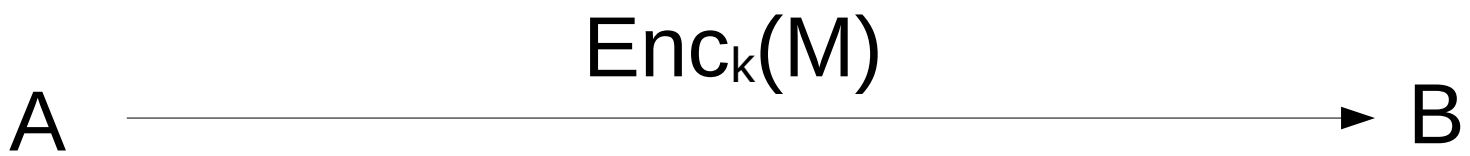
Criptografía y Seguridad

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

  Cuando la comunicacion se da entre dos empresas, tal vez se puede pedirle a una persona que viaje y entregue la clave.
- Multiples puntos → Cuando hay N personas que quieren comunicarse con M sitios web, se necesitan generar N*M/2 claves, y alguna manera de transmitirlo.
  - 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

      Durante mucho tiempo se uso esta alternativa. La idea es que existe un servicio KDC (key distribution center).

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 (desde el punto de vida de seguridad, no tanto disponibilidad)

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 => $k' \leftarrow \{ 0, 1 \}^n$
   Si b = 1 => k' = k
4) A obtiene Trans y k', y emite b' = { 0, 1}

Se le da al atacante k' (que es la clave k que se genero al ejecutar el intercambio de claves o tambien puede ser una secuencia aleatoria).
Si el atacante puede distinguir cual de las dos le llego, gana.
La idea es que si el atacante puede obtener informacion de la clave solo con ver los mensajes, entonces la clave esta comprometida.

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

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

Intercambio Diffie-Hellman

Es el mas usado por su simpleza y por las pruebas matematicas que lo respaldan

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 $k_b = h_1^y$

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

2) A elige un numero aleatorio entre 0 y q (el tamaño del grupo)
3) se envia PUBLICAMENTE G, q, g, h1

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: $Dec_{sk}(Enc_{pk}(m))=m$

Por que decimos que se cifra siempre con la clave publica y se descifra con la privada y no indistintamente?
Si se pudiera descifrar con la clave publica, cualquiera pudiera descifrar los mensajes.
Es importante que cualquier persona pueda cifrar los mensajes, pero solo el receptor pueda descifrarlos.
Por este motivo, las claves NO son intercambiables.

Prueba de indistinguibilidad

Equivale a CPA

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

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

Se genera un par clave publica-privada, y recibe la publica. El atacante genera dos mensajes, y recibe un mensaje cifrado.
El atacante tiene que adivinar cual de los mensajes recibio.
Dado que el atacante posee la clave publica, entonces puede generar mensajes a gusto, entonces para pasar la prueba el criptosistema debe ser NO DETERMINISTICO.

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

Hay bastante complejidad en la generacion de pares de claves.
Primero se elige p,q.
Se elige un valor e entre 0 y pho(n) que sea coprimo con phi(n).
Al e ser coprimo con d, va a tener un inverso.
Con eso emitimos dos claves

- 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)
- $Enc_{pk}(p) \equiv p^e \bmod n$

  Cifrar el mensaje es primero convertirlo a un numero, luego elevarlo a la e, y luego reducirlo a modulo n
- $Dec_{sk}(c) = c^d \bmod n$

  Descifrar el texto es tomar el mensaje cifrado, convertirlo a un numero, elevarlo a la d y reducirlo a modulo n. Finalmente pasar de numero a mensaje de vuelta.

Surge la pregunta de si ENC y DEC son inversas.
Si yo parto de un p, lo cifro, y lo descifro puedo recuperar el p original. La respuesta es que SI son inversas.
El problema que surge, es que la version "de libro" de RSA es DETERMINISTICA.

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

Le voy a agregar 11 bytes a todos los mensajes. Funciona como un padding que busca resolver varios problemas.
r hace que sea no deterministico.

Tamaño de claves

n = p * q (rsa-2048)

25195908475657893494027183240048398571429282126204032027777137836043662020707595556264018525880784406918290641249515082189298559149176184502808489120072844992687392807287776735971418347270261896375014971824691165077613379859095700097330459748808428401797429100642458691817195118746121515172654632282216869987549182422433637259085141865462043576798423387184774447920739934236584823824281198163815010674810451660377306056201619676256133844143603833904414952634432190114657544454178424020924616515723350778707749817125772467962926386356373289912154831438167899885040453640235273819513786365643912120103971228221207203570

El gamal (basado en DH)

Es el otro criptosistema mas difundido en la actualidad, que hoy en dia tiene mas uso que RSA

- **Generación de claves** (los mismos que DH)

  Conceptualmente, la idea detras del gamal es hacer el intercambio DH y pasar el mensaje al mismo tiempo

  - Seleccionar G, q, g (campo G de tamaño q)
  - $x \leftarrow Z_q$, $h = g^x$

    x -> numero al azar
    h -> generador del numero
  - $pk = (G, q, g, h)$, $sk = (G, q, g, x)$

    publicar dicho h
- $Enc_{pk}(m)$: $y \leftarrow Z_q$, salida: $c=(c_1, c_2)=(g^y, h^y * m)$

  se selecciona un numero al azar y
- $Dec_{sk}(c) = c_2/c_1^x$

  1) Se elige un numero al azar (entre 0 y la cant. de elementos de G)
  2) Emitir dos numeros como texto cifrado
     - g elevado al numero al azar
     - la clave publica elevada al numero al azar multiplicado por el mensaje

  Para descifrar eso, tomamos el segundo componente del texto cifrado y lo multiplicamos por el inverso del primer mensaje a la x. h a la y seria como la clave compartida

  Lo que ocurre en el gamal es que para cada nuevo mensaje que envio genero un nuevo y, entonces nunca repito la misma clave exacta. Notar que la y no forma parte del texto cifrado, y no la usamos a la hora de descifrar
- **Resultados conocidos:**
  - Si la prueba de decisión DH es dificil en G, El gamal es CPA-Secure

Diferencias con RSA

- El Gamal es probabilístico (no hay que agregarle un padding para resolver el problema de que RSA normal es deterministico)
- 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

RSA esta definido sobre el campo de los numeros modulo n. No es generalizable a cualquier campo. En cambio, el gamal esta definido de manera mucho mas abierta, y se puede usar en el campo de anillos de polinomios (que tenga una cantidad de elementos que sea potencia de 2) o curvas elipticas (se define como el subconjunto de puntos que pertenecen a una curva eliptica y caen en coordenadas enteras).
Con curvas elipticas podriamos tener el mismo nivel de seguridad teorico pero con operaciones mucho mas eficientes de ejecutar, y claves mas chicas.
Estos son los motivos por el cual se esta reemplazando RSA por el gamal.

Ejemplo El Gamal

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

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

Esta slide muestra como las curvas elipticas requieren de claves mas chicas que en campos numericos (como RSA)

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

Se invierte el uso de las claves. Antes cualquiera podria encriptar y uno solo podia desencriptar.
Ahora, uno solo puede "firmar" y todos pueden "verificar"

Esta firmado por mi y no depende de ninguna clave de la otra persona, entonces se lo puedo pasar a muchas personas sin necesidad de hacer el firmado multiples veces

Muchos paises tienen leyes de firma digitales, que establecen ciertos requisitos para la cual se le puede dar la misma importancia a una firma digital que a una firma a mano.
El requerimiento es que la clave publica que se usa para verificar tiene que estar avalada por algun organismo gubernamental. Para avalar una firma, simplemente el organismo la firma con su clave.

Firma digital

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

A partir de la clave secrete, genera una firma. La verificacion utiliza el mensaje, la firma y la clave publica.
La idea es justamente que todo el mundo podria verificar una etiqueta sin verificar la integridad (porque nosotros con el MAC teniamos que compartir la clave para que pueda verificar).
El problema con el MAC es que la persona que lo pudo verificar, podria desencriptar el mensaje, y generar un nuevo MAC.

Seguridad de una firma digital

- Sig-forge$_{A,\Pi}$

Practicamente lo mismo que MAC-Forge.
Se genera un par de claves. Se le da al atacante la posibilidad de obtener las firmas de los mensajes que quiera, y se le da la clave publica (para verifiar lo que quiera). Se le pide al atacante que emita un mensaje y una firma que pasen la verificacion, pero que no sea un mensaje para el cual haya pedido la firma en el paso anterior.

- 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)=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)

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

RSA-Signature

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

ERROR CONCEPTUAL:
- En ningun algoritmo en ningun caso se firma con una clave publica.
- En ningun algoritmo en ningun caso se encripta con una clave privada

- Generación de claves
  - Elegir p,q ← Números primos. **n** = p*q.
  - d ← (0, Φ(n)) | mcd(**e**, Φ(n)) = 1

Se elige un numero al azar entre 0 y el phi de esa base

  - Calcular **d** / e*d ≡ 1 mod Φ(n)
  - pk = (n, e), sk = (n, d)

Se genera un numero publico y uno privado

- $Sign_{sk}(m) \equiv m^d \bmod n$

Firmar es elevar el mensaje a la clave privada. Despues se hace mod n

- $Vrfy_{pk}(m, s) = ¿m = s^e \bmod n?$

Verificar es elevar la firma a la clave publica y hacerle mod n.

Este esquema, aunque común en la literatura, es inseguro

Problemas de RSA-Signature

- Considerar el adversario A:
  - s ← S (selecciona una firma al azar)
  - Calcula $m = s^e \bmod n$
  - Emite (m, s)

Calculamos la firma elevada a la clave publica. Con esa cuenta me invento un mensaje, y emitiendo eso puedo pasar el sig-forge.
El mensaje que generamos no tenia sentido, pero igual puedo pasar sig-forge.

- ¡Consigue pasar exitosamente la prueba de falsificación!
- Otro ataque:

s1 es la firma de m1

  - $s_1 = f(m_1)$, $s_2 = f(m_2)$
  - 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
  - $Sign_{sk}(m) \equiv H(m)^d \bmod n$
  - $Vrfy_{pk}(m, s) = ¿H(m) = s^e \bmod n?$

Ventaja adicional: en RSA comun, el tamano de los mensajes no puede ser arbitrariamente grande porque esta condicionado por n. Cuando nosotros introducimos el hash, nuestra funcion deja de tener este problema.

- Pero no posee una prueba de seguridad a menos que se asuma un modelo ideal de H

Es lo que se usa en la practica. Usa una funcion de hash criptografica (libre de colisiones). Primero se calcula el hash del mensaje y luego se lo eleva al coeficiente privado (para firmar). Para verificar, se eleva la firma a la clave publica mod n, y luego se chequea si es igual al hash del mensaje.

Si el hash que usamos tiene resistencia a preimagen, entonces un atacante no puede generar un mensaje que genere determinado hash. Por lo tanto, se puede demostrar que Hashed RSA genera una firma digital infalsificable (siempre y cuando nuestra funcion de hash tenga resistencia a preimagen).

Digital Signature Standard

- Generación de claves

Hoy en dia no se usa mas RSA-Hashed y se usa DSS, que es mucho mas complicada

  - 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 mod p
  - pk = (p, q, g, y), sk = (p, q, g, x)

DSS sigue la idea de utilizar una funcion de hash, y maneja un tamano doble de clave (porque se trabaja con dos grupos algebraicos en vez de uno). La idea por detras de esto es que se puede aplicar algo muy parecido a lo que era el gamal, pero en vez de utilizar una direccion de hash, trabajamos en un espacio algebraico de tamano grande (como 1024), y cuando termine voy a reducir el resultado a un campo algebraico mas chico (como 224).
En consecuencia, RSA genera una firma de tamano grande como 2048, y DSS logra trasladar estos numeros a mas chicos (como 256)

Digital Signature Standard

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

Lectura Recomendada

Capítulos 9-12

Introduction to Modern Cryptography
Katz & Lindell