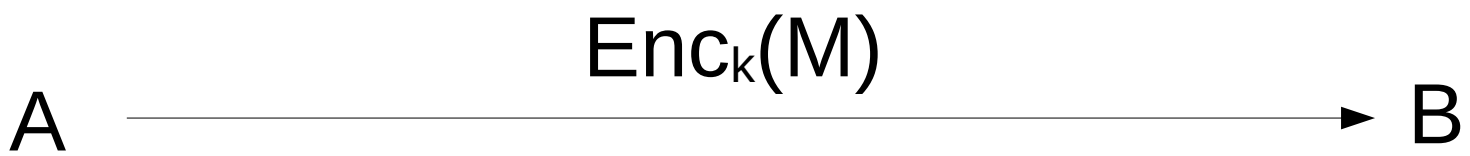
Criptografía y Seguridad

Criptografía: Cifrado asimétrico y Firma digital

Distribución de claves

- Un criptosistema CCA-Secure permite enviar información manteniendo:
  - Confidencialidad
  - Integridad



- Pero requiere que ambas partes conozcan una misma clave

Distribución de claves

- Las claves no pueden transmitirse por un canal inseguro.
- ¿Como se comparten las claves?
- Dos puntos → Uso de canal seguro
- Multiples puntos →
  - Una clave por cada combinación
    - Cada parte debe administrar n-1 claves
    - ¿Si hay n puntos cuantas claves se necesitan?
  - Un punto unico de confianza
    - También referido como Trusted third party

Distribución de claves

- KDC – Key Distribution Centers
  - Son entidades que centralizan intercambio de claves
  - Comparten una clave con cada entidad que participa (k_a, k_b, k_c, …)
  - Si A quiere comunicarse con C, envía un pedido al KDC
    - El KDC crea una nueva clave k_s, clave de sesión, y se la envía a A y C, cifrandola con k_a y k_c respectivamente
- Permiten tener n claves para n entidades
- Pero es un único punto de falla

Distribución de claves

- KDC – Key Distribution Centers

Criptografía asimétrica

- “We stand today on the brink of a revolution in cryptography”, Diffie-Hellman (1976).

Criptografía asimétrica

- Se basa en la asimetría de ciertos problemas
  - Por ejemplo, es fácil cerrar un candado
  - Pero se necesita una llave para abrirlo
- ¿Se puede crear un criptosistema donde existan dos contraseñas?
  - Una para cifrar, la otra para descifrar.
  - Incluso si un atacante posee la clave para cifrar, no puede recuperar el mensaje
    - Se puede publicar dicha clave a proposito !!!
    - También llamada criptografía de clave pública

Criptografía asimétrica

- Intercambio de claves (nuevo!)
  - Permite a dos partes generar una clave compartida en linea
- Cifrado asimétrico
  - Similar al cifrado simétrico
- Firma digital
  - Similar al MAC

Algebra, topología, aritmetica, ...

Repaso de algebra

- Grupo algebraico: G,+ (conjunto y operación)
  - Clausura
  - Asociatividad
  - Neutro
  - Inverso
- Subgrupo (G, G', +)
  - (G,+) es un grupo, (G', +) es un grupo
  - G contiene a G' y G' ≠Ø
- Grupo abeliano: G, +
  - (G, +) es un grupo algebraico
  - Conmutatividad

Repaso de algebra

- Grupos finitos cíclicos
  - G = ({g^n | n ε Z}, +)
    - (^ es aplicación sucesiva de +)
  - Todos los grupos de tamaño n son isomorfos
  - Grupo canónico Z_n = ({ 1, 2, 3, …, n-1 }, +)
  - g^n = e (neutro)
  - Generador: g / g es primo relativo a n
    - Hay Φ(n) generadores
    - Ord(g) = # elementos del subgrupo ciclico
    - g es elemento primitivo si ord(g) = n

Repaso de algebra

- Anillo: G, +, * (conjunto y dos operaciones)
  - (G, +) es un grupo abeliano
  - Clausura de *
  - Asociatividad de *
  - * distributiva sobre +
- Cuerpo o campo: G, +, *
  - (G, +, *) es un anillo
  - Conmutatividad de *
  - Neutro de *
  - Inverso de * en { G – neutro. de + }

Repaso de algebra

- Campo de Galois (campo finito)
  - (G, +, *)
  - |G| = n
  - Todos los campos de tamaño k son isomorfos
  - Todo campo finito tiene tamaño p^n con p primo y n entero
  - Campo canónico:
    - z_p = { k | k ε {Z_p – 0} ^ (k:p)=1 }, +, *), p primo
    - Tamaño: p - 1

Repaso de aritmética

- Artitmetica modular
  - a = b mod n ‹–› a – b = k * n
  - Se reduce al grupo canónico: 0 <= a < n
- Ejemplo:
  - 2+8 mod 5 = 0
  - 4 * 3 mod 7 = 5
- Si n es primo, tenemos un campo finito (Zp)
- Si n no es primo, tenemos un anillo
  - Si consideramos solo k / (k:n) = 1, tenemos un grupo multiplicativo (Z^*_n, *)

Repaso de aritmética

- Aritmética modular
  - Si $(k : n) = 1 \rightarrow \exists\ k^{-1} / k * k^{-1} = 1 \bmod n$
  - $a^{\Phi(n)} = 1 \bmod n$ ($\Phi(n)$ es el tamaño de Z^*_n)
    - Si p es primo, $a^{p-1} = 1 \bmod p$
- Calculo de $\Phi(n)$
  - $\Phi(n*m) = \Phi(n) * \Phi(m)$, si $(n : m) = 1$
  - $\Phi(p^a) = p^k - p^{k-1} = p^{k-1} * (p - 1)$, si p es primo

Intercambio de claves

- Es un protocolo $\Pi(n)$, ejecutado por dos partes:
  - Π: (n) $\rightarrow$ Tran, k_a, k_b
  - No tiene entrada (salvo el parámetro de seguridad)
  - La salida del protocolo es
    - Un conjunto de mensajes intercambiados
    - Una clave k_a conocida solo por una de las partes
    - Una clave k_b conocida solo por la otra parte
- Condición fundamental:
  - $k_a = k_b$

Seguridad frente a ataques pasivos

- Key-Exchange experiment: $KE_{A,\Pi}$
- Dado un adversario A, y una primitiva de intercambio de claves Π:

1) Se ejecuta Π, sea $k = k_a = k_b$
2) Se genera $b \leftarrow \{ 0, 1 \}$
3) Si $b = 0 \Rightarrow k' \leftarrow \{ 0, 1 \}^n$

   Si $b = 1 \Rightarrow k' = k$
4) A obtiene Trans y k', y emite $b' = \{ 0, 1\}$

- $KE_{A,\Pi} = 1$ si $b = b'$ (A gana)

Si $Pr[KE_{A,\Pi}=1] < 0.5 + \varepsilon \Rightarrow \Pi$ es seguro.

Intercambio Diffie-Hellman

1) A define G, q , g, donde G es un grupo, q el tamaño y g un generador.
2) A elije $x \leftarrow Z_q$ y calcula $h_1 = g^x$
3) A envía a B: (G, q, g, h1)
4) B elije $y \leftarrow Z_q$ y calcula $h_2 = g^y$
5) B envía a A: (h2)
6) A calcula $k_a = h_2^x$
7) B calcula $k_b = h_1^y$

$Zq = \{ 0, 1, \ldots, q-1 \}$

Si A y B se conocen de antemano, pueden tener predefinidos (G, q, g)

La seguridad de DH

- Primero, dado g^x y g^y, no debería ser posible obtener x o y.
  - Esto se conoce como el problema del logaritmo discreto, y no tiene solución eficiente
- Condición necesaria pero no suficiente.
- Se necesita la conjetura de decisión DH:
  - Dados g, g^x y g^y, un adversario no puede distinguir g^{xy} de un valor aleatorio
  - Nota: ¡La formulación se realizó muchos años después de la publicación del algoritmo!
  - Hoy se sabe que es un problema NP-Hard

Diffie Hellman en la prática

- La versión original requiere un canal de transmisión autentificado
    - O sea, un atacante que pueda modificar mensajes rompe la seguridad de DH
- Se complementa con firmas digitales

Criptosistema asimétrico

- Es una terna de algoritmos
  - Gen: () → pk, sk (public key, secret key)
  - Enc (cifrado): $Enc_{pk}(m)$
  - Dec (descifrado): $Dec_{sk}(c)$
- Propiedades
  - Para todo m y k válidos: $Dec_{sk}(Enc_{pk}(m))=m$

Prueba de indistinguibilidad

- Eavesdropping Indistinguishability test: $Eav_{A,\Pi}$
- Dado un adversario A, y un Criptosistema Π:
  - Se genera una clave: (pk, sk) $\leftarrow$ K
  1) A recibe pk y emite m_0 y m_1
  2) Se genera b $\leftarrow$ { 0, 1 }
  3) Se calcula c $\leftarrow$ $Enc_{pk}(m_b)$ y se le envía a A
  4) A emite b' = { 0, 1}
- $Eav_{A,\Pi}$ = 1 si b = b' (A gana)

Si $Pr[Eav_{A,\Pi}=1] < 0.5 + \varepsilon$ => Π es indisting.

Consecuencias

- Un atacante que conoce pk
  - Pude cifrar cualquier mensaje
  - O sea, tiene acceso a la función de cifrado
- Para un criptosistema asimétrico:
  - Si Π es indistingible para un adversario pasivo
    => Π es CPA-Secure
- Recordar que:
  - CPA-Secure REQUIERE cifrado no determinístico

“Textbook” RSA

- Generación de claves
  - Elegir p,q ← Números primos. **n** = p*q.
  - e ← (0, Φ(n)) | mcd(**e**, Φ(n)) = 1
  - Calcular **d** / e*d ≡ 1 mod Φ(n)
  - Pk = (n, e), sk = (n, d)
- $Enc_{pk}(p) \equiv p^e \bmod n$
- $Dec_{sk}(c) = c^d \bmod n$

Problemas

- Tal cual está formulado, el cifrado es determinístico ¿Que problemas trae?
- Si e y m son pequeños: $m^e < n$, y entonces se puede calcular el logaritmo
  - Durante mucho tiempo se utilizo e=3 para ahorrar tiempo
- Módulos repetidos
  - Si dos pares de claves comparten el mismo n, es posible recuperar n (y luego la clave privada)

Ejemplo RSA

- Parámetros muy pequeños a modo ilustrativo:
  - p=2.357, q=2.551, n=p*q=6.012.707
  - Φ(n) = (p-1)*(q-1) = 6.007.800
  - e=3.674.911 (elegida al azar)
  - Usando euclides extendido: d=422.191
- Cifrado de m=5.234.673
  - $e(m)= 5.234.673^{3.674.911} \bmod 6.012.707 = 3.650.502$
- Descifrado de m'= 3.650.502
  - $d(m')= 3.650.502^{422.191} \bmod 6.012.707 = 5.234.673$

PKCS 1 v1.5

- RSA Labs Public Key Cryptography Standard
- Define una versión con padding aleatorio de RSA:
  - Sea k la longitud de n en bytes
  - Solo se permiten cifrar mensajes de hasta n-11 bytes. Sea D la longitud de m en bytes
  - m' = 00000000 || 00000010 || r || 00000000 || m
  - Donde r = k – D – 3 bytes aleatorios != 0
    - La condición es para evitar ambigüedades
- Se cree que es CPA – Secure
  - Pero se encontraron ataques que muestran que no es CCA-Secure

Tamaño de claves

n = p * q (rsa-2048)

25195908475657893494027183240048398571429282126204032027777137836043662020707595556264018525880784406918290641249515082189298559149176184502808489120072844992687392807287776735971418347270261896375014971824691165077613379859095700097330459748808428401797429100642458691817195118746121515172654632282216869987549182422433637259085141865462043576798423387184774447920739934236584823824281198163815010674810451660377306056201619676256133844143603833904414952634432190114657544454178424020924616515723350778707749817125772467962926386356373289912154831438167899885040453640235273819513786365643912120103971228221207203570

El gamal (basado en DH)

- Generación de claves
  - Seleccionar G, q, g (campo G de tamaño q)
  - $x \leftarrow Z_q$, $h = g^x$
  - pk = (G, q, g, h), sk = (G, q, g, x)
- $Enc_{pk}(m)$: $y \leftarrow Z_q$, salida: $c=(c_1, c_2)=(g^y, h^y * m)$
- $Dec_{sk}(c) = c_2/c_1^x$

- Resultados conocidos:
  - Si la prueba de decisión DH es dificil en G, El gamal es CPA-Secure

Diferencias con RSA

- El Gamal es probabilístico
- Permite reutilizar los parámetros G, q, g
- No está limitado a campos numéricos
  - Permite usar anillos de polinomios
    - Tienen 2^n elementos → Fácil mapear mensajes
  - Permite usar curvas elipticas
    - El problema de decisión DH es más complejo

Ejemplo El Gamal

- (con parámetros muy pequeños)
  - G= Z^*_q, q = 2.357, g = 2
  - x = 1.751 -->g^x mod q= $2^{1.751}$ mod 2.357 = 1185
- Cifrado de m=2.035
  - Seleccionar y= 1.520 (aleatorio)
  - e(m) = ($2^{1.520}$ mod 2.357,2.035 * $1.185^{1.520}$ mod 2.357)
  - e(m) = (1.430, 697)
- Descifrado de m'=(1.430, 697)
  - d(m') = $1.430^{-1.751}$ * 697 mod 2.357 = 2.035

Cifrado asimétrico en números

- El nivel de seguridad es relativo a los tamaños de los conjuntos involucrados
  - En RSA: n=p*q
  - En El Gamal: n=q
- Cuando se utilizan campos numéricos, n >= 1024 bits
  - En la actualidad se recomiendan 1536 o 2048 bits
- Para campos de otro tipo, los números pueden variar
  - Por ejemplo, sobre curvas elípticas, n >= 320 bits

Firmas digitales

- Similares a los MACs
  - Su objetivo es la integridad
- Pero tienen ciertas ventajas:
  - Publicamente verificables
  - Es transferible: Puede enviarse simultaneamente a varios destinatarios o reenviarse y sigue siendo verificable
  - Proveen no repudio: Quien firma no puede negar haberlo hecho
    - Propiedad muy importante cuando está reglamentada juridicamente

Firma digital

- Es una terna de algoritmos
  - Gen: (n) $\rightarrow$ k = (sk, pk)
  - Sign (firma): s $\leftarrow$ $Sign_{sk}(m)$
  - Vrfy (verificación): b = $Vrfy_{pk}(m, s)$
- Propiedades
  - Para todo m y k válidos: $Vrfy_{pk}(m, Sign_{sk}(m)) = 1$

Seguridad de una firma digital

- $\text{Sig-forge}_{A,\Pi}$
- Dado un nivel de seguridad n, un adversario A, y una firma digital $\Pi(n)$:

  1) Se genera una clave $k=(sk, pk) \leftarrow K$
  2) A obtiene $f(x)=\text{Sign}_{sk}(x)$ y pk
  3) A realiza las evaluaciones que quiera de f(x)

     (Llamese Q al conjunto de las evaluaciones)
  4) A emite (m, s)

- $\text{Sig-forge}_{A,\Pi} = 1$ si $\text{Vrfy}_{pk}(m, s) = 1$ y m no pertenece a Q

RSA-Signature

- Identico a RSA-Encryption, pero invirtiendo los papeles de las claves:

- Generación de claves
  - Elegir p,q ← Números primos. **n** = p*q.
  - d ← (0, Φ(n)) | mcd(**e**, Φ(n)) = 1
  - Calcular **d** / e*d ≡ 1 mod Φ(n)
  - pk = (n, e), sk = (n, d)
- $Sign_{sk}(m) \equiv m^d \bmod n$
- $Vrfy_{pk}(m, s) = ¿m = s^e \bmod n?$

Este esquema,
aunque común en
la literatura, es inseguro

Problemas de RSA-Signature

- Considerar el adversario A:
  - $s \leftarrow S$ (selecciona una firma al azar)
  - Calcula $m = s^e \bmod n$
  - Emite (m, s)
- ¡Consigue pasar exitosamente la prueba de falsificación!
- Otro ataque:
  - $s_1 = f(m_1)$, $s_2 = f(m_2)$
  - Emite $(m_1 * m_2, s_1 * s_2)$
  - Ejercicio: Verificar que la salida pasa la verificación

Hashed RSA

- Busca solucionar los problemas anteriores:
- Introduce una función de hash libre de colisiones
  - $Sign_{sk}(m) \equiv H(m)^d \bmod n$
  - $Vrfy_{pk}(m, s) = ¿H(m) = s^e \bmod n?$
- Pero no posee una prueba de seguridad a menos que se asuma un modelo ideal de H

Digital Signature Standard

- Generación de claves
  - Seleccionar H(x): SHA1 o SHA2
  - Tamaño de claves: (L, N): (1024, 160), (2048, 224), (2048, 256) o (3072, 256).
  - q ← primo de tamaño N (bits)
  - p ← primo de tamaño P / (p – 1) = 0 mod q
  - g ← generador de orden q mod p
    - $g^{(p-1)/q} \neq 1$
  - Seleccionar p, q, g ($G=Z^*_p$)
  - x ← Z_q, $y = g^x \bmod p$
  - pk = (p, q, g, y), sk = (p, q, g, x)

Digital Signature Standard

- $Sign_{sk}(m)$: $k \leftarrow Z_q$, $r = (g^k \bmod p) \bmod q$
  - $s = [H(m) + x*r) * k^{-1} \bmod q$
  - $Sign_{sk}(m) = (r, s)$
- $Vrfy_{pk}(m, (r, s))$: $v1 = [H(m) * s^{-1}] \bmod q$
  - $v2 = r*s-1 \bmod q$
  - ¿$r = g^{u1}*y^{u2} \bmod p \bmod q$?

Lectura Recomendada

Capítulos 9-12

Introduction to Modern Cryptography
Katz & Lindell