

[Cifrar] = transformar msj \Rightarrow 2 parámetros = msj y clave
↳ si esta, la transformación no puede ser invertida

Principio de kerckhoffs: criptosist debe ser seguro incluso si todo sobre el sist., excepto la clave, es de público conocimiento

- Msj cifrado puede publicarse y enviarse x canales no seguros
- Seguridad reside en la clave q tiene tamaño < que msj

Cifrado por rotación

Clave k = num de 1 a 26 y cada letra = un num.

\Rightarrow se reemplaza cada letra x la q ocupa k posiciones a continua° en el alfabeto

Ej. k=2 \Rightarrow Encl('hola') = 'jqncl'

⚠ Muy inseguro \Rightarrow Sale x fuerza bruta (probando 26 posibles claves)

Cifrado de sustitución

Reemplazar un simbolo x otro \Rightarrow n! posibilidades \Rightarrow no sale x fuerza bruta.

PERO se mantiene propiedades estadísticas del lenguaje

- Obtener freq. estimada de cada simbolo del lenguaje
- Calcular freq de cada simbolo del cifrado
- Asumir que simbolos de mayor freq se corresponde cl el de mayor freq del lenguaje

Cifrado Vigenere

Sustitu° polialfabetica i.e. un mismo simbolo puede transformarse a ≠. Clave compuesta x n numeros. Aplicar Rot - x segun la clave.

Ej. k = 'cbbc' \Rightarrow Encl('clave') = 'embxzg'

Ataque

Test de Kasiski:

- Buscar secuencias repetidas
- Calcular dist entre secuencias
- Long clave = múltiplo del MCD entre las distancias halladas

Método de coincidencia mutua:

- Determinar letras + probables en cada bloque (# bloques = long. clave)
- Estimar rotaciones q llevan a cada grupo a equipararse con las estadísticas del lenguaje
- Probar rotaciones, eliminando combinaciones si sentido

Cifrado simétrico

Terna de algoritmos:

- Gen = generador de claves
- Enc_K(m) = cifrado
- Dec_K(c) = decifrado

Propiedades:

- Salida de Gen = espacio de claves K
- Entrada de Enc = espacio de msjs M
- Entrada de Dec = espacio de cifrados C
- $\forall m \text{ y } k \text{ validos} : \text{Dec}_K(\text{Enc}_K(m)) = m$

Ataques

Pasivos:

- De texto cifrado: A solo tiene cifrados y busca obtener los msjs
- De texto plano conocido: A conoce pares (m, c) el mismo clave y busca m' a partir de otro cifrado c'

Activos:

- De texto plano escogido: A obtiene cifrado de cualquier msj q deseé y busca obtener msj de un cifrado \neq
- De texto cifrado escogido: A obtiene decifrado de msjs q deseé y busca obtener decifrado de un msj \neq

Secreto perfecto

Un criptosist. es seguro si ningún adversario puede computar cualquier función del texto plano a partir del cifrado q posee \Rightarrow condición de secreto perfecto.

- $P(M = a)$: prob de q msj sea a
- $P(K = x)$: prob de q clave sea k
- $P(C = b)$: prob de q cifrado sea c

Un criptosist. tiene secreto perf si:

- \forall todo distribu° de probabilidades en M, cada msj m y cada cifrado c $/ P(C=c) > 0 : P(M=m/C=c) = P(M=m)$
i.e msj y cifrado son probabilistica° independientes.
- Si pl cualquier par de msjs m_1 y m_2 : $P(C=c/M=m_1) = P(C=c/M=m_2)$.

One Time Pad

- Gen: $k \leftarrow \{0,1\}^n$
- Enc: $\text{Enc}_k(m) = m \oplus k$
- Dec: $\text{Dec}_k(c) = c \oplus k$

Condiciones:

- $|k| \geq |M|$
- Nunca cifrar 2 msjs cl misma clave
- Clave aleatoria

Resultados interesantes

- Cualquier criptosist cl secreto perfecto es reducible a OTP
- Cualquier sist no reducible a OTP no posee secreto perfecto
- \Rightarrow Secreto perfecto muy impráctico, necesitamos otras construcciones

Criptosistemas de flujo

- Gen: $s \leftarrow S \Rightarrow$ Salida de un generador pseudoaleatorio
- $\text{Enc}_s(m) = G(s) \oplus m$
- $\text{Dec}_s(c) = G(s) \oplus c$

$$\Delta |S| \lll |M|$$

cuanto + larga la seed,
+ difícil de romper

Generadores pseudoaleatorios

- Algoritmos determinísticos
- Expanden una entrada llamada seed \Rightarrow Salida parece random

Sea $D = \{f : \{0,1\}^n \rightarrow \{0,1\}^n\}$ una flia de func $\Rightarrow G : \{0,1\}^s \rightarrow \{0,1\}^n$ ($s < n$) es gen. pseudoaleatorio cl respecto a D si: $\forall f \in D, P(f(G(r^s)) \neq f(r^n)) = \epsilon \Rightarrow$ despreciable
 \hookrightarrow Secuencia real⁺ aleatoria

Eavesdropping Indistinguishability Test: EAV_{A,π}

Dado un adversario $A(n)$ y un criptosist. $\pi(n)$,

- (1) A emite m_0 y m_1
- (2) Se genera $k \leftarrow k$
- (3) Se genera $b \leftarrow \{0,1\}$
- (4) Se calcula $c \leftarrow \text{Enc}_k(m_b)$ y se envía a A
- (5) A emite $b' = \{0,1\} \Rightarrow \text{EAV}_{A,\pi} = 1$ si $b = b'$ i.e. A gana
 \Rightarrow Si $P(\text{EAV}_{A,\pi} = 1) = 0.5 + \epsilon(n)$, π es indistinguible

\hookrightarrow Si $G(s)$ es un gen pseudoaleatorio, π es indistinguible. Si puedo distinguir $G(s)$ de una sec. aleatoria, π NO pasa EAV.

Estado de un criptosistema

- Seguro: cumple expectativas de modelo de seguridad
- Debilitado: 3 aduersarios cl prob no despreciables de éxito PERO esfuerzo muy alto
- Quebrado: 3 aduersarios cl prob no despreciables de éxito en tiempos practicables

Obs: criptosist. puede ser seguro y estar quebrado \Rightarrow la seguridad es relativa al escenario de ataque

Multiple Message Eavesdropping Test: Mula_{A,π}

Dado un adu A y un criptosist. π .

- (1) A emite $(m_{00}, m_{01}, \dots, m_{0l})$ y $(m_{10}, m_{11}, \dots, m_{1l})$
- (2) Se genera $k \leftarrow k$
- (3) Se genera $b \leftarrow \{0,1\}$
- (4) Se calcula $c_i = \text{Enc}_k(m_{bi})$ y se le envía a A
- (5) A emite $b' = \{0,1\} \Rightarrow \text{Mula}_{A,\pi} = 1$ si $b = b'$
 \Rightarrow Si $P(\text{Mula}_{A,\pi} = 1) = 0.5 + \epsilon$, π es indistinguible

Obs: Criptosist. de flujo NO son seguros bajo multiples cifrados

Cifrado probabilístico

Si fun. de cifrado es determinística, NO es segura bajo multiples cifrados \Rightarrow se agrega un valor a la func. generadora. Valor no debe repetirse p/ una misma clave = nonce o IV. Es una sec. aleatoria pública q se debe guardar p/ desencriptar.

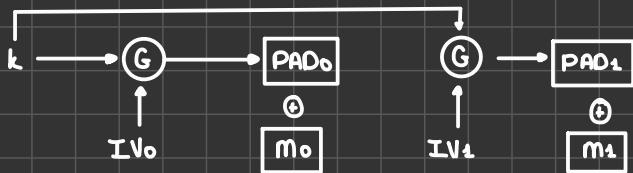
\Rightarrow Entrada al generador = (k, IV)

Modo sincronizado:

Hay que guardar dond estamos parados en el pad \Rightarrow Se usa una parte de k p/ cifrar cada msj.



Modo no sincronizado: pl cada msj un IV ≠



Chosen Plaintext Indistinguishability Test: CPA_{A,π}

Dado un adu A y un criptosist. π .

- (1) Se genera una clave $k \leftarrow k$
- (2) A obtiene $f(x) = \text{Enc}_k(x)$ y emite (m_0, m_1)
- (3) Se genera $b \leftarrow \{0, 1\}$
- (4) Se calcula $c = \text{Enc}_k(m_b)$ y se le envia a A
- (5) A emite $b' = \{0, 1\} \Rightarrow \text{CPA}_{A,\pi} = 1$ si $b = b'$
 \Rightarrow Si $P(\text{CPA}_{A,\pi} = 1) = 0.5 + \epsilon$, π es indistinguible

Propiedades CPA

- Criptosist. determinístico NO puede ser CPA secure
- Si criptosist. es CPA sec pl un msj, tmb lo es pl multiples msjs
- Un criptosist. CPA sec pl msjs de tamaño limitado debe ser extendido \Rightarrow Criptosist. de bloque

Primitivas de cifrado de bloque

Definidos pl msjs de tamaño fijo. Son primitivas y NO criptosist. PERO forman criptosist. Al combinarse $c_i \neq \text{modo}$. Al ser deterministicos, NO se deben usar como criptosist.

$$\Rightarrow k = \{0, 1\}^n \text{ y } C = M = \{0, 1\}^n$$

Si $|m| < |bloque|$: padding

- Simple pad: completar al ceros \Rightarrow necesito conocer $|m|$
- Des pad: agregar un bit 1 y luego 0s \Rightarrow Si $|m| = |bloque|$, se agrega un bloque extra

Si $|m| > |bloque|$: se divide msj en bloques, se extiende el último y se transforma cada bloque segun algun modo de encadenamiento

Modos de encadenamiento

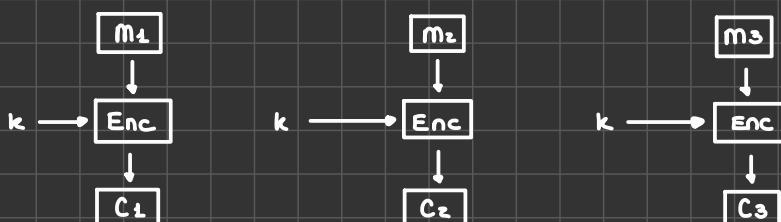
Extender primitiva de cifrado q x ahí no es CPA secure y si la encadenó con otras sí.

ECB: tratar al msj como un conj de bloques independientes

\Rightarrow 2 msjs idénticos siempre devuelven el mismo cifrado

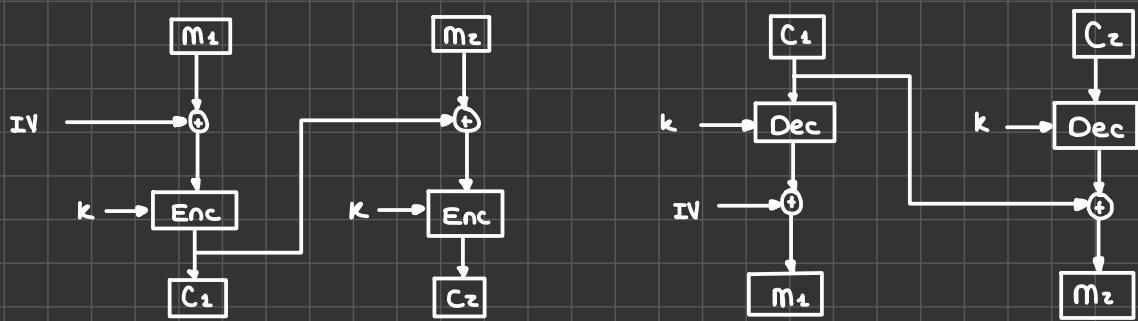
Un atacante puede detectar 2 msjs iguales, si un msj tiene data repetida, si 2 msjs comparten prefijo...

\Rightarrow Atacante obtiene mucha info^o, NO es CPA Secure



CBC: Salida de un bloque = entrada pl el prox

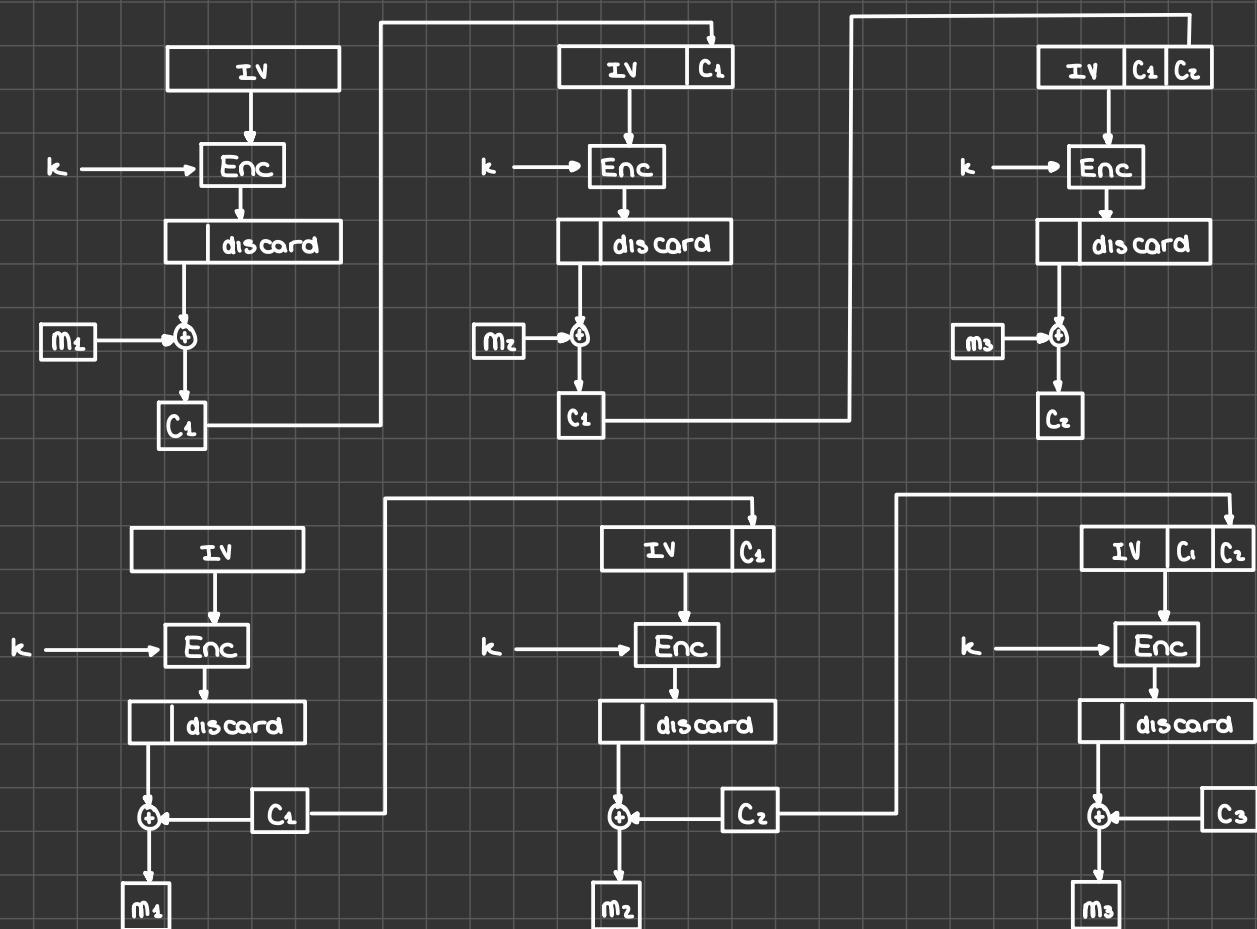
- Disminuye traspaso de info°
- Requiere IV aleatorio
- ⇒ CPA secure



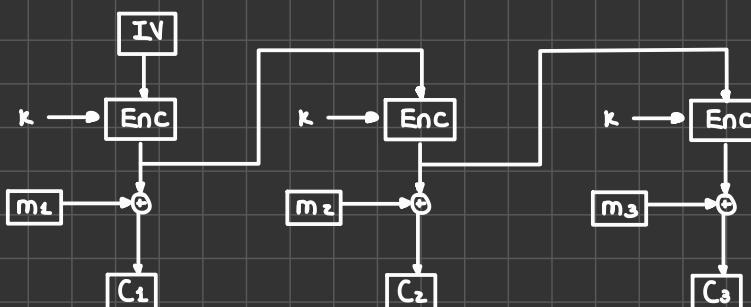
CFB: Utiliza solo función Enc ⇒ menos complejo

⇒ Se cifra IV y se xorea la salida

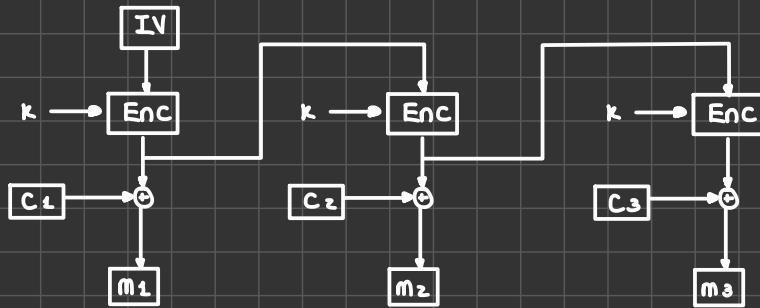
⚠ CPA Secure con IV random



OFB: Construye un criptosist. de flujo a partir de una primitiva de bloque

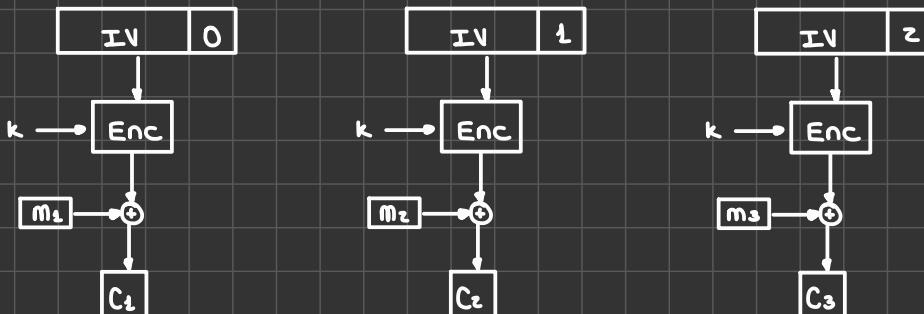


⚠ CPA Secure con IV random



Counter: contador pl para generar secuencia de bits de clave

- Permite acceso random
- Util pl para procesamiento paralelo



A CPA Secure si no repite (k, IV)

Data Encryption Standard (DES)

Trabaja con textos planos binarios.

- Entrada y Salida: 64 bits
- Clave: 64 bits (56 efectivos)

Pasos:

- Texto plano 64 bits pasa a una func. de permutación inicial
- Se divide la permutación en dos bloques (L y R)
- Cada bloque x un proceso de encriptación 16 veces
- Se unen bloques encriptados y se permuta una ultima vez sobre el cifrado final

Proceso de encriptación:

- Transformación de clave: permutación sobre clave y luego se divide en 2. Cada una se desplaza a lo largo n cant. de bits q no siempre es el mismo. La clave k_i de la ronda $i = \text{permutación}$ entre 2 mitades desplazadas y así obtenemos k_i pl cada i entre 1 y 16. Clave inicial 64 bits pero 8 son pl paridad \Rightarrow cada mitad tiene 32 bits y pl obtener k_i se seleccionan 48 de los 56 durante la permutación.
- Expansión: texto plano de 64 bits en 2 mitades de 32 bits \Rightarrow se toma R, se expande a 48 bits y se xorea cl la clave de la ronda actual
- Sustitución S-Box: se divide 48 bits en 8 bloques de 6 bits y se reemplaza cada bloque x su correspondiente en la S-Box
- Permutación P-Box: se toman 4 bits x bloques, se concatenan y se permutan
- XOR y swap: se xorea el valor resultante y la mitad izq de la ronda anterior (L_{i-1}) y luego la mitad izq actual toma el valor de la mitad derecha anterior ($L_i = R_{i-1}$)

A DES se puede atacar en 2^{63} intentos

Variantes de DES

3 - Des:

3 claves independientes $\Rightarrow C = \text{Enc}_{K_1}(\text{Dec}_{K_2}(\text{Enc}_{K_3}(m)))$

- 3 veces + lento q DES
- Seguridad del orden de 112 bits $< 3 \times 56 = 168$ bits x el Meet-In-The-Middle
- Inmune a criptoanálisis diferencial y lineal

AES : reemplazo de DES

- Bloques = 128 bits \Rightarrow matriz 4 bytes x 4 bytes
- Clave = 128, 192 o 256 bits

Operaciones:

- (1) Byte Sub : cada byte sustituido x otro byte sacado de la S-box. Ningun byte sustituido x si mismo ni su complemento.
- (2) Shift Rows : cada fila desplazada x cant. de veces
- (3) MixColumns : cada columna multiplicada x un matriz \Rightarrow cambia la posic. de cada byte de la columna
- (4) Add Round keys: el output de lo anterior se xorea cl la clave de la ronda actual

Pasos:

- Primera ronda solo Add Round keys cl clave original
- Rondas intermedias \Rightarrow todas las ops en ese orden
- Ultima ronda \Rightarrow todas las ops menos Mix Columns

Chosen Ciphertext Attack: CCA_{A,π}

Dado un nivel de seguridad n , un adversario $A(n)$ y un criptosist. $\pi(n)$.

1. Se genera $k \leftarrow k$
2. A obtiene $f(x) = \text{Enc}_k(x)$ y $g(x) = \text{Dec}_k(x)$
3. A emite (m_0, m_1)
4. Se genera $b \leftarrow \{0, 1\}$ y se envia $c \leftarrow \text{Enc}_k(m_b)$
5. A emite $b' = \{0, 1\}$ (A no puede ver $g(c)$) $\Rightarrow \text{CCA}_{A, \pi} = 1$ si $b = b'$
 $\Rightarrow \Pr[\text{CCA}_{A, \pi} = 1] < 0.5 + \epsilon$. π es indistinguible

⚠ Ningun criptosist. visto es CCA Secure. Es un problema q requiere control de integridad.

Message Authentication Code (MAC)

Primitiva criptografica pl identificar adulteraciones y hacer control de integridad.

- Gen: generador de claves \Rightarrow Salida define espacio de claves k
 - Mac (etiquetado): $t \leftarrow \text{Mac}_k(m) \Rightarrow$ func. no deterministica
 - Vrfy (verifica^o): $\text{Vrfy}_k(m, t) = \{0, 1\}$
- ↳ $\forall m$ y k validos, $\text{Vrfy}_k(m, \text{Mac}_k(m)) = 1$

Message Authentication Experiment: Mac-forge_{A,π}

Dado un nivel de seguridad n , un adversario $A(n)$ y un criptosist. $\pi(n)$.

1. Se genera $k \leftarrow k$
2. A obtiene $f(x) = \text{Mac}_k(x)$
3. A realiza las evaluaciones q quiera $c | f(x) \Rightarrow$ Sea Q el conjunto de evaluaciones
4. A emite (m, t) tal q $m \in Q \Rightarrow \text{Mac-forge}_{A, \pi} = 1$ si $\text{Vrfy}_k(m, t) = 1$
 $\Rightarrow \Pr[\text{Mac-forge}_{A, \pi} = 1] < \epsilon$, π es infalsificable ante un ataque de mensaje escogido

⚠ Si un MAC ignora un solo bit, es atacable.

Construir un MAC

Opcion: a partir de una primitiva de cifrado de bloque \Rightarrow CBC-MAC

↑ tamaño fijo de msjs

- Gen: $k \leftarrow \{0, 1\}^n$
- Mac: $m = m_1 || m_2 || \dots || m_j$, $t_0 = 00\dots 0$ y $t_i = F_k(t_{i-1} \oplus m_i) \Rightarrow \text{Mac}_k(m) = t_j$
- Vrfy: $\text{Vrfy}_k(m, t) = 1 \Leftrightarrow t = \text{Mac}_k(m)$

↑ func. pseudoaleatoria

Observaciones:

- Construcción infalsificable solo si msjs de misma longitud
- CBC-MAC no usa IV o usa IV NO aleatorio = crucial pl seguridad \Rightarrow IV aleatorio no sería seguro

Extensiones para mensajes arbitrarios:

- (1) $k' = F_k(lm)$ y $t_i = F_{k'}(t_{i-1} \oplus m_i) \Rightarrow \text{Mac}_k(m) = t_j$
- (2) $m' = lm || m \Rightarrow \text{Mac}_k(m) = \text{CBC-MAC}_{k'}(m')$
- (3) k_1, k_2 y $t' = \text{CBC-MAC}_{k_1}(lm) \Rightarrow t = F_{k_2}(t')$

Funciones de hash criptográficas

- Gen: $s \leftarrow S \Rightarrow s$ NO es clave, es un selector
 - Hash: $h = H_s(m) \in \{0, 1\}^L$ dnd $L = \text{long del hash}$
- ⇒ Analogas a MACs pero sin clave

Características

1. No inversibles: es imposible sacar entrada x conociendo la salida y
2. Deterministas: mismo msg, misma salida
3. Rapidez de computo
4. Alto grado de mutabilidad: cualquier varia^o en el msg hace q el hash sea completo^t ≠

[Colisión] = $x \neq x'$ y $h(x) = h(x')$ i.e si hay $n+1$ msgs y n salidas, \exists al menos 1 colisión

Propiedades

- Resistencia a preimágenes: $\forall y$, es computacional^t imposible hallar $x / h(x) = y$
- Resistencia a segundas preimágenes: $\forall x$ es computacional^t imposible hallar $x' \neq x / h(x') = h(x)$
- Resistencia a colisiones: es computacional^t imposible hallar $x, x' / h(x) = h(x')$

Collision Resistance: Hash-CollA,H

Dado un nivel de seguridad n , un adv A(n) y una fija de funciones de hash H(n):

1. Se selecciona una función s 4- S
2. A obtiene $Hs(x)$
3. A emite $x, x' \Rightarrow$ Hash-CollA,H = 1 si $x \neq x'$ y $Hs(x) = Hs(x')$ (A encuentra col^o)
 \Rightarrow Si $P(Hash-CollA,H = 1) < \epsilon$, la función H es libre de colisiones

Modelo generativo iterativo

1. Ajustar el tamaño del msg y agregar long de bloque
 2. Aplicar func. de compresión a cada bloque
- \Rightarrow Permite extender func. de hash de long fija resistente a colisiones a una de long variable tmb resistente a colisiones

Construcción

Sea (Gen_n, h) una func. hash de long fija c/ input $z.l(n)$ y output $l(n)$.

1. Gen = Gen_n \Rightarrow S 4- Gen_n
 2. Hs(x): input s y msg $x \in \{0,1\}^n$ de long $z^{l(n)} - 1$
 - Sea $L = |x|$ y $B = \lceil L/l \rceil$ i.e # de bloques en x \Rightarrow completar x c/ 0s pt q long(x) sea multiplo de l(n)
 - Definir $z_0 = 0^l$ y pt cada $i = 1, \dots, B$, hacer $z_i = hs(z_{i-1}, l|x_i|)$ siendo hs la func. hash de long fija
 3. Emitir salida z_B .
- \Rightarrow La seguridad esta dada x la func. de compresión hs. Una col^o en hs solo puede ocurrir si hay una col^o en hs.

Ejemplos de funciones de hash

MDS: aplicación iterativa \Rightarrow destruida xq puedo encontrar colisión

- Entrada: hasta 264 bits
- Salida: 128 bits

SHA-1: aplica^o iterativa y se construye sobre la base de MDS \Rightarrow debilitada

- Entrada: hasta 264 bits
- Salida: 160 bits

SHA-3: estandarizada ; aplica^o modelo esponja

- Entrada: hasta 264 bits
- Salida: 224, 256, 384 o 512 bits

Funciones de hash y MACs

Es posible construir un MAC a partir de una func. de hash.

HMAC: Dado $H(x)$ func. de hash libre de colisiones.

- Gen: $k \leftarrow k$, $s \leftarrow s$
- Mac: $t = H_s((k \text{ xor } opad) \parallel H_s((k \text{ xor } ipad) \parallel m))$
 - opad = 0x3636...36
 - ipad = 0x5c5c...5c

⇒ Es infalsificable (paso Mac-Forge)

Función de Hash vs. MAC

- Func. de hash = integridad ⇒ permite chequear si el msg fue alterado
- MAC = integridad y autenticidad
 - ↳ usa una private key lo cual permite al receptor asegurarse de q el msg no fue modificado y q quien lo envio es quien estaba esperando

Seguridad de funciones de hash

Sea $h: A \rightarrow B$.

- Preimagenes: dado $y \in B$, hallar $x \mid h(x) = y \Rightarrow x$ fuerza bruta requiere $|B|$ intentos
- Segundas preimagenes: dado $(x, y) \mid h(x) = y$, hallar $x' \mid h(x') = y \Rightarrow x$ fuerza bruta requiere $|B|$ intentos
- Colisiones: hallar $x, x' \mid h(x) = h(x') \Rightarrow x$ fuerza bruta requiere $|B|^{0.5}$

Privacidad e Integridad

1. Cifrar y autenticar: $C \leftarrow \text{Enc}_{k_1}(m)$ y $t \leftarrow \text{Mac}_{k_2}(m) \Rightarrow$ NO usar xq t puede brindar info^o de m
2. Autenticar, luego cifrar: $C \leftarrow \text{Enc}_{k_1}(m \parallel \text{Mac}_{k_2}(m)) \Rightarrow$ puede ser seguro pero requiere prueba de seguridad
3. Cifrar, luego autenticar: $t \leftarrow \text{Mac}_{k_2}(\text{Enc}_{k_1}(m)) \Rightarrow$ Seguro

⚠ $k_1 \neq k_2$ p q sean independientes y q los estados internos no se repitan

Cifrado Autenticado

Cifrado + control de integridad.

Sean Π^e (Gen, Enc, Dec) un criptosist CPA-Secure y Π^m (Gen^m, Mac, Vrfy) un MAC infalsificable.

- Gen: $k_1 \leftarrow \text{Gen}_e$ y $k_2 \leftarrow \text{Gen}^m$
- Enc: $t \leftarrow \text{Mac}_{k_2}(\text{Enc}_{k_1}(m))$
- Dec: Si $\text{Vrfy}_{k_2}(c, t) = 1 \Rightarrow m = \text{Dec}_{k_1}(c)$

⇒ Criptosist. resultante es CCA-Secure

Counter with CBC-MAC (CCM)

"Authenticate - then - encrypt" ⇒ $\text{Enc}_k(\text{CBC-MAC}_k(m) \parallel m)$

Es una prueba de seguridad específico. Si el IV y el nonce no coinciden ni se reutilizan, la construc^o es CCA-Secure usando la misma clave.

Galois/Counter Mode

- Provee cifrado autenticado
- AES en modo GCM es lo + recomendado
 - mucho + eficiente q CCM
 - detecta falsificación
 - nivel de seguridad esperado

AES-GCM devuelve el cifrado junto con el tag. Cualquier cambio en el texto produce un cambio en el cifrado que por ende provoca modificaciones en el tag. Por otro lado, el mensaje y el cifrado tienen el mismo largo.

Cifrado asimétrico y Firma digital

Un criptosist. CCA Secure garantiza confidencialidad e integridad **PERO** ambas partes deben conocer la clave. La clave no se puede transmitir x canal inseguro \Rightarrow ¿cómo se comparten?

- Dos pts : canal seguro
- Multiples pts:
 - Una clave x cada combinación \Rightarrow cada parte debe administrar $n-1$ claves
 - Un único pt de confianza = trusted third party

key Distribution Center (KDC)

Entidades q centralizan intercambio de claves. Comparten una clave cl cada entidad q participa (k_A, k_B, k_C, \dots).

Si A quiere comunicarse cl C:

- A envia pedido a KDC
- KDC crea clave de sesión k_S
- KDC envia k_S a A y a C, cifrandola cl k_A y k_C respectivamente[†]

Problema: único pt de fallo

Criptografía Asimétrica

Dos claves: una pública

- Una p/ cifrar
- Una p/ decifrar

Tipos:

- Intercambio de claves: permite a 2 partes generar una clave
- Cifrado asimétrico
- Firma digital: similar al MAC

Intercambio de claves

Protocolo $\Pi(n) \rightarrow$ Transcript, k_A, k_B , ejecutado x 2 partes.

- Entrada: ninguna
 - Salida: conj de msjs intercambiados, k_A conocida x una parte y k_B conocida x la otra
- \Rightarrow Permite calcular $k_A, k_B / k_A = k_B$

key - Exchange Experiment: $k_{EA,\Pi}$

Dado un adv A y una primitiva de intercambio de claves Π .

1. Se ejecuta Π y se genera $k = k_A = k_B$
 2. Se genera $b \in \{0,1\}$
 3. Si $b = 0 \Rightarrow k' \in \{0,1\}^n$. Si $b = 1 \Rightarrow k' = k$.
 4. A obtiene Trans y k' y emite $b' = \{0,1\} \Rightarrow k_{EA,\Pi} = 1$ si $b = b'$
 \hookrightarrow A tiene q decir si Trans genera o no la key
- \Rightarrow Si $P(k_{EA,\Pi} = 1) < 0.5 + \epsilon$, Π es seguro

Intercambio Diffie - Hellman

1. A define $G = \text{grupo}$, $q = \text{tamaño}$ y $g = \text{generador}$
2. A elige $x \in \mathbb{Z}_q$ y calcula $h_A = g^x$ don $\mathbb{Z}_q = \{0, 1, \dots, q-1\}$
3. A \rightarrow B: (G, q, g, h_A)
4. B elige $y \in \mathbb{Z}_q$ y calcula $h_B = g^y$
5. B \rightarrow A: (h_B)
6. A calcula h_B^x y B calcula h_A^y

Seguridad:

Dado g^x y g^y , no debería ser posible obtener x o y = problema del logaritmo discreto y no tiene solu^o eficiente.

Condición necesaria **PERO** no suficiente \Rightarrow necesita la conjetura de dec^o DH: dados g, g^x y g^y , un adv no puede distinguir g^{xy} de un valor aleatorio.

En la práctica: requiere un canal de transmⁱo autenticado i.e un adv q pueda modificar mensajes rompe la seguridad de DH \Rightarrow se complementa cl firmas digitales

Criptosistema asimétrico

- Gen: pk y sk (public y secret keys)
- Enc: $Enc_{pk}(m) \quad | \quad Dec_{sk}(Enc_{pk}(m)) = m$
- Dec: $Dec_{sk}(c)$

Eavesdropping Indistinguishability Test : EAV_{A,π}

Dado un adv A y un criptosist. π.

1. Se generan (pk, sk) $\leftarrow k$
2. A recibe pk y emite m_0 y m_1
3. Se genera $b \leftarrow \{0, 1\}$
4. Se calcula $c \leftarrow Enc_{pk}(mb)$ y se le envia a A
5. A emite $b' = \{0, 1\} \Rightarrow EAV_A, \pi = 1$ si $b = b'$
 \Rightarrow Si $P(EAV_A, \pi = 1) < 0.5 + \epsilon$, π es indistinguible

Consecuencias:

- Si A conoce pk , puede cifrar lo q quiere
- Si un criptosist. asimétrico π es indistinguible p/ un adv pasivo \Rightarrow π es CPA-Secure
- ⚠ Recordar q CPA-Secure requiere cifrado no determinístico

"Textbook" RSA

- Gen:
 1. Elegir p, q numeros primos $\Rightarrow n = pq$
 2. $e \leftarrow (0, \phi(n)) / \text{MCD}(e, \phi(n)) = 1$
 3. Calcular $d / ed \equiv 1 \pmod{\phi(n)}$
 - $\Rightarrow pk = (n, e)$ y $sk = (n, d)$
- Enc_{pk}(m) = $m^e \pmod{n}$
- Dec_{sk}(c) = $c^d \pmod{n}$

Problemas: el cifrado es determinístico

- Si e y m pequeños / $m^e < n \Rightarrow$ puedo calcular el log
- Modulos repetidos: si z pares de cuales comparten n , es posible hallar n y recuperar sk

Solución: PKCS 1 v1.5 \Rightarrow versión de RSA cl padding aleatorio \rightarrow long de n en bytes

$\Rightarrow m' = 00000000 \parallel 00000010 \parallel r \parallel 00000000 \parallel m$ donde $r = k - |m| - 3$ bytes aleatorios $\neq 0$

Obs: Solo se permiten cifrar msjs de hasta $n - 11$ bytes

⚠ Se cree CPA-Secure pero no CCA-Secure

El gamal

Basado en DH pero pensado p/ ser usado como criptosist.

- Gen:
 1. Seleccionar G, q y g
 2. $x \leftarrow \mathbb{Z}_q$ y $h = g^x$
 - $\Rightarrow pk = (G, q, g, h)$ y $sk = (G, q, g, x)$
- Enc_{pk}(m) = $(c_1, c_2) = (g^y, h^y \cdot m)$ dond $y \leftarrow \mathbb{Z}_q$
- Dec_{sk}(c) = c_2 / c_1^x

Características:

- Si la prueba de decisión DH es difícil en G , es CPA-Secure
- Es probabilístico
- Permite reutilizar G, g y q
- No está limitado a campos numéricos \Rightarrow permite usar anillos de polinomios y curvas elípticas

Cifrado asimétrico en números

Nivel de seguridad relativo a los tamaños de los conjuntos involucrados.

- RSA: $n = pq$

Garal: $n = q$

Cdo se utilizan campos numéricos, $n \geq 1024$ bits. Campos de otro tipo, n puede cambiar (Ej. curvas elípticas $n \geq 320$ bits).

Firmas digitales

- Gen: $k = (pk, sk)$
 - Sign: $s \leftarrow \text{Sign}_{sk}(m)$
 - Vrfy: $b \leftarrow \text{Vrfy}_{pk}(m, s)$
- $\} \quad \text{Vrfy}_{pk}(m, \text{Sign}_{sk}(m)) = 1$

Objetivo: integridad

Ventajas:

- Pública verificables
- Transferible \Rightarrow puede enviar a varios destinatarios o reenviarse y sigue siendo verificable
- Proveen no repudio i.e. quien firma no puede negar haberlo hecho

Signature Experiment: Sig-forge_{A,π}

Dado un nivel de seguridad n , un adv A y una firma digital π .

1. Se genera $(sk, pk) \leftarrow k$
2. A obtiene $f(x) = \text{Sign}_{sk}(x)$ y pk
3. A realiza las evaluaciones que quiere sobre $f(x) \Rightarrow$ sea Q el conj de evaluaciones
4. A emite $(m, s) / m \notin Q \Rightarrow \text{sig-forge}_{A, \pi} = 1$ si $\text{Vrfy}_{pk}(m, s) = 1$

RSA - Signature

- Gen:
 1. Elegir p y q números primos $\Rightarrow n = pq$
 2. $e \leftarrow (0, \phi(n)) / \text{MCD}(e, \phi(n)) = 1$
 3. Calcular $d / ed \equiv 1 \pmod{\phi(n)}$
 $\Rightarrow pk = (n, e)$ y $sk = (n, d)$
- $\text{Sign}_{sk}(m) \equiv m^d \pmod{n}$
- $\text{Vrfy}_{pk}(m, s) = ? m \equiv s^e \pmod{n}?$

⚠ NO es seguro

Ataques:

- (1) A selecciona $s \leftarrow S$, calcula $m \equiv s^e \pmod{n}$ y emite $(m, s) \Rightarrow$ consigue prueba de falsificación
- (2) A calcula $s_1 = f(m_1)$ y $s_2 = f(m_2)$ y emite (m_1, m_2, s_1, s_2)

Hashed RSA

- $\text{Sign}_{sk}(m) \equiv H(m)^d \pmod{n}$
 - $\text{Vrfy}_{pk}(m, s) = ? H(m) \equiv s^e \pmod{n}?$
- $\} \quad H = \text{func. de hash libre de colisiones}$

\Rightarrow NO posee prueba de seguridad a menos q se asuma un modelo ideal de H

Digital Signature Standard

- Gen:
 1. Seleccionar $H(x)$: SHA1 o SHA2
 2. Definir tamaño de claves (P, N) : (1024, 160), (2048, 224)...
 3. $q \neq$ primo de tamaño N (bits)
 4. $p \neq$ primo de tamaño $P / (p - 1) \equiv 0 \pmod{q}$
 5. $g \neq$ generador de orden $q \pmod{p} \quad g^{(p-1)/q} \neq 1$
 6. $x \neq \exists q \quad e \quad y = g^x \pmod{p}$
- $\text{Sign}_{pk}(m) = (r, s)$ dona $r = (g^k \pmod{p}) \pmod{q}$ y $s = (H(m) + x \cdot r)^{-1} \pmod{q}$
↳ $k \neq \exists q$
- Verfy $pk(m, (r, s)) = ? r = g^{u_1} \cdot y^{u_2} \pmod{p} \pmod{q}$? donde $u_1 = (H(m) + s^{-1}) \pmod{q}$ y $u_2 = (r + s^{-1}) \pmod{q}$

Protocolos

Ataques activos: Man In The Middle

- Omitir msjs
- Reescribir contenido de msjs
- Reordenar msjs
- Repetir msjs

⇒ Los esquemas de intercambio vistos no sirven contra ataques activos

Problema: Se confia en la asociación de las claves c/ la identidad de las personas

Infraestructura de claves (PKI)

- Busca asociar identidad a las claves
- Busca evitar problemas de suplantación de identidad (MitM o spoofing)
- NO aplicable a criptosist. Simétricos

Solución: certificados digitales

Certificados

Contienen al menos:

- Información de identidad (Ej. nombre)
- Clave pública asociada
- Fecha de emisión
- Intervalo de validez
- Tipo de uso: firma de msjs, cifrado de mails, firma de certificados, cifrado de sitios web

 Certificados firmados digitalmente por autoridad competente

Uso de certificados

A solicita certificado de B: provisto x B o cualquier entidad

- A puede verificar identidad del certificado
- A puede obtener Pk de B
- A puede verificar validez del certificado: tipo de uso y fecha de vigencia
- A puede verificar integridad del certificado: validar firma digital de la autoridad q lo certifica

Cadenas de firmas

Autoridades certificadoras tienen a su vez un certificado ⇒ se incluye certificado de autoridad en cada certificado.

AC raíces: pt de confianza del sist ⇒ autoridades q firman sus propios certificados

Concepto de AC raíces:

- Confiar en una ! AC
- Dicha autoridad delega en otras la capacidad de firmar certificados

Validación entre AC

No hay una ! AC raíz ⇒ si A y B tienen ACs diferentes, ¿cómo valida A el certificado de B?

- Opción 1: A confía en la AC de B
- Opción 2: las ACs se certifican entre sí ⇒ cada AC emite un certificado de la identidad de la otra

Certificados X.509

Estandar de certificados digitales. Forman cadenas de certificados.

Incluye: versión, num de serie, identificador de algoritmo de firma, nombre de emisor (AC), intervalo de validez, nombre de sujeto (Common Name), Pk del sujeto y firma (firma del hash de todo lo anterior)

Verificación:

1. Obtener Pk: verificar cadena de certificados

- Verificar integridad del certificado usando algoritmo especificado y P_k
- Verificar intervalo de validez: vigente y AC debe estar vigente
- Verificar identidad: comparar CN cl el q espera la aplicaº (depende del uso del certificado)
- Verificar uso del certificado: validar q este autorizado p/ el uso q se le quiere dar

Revocación de claves

- ¿Porque revocar? \Rightarrow Fue aueriguada x un atacante
- \Rightarrow Cambio anticipado (cambio de dueño de clave)

Problemas:

- No revocar claves q no deben ser revocadas
- Evitar q se revoquen claves si autorizaº
- Propagar revocaº p/ evitar futuras comunicaciones cl dicha clave

Listas de certificados revocados: Similar a listas de num de tarjeta de credito robados

- Lista actual: certificados vigentes y revocados
- Lista historico: certificados expirados y revocados

X.509:

- Solo el emisor de un certificado puede revocarlo
- Se agrega al CRL de la autoridad certificante global
- La lista se puede obtener p/ validar offline u online

PKI

- \Rightarrow Asocia una identidad a los PK
- \Rightarrow Cifrados asimetricos
- \Rightarrow No hay un pt de fallo

KDC

- \Rightarrow Centraliza el intercambio de claves
- \Rightarrow Cifrado simetrico
- \Rightarrow Unico pt de fallo

Intercambio Simetrico de claves

Equivalente simetrico a DH

Protocolo Needham - Schroeder

Protocolo de intercambio de claves simetrico \Rightarrow genera claves de sesion entre pares. Requiere de un serv. centralizado \Rightarrow cada entidad comparte una clave cl el KDC.

Primera opcion:

- A \rightarrow S : $\{A \rightarrow B\}k_{AS}$
- S \rightarrow A : $\{k_{AS} \parallel k_{BS}\}k_A$
- A \rightarrow B : $\{k_{AS} \parallel k_{BS}\}k_B$

Problemas:

- Repetición (Replay): un adv puede grabar msjs de A hacia B y reenviar $\{k_{AS} \parallel k_{BS}\}k_B$ y los msjs siguientes \Rightarrow B no tiene forma de saber q no esta hablando cl A
- Reuso de claves (key reuse): adv intercepta paso 2 y cada vez q A quiere iniciar una comunicaº cl B, le envia ese msj \Rightarrow A y B estarian usando la misma clave

Segunda opcion:

- A \rightarrow S : $A \rightarrow B \parallel N_A$
- S \rightarrow A : $\{A \parallel B \parallel N_A \parallel k_{AS}\}k_A \Rightarrow$ Na coincide cl el enviado i.e. no es repetido
- A \rightarrow B : $\{A \parallel k_{AS}\}k_B$
- B \rightarrow A : $\{N_B \parallel k_{AS}\}k_A$
- A \rightarrow B : $\{N_B \parallel k_{AS}\}k_B$

Pasos:

- A \rightarrow S : A, B, Na \Rightarrow A manda al KDC identificando a A y a B, indicandole q se quiere comunicar cl B
- S \rightarrow A : $\{Na, k_{AB}, B, \{k_{AB}, A\}k_{AS}\}k_A \Rightarrow$ S genera clave de sesion k_{AB} y manda una copia a A encriptada

bajo kes (clave entre B y S) pl q A manda a B. Tmb le envia una copia a A encriptada bajo kes (clave entre A y S). Como A probablemente pida claves pl comunicarse cl ≠ personas, el nonce na asegura a A q el msj es nuevo q S esta respondiendo a ese msj en particular. La inclu^o de B es pl indicar a A a quien le tiene q compartir la clave.

3. A → B : {k_{AB}, A}k_S ⇒ A manda clave de sesion a B q puede desencriptarla cl su clave privada
4. B → A : {N_B}k_{AB} ⇒ B manda un nonce encriptado cl k_{AB} pl amsg a A de un intento de comunica^o
5. A → B : {N_B-1}k_{AB} ⇒ A confirma la comunica^o y B sabe q no es una repeticion por N_B

Problema:

En 3., un atacante puede interceptar y mandar {A||ks3|ks} a B. Luego, B manda {N_B}k_S a A que en realidad serio el adv ⇒ Adv logra impersonar a A.

↳ ks de una sesion vieja y fuerza comunica^o cl B

Modificación Demming - Sacco : usar timestamps

1. A → S : A||B||Na
2. S → A : {A||B||Na||ks||{A||T||ks3|ks3}k_A}
3. A → B : {A||T||ks3|ks} ⇒ reduce ventana util de la ks
4. B → A : {N_B}k_S
5. A → B : {N_B-1}k_S

Canal Seguro

Protocolos de Canal seguro: buscan construir un camino de comunica^o punta a punta q brinde confidencialidad e integridad a partir de un canal inseguro (i.e dnd hay un atacante interceptando msjs).

↳ secure socket layer

SSL / TLS

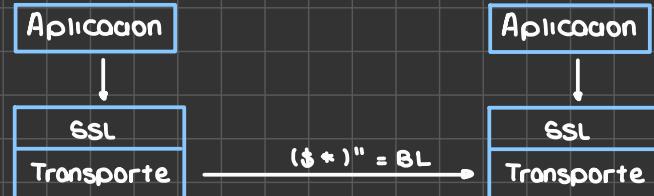
- Ofrece seguridad a nivel transporte
- Utiliza un canal de transporte confiable : TCP
- Provee confidencialidad, integridad y autentica^o de los participantes (origen y destino)

TLS (transport layer security): evolu^o de SSL , mejora deficiencias encontradas, estandar actual

Conexion insegura



Conexion con SSL



TLS Record

TLS recibe un msj a enviar ⇒ lo divide en bloques que luego comprime y calcula el hash de cada uno. Final⁺, encrypta cada bloque y hash y lo envia al servicio inferior (Ej. TCP).

Tipos de mensajes:

- TLS Handshake : config inicial, autentica^o y negocia^o de material criptografico
 - TLS Application Data: info^o del protocolo encapsulado
 - TLS Alert: informa de eventos y problemas
 - TLS Change cipher Spec: Concluye una renegocia^o de clave
- ↳ puede ocurrir en cualquier momento solicito x client o server

Intercambio de claves:

- RSA ✓
- DH, certificado RSA o DSS ✓
- DH, anonymous
- ECDH, DH sobre curvas elipticas
- kerberos

Hay variantes obsoletas q utilizan 512 bits.

↳ problemas de exporta^o de material criptografico

Cifrado simétrico:

- ChaCha20 (TLS 1.3)
- Triple DES (3DES - EDE - CBC)
- AES ; AES - CBC ; AES - CCM ; AES - GCM ✓
- IDEA CBC
- ARIA - CBC ; ARIA - CCM ; ARIA - GCM

Hay variantes obsoletas con claves de 40 bits.

Hash: HMAC-MD5 ; HMAC-SHA1 ; HMAC-SHA2 (256/384) ; AEAD (Authenticated Encryption with Associated Data)

Autenticación: firmas RSA y DSS

Sesión TLS

Asocia° entre 2 pares. Puede soportar múltiples conexiones.

Información:

- ID de sesión
- Certificado x.509 v3 del otro extremo (opcional)
- Método de compre° acordado
- Método de encripta° y MAC acordado
- 'Master Secret' : clave de sesión compartida (48 bytes)

⚠ Uso transparente x eficiencia
⇒ NO es un protocolo de sesión

Conexión de TLS

Describe como intercambiar datos.

Contiene:

- Secuencia de bits aleatoria de calidad criptográfica
- Claves de escritura ≠ pl cada lado
- Claves pl/MACs (hashes c/ claves) pl cada lado
- IVs necesarios
- Número de secuencia pl cliente y servidor

Cliente Hello: C → S : {Vc || r1 || S1o || Ciphers || Comps}

Server Hello: S → C : {V || r2 || S2o || Cipher || Comp}

- Vc = versión cliente ; V = min (Vcliente, Vservidor)
- r1, r2 = nonces (28 bytes rand)
- S1o = session ID (0 pl iniciar una nuevo)
- Ciphers = lista de stacks criptográficos disponibles ; Cipher = stack elegido
- Comps = métodos de compre° disponibles ; Comp = elegido

Certificate: S → C : {certificado3} (si server es autenticado)

ServerkeyExchange: S → C : {p || hash(r1||r2||p)}ks

Certificate request: S → C : {ctype || CAs} (si server es autenticado)

ServerHelloDone: S → C : {}

- r1, r2 = nonces previos pl evitar replay
- p = parámetros criptográficos (Ej. e, n pl RSA)
- ks = clave priv del server correspondiente a la pública del certificado
- CType = tipo de certificados permitidos dsa el lado algoritmo

Client certificate: C → S : {certificado3} (solo si fue solicitado)

Client key Exchange: C → S : {V || PRErand}ks (versión RSA)

C → S : {g^b mod p}ks (versión DH)

- V = versión informada previa x el client pl prevenir downgrade attacks
- g, p = parámetros DH informados x el server
- PRE = g^{ab} mod p

Change cipher spec: C → S : E3 ➔ a partir de aca, el client usa parametros negociados

Finish: C → S : { hash(master || opad) || hash(msgs || master || ipad)) }

Change cipher spec: S → C : E3 ➔ a partir de aca, el server usa parametros negociados

Finish: S → C : { hash(master || opad) || hash(msgs || master || ipad)) }

- opad = 01011100

- ipad = 00110110

- msgs = concat de todos los msjs intercambiados hasta el momento

TLS Alert

- Envia eventos fuera de banda: advertencias, fallas o evento fatal invalida conexion
- CloseNotify : evento q indica fin de sesion ➔ no se envian nuevos msjs o son ignorados
- Errores fatales : msj no esperado, MAC incorrecto, error al comprimir, error durante handshake, parametro ilegal
- Advertencias o errores : no hay cert o cert invalido, cert no soportado, expirado o revocado