MOOC de Criptología Matemática. El Sistema RSA

Leandro Marín

Módulo II. Sesión 4.
Dificultad Alta

1 Sistemas de Clave Pública

2 La Exponenciación Modular

3 El método RSA

Clave Privada y Clave Pública I

- Ya hemos visto los sistemas de cifrado como el AES o DES en los cuales hay una única clave de cifrado y descifrado.

Clave Privada y Clave Pública I

- Ya hemos visto los sistemas de cifrado como el AES o DES en los cuales hay una única clave de cifrado y descifrado.
- Estos sistemas son muy eficientes y válidos cuando entre en emisor y el receptor hay un acuerdo previo sobre cual es la clave que se va a utilizar.

Clave Privada y Clave Pública I

- Ya hemos visto los sistemas de cifrado como el AES o DES en los cuales hay una única clave de cifrado y descifrado.
- Estos sistemas son muy eficientes y válidos cuando entre en emisor y el receptor hay un acuerdo previo sobre cual es la clave que se va a utilizar.
- Sin embargo no siempre es esa la situación, en muchas ocasiones es necesario iniciar una comunicación segura entre dos interlocutores que no han compartido nunca una clave.

Clave Privada y Clave Pública I

- Ya hemos visto los sistemas de cifrado como el AES o DES en los cuales hay una única clave de cifrado y descifrado.
- Estos sistemas son muy eficientes y válidos cuando entre en emisor y el receptor hay un acuerdo previo sobre cual es la clave que se va a utilizar.
- Sin embargo no siempre es esa la situación, en muchas ocasiones es necesario iniciar una comunicación segura entre dos interlocutores que no han compartido nunca una clave.
- Esto se puede hacer creando dos claves separadas, una clave de cifrado que haremos pública y que permitirá a cualquier usuario enviarnos mensajes de forma segura y una clave privada que mantendremos en secreto y que utilizaremos para descifrar los mensajes recibidos.

Clave Privada y Clave Pública II

- Si alguien intercepta el mensaje, al no disponer de la clave privada sólo obtendrá una información inútil.

Clave Privada y Clave Pública II

- Si alguien intercepta el mensaje, al no disponer de la clave privada sólo obtendrá una información inútil.
- Evidentemente las claves públicas y privadas han de estar relacionadas, la cuestión es que alguien que disponga de la clave pública no debe poder obtener la privada (en un tiempo razonable).

Clave Privada y Clave Pública II

- Si alguien intercepta el mensaje, al no disponer de la clave privada sólo obtendrá una información inútil.
- Evidentemente las claves públicas y privadas han de estar relacionadas, la cuestión es que alguien que disponga de la clave pública no debe poder obtener la privada (en un tiempo razonable).
- Para eso se determinarán los tamaños de claves adecuados que permitan garantizar este hecho.

El RSA

- El sistema RSA es probablemente el más famoso y utilizado de los sistemas de clave pública, aunque no el único.

El RSA

- El sistema RSA es probablemente el más famoso y utilizado de los sistemas de clave pública, aunque no el único.
- Su nombre proviene de Rivest, Shamir y Adleman, tres investigadores del Instituto de Tecnología de Massachusetts que publicaron el algoritmo en 1977.

El RSA

- El sistema RSA es probablemente el más famoso y utilizado de los sistemas de clave pública, aunque no el único.
- Su nombre proviene de Rivest, Shamir y Adleman, tres investigadores del Instituto de Tecnología de Massachusetts que publicaron el algoritmo en 1977.
- La base del sistema es un cálculo mateático relativamente sencillo, pero aplicado a números muy grandes.

El RSA

- El sistema RSA es probablemente el más famoso y utilizado de los sistemas de clave pública, aunque no el único.
- Su nombre proviene de Rivest, Shamir y Adleman, tres investigadores del Instituto de Tecnología de Massachusetts que publicaron el algoritmo en 1977.
- La base del sistema es un cálculo mateático relativamente sencillo, pero aplicado a números muy grandes.
- Este método supuso una revolución en el mundo de la criptografía ampliando el abanico de protocolos que se podían desarrollar.

Planteamiento

- La base del algoritmo RSA es el cálculo de $x^k (mod\ n)$ para números x, k y n muy grandes (más de 1000 cifras binarias).

Planteamiento

- La base del algoritmo RSA es el cálculo de $x^k (mod\ n)$ para números x, k y n muy grandes (más de 1000 cifras binarias).
- Según los cálculos más optimistas la antigüedad del universo es de 13.835 millones de años, que medido en segundos no supera los 2^{60}.

Planteamiento

- La base del algoritmo RSA es el cálculo de $x^k (mod\ n)$ para números x, k y n muy grandes (más de 1000 cifras binarias).
- Según los cálculos más optimistas la antigüedad del universo es de 13.835 millones de años, que medido en segundos no supera los 2^{60}.
- Los cálculos del número de átomos del universo dicen que podrían estar en torno a los 10^{82}, número menor que 2^{120}.

Planteamiento

- La base del algoritmo RSA es el cálculo de $x^k (mod\ n)$ para números x, k y n muy grandes (más de 1000 cifras binarias).
- Según los cálculos más optimistas la antigüedad del universo es de 13.835 millones de años, que medido en segundos no supera los 2^{60}.
- Los cálculos del número de átomos del universo dicen que podrían estar en torno a los 10^{82}, número menor que 2^{120}.
- Ni aún en el caso de que cada átomo del universo pudiera hacer una múltiplicación por segundo a lo largo de toda la vida del universo, no llegaríamos a alcanzar las 2^{180} multiplicaciones, número insignificante frente a los 2^{1000} que necesitaríamos para calcular $x^k (mod\ n)$ mediante este método y un número k de 1000 cifras binarias.

Planteamiento

- La base del algoritmo RSA es el cálculo de $x^k (mod\ n)$ para números x, k y n muy grandes (más de 1000 cifras binarias).
- Según los cálculos más optimistas la antigüedad del universo es de 13.835 millones de años, que medido en segundos no supera los 2^{60}.
- Los cálculos del número de átomos del universo dicen que podrían estar en torno a los 10^{82}, número menor que 2^{120}.
- Ni aún en el caso de que cada átomo del universo pudiera hacer una múltiplicación por segundo a lo largo de toda la vida del universo, no llegaríamos a alcanzar las 2^{180} multiplicaciones, número insignificante frente a los 2^{1000} que necesitaríamos para calcular $x^k (mod\ n)$ mediante este método y un número k de 1000 cifras binarias.
- Es evidente que no podemos multiplicar $x \cdot x \cdot x \cdots$ un número de veces k si k es un número de 1000 cifras binarias, debemos buscar otra manera.

Resolución del Problema I

- Para calcular $x^k (mod\ n)$ lo primero que necesitamos son las cifras binarias del exponente k.

Resolución del Problema I

- Para calcular $x^k (mod\ n)$ lo primero que necesitamos son las cifras binarias del exponente k.
- Para ello haremos divisiones por dos y poniendo a la derecha el resto de la división.

Resolución del Problema I

- Para calcular $x^k (mod\ n)$ lo primero que necesitamos son las cifras binarias del exponente k.
- Para ello haremos divisiones por dos y poniendo a la derecha el resto de la división.
- Este proceso lo haremos hasta que lleguemos a 1. Supongamos por ejemplo $k = 277$, la tabla obtenida sería la siguiente:

paso	k	k_i
0	277	1
1	138	0
2	69	1
3	34	0
4	17	1
5	8	0
6	4	0
7	2	0
8	1	1

$$277 = 2^0 + 2^2 + 2^4 + 2^8$$

Resolución del Problema II

- Notemos ahora que las potencias x^{2^i} se puden calcular de forma recursiva el siguiente modo.

Resolución del Problema II

- Notemos ahora que las potencias x^{2^i} se puden calcular de forma recursiva el siguiente modo.
- Para $i = 0$, $x^{2^0} = x$, $x^{2^1} = x^2$, ..., si tenemos x^{2^i} calculado, elevándolo al cuadrado obtenemos $x^{2^i} \cdot x^{2^i} = x^{2 \cdot 2^i} = x^{2^{i+1}}$.

Resolución del Problema II

- Notemos ahora que las potencias x^{2^i} se puden calcular de forma recursiva el siguiente modo.
- Para $i = 0$, $x^{2^0} = x$, $x^{2^1} = x^2$, ..., si tenemos x^{2^i} calculado, elevándolo al cuadrado obtenemos $x^{2^i} \cdot x^{2^i} = x^{2 \cdot 2^i} = x^{2^{i+1}}$.
- La forma más sencilla es utilizar una tabla, que haremos tan larga como cifras binarias tenga k y en la iremos elevando al cuadrado el paso anterior. Esta operación se realizará en aritmética modular.

Resolución del Problema II

- Notemos ahora que las potencias x^{2^i} se puden calcular de forma recursiva el siguiente modo.
- Para $i = 0$, $x^{2^0} = x$, $x^{2^1} = x^2$, ..., si tenemos x^{2^i} calculado, elevándolo al cuadrado obtenemos $x^{2^i} \cdot x^{2^i} = x^{2 \cdot 2^i} = x^{2^{i+1}}$.
- La forma más sencilla es utilizar una tabla, que haremos tan larga como cifras binarias tenga k y en la iremos elevando al cuadrado el paso anterior. Esta operación se realizará en aritmética modular.
- Vamos a calcularlo para $x = 165$ módulo $n = 493$.

paso	k	k_i	x^{2^i}
0	277	1	$165 = x$
1	138	0	$110 = 165^2 (mod\ 493)$
2	69	1	$268 = 110^2 (mod\ 493)$
3	34	0	$339 = 268^2 (mod\ 493)$
4	17	1	$52 = 339^2 (mod\ 493)$
5	8	0	$239 = 52^2 (mod\ 493)$
6	4	0	$426 = 239^2 (mod\ 493)$
7	2	0	$52 = 426^2 (mod\ 493)$
8	1	1	$239 = 52^2 (mod\ 493)$

Resolución del Problema III

- Vamos a calcular una nueva columna con los valores $x^{k_i 2^i}$.

Resolución del Problema III

- Vamos a calcular una nueva columna con los valores $x^{k_i 2^i}$.
- Tenemos dos casos:

Resolución del Problema III

- Vamos a calcular una nueva columna con los valores $x^{k_i 2^i}$.
- Tenemos dos casos:
  1. Si $k_i = 1$ entonces $x^{k_i 2^i} = x^{2^i}$ y copiaremos la columna anterior.

Resolución del Problema III

- Vamos a calcular una nueva columna con los valores $x^{k_i 2^i}$.
- Tenemos dos casos:
  1. Si $k_i = 1$ entonces $x^{k_i 2^i} = x^{2^i}$ y copiaremos la columna anterior.
  2. Si $k_i = 0$ entonces $x^{k_i 2^i} = x^0 = 1$ y dejaremos el espacio en blanco que entenderemos como 1.

Resolución del Problema III

- Vamos a calcular una nueva columna con los valores $x^{k_i 2^i}$.
- Tenemos dos casos:
  1. Si $k_i = 1$ entonces $x^{k_i 2^i} = x^{2^i}$ y copiaremos la columna anterior.
  2. Si $k_i = 0$ entonces $x^{k_i 2^i} = x^0 = 1$ y dejaremos el espacio en blanco que entenderemos como 1.
- La solución final vendrá por el producto modular de todos los elementos que nos han quedado en la nueva columna (por eso el valor 1 lo hemos eliminado, porque multiplicar por 1 no altera el resultado).

Resolución del Problema III

- Vamos a calcular una nueva columna con los valores $x^{k_i 2^i}$.
- Tenemos dos casos:
  1. Si $k_i = 1$ entonces $x^{k_i 2^i} = x^{2^i}$ y copiaremos la columna anterior.
  2. Si $k_i = 0$ entonces $x^{k_i 2^i} = x^0 = 1$ y dejaremos el espacio en blanco que entenderemos como 1.
- La solución final vendrá por el producto modular de todos los elementos que nos han quedado en la nueva columna (por eso el valor 1 lo hemos eliminado, porque multiplicar por 1 no altera el resultado).
- Vamos a calcularlo para $k = 277$, $x = 165$ módulo $n = 493$.

paso	k	k_i	x^{2^i}	$x^{k_i 2^i}$
0	277	1	165	
1	138	0	110	110
2	69	1	268	
3	34	0	339	339
4	17	1	52	
5	8	0	239	239
6	4	0	426	426
7	2	0	52	52
8	1	1	239	

$$165^{277}(mod\ 493) = 110 \cdot 339 \cdot 239 \cdot 426 \cdot 52 = 326.$$

Demostración del Método

- Lo que hemos dado como solución final es el producto

$$x^{k_0 2^0} \cdot x^{k_1 2^1} \cdot x^{k_2 2^2} \cdots x^{k_{t-1} 2^{t-1}} \cdot x^{k_t 2^t}$$

Demostración del Método

- Lo que hemos dado como solución final es el producto

$$x^{k_0 2^0} \cdot x^{k_1 2^1} \cdot x^{k_2 2^2} \cdots x^{k_{t-1} 2^{t-1}} \cdot x^{k_t 2^t}$$

- Este es un producto de potencias con la misma base, por lo que es igual a la base elevada a la suma de los exponentes:

$$x^{k_0 2^0 + k_1 2^1 + k_2 2^2 + \cdots + k_{t-1} 2^{t-1} + k_t 2^t} = x^k$$

Demostración del Método

- Lo que hemos dado como solución final es el producto

$$x^{k_0 2^0} \cdot x^{k_1 2^1} \cdot x^{k_2 2^2} \cdots x^{k_{t-1} 2^{t-1}} \cdot x^{k_t 2^t}$$

- Este es un producto de potencias con la misma base, por lo que es igual a la base elevada a la suma de los exponentes:

$$x^{k_0 2^0 + k_1 2^1 + k_2 2^2 + \cdots + k_{t-1} 2^{t-1} + k_t 2^t} = x^k$$

- Pero el exponente que nos ha quedado es precisamente k por ser los valores k_i sus cifras en binario.

Implementación en sage

- Para hacer la implementación no es necesario almacenar todos los valores interemedios.

Implementación en sage

- Para hacer la implementación no es necesario almacenar todos los valores interemedios.
- Lo que haremos es utilizar una variable `acumulador` en la que multiplicar los valores $x^{k_i 2^i}$ según vayan siendo calculados.

Implementación en sage

- Para hacer la implementación no es necesario almacenar todos los valores interemedios.
- Lo que haremos es utilizar una variable `acumulador` en la que multiplicar los valores $x^{k_i 2^i}$ según vayan siendo calculados.
- El código podría ser el siguiente:

```
def expmod(x,k,n):
  acumulador = 1
  while k!=0:
    if k%2==1:
      acumulador = (acumulador * x) % n
    x = (x*x)%n
    k = k//2
  return acumulador
```

Generación de Claves

- Una clave RSA consiste en tres valores (e, d, n) de los cuales e y n son públicos y d es privado.

Generación de Claves

- Una clave RSA consiste en tres valores (e, d, n) de los cuales e y n son públicos y d es privado.
- El número n será un producto de dos primos grandes distintos, aproximadamente del mismo número de cifras y n debe tener un tamaño de 1024, aunque actualmente se considera adecuado ir ajustando los desarrollos a 2048 bits.

Generación de Claves

- Una clave RSA consiste en tres valores (e, d, n) de los cuales e y n son públicos y d es privado.
- El número n será un producto de dos primos grandes distintos, aproximadamente del mismo número de cifras y n debe tener un tamaño de 1024, aunque actualmente se considera adecuado ir ajustando los desarrollos a 2048 bits.
- Si p y q son los números primos que componen n, sabemos que $\varphi(n) = (p-1)(q-1)$.

Generación de Claves

- Una clave RSA consiste en tres valores (e, d, n) de los cuales e y n son públicos y d es privado.
- El número n será un producto de dos primos grandes distintos, aproximadamente del mismo número de cifras y n debe tener un tamaño de 1024, aunque actualmente se considera adecuado ir ajustando los desarrollos a 2048 bits.
- Si p y q son los números primos que componen n, sabemos que $\varphi(n) = (p-1)(q-1)$.
- El valor de e debe ser un número coprimo con $\varphi(n)$. Es habitual tomar como valor $e = 65537$ puesto que se trata de un número primo. Si no resultase coprimo con $\varphi(n)$ se cambiaría n.

Generación de Claves

- Una clave RSA consiste en tres valores (e, d, n) de los cuales e y n son públicos y d es privado.
- El número n será un producto de dos primos grandes distintos, aproximadamente del mismo número de cifras y n debe tener un tamaño de 1024, aunque actualmente se considera adecuado ir ajustando los desarrollos a 2048 bits.
- Si p y q son los números primos que componen n, sabemos que $\varphi(n) = (p-1)(q-1)$.
- El valor de e debe ser un número coprimo con $\varphi(n)$. Es habitual tomar como valor $e = 65537$ puesto que se trata de un número primo. Si no resultase coprimo con $\varphi(n)$ se cambiaría n.
- El valor de d se calcula por el algoritmo de euclides extendido de forma que $1 = ed + m\varphi(n)$ para algún $m \in \mathbb{Z}$. O dicho de otra forma, d es el inverso de e módulo $\varphi(n)$.

Precálculos y Relleno

- Para cifrar con RSA debemos convertir el mensaje en una serie de números x_i entre 0 y $n-1$.

Precálculos y Relleno

- Para cifrar con RSA debemos convertir el mensaje en una serie de números x_i entre 0 y $n-1$.
- Una forma sencilla de hacerlo es tomando bloques de bits que tengan una longitud menor de 1024 bits (ó 2048 si utilizamos claves de 2048 bits).

Precálculos y Relleno

- Para cifrar con RSA debemos convertir el mensaje en una serie de números x_i entre 0 y $n-1$.
- Una forma sencilla de hacerlo es tomando bloques de bits que tengan una longitud menor de 1024 bits (ó 2048 si utilizamos claves de 2048 bits).
- De esta forma cada uno de los bloques representará un número estrictamente menor que n.

Precálculos y Relleno

- Para cifrar con RSA debemos convertir el mensaje en una serie de números x_i entre 0 y $n-1$.
- Una forma sencilla de hacerlo es tomando bloques de bits que tengan una longitud menor de 1024 bits (ó 2048 si utilizamos claves de 2048 bits).
- De esta forma cada uno de los bloques representará un número estrictamente menor que n.
- En muchos casos el problama será al contrario, es decir, tendremos mensajes cortos que tendremos que alargar hasta los 1024 bits. Es importante utilizar algún tipo de método de relleno aleatorio del espacio sobrante.

Cifrado

- Para cada uno de los bloques x_i calcularemos $y_i = x_i^e (mod\ n)$ utilizando el algoritmo de exponenciación modular.

Cifrado

- Para cada uno de los bloques x_i calcularemos $y_i = x_i^e (mod\ n)$ utilizando el algoritmo de exponenciación modular.
- Si hemos utilizado $e = 65537$, este número en binario es 0b1000000000000001 y por lo tanto requiere únicamente 17 multiplicaciones modulares.

Cifrado

- Para cada uno de los bloques x_i calcularemos $y_i = x_i^e (mod\ n)$ utilizando el algoritmo de exponenciación modular.
- Si hemos utilizado $e = 65537$, este número en binario es 0b1000000000000001 y por lo tanto requiere únicamente 17 multiplicaciones modulares.
- El texto cifrado serán los valores y_i.

Descifrado

- Para descifrar los valores y_i utilizaremos la clave privada d y también la clave pública n, calculando $z_i = y_i^d (mod\ n)$ con el algoritmo de exponenciación modular.

Descifrado

- Para descifrar los valores y_i utilizaremos la clave privada d y también la clave pública n, calculando $z_i = y_i^d (mod\ n)$ con el algoritmo de exponenciación modular.
- Estos valores z_i cumplen que

$$z_i = y_i^d = x_i^{ed} = x_i^{1-\varphi(n)m} = x_i \underbrace{\left(x_i^{\varphi(n)}\right)^{-m}}_{1} = x_i (mod\ n)$$

Descifrado

- Para descifrar los valores y_i utilizaremos la clave privada d y también la clave pública n, calculando $z_i = y_i^d (mod\ n)$ con el algoritmo de exponenciación modular.
- Estos valores z_i cumplen que

$$z_i = y_i^d = x_i^{ed} = x_i^{1-\varphi(n)m} = x_i \underbrace{\left(x_i^{\varphi(n)}\right)^{-m}}_{1} = x_i (mod\ n)$$

- Por lo tanto hemos recuperado los valores x_i originales.

Proceso de Entrelazado

- Un problema común a los métodos de cifrado que parten en bloques el mensaje original es que un atacante pueda recomponer un mensaje utilizando pedazos de otros mensajes cifrados.

Proceso de Entrelazado

- Un problema común a los métodos de cifrado que parten en bloques el mensaje original es que un atacante pueda recomponer un mensaje utilizando pedazos de otros mensajes cifrados.
- Pensemos en el siguiente problema, un sistema para ordenar transferencias bancarias en el que sabemos que el primer bloque contiene los datos del beneficiario y el segundo bloque la cantidad.

Proceso de Entrelazado

- Un problema común a los métodos de cifrado que parten en bloques el mensaje original es que un atacante pueda recomponer un mensaje utilizando pedazos de otros mensajes cifrados.
- Pensemos en el siguiente problema, un sistema para ordenar transferencias bancarias en el que sabemos que el primer bloque contiene los datos del beneficiario y el segundo bloque la cantidad.
- Supongamos un atacante C que ha ordenado una transferencia correcta en su cuenta mediante un mensaje $m = (x_1, x_2)$ siendo esa transferencia de una candidad pequeña (digamos 100 euros). Ese atacante sabe que se ha emitido otro mensaje válido $\hat{m} = (\hat{x}_1, \hat{x}_2)$ que ordena transferir a la cuenta D 100000 euros. Conociendo la estructura y disponiendo de los mensajes cifrados (y_1, y_2) e $(\hat{y}_1, \hat{y}_2)$ el atacante podría introducir en el sistema el mensaje $(y_1, \hat{y}_2)$ que ordenara una transferencia de 100000 euros en su cuenta.

Proceso de Entrelazado

- Un problema común a los métodos de cifrado que parten en bloques el mensaje original es que un atacante pueda recomponer un mensaje utilizando pedazos de otros mensajes cifrados.
- Pensemos en el siguiente problema, un sistema para ordenar transferencias bancarias en el que sabemos que el primer bloque contiene los datos del beneficiario y el segundo bloque la cantidad.
- Supongamos un atacante C que ha ordenado una transferencia correcta en su cuenta mediante un mensaje $m = (x_1, x_2)$ siendo esa transferencia de una candidad pequeña (digamos 100 euros). Ese atacante sabe que se ha emitido otro mensaje válido $\hat{m} = (\hat{x}_1, \hat{x}_2)$ que ordena transferir a la cuenta D 100000 euros. Conociendo la estructura y disponiendo de los mensajes cifrados (y_1, y_2) e $(\hat{y}_1, \hat{y}_2)$ el atacante podría introducir en el sistema el mensaje $(y_1, \hat{y}_2)$ que ordenara una transferencia de 100000 euros en su cuenta.
- Para evitar este tipo de ataques de recomposición de mensajes a partir de trozos cifrados, se pueden entrelazar los bloques. Es decir, si el mensaje original es $(x_1, x_2, x_3, \cdots)$ ciframos realmente el mensaje $(x'_1, x'_2, x'_3, \cdots)$ formado mediante

$$x'_1 = x_1, x'_2 = x'_1 \oplus x_2, x'_3 = x'_2 \oplus x_3, x'_4 = x'_3 \oplus x_4, \cdots$$

siendo $\oplus$ la operación de *o exclusivo* o cualquier otra que sea reversible fácilmente.

Importancia del Relleno Aletorio

- Hemos indicado que si los mensajes son cortos, debemos rellenar aleatoriamente el espacio sobrante.

Importancia del Relleno Aletorio

- Hemos indicado que si los mensajes son cortos, debemos rellenar aleatoriamente el espacio sobrante.
- Supongamos que no lo hacemos y que nosotros utilizamos este sistema para emitir mensajes que siempre solo tienen una cifra decimal, $0, 1, 2, \cdots, 9$ cada uno.

Importancia del Relleno Aletorio

- Hemos indicado que si los mensajes son cortos, debemos rellenar aleatoriamente el espacio sobrante.
- Supongamos que no lo hacemos y que nosotros utilizamos este sistema para emitir mensajes que siempre solo tienen una cifra decimal, $0, 1, 2, \cdots, 9$ cada uno.
- Aunque los mensajes cifrados tengan 1024 bits, en realidad sólo habrá 10 diferentes, y un atacante podría mediante cualquier análisis estadístico o de otro tipo, determinar cual es cual.

Importancia del Relleno Aletorio

- Hemos indicado que si los mensajes son cortos, debemos rellenar aleatoriamente el espacio sobrante.
- Supongamos que no lo hacemos y que nosotros utilizamos este sistema para emitir mensajes que siempre solo tienen una cifra decimal, $0, 1, 2, \cdots, 9$ cada uno.
- Aunque los mensajes cifrados tengan 1024 bits, en realidad sólo habrá 10 diferentes, y un atacante podría mediante cualquier análisis estadístico o de otro tipo, determinar cual es cual.
- Por lo tanto la seguridad del RSA estaría comprometida.

Ataques por Fuerza Bruta

- Lo que siempre conoce un atacante es n y e, y lo que desconoce es d. Un ataque por fuerza bruta requiere poder calcular d a partir de n y e sin más información que esa.

Ataques por Fuerza Bruta

- Lo que siempre conoce un atacante es n y e, y lo que desconoce es d. Un ataque por fuerza bruta requiere poder calcular d a partir de n y e sin más información que esa.
- Pero para poder hacer eso necesitamos conocer el valor de $\varphi(n) = (p-1)(q-1) = pq - p - q + 1 = n + 1 - (p+q)$. Como n es público, conocer $\varphi(n)$ es equivalente a conocer $p+q$, pero si conocemos $p+q$, poniendo la ecuación $X^2 - (p+q)X + n = 0$ y resolviendo con la fórmula de segundo grado podemos deducir el valor de p y de q.

Ataques por Fuerza Bruta

- Lo que siempre conoce un atacante es n y e, y lo que desconoce es d. Un ataque por fuerza bruta requiere poder calcular d a partir de n y e sin más información que esa.
- Pero para poder hacer eso necesitamos conocer el valor de $\varphi(n) = (p-1)(q-1) = pq - p - q + 1 = n + 1 - (p+q)$. Como n es público, conocer $\varphi(n)$ es equivalente a conocer $p+q$, pero si conocemos $p+q$, poniendo la ecuación $X^2 - (p+q)X + n = 0$ y resolviendo con la fórmula de segundo grado podemos deducir el valor de p y de q.
- Dicho de otro modo, conocer $\varphi(n)$ en este contexto es equivalente a conocer la factorización de n en producto de primos.

Ataques por Fuerza Bruta

- Lo que siempre conoce un atacante es n y e, y lo que desconoce es d. Un ataque por fuerza bruta requiere poder calcular d a partir de n y e sin más información que esa.
- Pero para poder hacer eso necesitamos conocer el valor de $\varphi(n) = (p-1)(q-1) = pq - p - q + 1 = n + 1 - (p+q)$. Como n es público, conocer $\varphi(n)$ es equivalente a conocer $p+q$, pero si conocemos $p+q$, poniendo la ecuación $X^2 - (p+q)X + n = 0$ y resolviendo con la fórmula de segundo grado podemos deducir el valor de p y de q.
- Dicho de otro modo, conocer $\varphi(n)$ en este contexto es equivalente a conocer la factorización de n en producto de primos.
- Pero los algoritmos más rápidos de factorización conocidos no permiten hacer este proceso en un tiempo razonable si las claves tienen el tamaño indicado, por eso este sistema es seguro. Los métodos de factorización se han estudiado intensamente desde hace muchos años y es altamente improbable que se pudiera descubrir uno que factorizara números de este tamaño.