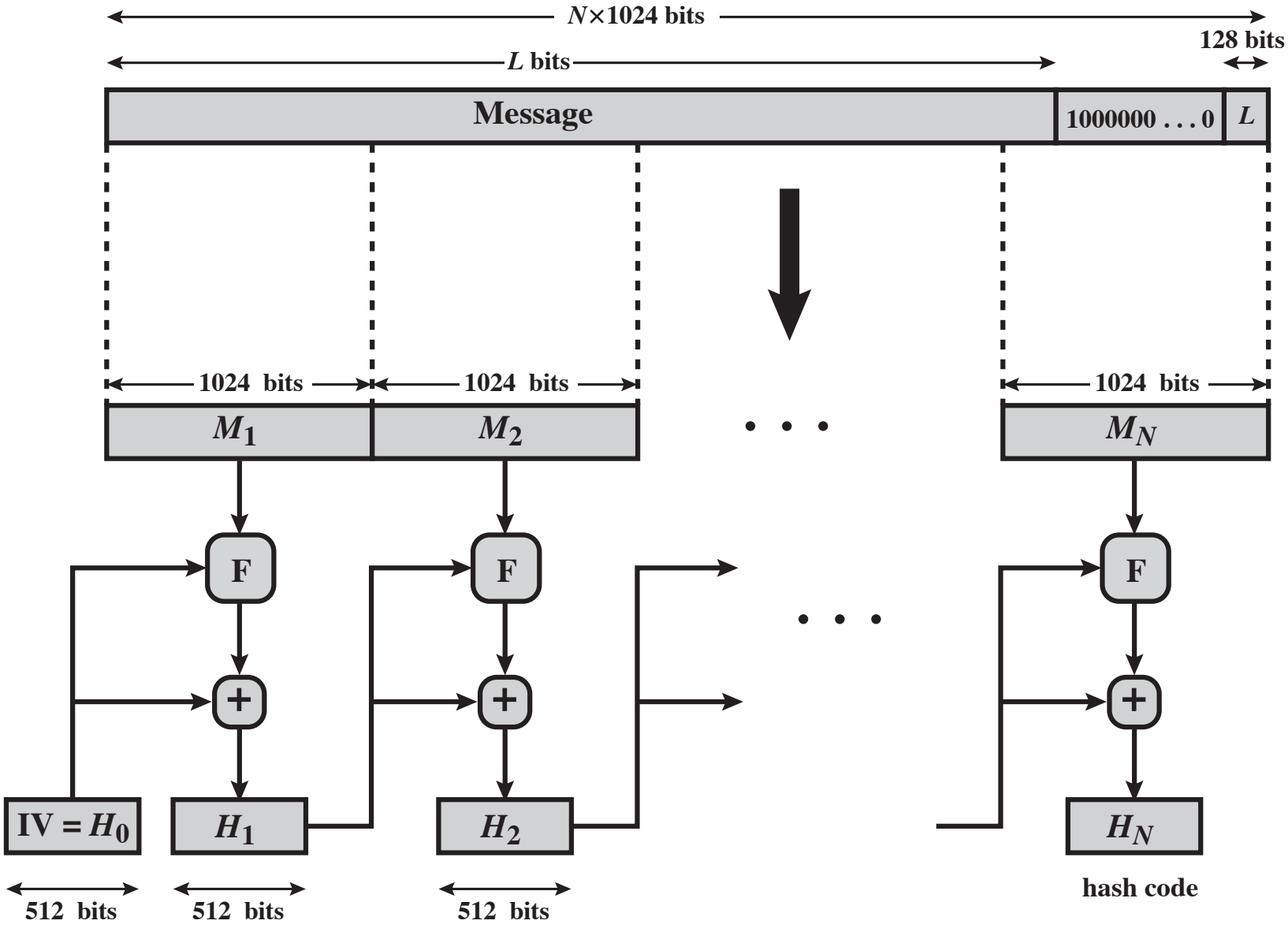
Parámetros de SHA

SHA-2 ⟷

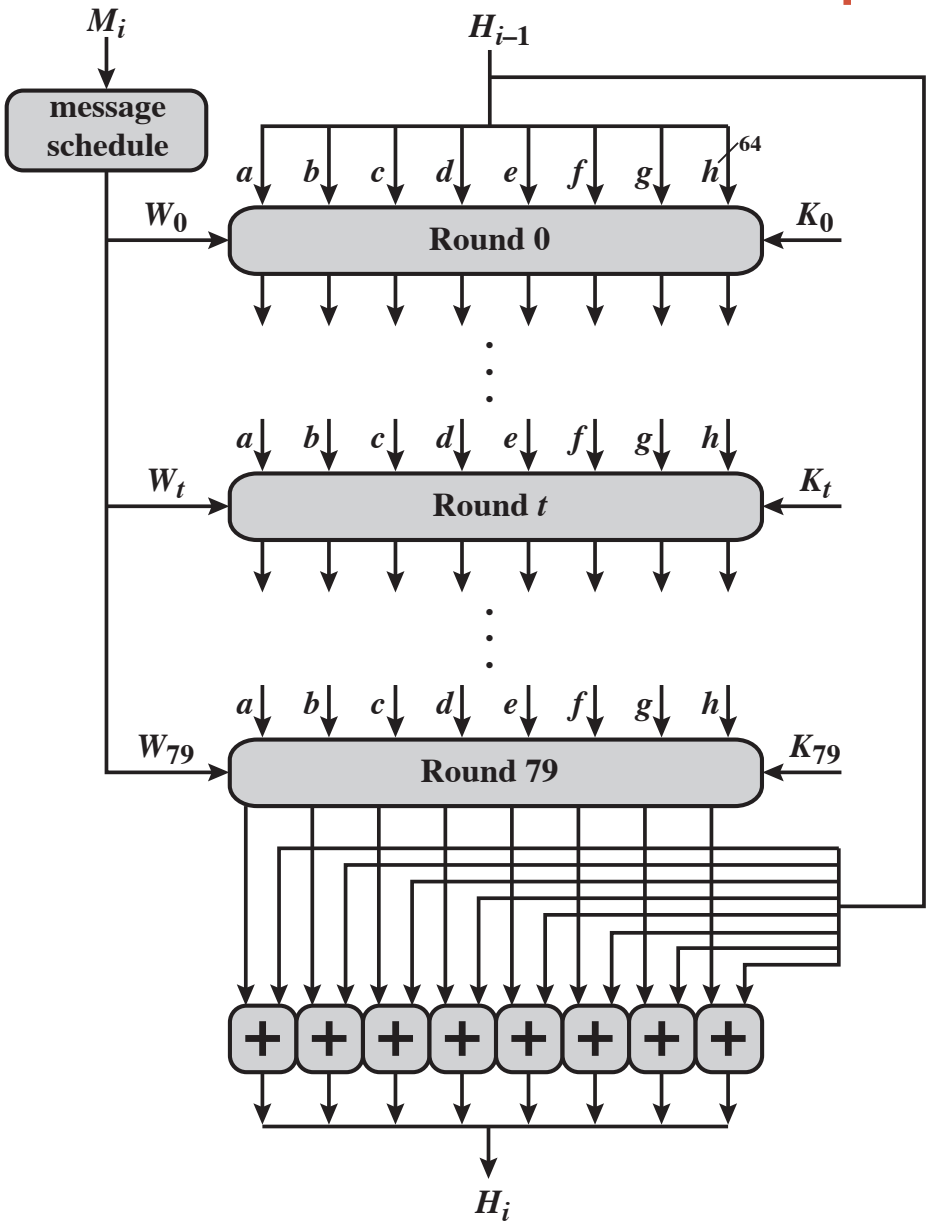
	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Message Digest Size	160	224	256	384	512
Message Size	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block Size	512	512	512	1024	1024
Word Size	32	32	32	64	64
Number of Steps	80	64	64	80	80

Nota: Todos los tamaños expresados en bits

SHA-512: Procesamiento de un mensaje

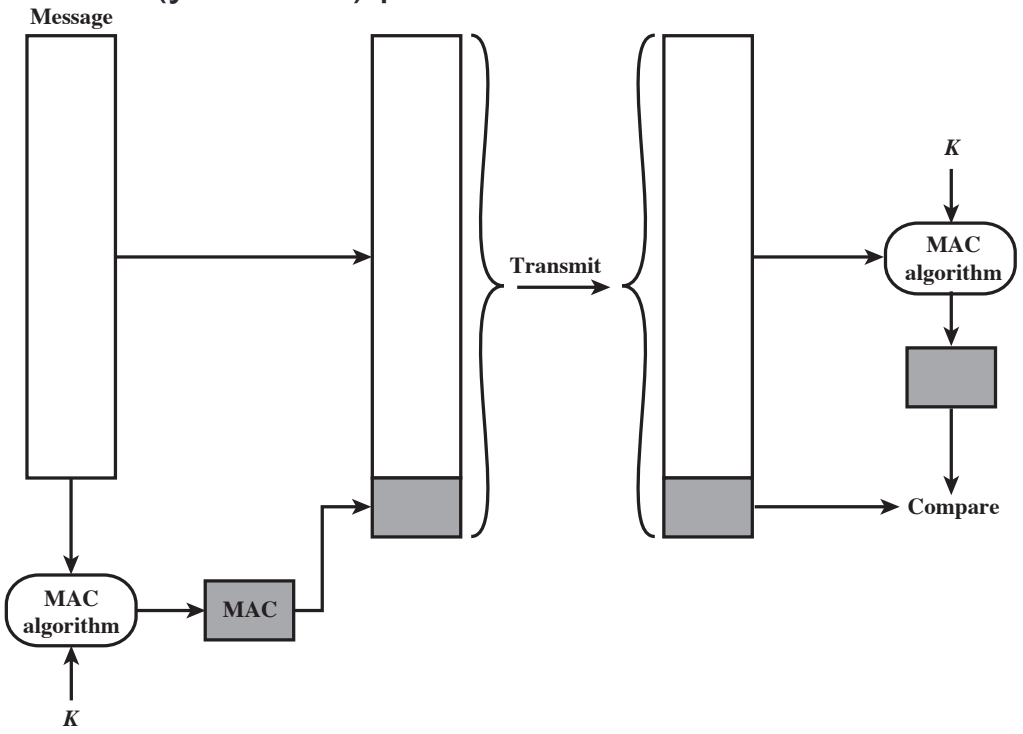


SHA-512: Procesamiento un bloque (1024 bits)



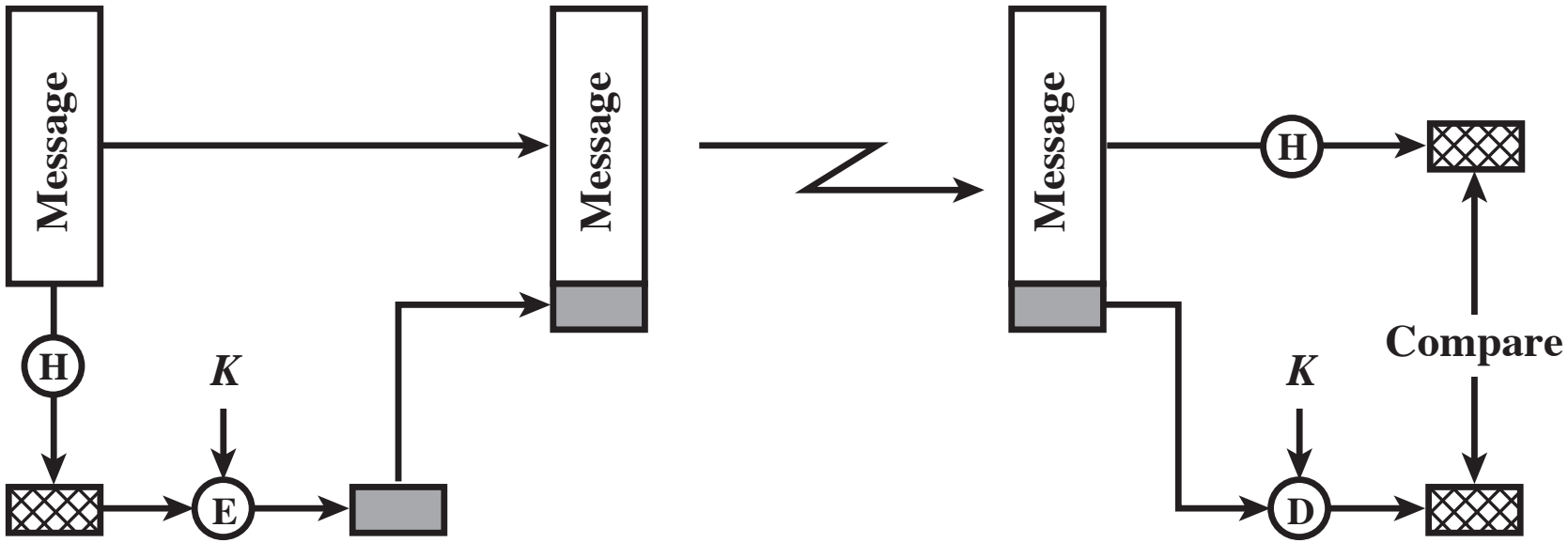
Autenticación de mensajes

- El cifrado simétrico por sí solo no proporciona integridad de datos ni autenticación de su origen
  - Incluso cifrados, los mensajes puedes ser modificados de diversas formas
    - Los paquetes pueden ser alterados (*bit flipping*), reordenados o retardados
- Se debe adjuntar una etiqueta o código de autenticación de mensajes (*Message Authentication Code*, MAC) a cada mensaje o paquete
  - El mensaje mismo (y el MAC) puede ser cifrado o no



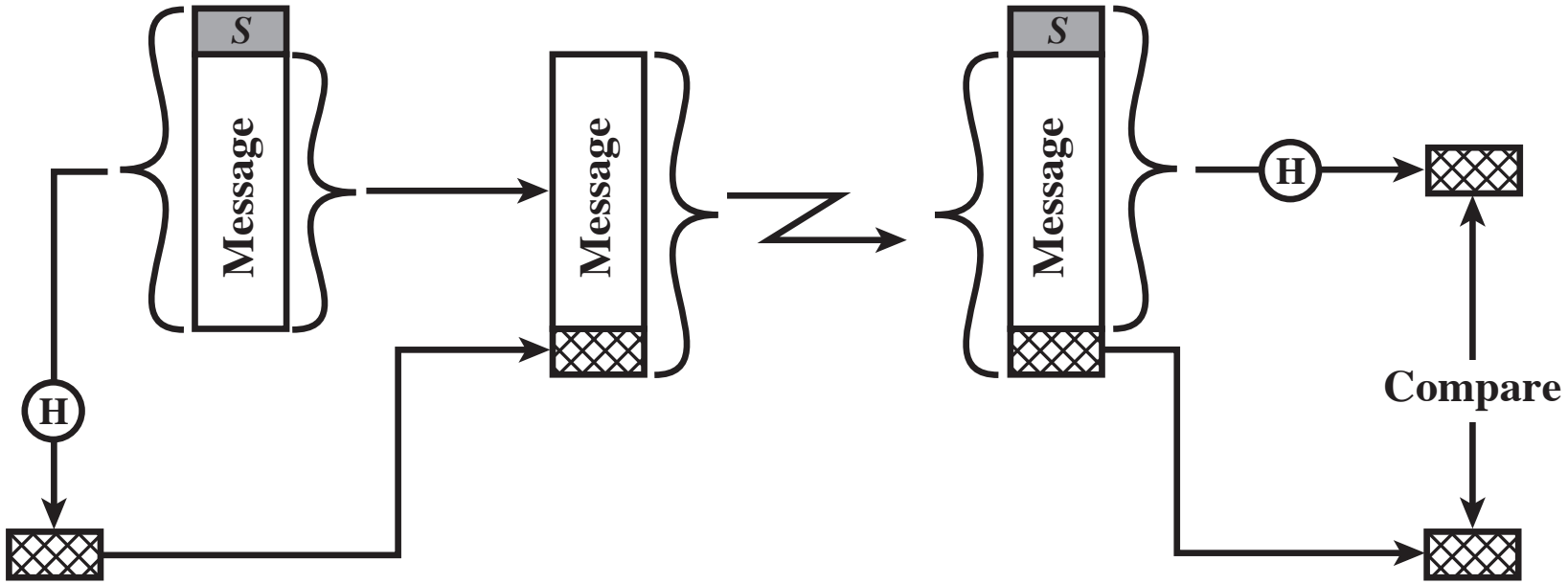
Autenticación de mensajes con una función resumen y cifrado simétrico

- Similar a la firma digital con criptografía de clave pública



Autenticación de mensajes con una función resumen sin cifrado

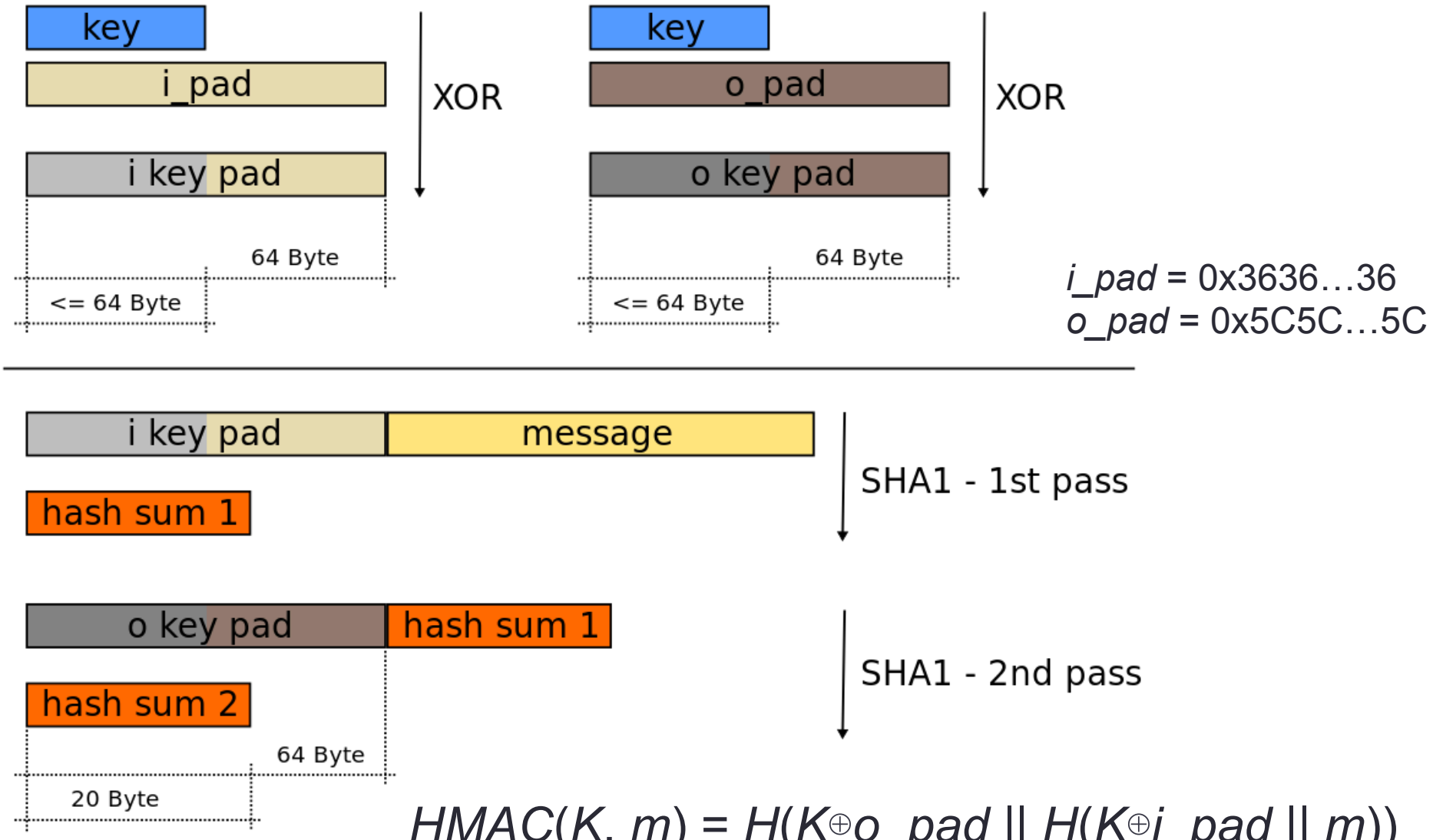
- Similar al *hashing* de contraseñas con sal



Keyed-Hash Message Authentication Code (HMAC)

- Interesa tener un MAC basado en una función resumen
  - Las funciones resumen generalmente se ejecutan más rápido en *software* que los algoritmos de cifrado simétrico
  - El código está ampliamente disponible
- SHA no se diseñó para usarse como MAC
  - Adaptaciones triviales pueden presentar vulnerabilidades fácilmente explotables (*length-extension attack*)
- De entre todas las propuestas, HMAC fue la que más respaldo recibió
  - Estándar publicado por NIST en 2002 [FIPS 198], como una generalización de especificaciones previas [RFC 2104]
  - Elegido como MAC de implementación obligatoria para IPsec
  - Usado en otros protocolos de Internet, como TLS

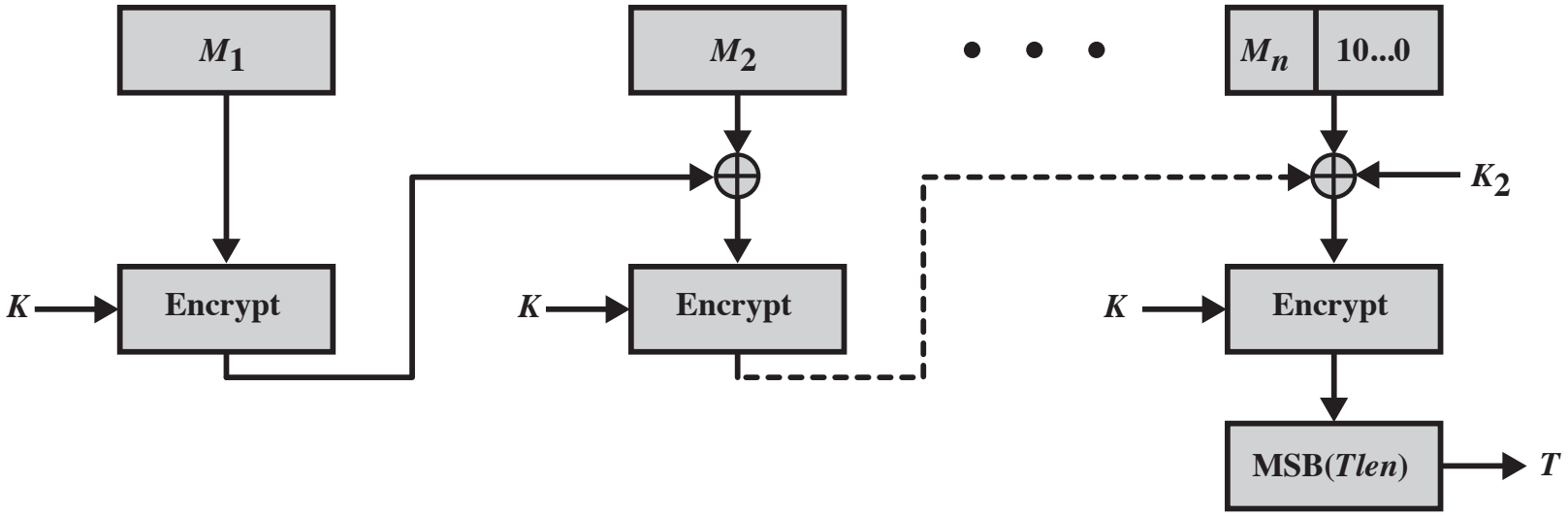
Estructura de HMAC con SHA-1



$$HMAC(K, m) = H(K \oplus o\_pad \,||\, H(K \oplus i\_pad \,||\, m))$$

Cipher-Based Message Authentication Code (CMAC)

- Recomendación publicada por NIST [SP-800-38B]
  - También publicada por IETF [RFC 4493]
- Ligera modificación de CBC-MAC [FIPS 113]
  - CBC-MAC usa cifrado de bloque en modo CBC y se queda solo con el último bloque
  - CMAC corrige deficiencias para mensajes de tamaño variable



Counter with Cipher Block Chaining-Message Authentication Code (CCM)

- Recomendación publicada por NIST [SP 800-38C]
  - También publicado por IETF [RFC 3610]
  - Usado en IEEE 802.11i (WPA2), IPsec y TLS 1.2
- Proporciona cifrado autenticado (*authenticated encryption)*
  - Término usado para describir sistemas de cifrado que protegen simultáneamente la confidencialidad, la integridad y la autenticidad de origen de las comunicaciones
- Usa una sola clave tanto para el algoritmo de cifrado como para el de MAC, combinando:
  - AES en modo CTR para cifrado
  - CMAC para autenticación de mensajes

Resumen

- Funciones resumen
  - Requisitos
  - Seguridad
  - SHA-1 y SHA-2
- Autenticación de mensajes
  - HMAC
  - CMAC
  - CCM

2.4 CRIPTOGRAFÍA DE CLAVE PÚBLICA

2. Comunicaciones seguras

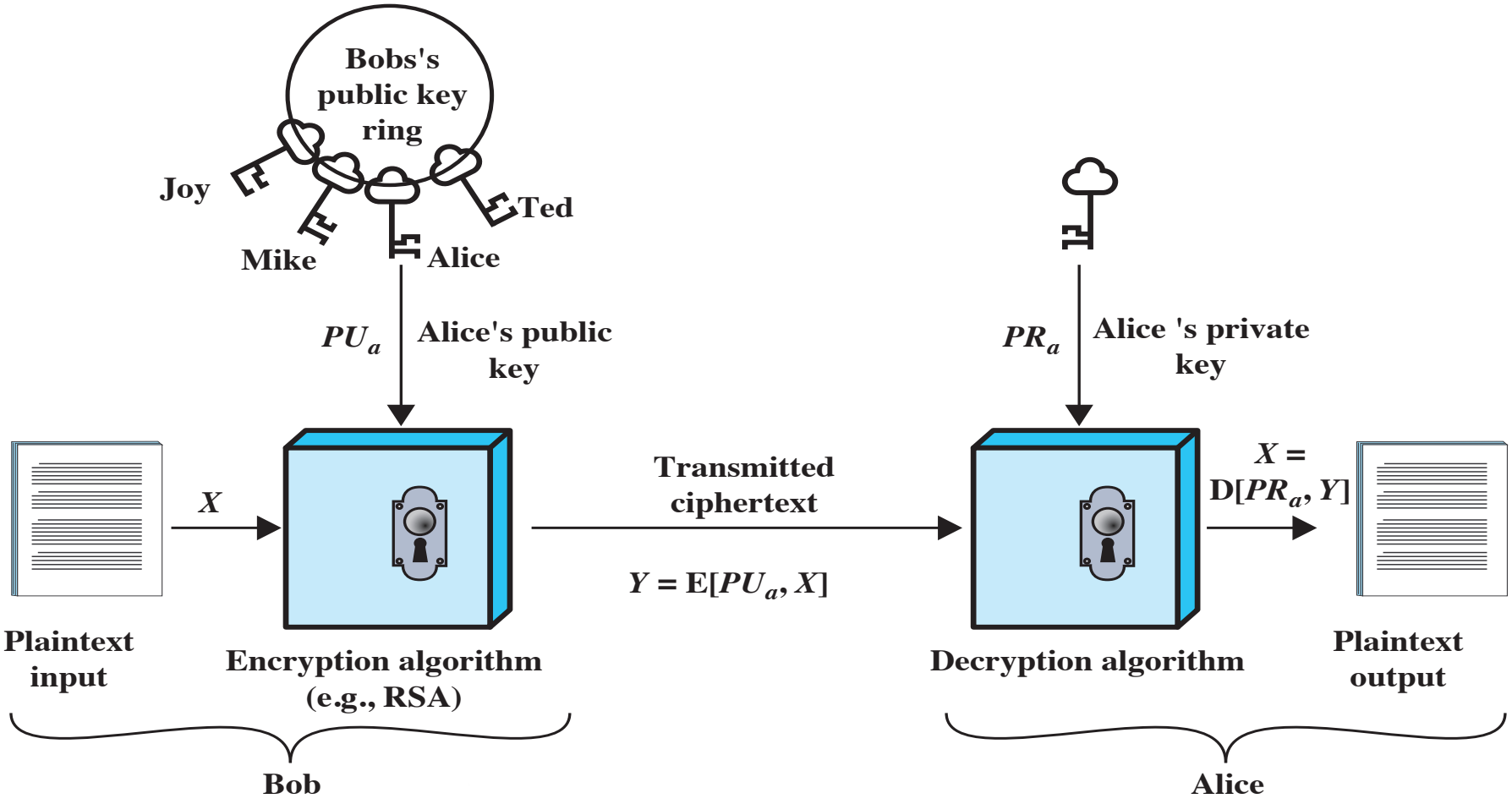
Criptografía de clave pública

- Fue propuesta, al menos públicamente, por primera vez por Diffie y Hellman
  - *"New Directions in Cryptography"*, IEEE Trans. Information Theory, 1976
- Basada en funciones matemáticas en lugar de usar operaciones sobre patrones de bits
- Es asimétrica, ya que usa dos claves distintas
- Algunos conceptos erróneos:
  - Es más segura frente a criptoanálisis que el cifrado convencional
  - Es una técnica que propósito general que deja el cifrado convencional obsoleto
  - La distribución de claves es trivial

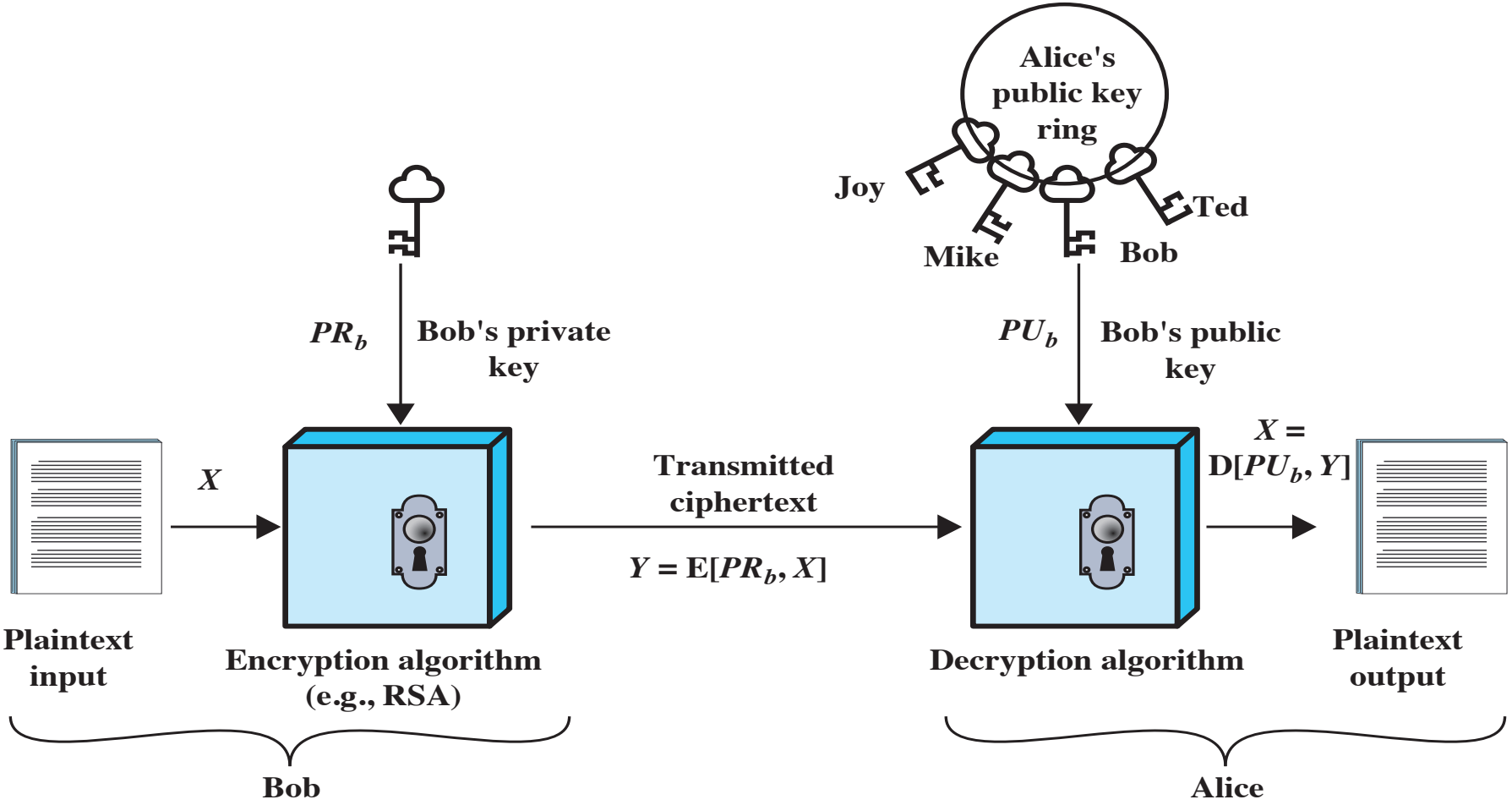
Aplicaciones de criptosistemas de clave pública

- Una clave se mantiene privada y la otra se hace pública
- Dependiendo de la aplicación, el emisor usa su clave privada, la clave pública del receptor o ambas para realizar algún tipo de función criptográfica
- Aplicaciones
  - Cifrado/descifrado: El emisor cifra un mensaje con la clave pública del receptor
  - Firma digital: El emisor cifra un resumen de un mensaje con su clave privada
  - Intercambio de clave: Dos partes cooperan para establecer una clave de sesión

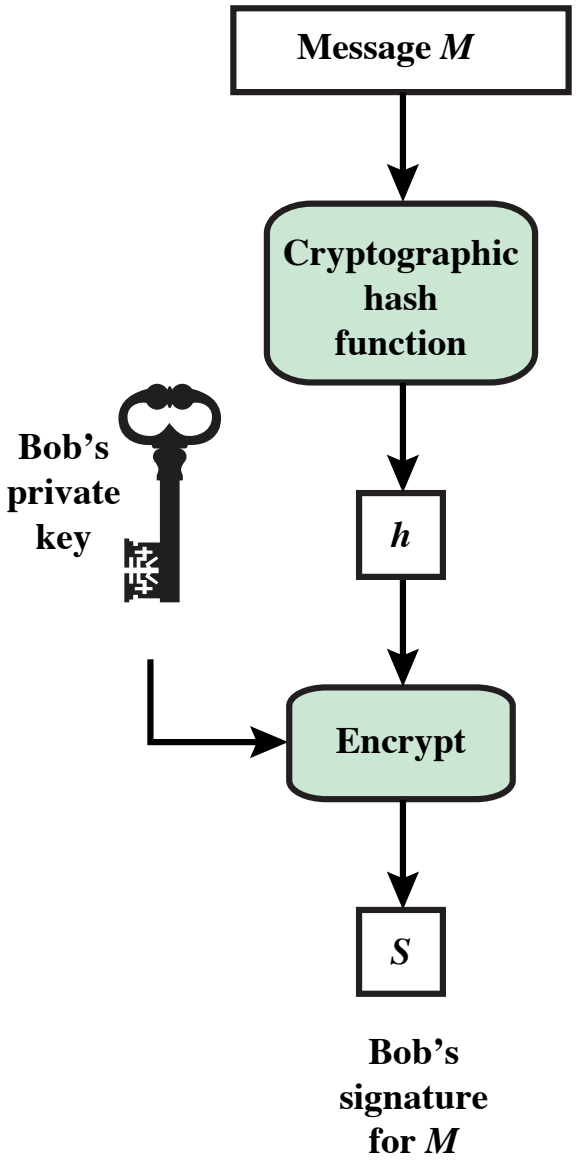
Cifrado con la clave pública del receptor

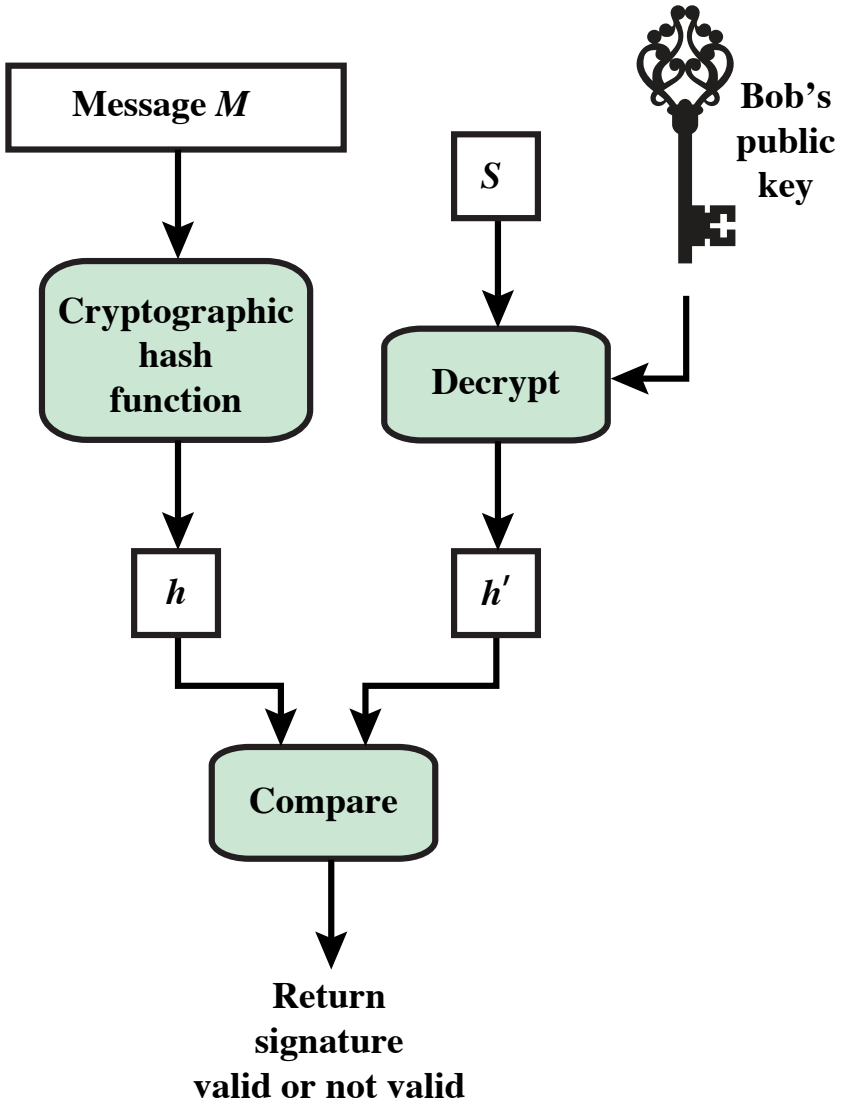


Cifrado con la clave privada del emisor



Firma digital





Intercambio de clave

- Los algoritmos de clave pública no son apropiados para cifrar grandes ficheros o flujos de datos:
  - Tratan los datos de entrada como un número (cuyo tamaño depende del de la clave)
  - Son mucho más lentos que los algoritmos de clave secreta
  - Los patrones (en texto claro, protocolos…) ayudan al criptoanálisis
- Por tanto, el cifrado de clave pública solo debe usarse para intercambiar una clave secreta aleatoria de forma segura
- Por ejemplo, para cifrar un fichero grande, el emisor:
  1. Genera una clave de sesión aleatoria
  2. Cifra la clave de sesión usando la clave pública del receptor
  3. Cifra los datos usando la clave de sesión
  4. Empaqueta y envía la clave de sesión cifrado con los datos cifrados
- El receptor descifra la clave de sesión con su clave privada y después descifra los datos con la clave de sesión
- En comunicaciones seguras, las claves se intercambian de forma similar o mediante un proceso de acuerdo o derivación

Algoritmos de clave pública

Algoritmo	Cifrado	Firma digital	Intercambio de clave
RSA	Sí	Sí	Sí
Diffie-Hellman	No	No	Sí
DSA	No	Sí	No
Curva elíptica	Sí	Sí	Sí

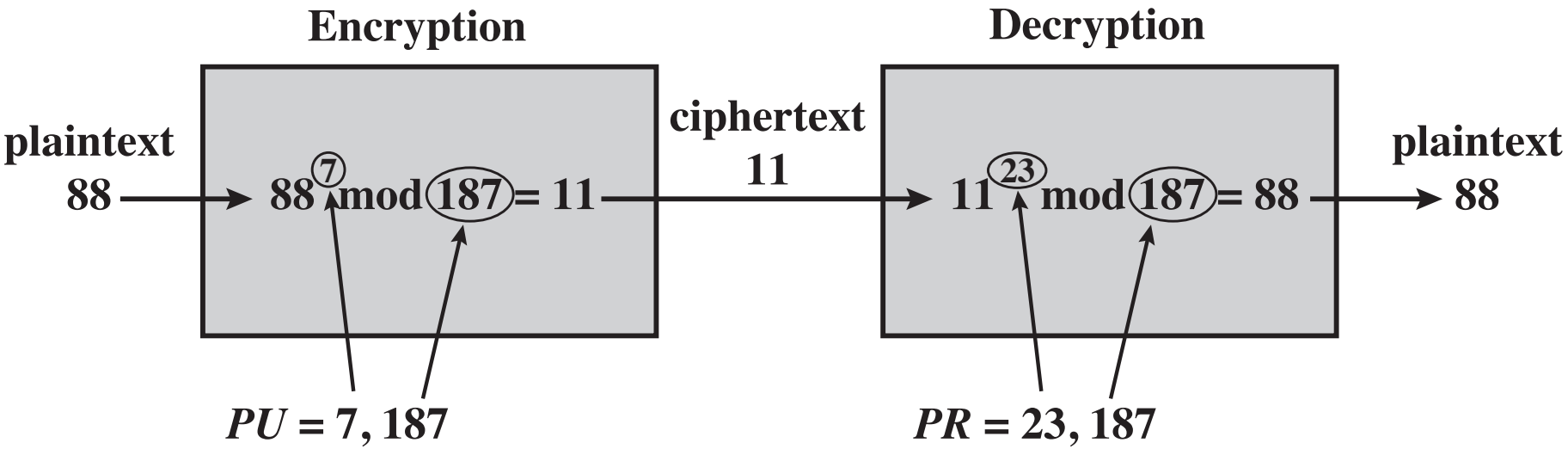
RSA

- Uno de los primeros esquemas de clave pública desarrollado en 1977 por Ron Rivest, Adi Shamir y Len Adleman en el MIT
  - *"A Method for Obtaining Digital Signatures and Public-Key Cryptosystems"*, Comm. ACM, 1998
  - Uno de los esquemas más aceptados e implementados
- Cifrado de bloque en el que el texto plano y cifrado son enteros entre 0 y n - 1 para un n
- La seguridad se basa en dos problemas matemáticos, para los que no existe un algoritmo eficiente (*hard*):
  - Factorización de enteros (descomposición en factores primos)
  - El problema RSA (revertir la operación de cifrado RSA)
- En general, se supone que RSA es seguro si n es suficientemente grande

El algoritmo RSA

- Generación de claves
  - Seleccionar dos números primos distintos p y q
  - Calcular $n = p \cdot q$ y $\phi(n) = (p - 1) \cdot (q - 1)$
  - Seleccionar un entero e tal que $1 < e < \phi(n)$ y $\text{mcd}(\phi(n), e) = 1$
    - Se suele tomar e = 65,537 (0x10001)
  - Calcular d tal que $d \cdot e \bmod \phi(n) = 1$
  - La clave pública es $\{e, n\}$
  - La clave privada es $\{d, n\}$
- Cifrado
  - Dado un texto claro $M < n$, el texto cifrado es $C = M^e \bmod n$
- Descifrado
  - Dado un texto cifrado C, el texto claro es $M = C^d \bmod n$

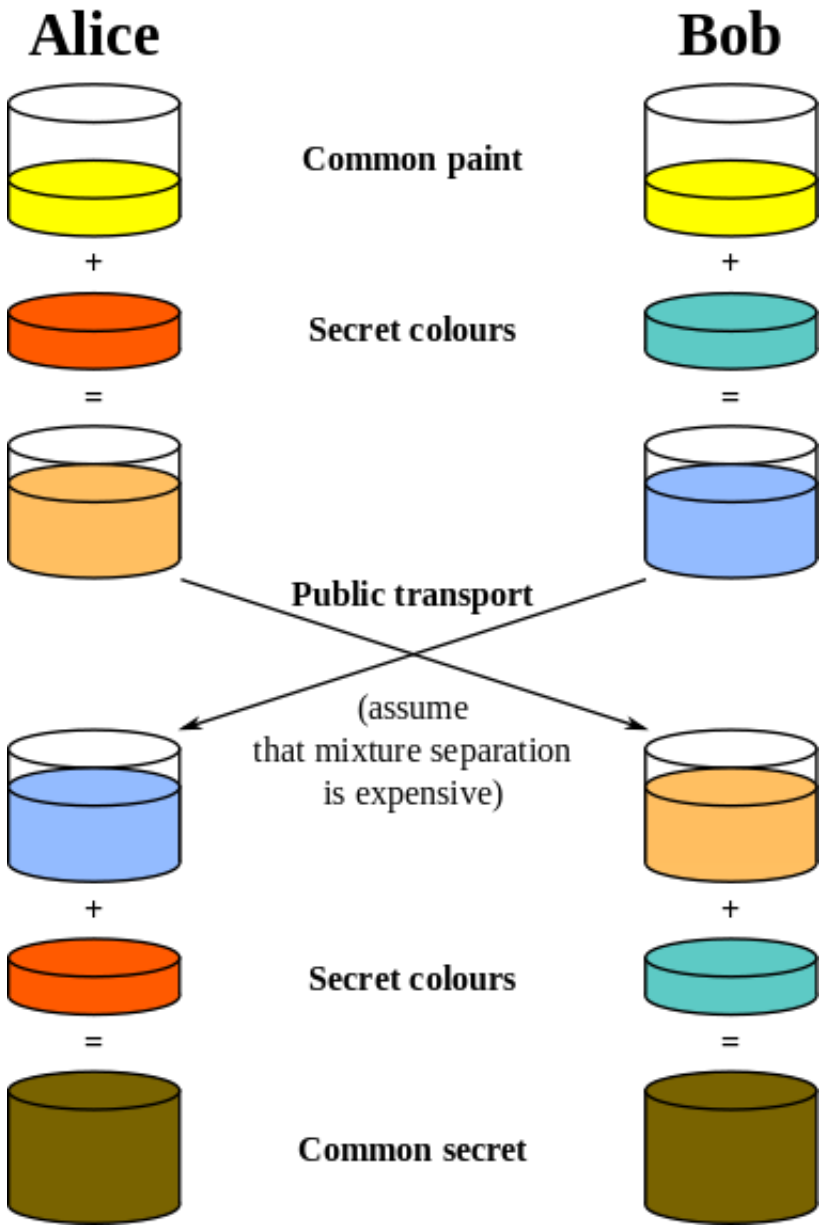
Ejemplo


Encryption
Decryption
plaintext
88
88 7 mod 187 = 11
ciphertext
11
11 23 mod 187 = 88
plaintext
88
PU = 7, 187
PR = 23, 187

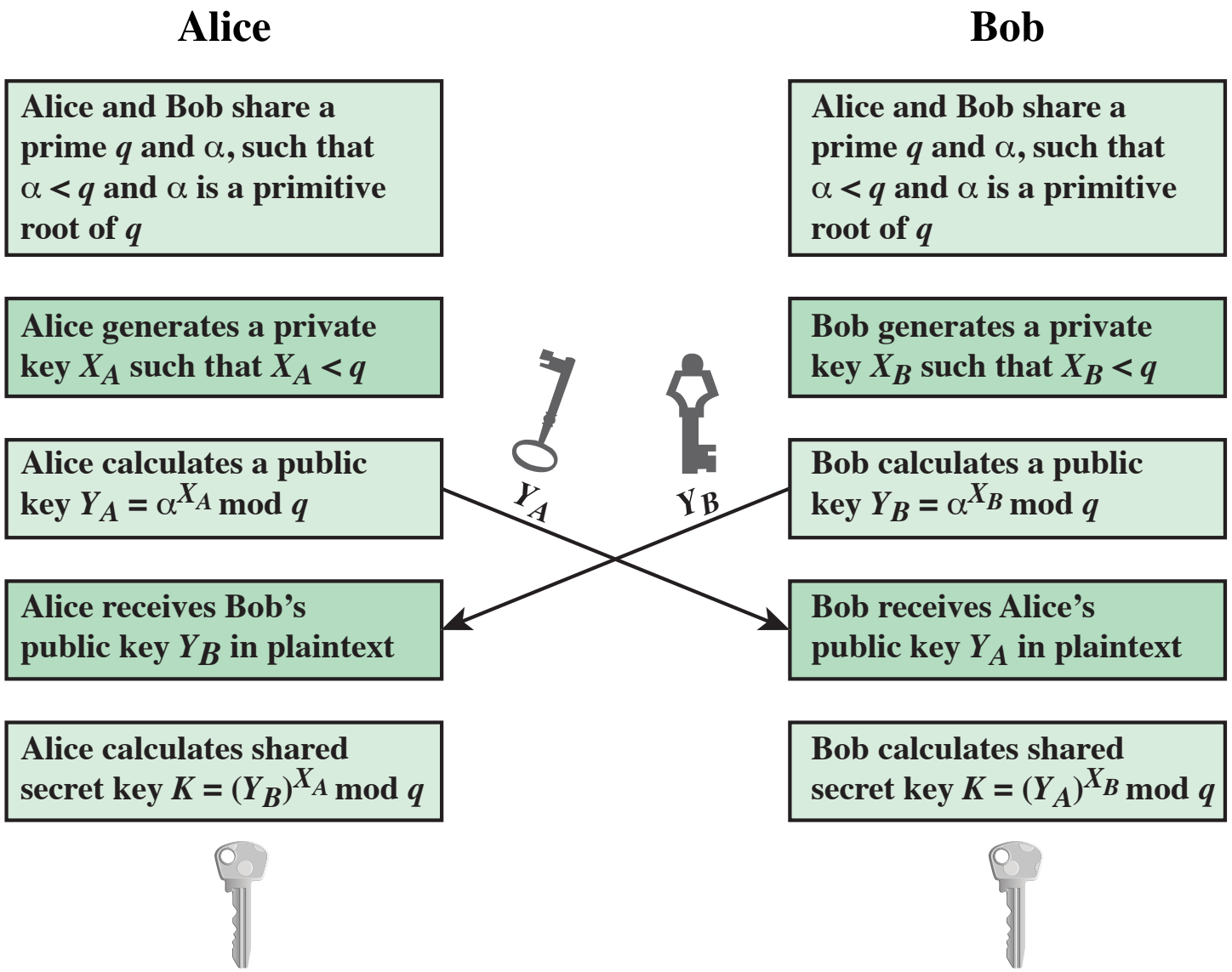
Diffie-Hellman

- Primer algoritmo de clave pública publicado
  - *"New Directions in Cryptography"*, IEEE Trans. Information Theory, 1976
- Su propósito es permitir a dos partes derivar una clave secreta de forma segura que pueda ser después usada para cifrar mensajes
  - El algoritmo se limita al intercambio de claves
- Muchos productos usan esta técnica
- Su seguridad depende de la dificultad de calcular logaritmos discretos

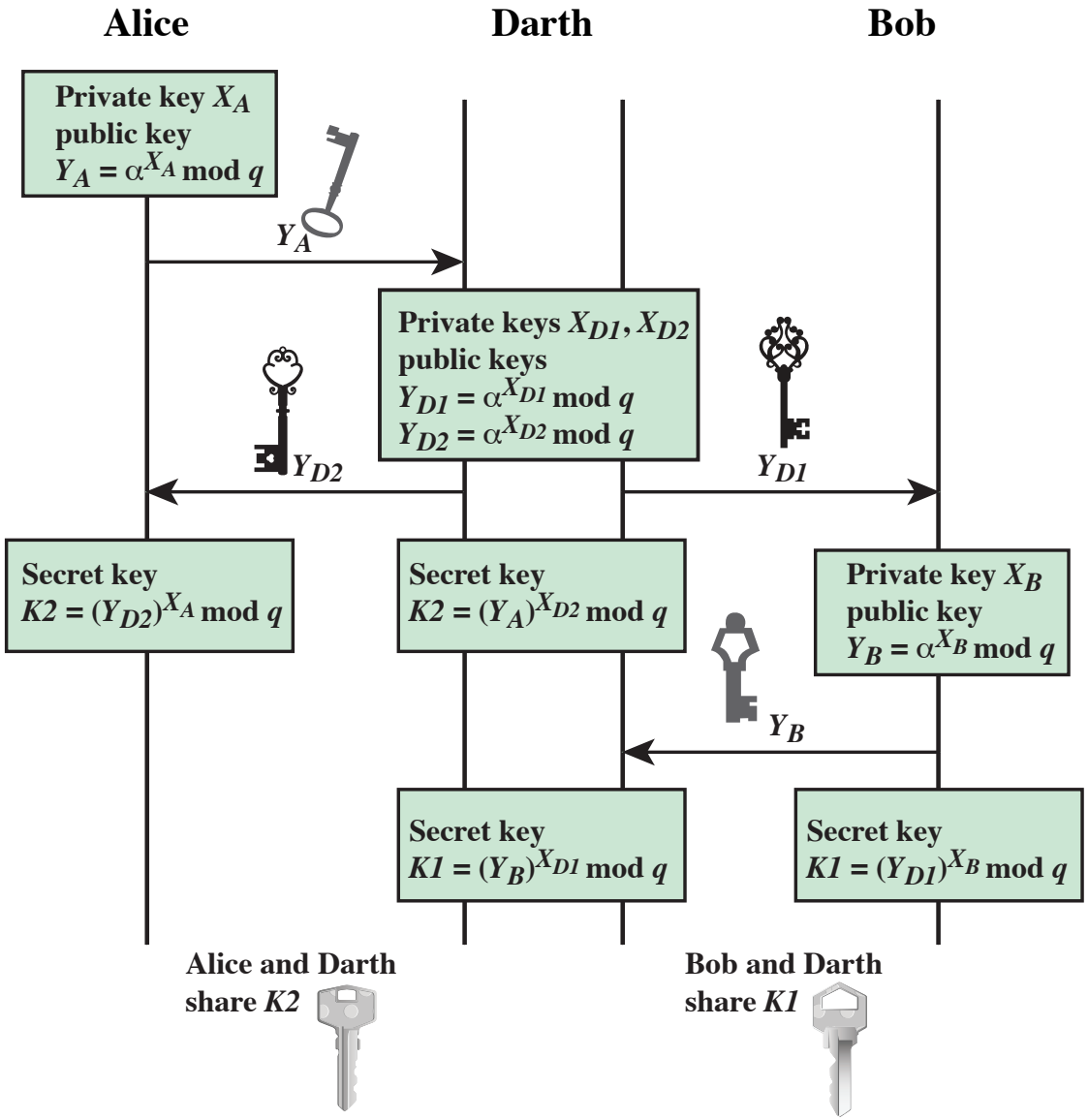
Idea general



Algoritmo de Diffie-Hellman



Ataque de intermediario



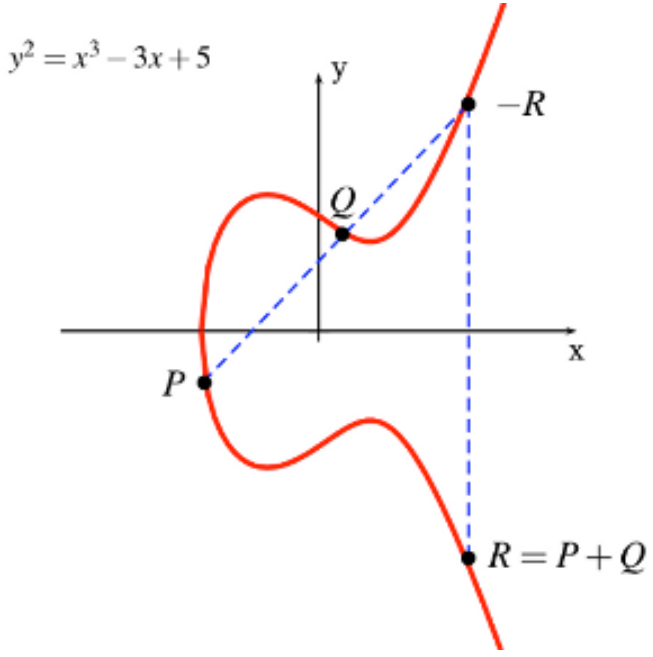
Digital Signature Algorithm (DSA)

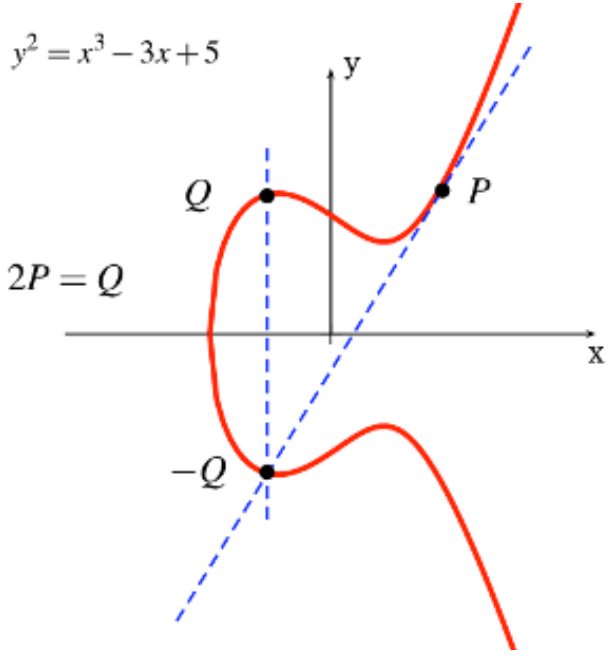
- *Digital Signature Standard* (DSS) propuesto por el NIST en 1991 y publicado en 1994 [FIPS PUB 186]
  - Revisado en 1998, 2000, 2009 y 2013 [FIPS PUB 186-4]
- Variación del esquema de firma de Elgamal
  - Propuesto en 1984 por el criptógrafo egipcio Taher Elgamal
  - Basado en DH
- Hace uso de SHA-1
  - El DSS actual ha aprobado el uso de SHA-2
- Solo proporciona la función de firma digital
  - A diferenca de RSA, no proporciona cifrado ni intercambio de claves

Criptografía de curvas elípticas

- Se basa en una curva elíptica sobre un cuerpo finito y la operación de multiplicación escalar
  - Sustituye a los logaritmos discretos en DH y DSA
  - Su principal ventaja es que ofrece la misma seguridad que RSA con menor tamaño de clave, lo que reduce el tiempo de procesamiento

$$y^2 = x^3 + ax + b$$





Resumen

- Criptografía de clave pública
  - Cifrado/descifrado
  - Firma digital
  - Intercambio de claves
- Algoritmos de clave pública
  - RSA
  - DH
  - DSA
  - Curvas elípticas

2.5 CERTIFICADOS DIGITALES Y MODELOS DE CONFIANZA

2. Comunicaciones seguras

Certificado digital

- Documento electrónico que usa la firma digital de una tercera parte de confianza para vincular una clave pública con una identidad (nombre de una persona u organización, dirección, e-mail…)
- Usada para verificar que una clave pública pertenece a un individuo, organización o servicio
  - Proporciona no repudio

Modelos de confianza

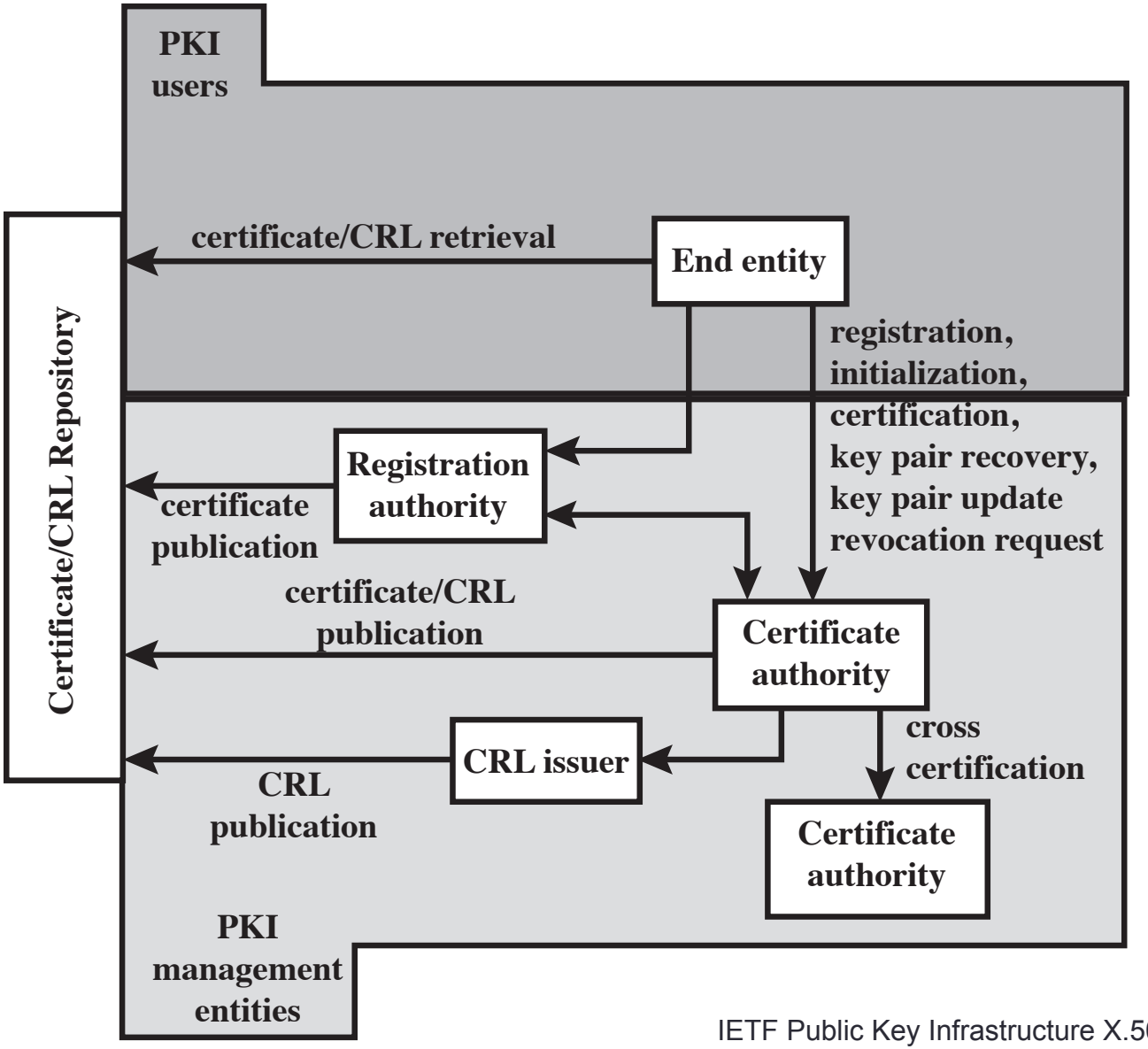
- **Confianza directa**: se confía en una clave porque se sabe de dónde procede
  - Los otros modelos usan este de alguna manera
  - Usada en SPKI
- **Confianza jerárquica o cadena de confianza**: hay varios certificados raíz (*trust anchors*) firman certificados finales o certificados intermedios que pueden a su vez firmar otros
  - Los certificados se verifican siguiendo la cadena de firmas hacia atrás, hasta que se encuentra un certificado raíz de confianza (directa)
  - Usada en PKI X.509
- **Red de confianza (*Web of trust*)**: engloba ambos modelos
  - Se puede tener confianza directa en un certificado o porque está firmado por otro de confianza (*trusted introducer*)
  - Usada en PGP

X.509

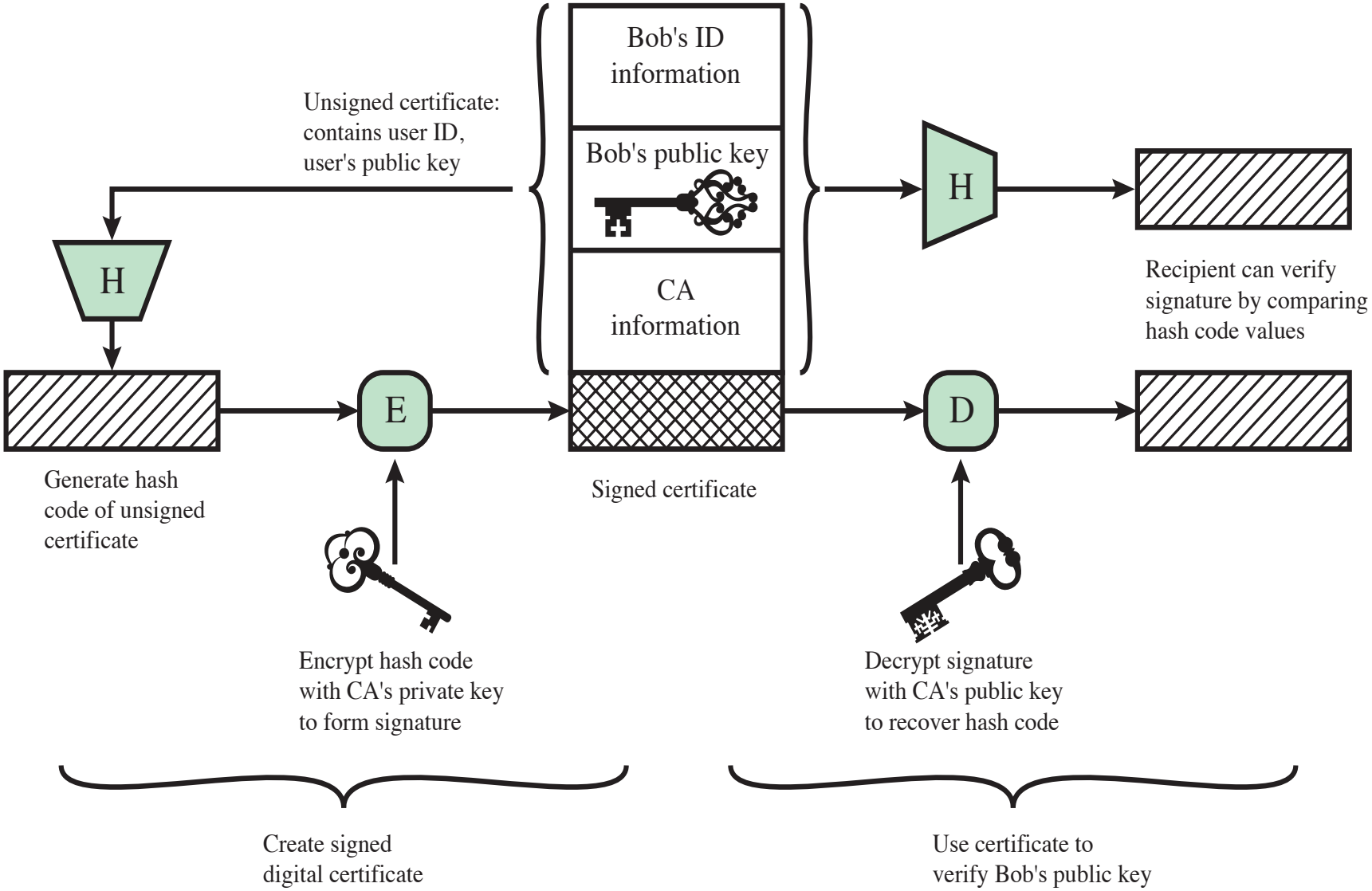
- Recomendación de ITU-T para una infraestructura de clave pública (*Public Key Infrastructure*, PKI), aprobada en 1988
  - Parte de la serie X.500 de recomendaciones para definir un servicio de directorio
  - Basada en el uso de criptografía de clave pública y firmas digitales
- Definición de PKI [RFC 4949]:

  "the set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates based on asymmetric cryptography"
- X.509 especifica, entre otras cosas:
  - Formatos estándar para certificados y listas de revocación
  - Algoritmo de validación de rutas de certificación

Arquitectura PKI X.509 (PKIX)



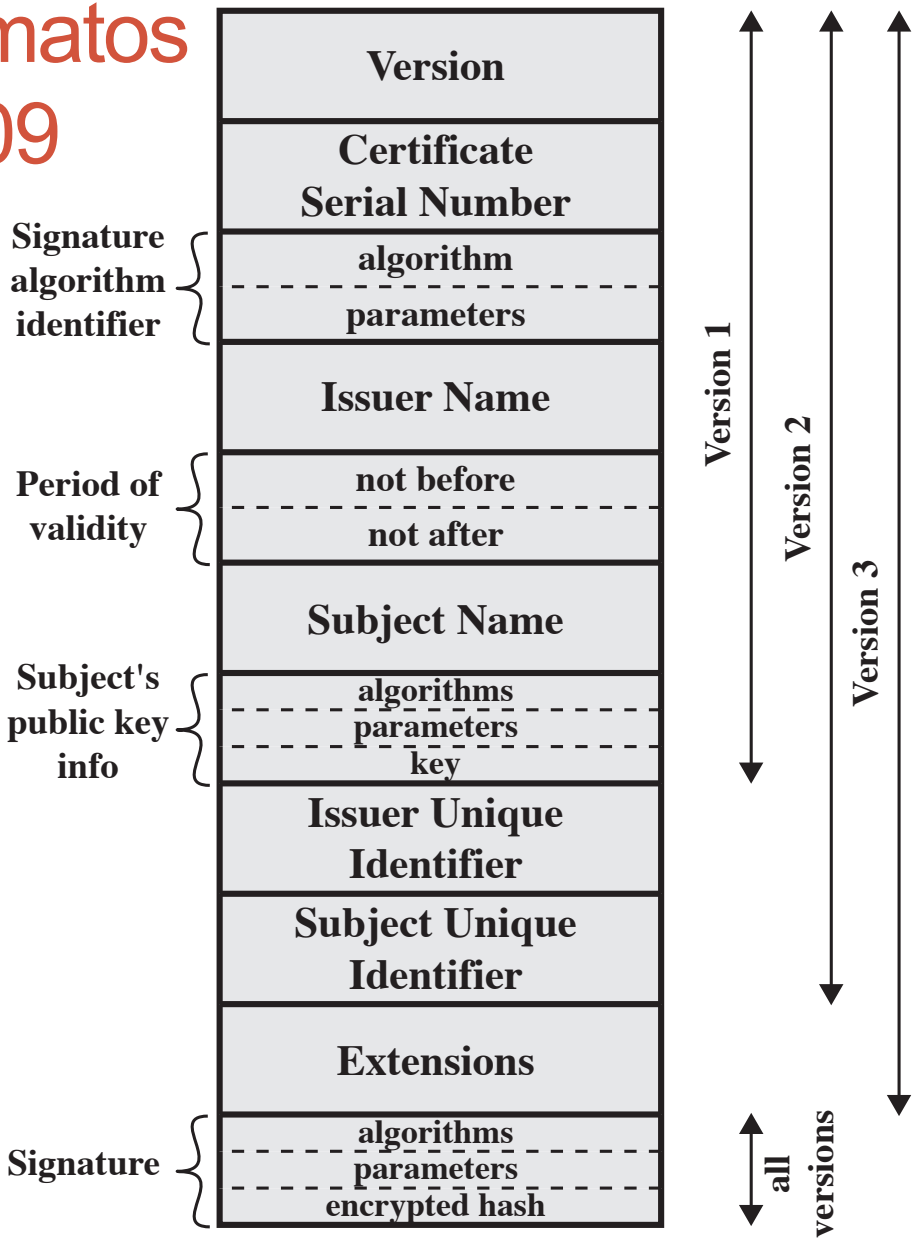
Creación y verificación de certificados



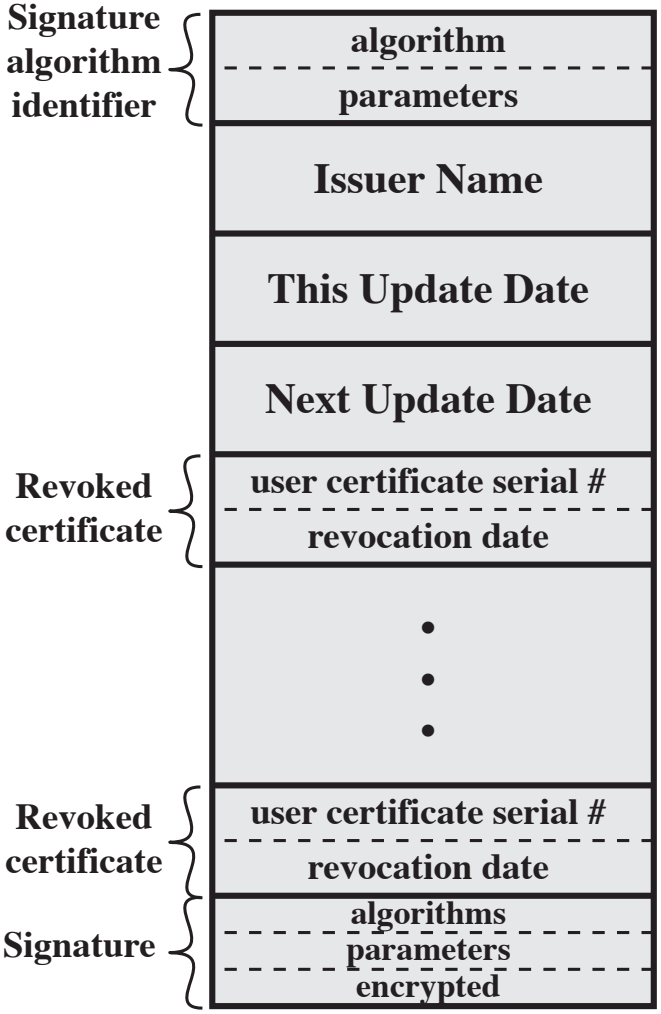
Revocación de certificados

- Cada certificado incluye un periodo de validez
  - Se emite un nuevo certificado justo antes de que expire el antiguo
- Puede ser necesario revocar un certificado antes de que expire debido a las siguientes razones:
  - Se cree que la clave privada ha sido comprometida
  - El usuario ya no está certificado por esta CA
    - El nombre de sujeto ha cambiado, el certificado ha sido sustituido, o el certificado no se emitió de conformidad con las políticas de la CA
  - Se cree que el certificado de la CA ha sido comprometido
- Cada CA mantiene una lista (*Certificate Revocation List*, CRL) con todos los certificados revocados y aún no expirados emitidos por esa CA
  - Cuando se recibe un certificado, se debe comprobar que no aparezca en esa lista

Formatos X.509



(a) X.509 Certificate



(b) Certificate Revocation List

Pretty Good Privacy (PGP)

- Creado en 1991 por Phil Zimmermann
  - Renombrado después como GnuPG o GPG (*GNU Privacy Guard*)
- En 1997 se forma el grupo de trabajo sobre OpenPGP en el IETF
- La propuesta de estándar OpenPGP [RFC 4880], derivada de PGP, es un formato no propietario para cifrar correo electrónico usando criptografía de clave pública
  - Define formatos estándar para los mensajes cifrados, firmas y certificados para intercambiar claves públicas

Red de confianza (*Web of Trust*) PGP

- Planteado por primera vez en 1992 en la versión 2.0 del manual de PGP:

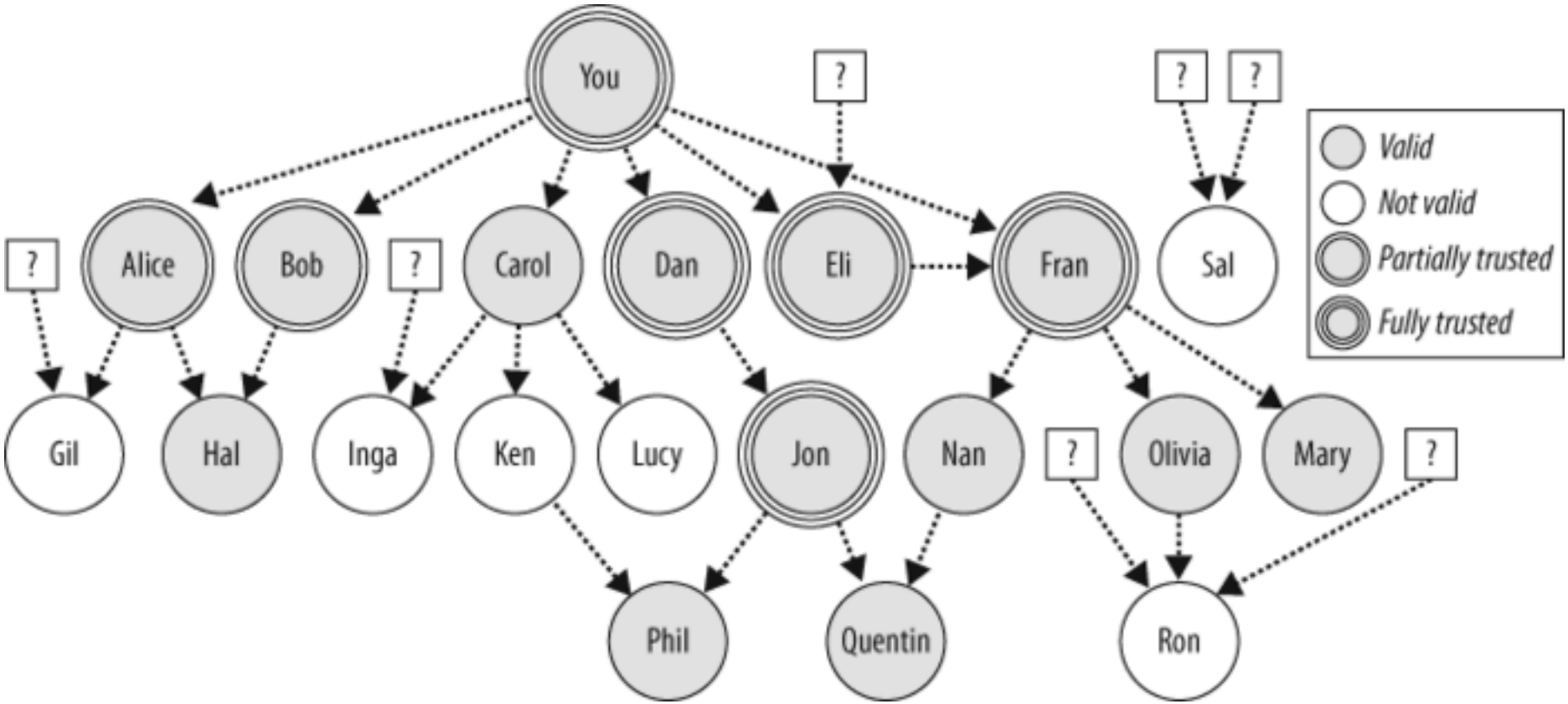
"As time goes on, you will accumulate keys from other people that you may want to designate as trusted introducers. Everyone else will each choose their own trusted introducers. And everyone will gradually accumulate and distribute with their key a collection of certifying signatures from other people, with the expectation that anyone receiving it will trust at least one or two of the signatures. This will cause the emergence of a decentralized fault-tolerant web of confidence for all public keys"

- Modelo de confianza descentralizado
  - Hay muchas redes de confianza independientes, y un usuario puede ser parte de varias de ellas y servir de enlace entre ellas
- Los certificados son parecidos a los de X.509
  - Incluyen una auto-firma y pueden tener varias firmas de otros individuos

Cálculo de la validez de las claves PGP

- La **validez** de la clave se refiere a la confianza en la asociación clave-identidad
- La **confianza** en la clave se refiere a la confianza en cómo su propietario firma otras claves
  - Pueden ser *desconocida*, *ninguna*, *marginal*, *completa* o *absoluta*
- Una clave es (*completamente*) *válida* si:
  1. Está firmada por suficientes claves *válidas*, que significa que
     - El usuario la ha firmado personalmente (*confianza absoluta*)
     - O ha sido firmada por una clave con *confianza completa*
     - O ha sido firmada por tres claves con *confianza marginal*
  2. La ruta de firmas es de cinco pasos como máximo
- Los números anteriores pueden ser ajustados
  - Son los valores por defecto en GnuPG que, además, tiene varios modelos de confianza: *pgp*, *classic*, *direct* y *always*

Ejemplo de red de confianza



Solamente se necesitan dos claves con confianza marginal

Resumen

- Certificados digitales
- Modelos de confianza
  - Confianza directa
  - Confianza jerárquica
  - Red de confianza
- X.509
  - Arquitectura PKIX
  - Creación y verificación de certificados
  - Revocación
  - Formatos
- PGP
  - PGP, GPG y OpenPGP
  - Red de confianza
  - Cálculo de la validez de las claves

2.6 APLICACIONES PARA COMUNICACIONES SEGURAS

2. Comunicaciones seguras

OpenSSL

- OpenSSL es una librería criptográfica que implementa los protocolos *Secure Sockets Layer* (SSL v2/v3) y *Transport Layer Security* (TLS v1) y estándares criptográficos relacionados
- `openssl` es un comando para usar las funciones criptográficas de la librería OpenSSL
  - Creación y gestión de claves públicas, privadas y parámetros
  - Operaciones con criptografía de clave pública
  - Creación de certificados, CSRs y CRLs X.509
  - Cálculo de resúmenes de mensajes
  - Cifrado y descifrado con criptografía de clave secreta
  - Pruebas de clientes y servidores SSL/TLS
  - Gestión de correo firmado o cifrado con S/MIME
  - Petición, generación y verificación de marcas de tiempo

Comandos openssl

- `enc` Cifrado y descifrado con clave secreta
- `dgst` Cálculo de resúmenes de mensajes
- `genpkey` Generación de pares de claves y parámetros
- `pkeyparam` Gestión de parámetros de claves
- `pkey` Gestión de claves públicas y privadas
- `pkeyutl` Operaciones con criptografía de clave pública
- `ca` Gestión de autoridades de certificación (CA)
- `req` Gestión de CSRs X.509
- `x509` Gestión de certificados X.509
- `verify` Verificación de certificados X.509
- `crl` Gestión de CRLs
- `s_client` Cliente SSL/TLS genérico
- `s_server` Servidor SSL/TLS genérico
- `smime` Procesado de correo S/MIME

GnuPG (*GNU Privacy Guard*)

- Implementación completa y libre del estándar OpenPGP [RFC 4880] del proyecto GNU
  - Antes conocido como PGP (*Pretty Good Privacy*)
- Permite **cifrar** y **firmar** datos, proporciona un sistema de **gestión de claves** versátil y módulos de acceso a todo tipo de **directorios** de claves públicas
- Proporciona un conjunto de herramientas de línea de comandos que puede ser integrada en otras herramientas
  - La herramienta `gpg2` implementa la parte OpenPGP, proporcionando servicios de cifrado y firma digital
  - La herramienta `gpgsm` proporciona soporte para S/MIME

Comandos gpg2

`--symmetric` o `-c`	Cifrado y descifrado con clave secreta
`--print-md <algo>`	Resumen de mensaje
`--gen-key`	Generación de pares de claves
`--encrypt` o `-e`	Cifrado con clave pública/privada
`--decrypt` o `–d`	Descifrado con clave pública/privada
`--sign` o `-s`	Firma digital de datos
`--verify`	Verificación de una firma
`--sign-key <name>`	Firma de una clave pública

Resumen

- Introducción
- Criptografía simétrica
  - Cifrado de bloque
  - Cifrado de flujo
- Funciones resumen
  - Autenticación de mensajes
- Criptografía asimétrica
  - Cifrado y descifrado
  - Firma digital
  - Intercambio de claves
- Certificados digitales
  - Modelos de confianza
- Aplicaciones
  - OpenSSL
  - GnuPG