5-Criptografía Pública

Esquema genérico de la criptografía pública

- En este caso tenemos una clave que está formada por clave pública + clave privada.
- Es extremadamente segura, pero es computacionalmente muy costosa (hay operaciones matemáticas complicadas).
- Se utilizan problemas del tipo NP para esta criptografía:
  - Por ejemplo factorización de números primos.
- La criptografía publica surgió como consecuencia del problema de distribución de claves:
  - Diffie, Hellman y Merkle sentaron las bases teóricas dando la idea teórica para el intercambio de claves en 1976:
    - Diffie, Whitfield; Hellman, Martin (1976-11-01). "New directions in cryptography". *IEEE Transactions on Information Theory*. **22** (6): 644–654. CiteSeerX 10.1.1.37.9720. doi:10.1109/TIT.1976.1055638. ISSN 0018-9448
- Rivest, Shamir y Adleman (RSA), pusieron las matemáticas a la idea anterior utilizando la potenciación en aritmética modular y la factorización de números primos en 1977.

Esquema genérico de la criptografía pública

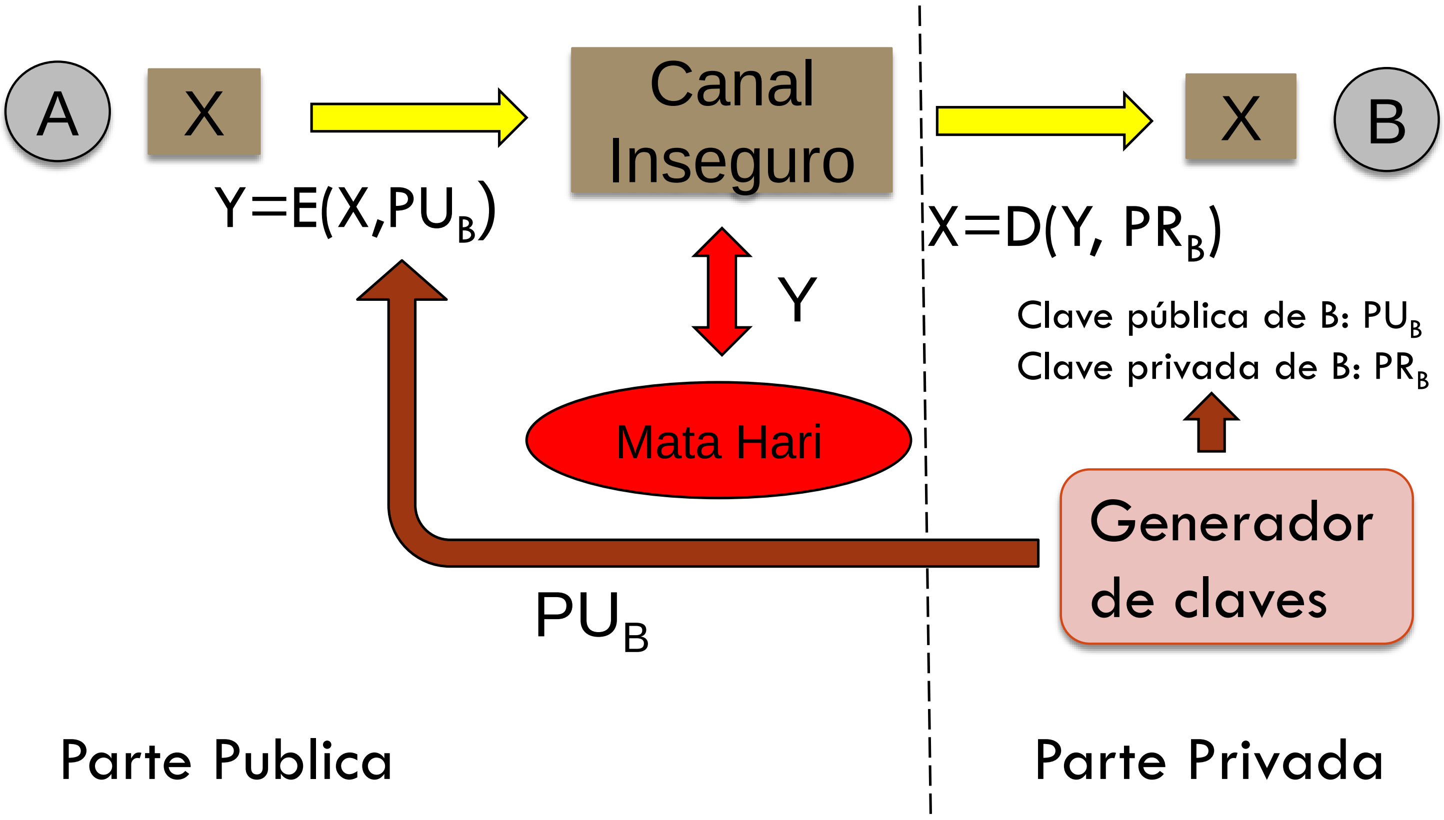
- Texto Plano:
- Texto Cifrado:
- Algoritmo de cifrado:
- Algoritmo de descifrado:
- Claves públicas: PU_A, PU_B: También se conocen como exponentes de cifrado, e.
- Claves privadas: PR_A, PR_B: También se conocen como exponentes de descifrado, d.

- Propiedades básicas:
  1. El algoritmo criptográfico cumple la propiedad fundamental que derivar la clave privada de la clave pública es computacionalmente imposible.
  2. Clave para el cifrado y una diferente pero relacionada con esta para el descifrado.
  3. Es computacionalmente imposible determinar la clave de descifrado dado sólo el conocimiento del algoritmo de cifrado y la clave de cifrado.

Esquema genérico de la criptografía pública

- Cada usuario genera un par de claves que se utilizará para el cifrado y el descifrado de los mensajes (PU_A, PU_B, PR_A, PR_B).
- Cada usuario publica su clave pública.
- Cada usuario mantiene su clave privada completamente secreta.
- Si **A**licia quiere enviar un mensaje confidencial a **B**ernardo entonces cifra el mensaje con la clave pública de **B**ernardo (PU_B).
- **B**ernardo descifra el mensaje cifrado enviado mediante sus clave privada (PR_B).
- Ningún otro destinatario puede descifrar el mensaje porque sólo **B**ernardo sabe su clave privada.
- Para enviar un mensaje **B**ernardo a **A**licia se realiza el mismo procedimiento, pero en el otro sentido.

Esquema genérico de la criptografía pública: Confidencialidad



Fundamentos genéricos de la criptografía pública

- Estas condiciones anteriores para que funcione la criptografía asimétrica se traducen en los conceptos de funciones **One-Way** y **Trap-door**.
- Función One-Way:
  - $Y = f(X)$ fácil computacionalmente hablando
  - $X = f^{-1}(Y)$ es imposible computacionalmente hablando
- Función Trap-door: es una familia de funciones invertibles que cumplen:
  - $Y = f_k(X)$ fácil, si k y X son conocidos
  - $X = f_k^{-1}(Y)$ fácil, si k y Y son conocidos
  - $X = f_k^{-1}(Y)$ inviable, si Y es conocido pero no k, a k se le suele denominar dertificado.
- Por ejemplo: la aritmética modular como principio de diseño de las funciones de una sola vía:
  - x - 1 2 3 4 5 6
  - 3^x - 3 9 27 81 243 729
  - 3^x mod 7 - 3 2 6 4 5 1 (con aritmética modular para sacar x solo por fuerza bruta, $f(x)=453^x \bmod 21997$, si $f(x)=5787$ ¿Qué vale x? Tendríamos que calcular todas las posibilidades, solo unos segundos para calcular f(x) pero cuesta mucho en invertir.

RSA (Rivest, Shamir y Adleman): Generación de Claves

- Tenemos que estudiar dos partes diferenciadas para este algoritmo:
  - Generación de claves.
  - Función de cifrado.
- Para la generación de claves en la zona privada se siguen los siguientes pasos:
  1. Crear dos números primos p y q:
     - $p \neq q$.
     - Longitud en bits de p $\approx$ Longitud en bits de q.
  2. Se crea el módulo del RSA, como n=pq, y se calcula la función de Euler $\varphi(n) = \varphi(pq) = \varphi(p)\varphi(q) = (p-1)(q-1)$.
  3. Se crea el exponente de cifrado e tal que $\mathrm{mcd}\big(e, \varphi(n)\big) = 1 \;\Rightarrow e \in Z^*_{\varphi(n)}$.
  4. Ahora se crea el exponente de descifrado d tal que $ed \equiv 1 \, mod \; \varphi(n)$.
  5. Ahora ya tenemos generadas las claves:
     - Clave pública: (n, e), publicándose y haciéndose visible.
     - Clave privada: d (e indirectamente p y q, la clave de descifrado se puede generar porque se conoce estos dos primos).

RSA (Rivest, Shamir y Adleman): Generación de Claves

- El truco para que esto funcione es que la factorización de números primos es un problema complejo.
- En 1977 Martín Gardner escribió un artículo titulado “Un nuevo tipo de cifra que costaría millones de años descifrar” en su sección “Juegos matemáticos” de “*Scientific American*”.
- Gardner lanzó un desafío a sus lectores: dio un número N producto de dos primos p y q.
- N=1143816257578888676692357799761466120102182961242362562561842935706935245733897830597123563151105058989075147599290026879543.541.
- El premio era de 100 dólares por sacar p y q.
- Pasaron 17 años para sacar p y q.
- El 26 de abril de 1994, un equipo de seiscientos voluntarios anunció que los factores de N eran:
  - p = 3490529510847650949147849619903898133417764638493387843990820577.
  - q = 32769132993266709549961988190834461413177642967992942539798288533.
- Esto es solo un ejemplo de la complejidad de este problema.

RSA (Rivest, Shamir y Adleman): Función de cifrado

- La función de cifrado, tiene las siguientes fases:
  - Buscar la clave pública del receptor, es decir (n, e) de B.
  - Tenemos que representar el mensaje como un entero en el intervalo [0, n-1]. Así codificamos el mensaje en base n (recordemos que el mensaje M es un número):
    - $M = x_l n^l + x_{l-1} n^{l-1} + \cdots + x_1 n^1 + x_0$, con $x_i \in Z_n$.
    - Sabemos que $n^l \leq M \leq n^{l+1}$.
    - Lo que ciframos son los $l + 1$ números x_i, es decir x_l, x_{l-1}, …, x_1, x_0.
  - Ciframos como $y_i = {x_i}^e \, mod \, n$ con $n = pq$ y $\mathrm{mcd}\big(e, \varphi(n)\big) = 1$.
  - Así el receptor lee el mensaje como:
    - $x_i = {y_i}^d \, mod \, n$.
- Como conclusión la función $y = x^e mod \, n$ es una función **One-Way**, siendo el certificado para la **Trap-Door** la clave d.

RSA (Rivest, Shamir y Adleman): Ejemplo Numérico

- Seleccionamos dos número primos, $p = 17$ and $q = 11$.
- Calculamos $n = pq = 17 \times 11 = 187$.
- Calculamos la función de Euler $F(n) = (p - 1)(q - 1) = 16 \times 10 = 160$.
  - $F(n)=|\{m \in N \mid m \leq n \text{ se cumple } mcd(m,n)=1\}|$
  - TFA: Todo entero positivo $n > 1$ puede ser representado exactamente de una única manera como un producto de potencias de números primos:
    - $n=p_1^{a_1} p_2^{a_2} \ldots p_k^{a_k}$, con p_i primos distintos. Por ejemplo $1000 = 2^3 \times 5^3$.
  - $F(n)= (p_1-1)\, p_1^{(a_1-1)} \ldots\ldots (p_k-1)\, p_k^{(a_k-1)}$
- Seleccionamos un e primo relativo con $F(n) = 160$; así por ejemplo elegimos $e=7$.
- Determinamos $d=e^{-1} \pmod{160}$, es decir $d = 23$ (mediante el algoritmo de Euclides extendido).
- Clave pública $PU = \{7, 187\}$.
- Clave privada $PR = \{23, 187\}$.

RSA (Rivest, Shamir y Adleman): Ejemplo Numérico

- Supongamos que el mensaje que queremos mandar es M=88.
- Para cifrar $C = 88^7 \bmod 187$:
  - $88^7 \bmod 187 = [(88^4 \bmod 187) \times (88^2 \bmod 187) \times (88^1 \bmod 187)] \bmod 187$
  - $88^1 \bmod 187 = 88$
  - $88^2 \bmod 187 = 7744 \bmod 187 = 77$
  - $88^4 \bmod 187 = 59{,}969{,}536 \bmod 187 = 132$
  - $88^7 \bmod 187 = (88 \times 77 \times 132) \bmod 187 = 894{,}432 \bmod 187 = 11$
- Para descifrar $M = 11^{23} \bmod 187$:
  - $11^{23} \bmod 187 = [(11^1 \bmod 187) \times (11^2 \bmod 187) \times (11^4 \bmod 187) \times (11^8 \bmod 187) \times (11^8 \bmod 187)] \bmod 187$
  - $11^1 \bmod 187 = 11$
  - $11^2 \bmod 187 = 121$
  - $11^4 \bmod 187 = 14{,}641 \bmod 187 = 55$
  - $11^8 \bmod 187 = 214{,}358{,}881 \bmod 187 = 33$
  - $11^{23} \bmod 187 = (11 \times 121 \times 55 \times 33 \times 33) \bmod 187 = 79{,}720{,}245 \bmod 187 = 88$

RSA (Rivest, Shamir y Adleman): Observaciones

- Se ha conjeturado que romper el criptosistema RSA es polinomialmente equivalente a factorizar n, pero aún no se ha probado.
- Pero parece razonable ya que si sacamos n=pq, y se calcula la función de Euler $\varphi(n) = (p-1)(q-1)$.
- Por tanto podemos computar d tal que tal que $ed \equiv 1 \, mod \, \varphi(n)$.
- Así la complejidad del algoritmo (si no tenemos d), equivale a factorizar $\mathrm{n} = pq$.
- No obstante si n es suficientemente grande este problema es computacionalmente imposible.
- Ahora tenemos que ver varias cosas:
  - Como hacemos la potenciación en aritmética modular n.
  - Como generamos número primos grandes.
  - Como demostramos la inyectividad del RSA: $x^{ed} \equiv x \, mod \, n$.

RSA (Rivest, Shamir y Adleman): Potenciación modular

- Supongamos que queremos calcular $t = Z^e \, mod \, n$, y la representación binaria del exponente, donde l es el número de bits del mismo:

$$t = Z^e \, mod \, n = Z^{\sum_{i=0}^{l-1} a_i 2^i} mod \, n = \prod_{i=0}^{l-1} Z^{a_i 2^i} \, mod \, n = \prod_{a_i \neq 0}^{l-1} Z^{a_i 2^i} \, mod \, n = \prod_{a_i \neq 0}^{l-1} [Z^{a_i 2^i} \, mod \, n] \, mod \, n.$$

- Así en el productorio solo cuentan aquellos factores que $a_i = 1$.
- Además para calcular $i = 2$, nos apoyamos en $i = 2$ y así sucesivamente:

$i = 0$	$Z \, mod \, n$
$i = 1$	$Z^2 \, mod \, n$
$i = 2$	$(Z^2)^2 \, mod \, n$
$i = 3$	$((Z^2)^2)^2 \, mod \, n$

Recordar que partimos de la representación binaria del exponente:

$$e = \sum_{i=0}^{l-1} a_i \, 2^i$$

RSA (Rivest, Shamir y Adleman): Potenciación modular

- Así con estas ideas podemos desarrollar el siguiente algoritmo para potenciación modular.
- Para la potenciación modular podemos usar la propiedad:

  (a x b) mod n=[(a mod n) x (b mod n)] mod n
- Hay que darse cuenta que los números son muy grandes y tenemos que utilizar un algoritmo eficiente para la potenciación modular como por ejemplo este.

$\boxed{\text{POTENCIACIÓN MODULAR } (Z, e, n)}$

$X = 1$

FOR $i = \ell-1 \rightarrow 0$

$X = X^2 \bmod n$

IF $(a_i == 1)$

THEN $X = (X \cdot Z) \bmod n$

RETURN (X)

$$\boxed{e = \sum_{i=0}^{\ell-1} a_i 2^i}$$

DESCOMPOSICIÓN BINARIA DEL EXPONENTE

INPUT $\left\{ \begin{matrix} Z \\ e \\ n \end{matrix} \right.$ $\left\{ \begin{matrix} \text{COMPUTA EFICIENTEMENTE} \\ X = Z^e \bmod n \end{matrix} \right\}$ OUTPUT $Z^e \bmod n$

EJEMPLO: $Z^{11_{10}} \bmod n = Z^{(1\times 2^3 + 0\times 2^2 + 1\times 2^1 + 1\times 2^0)}$

$X = 1$

$\boxed{i=3} \rightarrow X = 1^2 \bmod n$

$a_3 = 1 \rightarrow X = 1^2 \times Z \bmod n$

$\boxed{i=2} \rightarrow X = Z^2 \bmod n$

$\boxed{a_2 = 0} \rightarrow$ NO HACE NADA

$\boxed{i=1} \rightarrow X = (Z^2)^2 \bmod n$

$a_1 = 1 \rightarrow X = (Z^2)^2 \times Z \bmod n$

$\boxed{i=0} \rightarrow X = ((Z^2)^2 \times Z)^2$

$a_0 = 1 \rightarrow X = ((Z^2)^2 \times Z)^2 \times Z \bmod n$

$\boxed{X = Z^{11_{10}}}$

RSA: Generación de números primos grandes

- Hasta el momento no hay técnicas deterministas para producir número primos grandes arbitrarios.
- El procedimiento en general es escoger un número impar aleatorio del deseado orden de magnitud, y se prueba si es primo (test de primalidad).
- Si no lo es se elige otro número aleatorio para volver a probar.
- Vamos a estudiar el más utilizado que es el test de Miller-Rabin.
- Aunque hay otros muchos de primalidad, que por ejemplo vienen descritos en:
- Libro de criptografía aplicada: A. J. Menezes; P. C. van Oorschot; S. A. Vanstone (1997). Handbook of Applied Cryptography

RSA: Generación de números primos grandes

- Los tests de primalidad son costosos, pero hay que tener en cuenta que se realizan muy infrecuentemente: cuando hay que generar las claves o cambiarlas.
- Miller-Rabin es un algoritmo probabilístico que se denomina del tipo *yes-biased Montecarlo*:
  - Una respuesta SI es siempre correcta.
  - Una respuesta NO no es siempre correcta.
- ¿Pero cuál es la pregunta que se hace el algoritmo?
  - ¿Es el número n (impar) compuesto?
- Si el algoritmo dice SI seguro que es compuesto.
- Si el algoritmo dice NO, es decir no responde, puede que si sea o puede que no sea compuesto (es decir primo).
- Cuando muchas veces el algoritmo NO dice compuesto va aumentando la probabilidad de que sea primo.

RSA: Generación de números primos grandes

- El test de Miller-Rabin (MR) se basa en el teorema pequeño de Fermat (TPF):
  - Si $a \in Z_p^*$ tal que p es primo $\Rightarrow a^{p-1} \equiv 1 \, mod \, p$.
  - El teorema no se cumple a la inversa: Es decir si $a^{p-1} \equiv 1 \, mod \, p$ con $a \in Z_p^* \Rightarrow$ p no tiene que ser primo.
  - Los números compuestos que satisfacen $a^{p-1} \equiv 1 \, mod \, p$ se llaman números de Carmichael. Por ejemplo N=561 satisface la congruencia, pero N = 561= 3x11x17.
- Vamos a utilizar el siguiente razonamiento:
  - Dado un N, si existe un entero x tal que $x^2 \equiv 1 \, mod \, N$, pero con $x \neq \pm 1 \, mod \, N \Rightarrow N$ es un número compuesto seguro (cuando N primo solo triviales).
  - En este caso a x se le denomina raíz no trivial de la ecuación de congruencia:
    - $x^2 \equiv 1 \, mod \, N.$
  - Es decir si encontramos una raíz no trivial de esta ecuación, entonces N es un número compuesto seguro.
- El algoritmo de MR intentará buscar raíces no triviales en $mod \, N$ para diferentes x's y cuando no encuentre estas raíces no triviales N tiene muchas posibilidades de que sea primo.

RSA: Generación de números primos grandes

- Así para un supuesto impar p la forma de encontrar raíces no triviales para asegurar que es compuesto es calculando los siguientes residuos en aritmética modular p.
  - Supongamos que descomponemos en potencias de dos $p-1=2^k m$, con $k>0$ y m impar.
  - $a^m \, mod \, p, a^{2m} mod \, p, a^{2^2 m} mod \, p, \ldots, a^{2^k m} mod \, p.$
  - Observar que vamos elevando al cuadrado sucesivamente los residuos, para intentar encontrar una solución no trivial de la ecuación $x^2 \equiv 1 \, mod \, p$.
  - Si al elevar uno de ellos distinto de ± 1 al cuadrado obtenemos un 1, p es compuesto seguro.
- Por otro lado, observar que el último residuo si p es primo es 1 por TPF.
- Pero tiene que existir un 1 en alguno de los residuos anteriores, ya que si no existiese al elevarlo al cuadrado da 1 y por tanto es una raíz no trivial (p compuesto seguro), que no puede ser ya que estamos suponiendo p primo.

RSA: Generación de números primos grandes

- Vamos a parametrizar la secuencia de residuos como $a^{2^j m}$.
- Supongamos que descomponemos en potencias de dos $p - 1 = 2^k m$, con $k > 0$ y m impar (como ya dijimos).
- Si p fuese primo por TPF se cumple $a^{p-1} \equiv 1 \, mod \, p \Rightarrow a^{2^k m} \equiv 1 \, mod \, p$ (último residuo de la secuencia).
- Supongamos el j más pequeño que cumple $a^{2^j m} \equiv 1 \, mod \, p$.
- Aquí podemos distinguir dos casos:
  - Caso 1: $j = 0$

  $a^m \equiv 1 \, mod \, p$, es decir que p$|(a^m - 1)$
  - Caso 2: $1 \leq j \leq k$

  $a^{2^j m} \equiv 1 \, mod \, p \Rightarrow a^{2^j m} - 1 \equiv 0 \, mod \, p \Rightarrow$
  $(a^{2^{j-1} m} - 1)(a^{2^{j-1} m} + 1) \equiv 0 \, mod \, p$,
  esto significa que a la fuerza p$|(a^{2^{j-1} m} + 1)$ (ya que si no estamos en el caso 1).
  Por lo tanto $a^{2^{j-1} m} \equiv -1 \, mod \, p \Rightarrow a^{2^{j-1} m} \equiv (p - 1) \, mod \, p$.

RSA: Generación de números primos grandes

- Ya tenemos suficiente información para definir el test MR y enunciarlo:
- Miller-Rabin(p) /* Se pregunta por p si es compuesto */
  1. Elegimos un número aleatorio a.
  2. Expresamos $p-1=2^t m$, con $t>0$ y m impar.
  3. $$\boxed{a^m \bmod p}\begin{cases} 1 & no\ responde \\ p-1 & no\ responde \\ x & \boxed{elevamos\ al\ cuadrado} \end{cases}$$

$$\boxed{a^{2m} \bmod p}\begin{cases} 1 & 100\%\ compuesto \\ p-1 & no\ responde \\ x & \boxed{elevamos\ al\ cuadrado} \end{cases}$$

$$\boxed{\ldots\, t-1 \text{ veces } \ldots}$$

$$\boxed{a^{2^{(t-1)}m} \bmod p}\begin{cases} 1 & 100\%\ compuesto \\ p-1 & no\ responde \\ x & 100\%\ compuesto \end{cases}$$

RSA: Generación de números primos grandes

- **Ejemplo 1:**
  - $p = 13$ (primo)
  - $p - 1 = 12 = 2^2 * 3$
  - Supongamos $a = 4$

  $4^3 \ mod\ 13 = 12 = p - 1 \Rightarrow$ como no responde posible primo.
  - Supongamos $a = 2$

  $2^3 \ mod\ 13 = 8$

  $(2^3)^2 \ mod\ 13 = 12 = p - 1 \Rightarrow$ como no responde posible primo.
- **Ejemplo 2:**
  - $p = 221$ (compuesto)
  - $p - 1 = 220 = 2^2 * 55$
  - Supongamos $a = 5$

  $5^{55} \ mod\ 221 = 112$

  $(5^{55})^2 \ mod\ 221 = 168 \Rightarrow$ compuesto 100%.
  - Supongamos $a = 21$

  $21^{55} \ mod\ 221 = 200$

  $(21^{55})^2 \ mod\ 221 = 220 \Rightarrow$ como no responde posible primo (pero aquí se equivoca claramente).

RSA: Consideraciones prácticas de Miller-Rabin

- Generar un número aleatorio N grande de n bits deseados.
- Colocar un 1 en el primer bit y en el último bit (para que sea del tamaño deseable e impar).
- Dividir N por los 2000 primero primos. De esta forma se elimina de manera muy rápida el posible número N primo (el 99.8% de los número impares no primos son divisibles por algún primo de los 2000 primero.
- Ahora ejecutamos el algoritmo Miller-Rabin(N), explicado anteriormente.
- Si el test de primalidad falla (i.e. N es compuesto), incrementamos N en dos unidades y volvemos a ejecutar Miller-Rabin($N + 2$).
- ¿Cuántas veces ejecutamos Miller-Rabin de tal forma que no responda (posible primo)?
- Esto viene determinado por la probabilidad de equivocación de MR:
  - $P(N\ compuesto\ |MR\ responde\ m\ veces\ posible\ primo) \approx \frac{1}{1+\frac{4^m}{n ln2}}$.
  - Hay que escoger esta probabilidad lo más pequeña posible: así por ejemplo si queremos un número primo de 1024 bits, si MR nos dice posible primo 16 veces, la probabilidad de equivocarnos será $1{,}6526 \times 10^{-7}$.

RSA: Ataque de las Vegas al RSA

- Si por alguna razón se captura el exponente de descifrado d, uno pensaría que con generar otro d sería suficiente ya que no me ha capturado p y q.
- Pero no es así, hay un algoritmo que de d me lleva a p y q: Algoritmo de las Vegas.
- Es decir este algoritmo factoriza $n = pq$.
- La base matemática de este algoritmo es similar a la de Miller-Rabin.
- Intenta de igual forma resolver la ecuación: $x^2 \equiv 1\, mod\, n$, con $n = pq$.
- Así resolvemos el sistema $\begin{cases} x^2 \equiv 1\, mod\, p \\ x^2 \equiv 1\, mod\, q \end{cases}$

Si $p > 2 \Rightarrow \begin{cases} x \equiv 1\, mod\, p \\ x \equiv -1\, mod\, p \end{cases}$

Si $p > 2 \Rightarrow \begin{cases} x \equiv 1\, mod\, q \\ x \equiv -1\, mod\, q \end{cases}$

Como p y q primos solo existen las soluciones triviales.

RSA: Ataque de las Vegas al RSA

- Así el sistema $\begin{cases} x^2 \equiv 1 \mod p & (1) \\ x^2 \equiv 1 \mod q & (2) \end{cases}$

Si $p > 2 \Rightarrow \begin{cases} x \equiv 1 \mod p \\ x \equiv -1 \mod p \end{cases}$

Si $p > 2 \Rightarrow \begin{cases} x \equiv 1 \mod q \\ x \equiv -1 \mod q \end{cases}$

- Puede tener 4 posibles soluciones:
  - $x = 1$ en (1) y (2).
  - $x = -1$ en (1) y (2).
  - $\begin{cases} x \equiv 1 \mod p & (1) \\ x \equiv -1 \mod q & (2) \end{cases}$
  - $\begin{cases} x \equiv -1 \mod p & (1) \\ x \equiv 1 \mod q & (2) \end{cases}$

Soluciones triviales.

$$x = [q(q^{-1} \mod p) - p(p^{-1} \mod q)] \mod n$$

Soluciones NO triviales (El teorema del resto Chino nos aseguran que existen.

$$x = [-q(q^{-1} \mod p) + p(p^{-1} \mod q)] \mod n$$

RSA: Ataque de las Vegas al RSA

- Si obtenemos una **solución NO trivial** ($x \neq \pm 1 \; mod \; n$) de la ecuación $x^2 \equiv 1 \; mod \; n$, con $n = pq$, implica una serie de cosas muy interesantes:
  - n tiene que dividir a $(x+1)(x-1)$ a la vez.
  - No puede dividir a $(x+1)$ o $(x-1)$ individualmente porque, ya que si divide a uno de ellos estaríamos en el caso de las soluciones triviales (que no es nuestro caso):
    - Por ejemplo si $n \mid (x-1) \Rightarrow (x-1) \equiv 0 \; mod \; n \Rightarrow x \equiv 1 \; mod \; n$ (solución trivial).
  - Por tanto como n tiene que dividir a $(x+1)(x-1)$ a la vez, no hay otra solución que $p \mid (x+1)$ y $q \mid (x-1)$, o al contrario ($q \mid (x+1)$ y $p \mid (x-1)$):
    - $\begin{cases} mcd\,(\mathrm{n}, x+1) = p \text{ ó } q, \text{ó} \\ mcd\,(\mathrm{n}, x-1) = p \text{ ó } q. \end{cases}$ Esto se resuelve por el algoritmo de Euclides.

RSA: Ataque de las Vegas al RSA

- Ahora bien ¿Cómo se encuentran las raíces no triviales de la ecuación?
- Se utiliza el teorema generalizado de Euler que dice (TGE):
  - Si $a \in Z_n^*$ tal que n es no necesariamente primo $\Rightarrow a^{\varphi(n)} \equiv 1 \, mod \, n$.
- Sabemos que la definición del RSA es $ed \equiv 1 \, mod \, \varphi(n) \Rightarrow ed - 1 \equiv 0 \, mod \, \varphi(n) \Rightarrow ed - 1 = k\varphi(n)$.
  - Un número par es divisible por 2 y como $\varphi(n) = (p-1)(q-1)$, que es producto de dos pares es divisible por 2 también.
  - Así siempre se puede expresar $ed - 1 = k\varphi(n) = 2^t m$, con t> 0 y m impar.
- Así, como hemos capturado el exponente de descifrado d, podemos poner $\mathrm{e}d - 1 = 2^t m$, con $t > 0$ y m impar.
- Para $a \in Z_n^*$ tenemos que $a^{2^t m} mod \, n \equiv a^{ed-1} mod \, n \equiv a^{\varphi(n)k} mod \, n \equiv \left(a^{\varphi(n)}\right)^k mod \, n \equiv (1)^k mod \, n \equiv 1 \, mod \, n$.
- Es decir que si $a \in Z_n^* \Rightarrow a^{2^t m} \equiv 1 \, mod \, n$, que utilizaremos para calcular las soluciones NO triviales de la ecuación $x^2 \equiv 1 \, mod \, n$.

RSA: Ataque de las Vegas al RSA

- Ya tenemos suficiente información para definir el algoritmo de las Vegas y enunciarlo:
- Vegas(d) /* Calcula p y q para $n = pq$ */
  1. Expresamos $ed - 1 = 2^t m$, con t> 0 y m impar.
  2. Elegimos un número aleatorio w, con $1 < w < n - 1$.
  3. Si el $\mathrm{mcd}(w, n) = h > 1 \Rightarrow$ ó $p = h$, ó q $= h$.
  4. Expresamos ed$-1 = 2^t m$, con $k > 0$ y m impar.
  5. $$\boxed{w^m \; mod\; n} \begin{cases} 1 & no\ responde\ (FIN) \\ n-1 & no\ responde\ (FIN) \\ \boxed{x \quad elevamos\ al\ \ cuadrado} \end{cases}$$

$$\boxed{w^{2m} mod\; n} \begin{cases} 1 & mcd\ (\mathrm{n}, x+1) = p\ \text{ó}\ q \\ p-1 & no\ responde\ (FIN) \\ \boxed{x \quad elevamos\ al\ \ cuadrado} \end{cases}$$

$\boxed{\ldots t - 1 \text{ veces} \ldots}$

$$\boxed{w^{2^{(t-1)}m} mod\; n} \begin{cases} 1 & mcd\ (\mathrm{n}, x+1) = p\ \text{ó}\ q \\ p-1 & no\ responde\ (FIN) \\ x & mcd\ (\mathrm{n}, x+1) = p\ \text{ó}\ q \end{cases}$$

RSA: Generación de números primos grandes

- **Ejemplo:**
  - Supongamos $\mathrm{n} = 35$, y no sabemos p y q.
  - Se sabe que el exponente de cifrado $\mathrm{e} = 5$.
  - Se ha capturado el exponente de descifrado $\mathrm{d} = 5$.
  - ¿Qué vale p y q?
  - $ed - 1 = 2^t m = 24 = 2^3 * 3$
  - Supongamos $\mathrm{w} = 3$, y $mcd(35, 3) = 1$, continuamos (si hubiéramos escogido $\mathrm{w} = 10$, no continuábamos ya que $mcd(35, 10) = 5$).
  - Empezamos el cálculo de los cuadrados:
    - $w^m \; mod \; n \Rightarrow 3^3 \equiv 27 \; mod \; 35$
    - $(3^3)^2 \equiv 29 \; mod \; 35$
    - $((3^3)^2)^2 \equiv 1 \; mod \; 35 \Rightarrow$ Por lo tanto hemos encontrado una raíz no trivial ($x \equiv 29 \; mod \; 35$) y podemos factorizar:
      - $mcd\;(\mathrm{n}, x + 1) = p \text{ ó } q \Rightarrow mcd\;(35, 29 + 1) = p \text{ ó } q = 5 \Rightarrow q \text{ ó } p = 7.$

RSA: Probabilidad de equivocación de Miller-Rabin

- El error cometido por la prueba Miller-Rabin se mide por la probabilidad de que un número compuesto sea declarado probable primo.
- Como ya sabemos, cuantas más bases se prueben, mejor será la precisión de la prueba.
- Se puede demostrar que si N es compuesto, entonces como máximo 1⁄4 de las bases a no consiguen responder compuesto para N.
  - Rabin, Michael O. (1980), *"Probabilistic algorithm for testing primality", Journal of Number Theory,* ***12*** *(1): 128–138,* doi:10.1016/0022-314X(80)90084-0.
  - Schoof, René (2004), "Four primality testing algorithms", *Algorithmic Number Theory: Lattices, Number Fields, Curves and Cryptography*, Cambridge University Press, ISBN 978-0-521-80854-5.
- Supongamos estos dos eventos:
  - A: N impar compuesto.
  - B: Miller-Rabin responde m veces que N es posible primo.
- Como hemos dicho se sabe que $P(B|A) \leq \left(\frac{1}{4}\right)^m$.
- Pero la probabilidad que nos interesa es $P(A|B)$, que es la conocida probabilidad de equivocación del algoritmo de Miller-Rabin.
- Esta probabilidad la podemos calcular mediante el teorema de Bayes y la conjetura de Gauss (hoy en día teorema de los número primos):
  - Zagier, Don (1997). "Newman's short proof of the prime number theorem". *American Mathematical Monthly*. **104** (8): 705–708. doi:10.2307/2975232. JSTOR 2975232. MR 1476753.

RSA: Probabilidad de equivocación de Miller-Rabin

- Por el teorema de Bayes $\mathrm{P}(A|B) = \frac{P(B|A)P(A)}{P(B)}$
- $P(B) = P(B|A)P(A) + P(B|\bar{A})P(\bar{A})$, darse cuenta que el nuevo evento es $\bar{A}$: N primo.
- Por favor notar que: $P(A) = 1 - P(\bar{A})$.
- También darse cuenta que $P(B|\bar{A}) = 1$ y $P(B|A) \leq \left(\frac{1}{4}\right)^m$.
- $P(A)$ y $P(\bar{A})$ las podemos sacar mediante la conjetura de Gauss:
  - El número de primos que hay en el intervalo de $[0, N]$ (i.e. $\pi(N)$), es aproximadamente $\frac{N}{LnN}$, es decir $\lim_{N\to\infty} \frac{\pi(N)}{\frac{N}{LnN}} \to 1$.
- Supongamos que la representación de n bits para representación binaria del número que estamos probando si es primo, con total de 2^n números representados.
- El número de primos en el intervalo de $[0, 2^n]$ es aproximadamente $\frac{2^n}{Ln2^n} = \frac{2^n}{nLn2}$.
- El número de primos en el intervalo de $[0, 2^{n-1}]$ es aproximadamente $\frac{2^{n-1}}{Ln2^{n-1}} = \frac{2^{n-1}}{(n-1)Ln2}$.
- El número de primos de longitud exactamente n bits será igual al número de primos en el intervalo de $[0, 2^n]$ menos el número de primos en el intervalo de $[0, 2^{n-1}]$:
  - $\frac{2^n}{nLn2} - \frac{2^{n-1}}{(n-1)Ln2} = \frac{2^{n-1}}{Ln2}\left(\frac{2}{n} - \frac{1}{n-1}\right) \approx \frac{2^{n-1}}{nLn2}$ (cuando n>>1).

RSA: Probabilidad de equivocación de Miller-Rabin

- Así el número de primos de longitud exactamente n bits cuando $n \gg 1$ es: $\dfrac{2^{n-1}}{nLn2}$.

Casos favorables: Es el número de primos de longitud exactamente n bits

- Por tanto $P(\bar{A})$ que es la probabilidad de encontrar un primo de exactamente longitud n bits es
  - $P(\bar{A}) = \dfrac{\frac{2^{n-1}}{nLn2}}{2^{n-1}} = \dfrac{1}{nLn2}$.

Casos totales: Es 2^{n-1}porque un bit está a uno fijo por ser impar.

Multiplicando arriba y abajo por $nLn2$.

- Por tanto ya lo tenemos todo:
  - $\mathrm{P}(A|B) = \dfrac{P(B|A)P(A)}{P(B)} = \dfrac{P(B|A)P(A)}{P(B|A)P(A)+P(B|\bar{A})P(\bar{A})} = \dfrac{\left(\frac{1}{4}\right)^m (1-\frac{1}{nLn2})}{\left(\frac{1}{4}\right)^m (1-\frac{1}{nLn2})+\frac{1}{nLn2}} =$

$\dfrac{\left(\frac{1}{4}\right)^m (nLn2-1)}{\left(\frac{1}{4}\right)^m (nLn2-1)+nLn2-1} = \dfrac{1}{1+\frac{4^m}{nln2-1}} \approx \dfrac{1}{1+\frac{4^m}{nln2}}$ (cuando n>>1).

Dividiendo arriba y abajo por $\left(\frac{1}{4}\right)^m (nLn2-1)$.

RSA: Probabilidad de equivocación de Miller-Rabin

- Como consecuencia de la expresión de la probabilidad $\mathrm{P}(A|B)$:

$$P(N\ compuesto\ |MR\ responde\ m\ veces\ posible\ primo) \approx \frac{1}{1+\frac{4^m}{nln2}}$$

- Para un n fijo, cuando más grande sea m mayor probabilidad tenemos de que N sea primo.
- Para un m fijo, cuando más grande sea n menor probabilidad tenemos de que N sea primo.
- Así algunos ejemplos numéricos de esta probabilidad

$\mathrm{P}(A\|B)$	m	n
$0,0107$	8	1024
$1,6526 * 10^{-7}$	16	1024
$8,2630 * 10^{-8}$	16	512
$8,8477 * 10^{-17}$	32	1024

RSA: Inyectividad

- Para demostrar la inyectividad del RSA, necesitamos primero demostrar teorema pequeño de Fermat (TPF) y teorema generalizado de Euler (TGE):
- Teorema pequeño de Fermat (TPF): Si $a \in Z_p^*$ tal que p es primo $\Rightarrow a^{p-1} \equiv 1 \, mod \, p$:

1. Supongamos $Z_p^* = \{1, 2, \dots, p-1\} \Rightarrow aZ_p^* = \{1a, 2a, \dots, (p-1)a\}$.
2. Pero sabemos que $ax_i \neq ax_j$, siempre que i$\neq j$ debido a que $\mathrm{mcd}(a, p) = 1$.
3. Por tanto el conjunto $aZ_p^* = \pi(Z_p^*)$. Por lo tanto al multiplicar los elementos de cada conjunto aZ_p^* y Z_p^* obtenemos:
   $\prod_{i=1}^{p-1} i \, mod \, p = \prod_{i=1}^{p-1} ai \;\, mod \, p \Rightarrow (p-1)! \equiv a^{p-1}(p-1)! \;\, mod \, p$
4. Como $(p-1)! \in Z_p^* \Rightarrow \exists (p-1)!^{-1}$ y por tanto podemos poner:
   $a^{p-1} \equiv (p-1)!^{-1}\,(p-1)! \;\, mod \, p \Rightarrow a^{p-1} \equiv 1 \, mod \, p.$

RSA: Inyectividad

- Teorema Generalizado de Euler (TGE): Si $a \in Z_n^*$ tal que n no es necesariamente primo $\Rightarrow a^{\varphi(n)} \equiv 1\, mod\, n$:
  1. Supongamos $Z_n^* = \{b_1, b_2, \dots, b_{\varphi(n)}\} \Rightarrow aZ_n^* = \{ab_1, ab_2, \dots, ab_{\varphi(n)}\}$.
  2. Pero sabemos que $ab_i \neq ab_j$, siempre que i$\neq j$, por la razón que $a \in Z_n^*$ y se cumple $\mathrm{mcd}(a, n) = 1$.
  3. Por tanto el conjunto $aZ_n^* = \pi(Z_n^*)$. Por lo tanto al multiplicar los elementos de cada conjunto aZ_n^* y Z_n^* obtenemos:
     $\prod_{i=1}^{\varphi(n)} b_i\, mod\, n = \prod_{i=1}^{\varphi(n)} ab_i\ mod\, n \Rightarrow$
     $\prod_{i=1}^{\varphi(n)} b_i\, mod\, n = a^{\varphi(n)} \prod_{i=1}^{\varphi(n)} b_i\ mod\, n.$
  4. Por la propiedad de cierre $\mathrm{Z} = \prod_{i=1}^{\varphi(n)} b_i\, mod\, p \in Z_n^* \Rightarrow \exists\, Z^{-1}$ y por tanto podemos poner:
     $$a^{\varphi(n)} \equiv \left(\prod_{i=1}^{\varphi(n)} b_i\right)^{-1} \left(\prod_{i=1}^{\varphi(n)} b_i\right) mod\, n \Rightarrow a^{\varphi(n)} \equiv 1\, mod\, n.$$

RSA: Inyectividad (Paréntesis: inversa con TGE)

- Teorema Generalizado de Euler (TGE): Si $a \in Z_n^*$ tal que n no es necesariamente primo $\Rightarrow a^{\varphi(n)} \equiv 1 \; mod \; n$:
  - A través de este teorema podemos calcular el inverso multiplicativo de un número.
  - Solo hay que darse cuenta que $a^{\varphi(n)}$ lo podemos poner como $a^{\varphi(n)} = aa^{\varphi(n)-1}$.
  - Por tanto $a^{\varphi(n)} \equiv 1 \; mod \; n \Rightarrow aa^{\varphi(n)-1} \equiv 1 \; mod \; n \Rightarrow$

$$\boxed{a^{-1} \equiv a^{\varphi(n)-1} mod \; n, \text{ si } a \in Z_n^*.}$$

  - ¿Por qué no es efectivo este método para calcular el inverso multiplicativo de un número?

RSA: Inyectividad

- Ahora utilizando estos dos teoremas, TPF y TGE vamos a demostrar la inyectividad del RSA: $x^{ed} \equiv x \, mod \, n$.
- Vamos a distinguir dos casos.
- **CASO 1:** $x \in \ Z_n^*$, es decir $\mathrm{mcd}(x, n) = 1$, es decir x es coprimo con n, es decir el único número que divide a x y n es el 1.
  1. $ed \equiv 1 \, mod \, \varphi(n) \Rightarrow ed = 1 + k\varphi(n).$
  2. $x^{ed} \, mod \, n = x^{(1+k\varphi(n))} mod \, n = x x^{k\varphi(n)} mod \, n =$ $x(x^{\varphi(n)})^k mod \, n.$
  3. $x \in \ Z_n^* \Rightarrow$ por el TGE se cumple que $x^{\varphi(n)} \equiv 1 \, mod \, n.$
  4. $x(x^{\varphi(n)})^k mod \, n = x(1)^k mod \, n = x \, mod \, n \Rightarrow x^{ed} \equiv x \, mod \, n.$

RSA: Inyectividad

- **CASO 2:** $x \notin Z_n^*$, es decir $\mathrm{mcd}(x,n) > 1$, es decir x NO es coprimo con n, es decir existen dos números que pueden dividir a x y n: que son los primos p o q.
  1. Solo puede dividir uno de ellos, si dividieran los dos a la vez sería cuando $x = n$. Pero este caso no se puede dar ya que el mensaje para cifrar es siempre menor n.
  2. Supongamos que x es primo relativo con q (es decir $x \in Z_q^*$).
  3. Así tenemos $\mathrm{mcd}(x,q) = 1 \Rightarrow \mathrm{mcd}(\underbrace{xx \ldots x}_{p-1 \text{ veces}}, q)$=1.
  4. Así en particular x^{p-1} es coprimo con q.
  5. Por tanto por el TPF, como $x^{p-1} \in Z_q^* \Rightarrow (x^{p-1})^{q-1} \equiv 1 \bmod q \Rightarrow \boxed{x^{\varphi(n)} \equiv 1 \bmod q}$.
  6. Por otro lado, la definición de RSA nos dice $ed \equiv 1 \bmod \varphi(n) \Rightarrow ed = 1 + k\varphi(n)$.
  7. $x^{ed} \bmod q = x^{(1+k\varphi(n))} \bmod q = xx^{k\varphi(n)} \bmod q = x(x^{\varphi(n)})^k \bmod q = x \bmod q$.
  8. Por tanto tenemos $x^{ed} \equiv x \bmod q \Rightarrow x^{ed} - x = k_1 q$.
  9. Como hemos supuesto que x es primo relativo con q (es decir $x \in Z_q^*$), entonces el factor p es el que divide a x, y en particular $x^{ed} - x = k_2 p$.
  10. Combinando los dos resultados anteriores $\begin{cases} x^{ed} - x = k_1 q \\ x^{ed} - x = k_2 p \end{cases} \Rightarrow x^{ed} \equiv x \bmod n$.

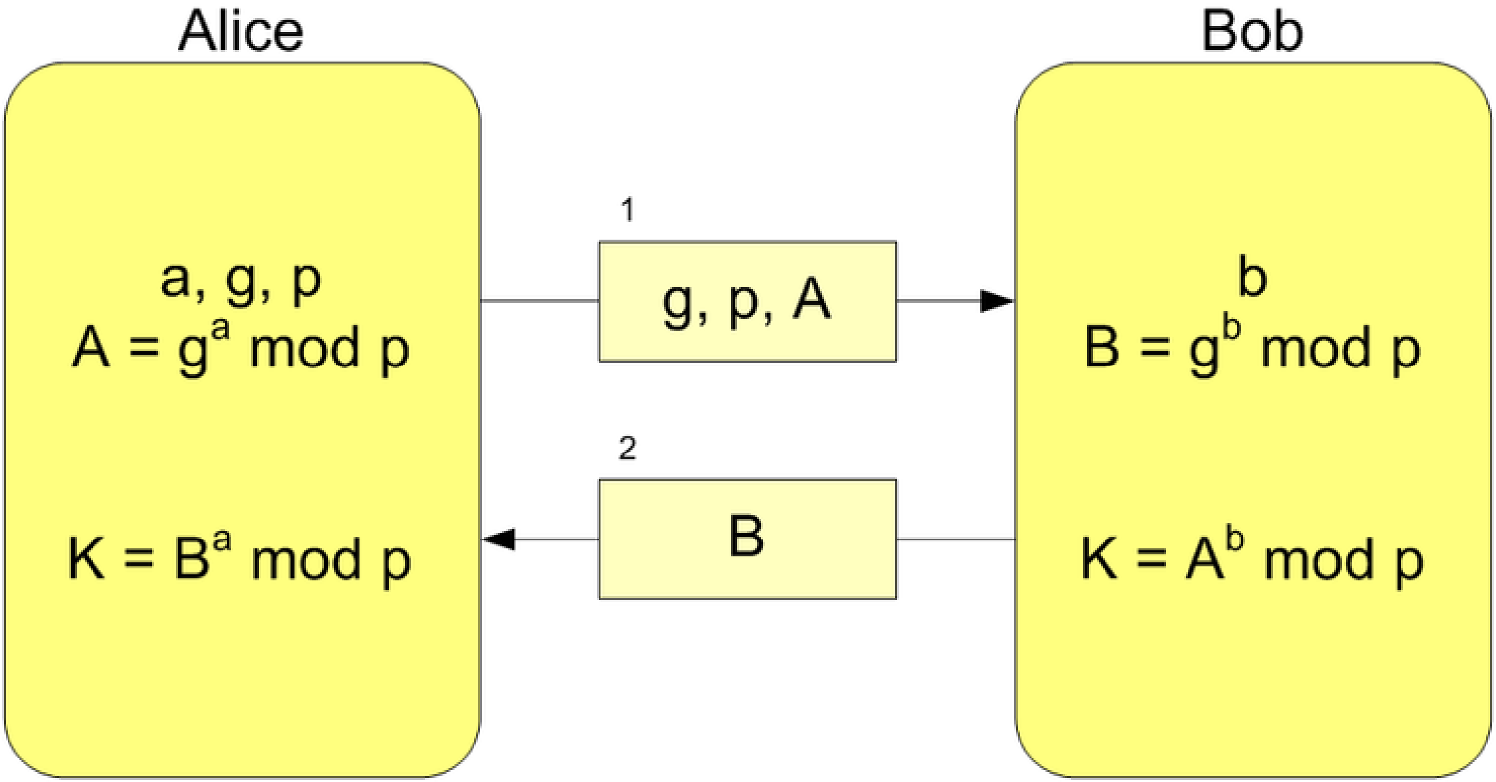
$\boxed{\varphi(n) = (p-1)(q-1)}$

Cifrados Asimétricos: Intercambio de Claves Diffie-Hellman

- Podemos definir los logaritmos discretos como sigue:
- Una raíz primitiva ***a*** de un número primo ***p*** es aquella que cuyas potencias en módulo p generan **todos** los números enteros de 1 a p-1.
  - Es decir, si a es una raíz primitiva de p todos entonces estos números son distintos:
    - a mod p, a^2 mod p,. . . , a^{p-1} mod p (i.e. el resultado es una permutación del conjunto {1…p-1})
- Para cualquier número entero **b** y una raíz primitiva **a** de un número primo **p**, podemos encontrar un exponente único **i** tal que $\mathbf{b = a^i}$ **(mod p)**, donde el exponente se encuentra en $0 \leq i \leq (p - 1)$.
- El exponente **i** se conoce como el logaritmo discreto de **b** para la base **a**, en aritmética **mod p.**
- Expresamos este valor como $\mathbf{i=dlog_{a,\,p}(b)}$.
- Ya sabemos que el calculo del logaritmo discreto es computacionalmente inviable, al igual que pasaba en RSA. En esta computación inviable se basa el intercambio de claves DH.

Cifrados Asimétricos: Intercambio de Claves Diffie-Hellman

Pensar para casa que problemas puede tener este esquema de intercambio de claves en la seguridad del protocolo.



$$K = A^b \bmod p = (g^a \bmod p)^b \bmod p = g^{ab} \bmod p = (g^b \bmod p)^a \bmod p = B^a \bmod p$$

Imagen extraída de https://es.wikipedia.org/wiki/Diffie-Hellman

Cifrados Asimétricos: Intercambio de Claves Diffie-Hellman

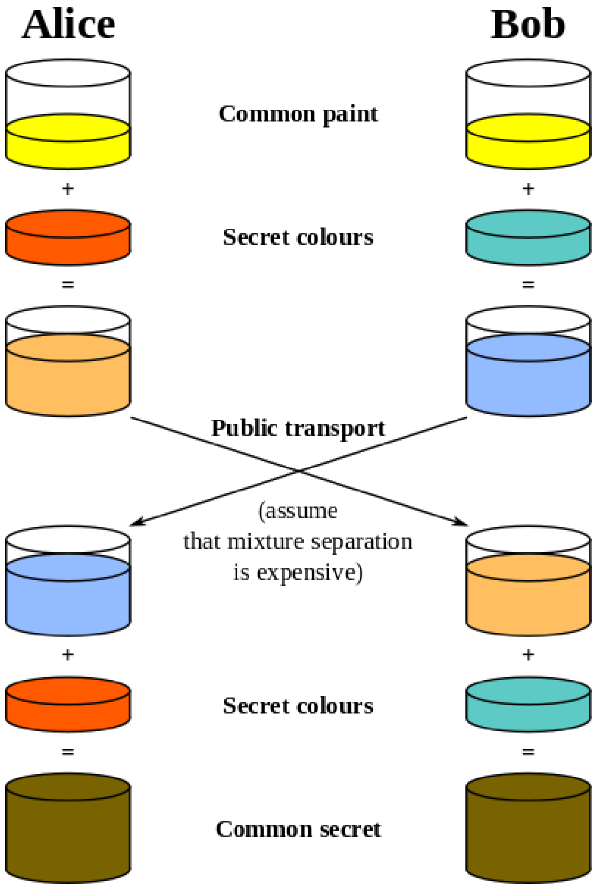


Imagen extraída de http://en.wikipedia.org/wiki/File:Diffie-Hellman_Key_Exchange.svg

- Alice y Bob acuerdan el primo p = 23 y la g = 5 base (es una raíz primitiva del numero primo p) .
- Alice elige un secreto **a = 6**, y envía a Bob $A = g^a \bmod p$
  - $A = 5^6 \bmod 23$
  - $A = \mathbf{15{,}625} \bmod 23$
  - $A = 8$
- Bob elige un secreto **b = 15**, y envía a Alice $B = g^b \bmod p$
  - $B = 5^{15} \bmod 23$
  - $B = \mathbf{30{,}517{,}578{,}125} \bmod 23$
  - $B = 19$
- Alice calcula el secreto compartido $s = B^a \bmod p$
  - $s = 19^6 \bmod 23$
  - $s = \mathbf{47{,}045{,}881} \bmod 23$
  - $s = 2$
- Bob calcula el secreto compartido $s = A^b \bmod p$
  - $s = 8^{15} \bmod 23$
  - $s = \mathbf{35{,}184{,}372{,}088{,}832} \bmod 23$
  - $s = 2$
- Alice y Bob ahora comparten el secreto **s = 2.**

Debilidades del RSA

- Generalmente existen tres tipos de ataques del RSA:
  - Fuerza bruta: probando todas las posibles claves privadas.
  - Ataques matemáticos: con diferentes enfoques pero siempre equivalentes a factorizar números primos.
  - Ataques de tiempo: dependen del tiempo de ejecución del cifrado.
    - Kocher, P.C. (1996). Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In: Koblitz, N. (eds) Advances in Cryptology — CRYPTO '96. CRYPTO 1996. Lecture Notes in Computer Science, vol 1109. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-68697-5_9
- Artículo clásico de Dan Boneh sobre Ataques y debilidades de RSA (Dan Boneh. Twenty Years of Attacks on the RSA Cryptosystem, 1999. NOTICES OF THE AMS, vol 46, pp 203-213):
  - http://crypto.stanford.edu/~dabo/papers/RSA-survey.pdf.
- Material interesante:
  - https://www.ellaberintodefalken.com/2014/09/ataques-debilidades-rsa.html.

Debilidades del RSA

- Respecto los ataques matemáticos vamos a ver 4 ataques que son los típicos y más sencillos:
  1. Conocimiento de $\varphi(n)$, resolución de una ecuación de segundo grado.
  2. Conocimiento del exponente de descifrado d (algoritmo de las Vegas, ya lo vimos en transparencias anteriores).
  3. Ataque por módulo del RSA común:
     - Se puede averiguar el mensaje cifrado, sin factorizar $\mathrm{n} = pq$.
  4. Ataque de exponente común y pequeño (mediante coincidencia de exponentes de cifrado):
     - Se utiliza el teorema del resto chino.
     - Se puede averiguar el mensaje cifrado, sin factorizar $\mathrm{n} = pq$.

Debilidades del RSA: Conocimiento de $\varphi(n)$

- Recordemos que la clave pública del RSA es (n, e), pero si se conoce por alguna razón $\varphi(n)$ es lo mismo que factorizar $\mathrm{n} = pq$.
- $\varphi(n) = \varphi(pq) = (p-1)(q-1) = pq - p - q + 1.$
- $\mathrm{n} = pq \Rightarrow q = \frac{n}{p}.$
- $\varphi(n) = p\frac{n}{p} - p - \frac{n}{p} + 1 \Rightarrow$

$$\mathrm{p}\varphi(n) = pn - p^2 - n + p \Rightarrow$$

$$\boxed{p^2 + \mathrm{p}(\varphi(n) - n - 1) + n = 0.}$$

- Por tanto resolvemos esta ecuación para p y luego sacamos q.

Debilidades del RSA: Conocimiento de $\varphi(n)$

- Ejemplo: Supongamos se ha capturado $\varphi(n) = 4382136$ por alguna razón y que $n = 4386607$ (es público).
- Teniendo en cuenta el razonamiento anterior tenemos $\Rightarrow$

$$p^2 + \mathrm{p}(4382136 - 4386607 - 1) + 4386607 = 0 \Rightarrow$$
$$p^2 - \mathrm{p}\ 4472 + 4386607 = 0 \Rightarrow$$
$$p = \frac{4472 \pm \sqrt{4472^2 - 4 * 4386607}}{2} = \begin{cases} 3019 \\ 1453 \end{cases}$$

$$\boxed{\mathrm{n} = 3019 * 1453 = 4386607.}$$

Debilidades del RSA: Módulo del RSA común

- Imaginemos que se generan varias claves públicas y privadas con un mismo $\mathrm{n} = pq$, para varios usuarios "*Bob*", y además "*Alice*" quiere enviar el mismo mensaje M para esos usuarios.
  1. Así por ejemplo se crean dos exponentes para los usuarios "*Bob*", e_1 y e_2, que satisfacen que el $\mathrm{mcd}\big(e_1, \varphi(n)\big) = 1$ y $\mathrm{mcd}\big(e_2, \varphi(n)\big) = 1$.
  2. Ahora "*Alice*" con sus claves públicas envía el mensaje M cifrado para los dos usuarios:

$$\begin{cases} y_1 = M^{e_1} \, mod \, n \\ y_2 = M^{e_2} \, mod \, n \end{cases}.$$

- Esto tiene un agujero de seguridad en ciertas situaciones y se puede averiguar M sin necesidad de factorizar $n = pq$.

Debilidades del RSA: Módulo del RSA común

- En el caso que e_1 y e_2 sean coprimos entre sí (que es bastante probable): $\mathrm{mcd}(e_1, e_2) = 1$, se puede averiguar M.
- Recordemos en el algoritmo de Euclides podemos poner todos los restos r_i en función solo de r_0 y r_1 , de manera recursiva obteniendo así la expresión del tipo para el último resto $r_n = 1$, ya que que es $\mathrm{mcd}(e_1, e_2) = 1$, del tipo:
  - $\boxed{r_n = 1 = e_1\, u_n + e_2\, v_n}$ (Algoritmo extendido de Euclides).
- En general tenemos que para cualquier pareja de exponentes que sean coprimos entre ellos existe una expresión del tipo:
  - $e_1 u + e_2 v = 1$, donde u y v, se calculan muy rápido mediante las recurrencias del algoritmo extendido de Euclides.
- En este contexto el mensaje M se calcula como:
  - $\boxed{M = {y_1}^u * {y_2}^v \ mod\ n.}$ ¿Por qué? $\Rightarrow$

$$\begin{cases} y_1 = M^{e_1}\ mod\ n \\ y_2 = M^{e_2}\ mod\ n \end{cases}$$

$$\boxed{{y_1}^u * {y_2}^v = (M^{e_1})^u * (M^{e_2})^v = M^{(e_1 u + e_2 v)} = M}$$

$$\boxed{e_1 u + e_2 v = 1}$$

Debilidades del RSA: Ataque de exponente común y pequeño

- Para este tipo de ataque necesitamos el Teorema del Resto Chino (TRC).
- Supongamos el siguiente sistema de ecuaciones de congruencias:
  - $\begin{cases} x \equiv a_1 \, mod \, m_1 \\ x \equiv a_2 \, mod \, m_2 \\ \quad \ldots \\ x \equiv a_n \, mod \, m_n \end{cases}$, siendo los m_i primos entre sí: $\mathrm{mcd}(m_1, m_2, \ldots, m_n) = 1 \Rightarrow$

    $\exists$ una solución que es: $x = \sum_{i=1}^{n} a_i c_i \; mod \, M$, con

    $M = m_1 * m_2 * \cdots * m_n$

    $M_i = \frac{M}{m_i} \Rightarrow y_i = {M_i}^{-1} \, mod \, m_i$

    $c_i = y_i * M_i$ sin aplicar ningún módulo.
  - Para probar el TRC, solo hay que sustituir la solución en el sistema de congruencias.

Debilidades del RSA: Ataque de exponente común y pequeño

- El exponente de descifrado e se suele escoger pequeño para que el cifrado sea más efectivo computacionalmente.
- Así puede ser que e coincida en las claves para varios usuarios.
- Si esto pasa se puede averiguar el mensaje cifrado, sin factorizar $n = pq$.
- Supongamos que "*Alice*" manda el mismo mensaje P a diferentes "*Bob*", cuyos exponentes de descifrado e son iguales, con módulos de RSA $m_1, m_2, \dots, m_n$:
- $\begin{cases} x = P^e mod\ m_1 \equiv a_1\ mod\ m_1 \\ x = P^e mod\ m_2 \equiv a_2\ mod\ m_2 \\ \qquad\qquad \dots \\ x = P^e mod\ m_n \equiv a_n\ mod\ m_n \end{cases}$, si los m_i primos entre sí: $\mathrm{mcd}(m_1, m_2, \dots, m_n) = 1 \Rightarrow$

  por el TRC ∃ una solución que es: $x = \sum_{i=1}^{n} a_i c_i\ \ mod\ M,$ con

  $M = m_1 * m_2 * \cdots * m_n$

  $M_i = \dfrac{M}{m_i} \Rightarrow y_i = {M_i}^{-1}\ mod\ m_i \qquad \Rightarrow \boxed{x = P^e \Rightarrow P = \sqrt[e]{x}.}$

  $c_i = y_i * M_i$ sin aplicar ningún módulo.

Debilidades del RSA: Ataque de exponente común y pequeño

- Ejemplo: Supongamos que hay tres receptores: $B_1 \Rightarrow (e = 3,\, m_1 = 46)$, $B_2 \Rightarrow (e = 3,\, m_2 = 51)$, y $B_3 \Rightarrow (e = 3,\, m_2 = 55)$.
- Hemos interceptado los mensajes cifrados que han llegado a los 3 receptores cifrados por "*Alice*": $B_1 \Rightarrow (x = 34), B_2 \Rightarrow (x = 31), B_1 \Rightarrow (x = 10)$.
- $\begin{cases} x \equiv 34\ mod\ 46 \\ x \equiv 31\ mod\ 51 \\ x \equiv 10\ mod\ 55 \end{cases}$, como $\mathrm{mcd}(46, 54, 55) = 1 \Rightarrow$

por el TRC ∃ una solución que es: $x = \sum_{i=1}^{n} a_i c_i \;\; mod\ M$, con

$M = 46 * 51 * 55 = 129030$

$$M_1 = \frac{M}{m_1} = 51 * 55 = 2805 \Rightarrow y_1 = {M_1}^{-1}\ mod\ m_1 = 45 \Rightarrow c_1 = \ y_1 * M_1 = 45 * 2805 = 126225$$

$$M_2 = \frac{M}{m_2} = 46 * 55 = 2530 \Rightarrow y_2 = {M_2}^{-1}\ mod\ m_2 = 28 \Rightarrow c_1 = \ y_1 * M_1 = 28 * 2530 = 70840$$

$$M_3 = \frac{M}{m_3} = 46 * 51 = 2346 \Rightarrow y_3 = {M_3}^{-1}\ mod\ m_3 = 26 \Rightarrow c_1 = \ y_1 * M_1 = 26 * 2346 = 60996$$

$$x = \sum_{i=1}^{n} a_1 c_i \;\; mod\ M = (34 * 126225 + 35 * 70840 + 10 * 60996)\ mod\ \ 129030$$

$$x \equiv 1000\ mod\ \ 129030 \Rightarrow x = P^e \Rightarrow P = \sqrt[e]{x} \Rightarrow 1000 = P^3 \Rightarrow \boxed{P = \sqrt[3]{1000} = 10.}$$