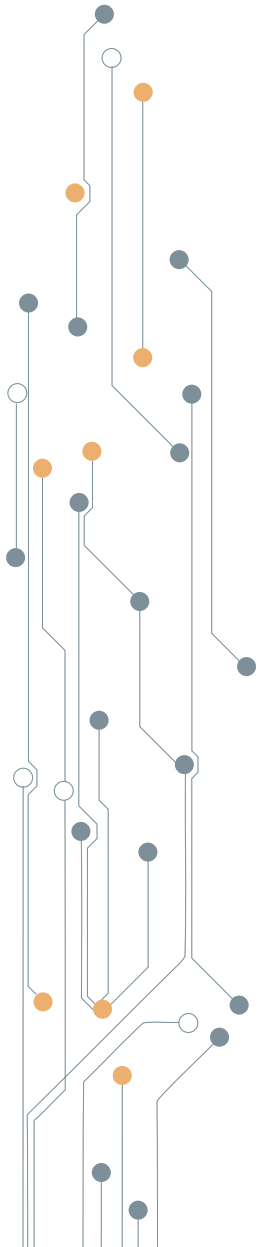
Algoritmos de cifra asimétrica

Telefónica EDUCACIÓN DIGITAL

Índice



1 | Esquema y fundamentos de la cifra asimétrica 3

2 | Intercambio de clave de Diffie y Hellman 8

3 | El algoritmo RSA 11

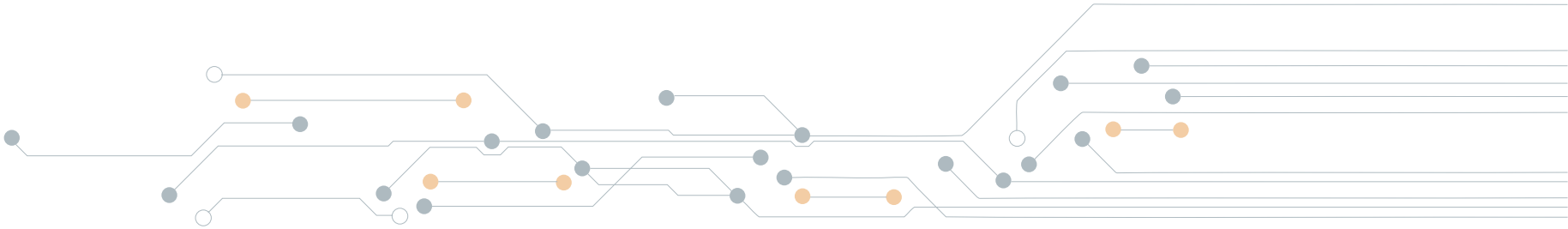
4 | El algoritmo de Elgamal 45

1. Esquema y fundamentos de la cifra asimétrica

La cifra asimétrica, conocida también como cifra por clave pública debido a que una de las dos claves de cada interlocutor es pública y, por lo tanto, no secreta, tiene sus inicios en noviembre de 1976 cuando dos investigadores de la Universidad de Stanford, Whitfield Diffie y Martin Hellman, proponen un nuevo esquema de cifra en su artículo New directions in cryptography. El nuevo sistema está basado en que la clave deja de ser única, y por tanto secreta, al estar ésta compuesta por dos partes, una pública y conocida por todo el mundo, y la otra secreta, o privada, y conocida sólo por su propietario. Mediante propiedades matemáticas de los cuerpos finitos, ambas claves son inversas entre sí y sólo puede a partir de una calcular la otra el dueño de esa clave. Cualquier usuario que conozca la clave pública de otro usuario o sistema, le será computacionalmente difícil descubrir la clave privada correspondiente, porque deberá enfrentarse a un problema de tipo NP (No Polinomial) que para números grandes se vuelve intratable.

Como cada usuario posee dos claves, una pública y la otra privada, inversas entre sí dentro de un cuerpo, de forma que lo que una cifra la otra lo descifra, podremos realizar las siguientes operaciones:

1) Cifrar con la clave pública del destino.
2) Cifrar con la clave pública del emisor.
3) Cifrar con la clave privada del emisor.



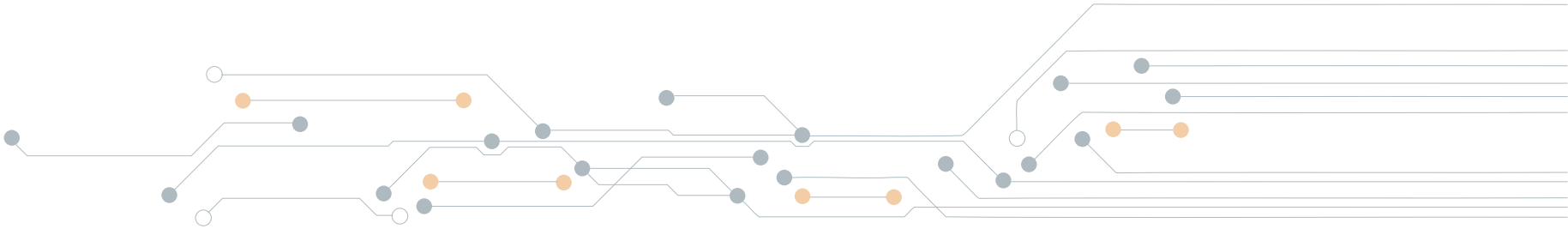
De las tres operaciones indicadas anteriores, en la práctica sólo se utilizan la primera y la tercera, como veremos a continuación.

a. Cifra con la clave pública del destinatario. Si se cifra por ejemplo un número N con la clave pública del destino, entonces solamente ese destinatario (y no otro) será capaz de descifrar el criptograma enviado, pues sólo él o ella están en posesión de la clave privada inversa de la pública, que deshace la cifra y recupera así el número N. Por lo tanto, en este caso el mensaje N se habrá enviado de forma secreta o confidencial. Esto es lo que en la práctica se conoce como intercambio seguro de clave. Resumiendo, cifrando con la clave pública del destino, se obtiene la confidencialidad.

b. Cifra con la clave privada del emisor. Si en este caso el emisor cifra por ejemplo un número N con su clave privada, podrá descifrar ese criptograma y recuperar el número N todo aquel que conozca la clave pública del emisor. El mensaje N en este caso no es secreto porque lo pueden recuperar todos los usuarios que conozcan la clave pública del emisor pero, sin embargo, quienes lo descifran pueden estar seguros de que ese mensaje es de ese emisor y no de otra persona, porque nadie excepto ella conoce esa clave privada. En este caso, en vez de confidencialidad lo que se obtiene con la cifra es autenticidad del emisor. Si a ello unimos que lo que normalmente se cifra en el extremo emisor es el hash de un documento, entonces el proceso se conoce como firma digital y en el extremo receptor, además de esa autenticidad, se puede comprobar la integridad del mensaje debido a las propiedades de las funciones hash.

Observa que se puede cifrar también con la clave pública del emisor, pero en este caso sólo podría descifrar el criptograma el mismo emisor usando su propia clave privada, inversa de la anterior. Esto es lo que se conoce como cifrado convencional o cifrado local, característico por ejemplo si se quiere cifrar un documento de forma local en el disco duro, pero eso ya lo hacen los algoritmos de cifra simétrica con una tasa de cifra mil veces superior. Por ello, esta opción no se usa.

Obviamente, como emisores nunca podremos cifrar con la clave privada del destino, pues la desconocemos y además en la práctica ésta será imposible de adivinar debido a su tamaño de miles de bits.



Aunque de forma genérica se hable de texto en claro o de mensaje M, lo cierto es que con estos sistemas asimétricos sólo se cifran números; más aún, números de unas pocas centenas de bits, como podrían ser una clave secreta o de sesión de un algoritmo simétrico como AES que se intercambia entre dos interlocutores, cliente y servidor por ejemplo, o bien el valor de un hash SHA-256 que se va firmar digitalmente. Si se desea, se podrían cifrar textos, documentos o archivos con este sistema asimétrico, formando bloques de x bits siempre menores que el cuerpo de cifra, pero ello sólo como ejercicio de laboratorio, nunca como una cifra real o práctica.

Las figuras 8.1 y 8.2 muestran los esquemas de cifra asimétrica usando la clave pública de destino para lograr confidencialidad y usando la clave privada de emisor para lograr integridad.

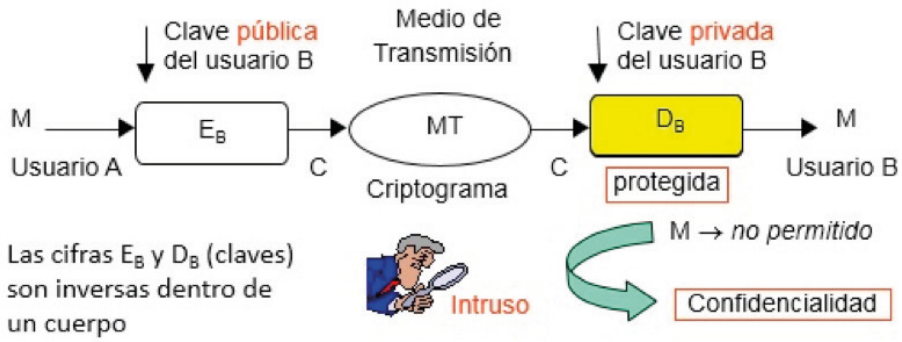


Figura 8.1. Cifra asimétrica con clave pública del destino: confidencialidad.

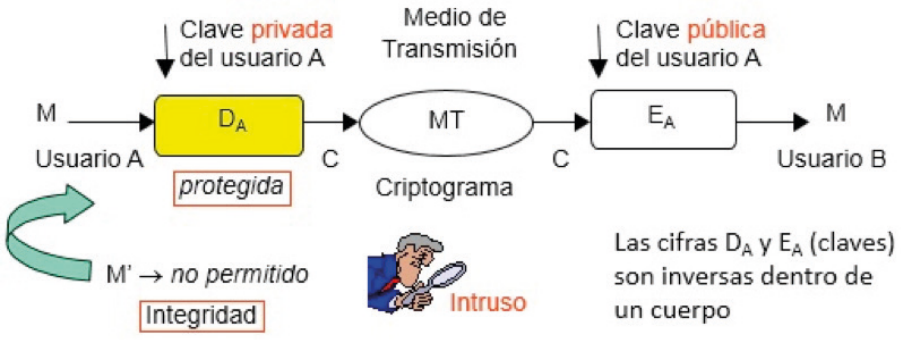


Figura 8.2. Cifra asimétrica con clave privada del emisor: integridad.

De las figuras 8.1 y 8.2 se deduce que para estos sistemas de cifra asimétrica la confidencialidad y la integridad se logran por separado, al contrario que en los sistemas de cifra simétrica en que la confidencialidad y la integridad se lograban de forma simultánea. Esto último porque si la clave de cifra simétrica K es segura y no está en entredicho, es obvio que lo que A envía a B, y viceversa, es confidencial (nadie más posee la clave) y es, además, es íntegro el mensaje y auténtico el emisor, porque no hay más claves que la que comparten A y B.

Lo anterior es muy importante porque nunca antes en criptografía se había podido separar ambos principios de la seguridad en una cifra, confidencialidad e integrad. Por lo tanto, en sistemas de cifra asimétrica podremos enviar un mensaje de forma sólo confidencial, enviarlo sólo con integridad, o bien enviarlo de forma que cumpla ambos principios, que sea confidencial y que además posea integridad.

Surgen aquí dos preguntas interesantes. ¿Se puede hacer una cifra doble en un sistema asimétrico? En caso afirmativo, si se puede cifrar un mensaje con ambas características, que tenga confidencialidad e integridad, ¿qué cifra hacemos primero? ¿Primero la cifra con la clave pública del destino y después la cifra con la clave privada del emisor? ¿O al revés?

La respuesta a estas cuestiones es que sí puede hacerse una cifra doble y que es recomendable primero cifrar con la clave pública del destino y después cifrar con la clave privada de emisión. Obviamente en el primer caso se usará el cuerpo de cifra o módulo del destino y en el segundo caso se usará el cuerpo de cifra o módulo del emisor, ambos públicos. Para recuperar la información en destino, se deshace la cifra de forma inversa; es decir, primero se descifra lo último cifrado en emisión y después se descifra lo primero que se ha realizado en emisión.

Siguiendo la nomenclatura genérica de las figuras 8.1 y 8.2, si A envía a B un mensaje cifrado y firmado, tendríamos el siguiente escenario:

a. Cifra del mensaje M por el usuario A con clave pública de B	EB(M) = C	(se aporta confidencialidad)
b. Cifra del criptograma C por el usuario A con clave privada de A El usuario A envía al usuario B el criptograma dupla {C, C'}	DA(C) = C'	(se aporta integridad)
c. Descifrado del criptograma C' por el usuario B con clave pública de A	EA(C') = C	(se comprueba la integridad)
d. Descifrado del criptograma C por el usuario B con clave privada de B	DB(C) = M	(se recupera el secreto)

Esto se hace así porque en la operación de firma digital, es decir, la segunda operación que hace A y por tanto la primera operación que deberá realizar B, para comprobar que esa firma es válida deberá obtenerse C. Si no se obtiene ese valor C, esto significa que o bien el criptograma intermedio C no es íntegro o bien se ha firmado C por otro usuario, otra clave privada, y no con la clave privada de A como se ha hecho el paso b). Por lo tanto, ya no merece la pena continuar descifrando C porque lo recibido no es íntegro. Si consideramos además que en la práctica la operación de comprobar una firma digital con la clave pública del firmante es mucho más rápida que la operación de descifrar un criptograma con la clave privada del receptor, más a nuestro favor de seguir este protocolo.

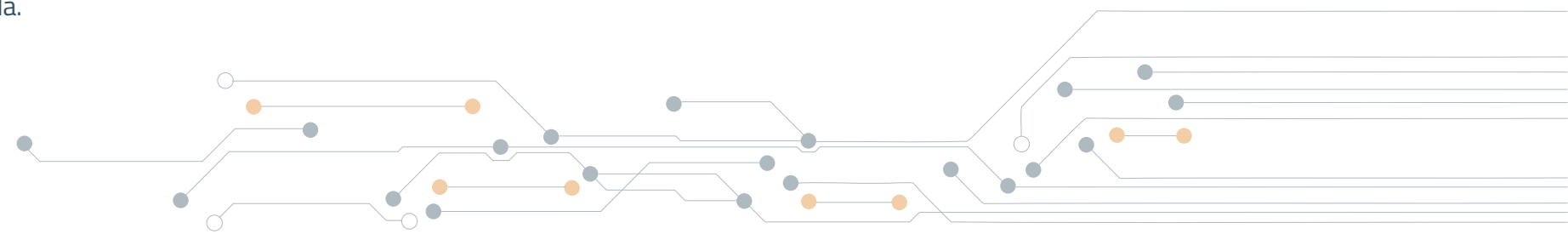
Pregunta: ¿Qué pasaría si el usuario A envía al usuario B sólo el último valor C y no la dupla {C, C'}?

Solución: en este caso B debería descifrar C' con la clave pública de A y el resultado descifrarlo con su clave privada, pero no tendría ninguna certeza de que en el primer descifrado efectivamente haya comprobado la firma de A sobre el criptograma C, y que en el segundo descifrado haya recuperado el mensaje secreto M descifrando C y no otro valor. No obstante, nada impide que podamos realizar esta operación, con la salvedad indicada.

En este escenario en que se recibe un único valor tras la doble cifra asimétrica, lo que nunca deberá hacer el receptor B es realizar las operaciones en el mismo sentido que las hizo A, siempre deberá ser en sentido inverso. Esto es, si A primero cifra y después firma, B entonces primero deberá comprobar la firma y después descifrar el criptograma. Y si A primero firma y después cifra, entonces B primero deberá descifrar y después comprobar la firma.

Otra particularidad de la cifra asimétrica es que en la práctica se cifran siempre números, no mensajes. Lo normal es cifrar números de unas pocas centenas de bits, como pueden ser una clave de sesión de cifra simétrica de AES, RC4, 3DES, etc., para un intercambio de clave, o bien una función hash para una firma digital.

No obstante, sí es posible cifrar mensajes, codificando éstos previamente por ejemplo mediante su código ASCII, pero no es lo habitual. En ese caso, formaremos bloques de texto de n-1 bits, siendo n el número de bits del cuerpo de cifra, o bien de x-1 bytes, siendo x el número de bytes del cuerpo de cifra. Esto porque lógicamente los números a cifrar deben ser parte de los elementos o restos de ese cuerpo de cifra.



2. Intercambio de clave de Diffie y Hellman

En noviembre de 1976, dos investigadores de la Universidad de Stanford, Whitfield Diffie y Martin Hellman, proponen un algoritmo para intercambiar una clave secreta de manera computacionalmente segura, usando para ello funciones matemáticas de un solo sentido o unidireccionales. Esto significa que, si bien el sistema puede romperse matemáticamente, para los valores estándar de miles de bits que se usan en la práctica, el problema se vuelve intratable, tanto en recursos de cálculo como en tiempo. Y ahí radica su seguridad.

Diffie y Hellman usaron en su propuesta la función de un solo sentido conocida como el Problema del Logaritmo Discreto, cuyo enunciado recordamos a continuación. Encontrar el resultado de la expresión $\alpha^{\beta} \bmod p = x$ conocidos el generador α, x y el primo p, es sencilla y rápida, incluso para números grandes. Sin embargo, conociendo x, α y p, encontrar ahora el valor del exponente $\beta = \log_{\alpha} x \bmod p$, se convierte en un problema de muy difícil solución para números grandes. Si bien la operación de calcular el logaritmo de x en base α es muy simple, el hecho de aplicar después la reducción módulo p al resultado, aporta una gran complejidad al problema.

El valor α debe ser unan raíz primitiva o generador del primo p. Una raíz primitiva o generador dentro de un primo p, es aquel número elemento de ese primo que genera todos los restos al hacer la operación de elevarlo a dichos restos y reducir módulo p. Es decir, si decimos que α es una raíz primitiva del primo p, entonces $\alpha^1 \bmod p$, $\alpha^2 \bmod p$, $\alpha^3 \bmod p$, ... $\alpha^{p-2} \bmod p$ y p-1 mod p, entregarán resultados distintos desde el 1 hasta p-1 y, por tanto, generará todos los restos del primo p.

Por ejemplo en el primo p = 17, el número 3 es una raíz primitiva porque entrega los resultados que se ven en la siguiente tabla, todos los restos de ese primo p desde el 1 hasta el 16.

$3^1 \bmod 17 = 3$	$3^2 \bmod 17 = 9$	$3^3 \bmod 17 = 10$	$3^4 \bmod 17 = 13$
$3^5 \bmod 17 = 5$	$3^6 \bmod 17 = 15$	$3^7 \bmod 17 = 11$	$3^8 \bmod 17 = 16$
$3^9 \bmod 17 = 15$	$3^{10} \bmod 17 = 8$	$3^{11} \bmod 17 = 7$	$3^{12} \bmod 17 = 4$
$3^{13} \bmod 17 = 15$	$3^{14} \bmod 17 = 2$	$3^{15} \bmod 17 = 6$	$3^{16} \bmod 17 = 1$

Tabla 7.1. Fortaleza de las funciones hash a los ataques por paradoja del cumpleaños.

Si no se elige α como una raíz primitiva, el algoritmo de Diffie y Hellman sigue funcionando, pero se vuelve menos seguro, en tanto no se dará la condición de que una clave pública (el resultado en la Tabla 8.1) tiene una única clave secreta (el exponente en la Tabla 8.1), ya que no se generarán todos los restos como en la tabla 8.1, sino anillos de números que se repiten y de un tamaño evidentemente menor que el cuerpo p.

El algoritmo propuesto por Diffie y Hellman es el siguiente. Alicia y Bernardo, que se encuentran distantes, eligen un primo p y un número αdentro de ese primo, conocido como generador del primo. Ambos valores son públicos. Acto seguido, cada uno elige un número secreto que no conoce el otro. Así, Alicia elige el número a y Bernardo elige el número b. A continuación realizan el siguiente protocolo:

1. Alicia usa su clave secreta **a**, calcula $\alpha^a \bmod p = x_1$ y se lo envía a Bernardo.

2. Bernardo usa su clave secreta **b**, calcula $\alpha^b \bmod p = x_2$ y se lo envía a Alicia.

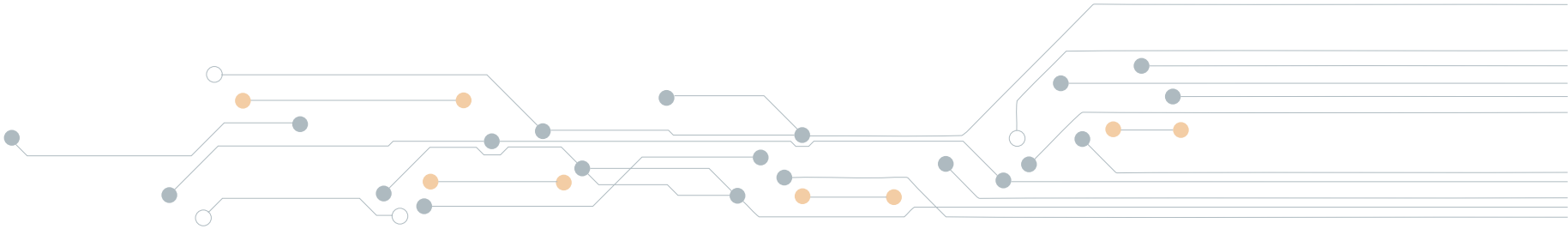
3. Bernardo recibe $x_1 = \alpha^a \bmod p$ y con su clave secreta **b** calcula $x_1^b \bmod p = (\alpha^a)b \bmod p$.

4. Alicia recibe $x_2 = \alpha^b \bmod p$ y con su clave secreta **a** calcula $x_2^a \bmod p = (\alpha^b)^a \bmod p$.

5. La clave intercambiada entre Alicia y Bernardo será $\alpha^{ab} \bmod p = \alpha^{ba} \bmod p = K$.

Como se observa, una vez terminado el protocolo, ambos comparten un mismo valor K que puede usarse como clave de sesión de forma segura. Esto porque cualquier intruso que conozca los datos públicos p y α, así como los valores intercambiados x_1 y x_2, para poder encontrar los valores secretos de **a** de Alicia y **b** Bernardo, y por tanto poder calcular $\alpha^{ab} \bmod p$, se enfrentará al problema del logaritmo discreto, que para valores de p ya cercanos a los mil bits se vuelve intratable.

Existe una variante del algoritmo que permite intercambiar entre cliente y servidor una clave K previamente generada por el cliente, y por tanto no requiere que el protocolo se ejecute simultáneamente como en el caso anterior.

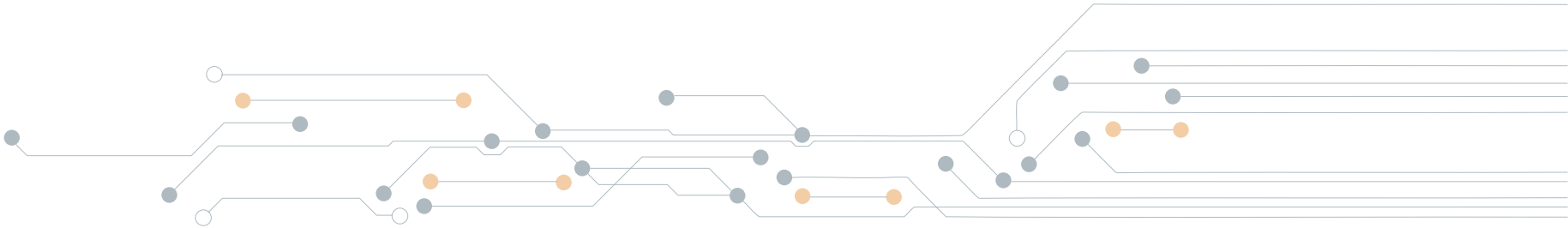


Si Alicia desea enviar a Bernardo una clave de sesión K, y las claves son las que se indican, se procede como se indica:

Claves de Bernardo: Clave pública: $C_{púbB}$, α_B, p_B. Clave privada: b

Claves de Alicia: Clave pública: $C_{púbA}$, α_A, p_A. Clave privada: a

- Alicia hace lo siguiente:
  - Calcula una clave: $K_{AB} = (C_{PúbB})^a \bmod p_B = (\alpha_B{}^b \bmod p_B)^a \bmod p_B$.
  - $K_{AB} = \alpha_B{}^{ba} \bmod p_B$. De ese valor en el cuerpo de p_B (de al menos 1.024 bits), se eligen un conjunto de bits para una clave de cifra simétrica, por ejemplo 256 bits para el AES-256 como clave K.
  - **A**licia cifra el mensaje con la clave simétrica K_{AB}.
  - Envía a Bernardo el mensaje cifrado con la clave K_{AB}.
  - Envía además a Bernardo el siguiente valor: $\alpha_B{}^a \bmod p_B$.
  -
- Bernardo hace lo siguiente:
  - El valor recibido $\alpha_B{}^a \bmod p_B$ lo eleva a su clave privada b.
  - Obtiene $(\alpha_B{}^a \bmod p_B)^b \bmod p_B = \alpha_B{}^{ab} \bmod p_B = \alpha_B{}^{ba} \bmod p_B = K_{AB}$. Selecciona los mismos 256 bits que en origen y obtiene K.
  - Con esa clave simétrica descifra el correo cifrado.



3. El algoritmo RSA

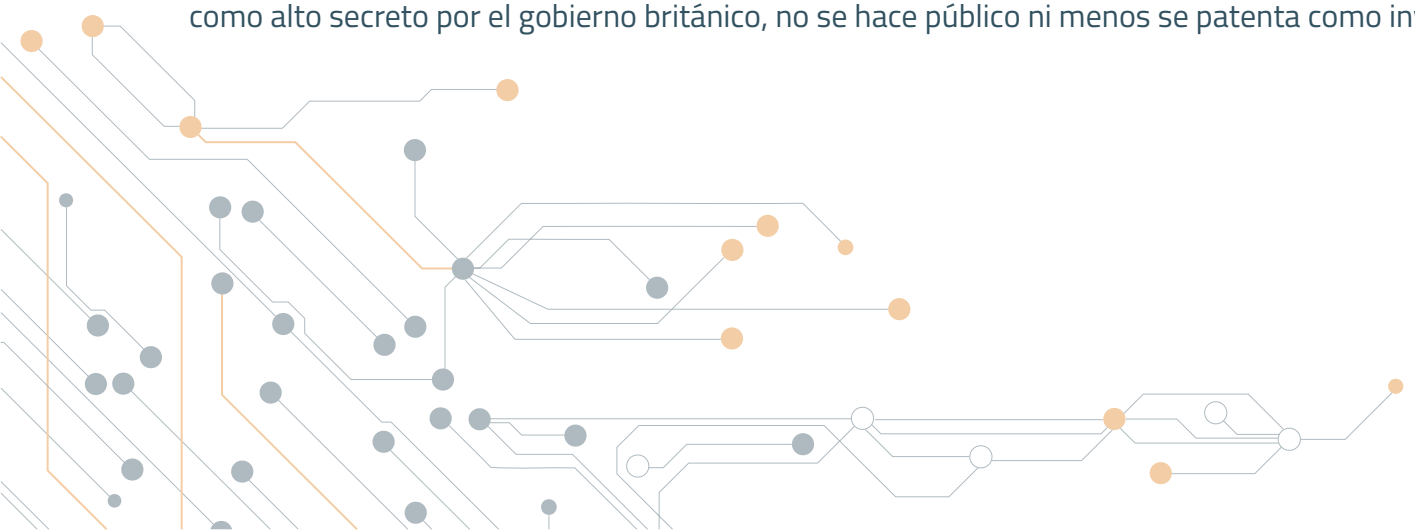
Los inicios

Aunque la propuesta de Diffie y Hellman se convierte en un hito que revoluciona el mundo de la criptografía en aquellos años, hasta ese momento sólo de tipo simétrica y de clave secreta, el sistema no permitía realizar una cifra real de información y menos la firma digital sobre un documento, sólo el intercambio seguro de una clave.

Es en febrero de 1978, es decir poco más de un año después de la propuesta de Diffie y Hellman, cuando otros tres investigadores norteamericanos, Ron Rivest, Adi Shamir y Leonard Adleman, en este caso del Instituto Tecnológico de Massachusetts MIT, proponen un sistema de cifra que llevará las iniciales de sus apellidos. El algoritmo se patentará como RSA.

A diferencia del intercambio de clave de DH, que basaba su fortaleza en la dificultad computacional de calcular logaritmos discretos en primos muy grandes, RSA basa su fortaleza en la dificultad computacional de factorizar un número compuesto muy grande, producto de dos primos grandes, y encontrar por tanto tales factores primos. Ambos problemas tienen una complejidad algorítmica similar y son inabordables para la capacidad mundial de cómputo en nuestros días cuando se trata de valores por encima de los mil bits.

Sin embargo, aunque Rivest, Shamir y Adleman son los autores de RSA, el mismo algoritmo de cifra asimétrico basado en la dificultad de factorizar números grandes fue descubierto mucho antes. En el año 1969 el Government Communications Headquarters GCHQ en Gran Bretaña comienza a trabajar en la idea de poder distribuir claves a través de una cifra no simétrica, llegando en el año 1973 -cinco años antes- a la misma conclusión que los creadores de RSA. Una investigación dirigida por el matemático Clifford Cocks que al ser un trabajo considerado como alto secreto por el gobierno británico, no se hace público ni menos se patenta como invento.



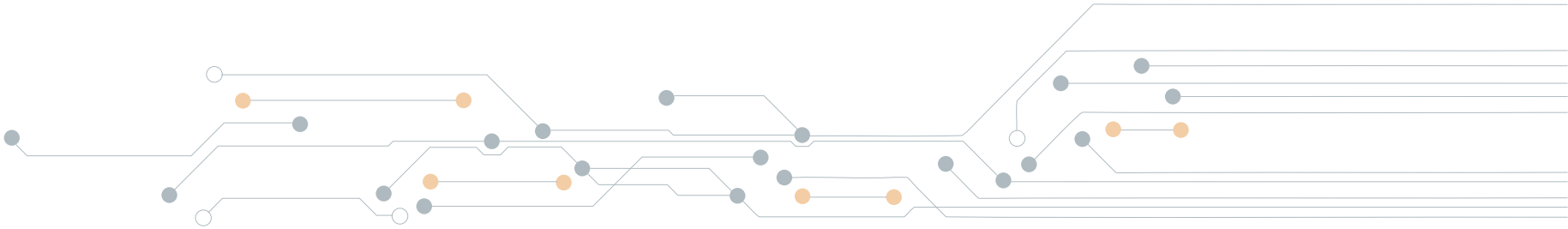
La seguridad de RSA

La seguridad del algoritmo RSA se basa en la dificultad computacional que conlleva encontrar los dos factores primos de un número compuesto muy grande, resultado del producto de éstos, donde sus primos también son números grandes. Esto es lo que matemáticamente se conoce como el problema de la factorización entera, uno de los problemas denominados No Polinomiales o de tipo NP, muy usados en la criptografía.

Se trata de un problema que en un sentido el cálculo es muy fácil y rápido (por ejemplo multiplicar dos números primos) pero que en sentido contrario (por ejemplo, encontrar esos dos factores conocido el producto) se vuelve computacionalmente intratable a medida que la entrada es cada vez mayor. Es decir, requiere de unos recursos informáticos excesivos y, por tanto, de un tiempo de cálculo exorbitante.

Aunque no sea matemáticamente exacto, se podría aceptar que la primera operación es lineal en el sentido de que la cantidad de operaciones a realizar es directamente proporcional al tamaño de los dos operandos; en cambio, la segunda operación es de tipo exponencial de manera que, por ejemplo, aumentando al doble el tamaño de la entrada, el número de cálculos a realizar no aumenta al doble sino mucho más, reflejándose esto en una curva de tipo exponencial del tiempo de cálculo en función del tamaño de la entrada. No obstante la operación directa de multiplicación de dos números primos muy grandes es sencilla y consume escasos recursos computacionales.

Pero, ¿qué entendemos por un número muy grande aplicado a la criptografía? En el caso de RSA y ya en el año 2010, los valores mínimos de esos dos primos eran de 512 bits (unos 155 dígitos) en certificados digitales X.509 y, por tanto, su producto es un número de 1.024 bits (unos 310 dígitos). Aunque sólo se ha llegado a factorizar a fecha actual números algo menos que 800 bits (768 bits en 2009), esto recomienda el aumento de esos primos por ejemplo a 1.024 bits cada uno, con lo que se obtiene un producto o cuerpo de cifra de 2.048 bits, que es el valor que hoy se usa en diversos programas, algunas aplicaciones, certificados digitales y pasarelas seguras de Internet.

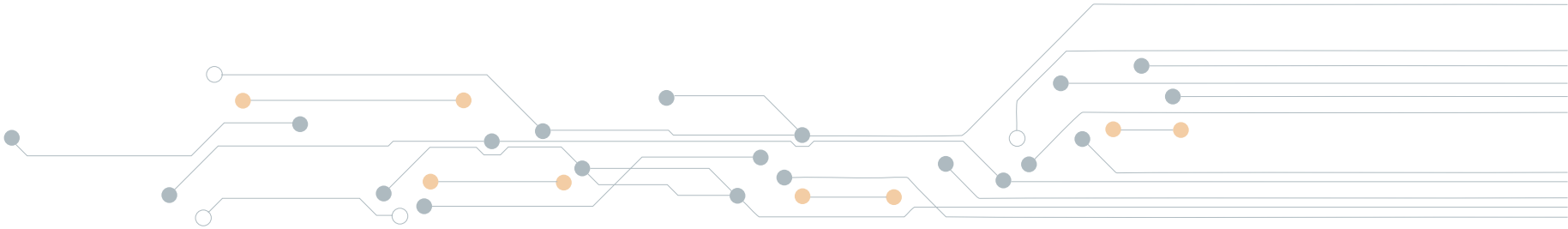


Generando una clave RSA

El protocolo detallado para la generación de una clave RSA, pública y privada, es el siguiente:

1. Los usuarios Alicia y Bernardo eligen cada uno dos primos p y q de valores iguales o superiores a 512 bits, hoy recomendable 1.024.
2. Los valores de los primos p y q serán un secreto, sólo conocido por los propietarios de esas claves.
3. Multiplican esos primos y obtienen el módulo de cifra n = p x q.
4. Alicia y Bernardo hacen público el módulo de cifra n.
5. En el caso de Alicia ese módulo será $n_A = p_A \times q_A$ y en el caso de Bernardo será $n_B = p_B \times q_B$.
6. Cada usuario calculará el Indicador de Euler Φ de ese módulo n, que en este caso de dos primos es $\Phi n = (p - 1)*(q - 1)$.
7. Así, Alicia calcula $\alpha\Phi_{nA} = (p_A - 1)*(q_A - 1)$ y Bernardo calcula $\Phi_{nB} = (p_B - 1)*(q_B - 1)$.
8. Ese valor va a ser su secreto o trampa, un número que sólo conoce su dueño al depender de p y de q.
9. A partir de ese valor trampa Φ_n se calculará ahora la clave pública e y la correspondiente clave privada d.
10. Cada usuario elegirá un valor de clave pública e dentro el siguiente intervalo: $1 < e < \alpha\Phi_n$. En la práctica, esto no es verdad puesto que todos usarán como clave pública e el número 4 de Fermat, es decir $2^{2^4} + 1 = 65.537$.

11. Para asegurarse que exista el inverso multiplicativo, y por tanto la clave privada d, debe cumplirse que mcd $[e,\Phi_n] = 1$.
12. Ese valor e será la segunda parte de su clave pública, además del módulo n.
13. Alicia elige como clave pública $1 < e_A < \alpha\Phi_{nA}$ y Bernardo elige como clave pública $1 < e_B < \Phi_{nB}$.
14. Usando ahora el Algoritmo Extendido de Euclides, cada usuario calcula su clave privada d en ese cuerpo trampa.
15. Alicia calcula $d_A = inv\ (e_A, \Phi_{nA})$ y Bernardo calcula $d_B = inv\ (e_B, \Phi_{nB})$.
16. Luego, cada usuario tiene dos valores que forman su clave pública, n y e, y un valor secreto que es su clave privada d.
17. Además del valor d, también guardarán en secreto p y q, que le servirán para acelerar la operación de descifrado mediante el teorema chino del resto.



Cifrando y descifrando con RSA

Por lo tanto, si Alicia desea enviar un número secreto (con confidencialidad) a Bernardo, realizará las siguientes operaciones. Como Alicia conoce la clave pública de Bernardo n_B y e_B, calcula $M^{eB} \bmod n_B = C$, enviando a Bernardo este criptograma C.

Dado que Bernardo es el único que conoce su clave privada dB, sólo él podrá realizar la siguiente operación: $C^{dB} \bmod n_B$.

Observa que $C^{dB} \bmod n_B = [M^{eB} \bmod n_B]^{dB} \bmod n_B = [M^{eB}]^{dB} \bmod n_B$.

Puesto que en la operación [MeB]dB mod nB los exponentes se anulan entre sí porque son inversos en Φ_{nB}, en recepción Bernardo recuperará el mensaje M enviado por Alicia dentro del cuerpo n_B. En realidad, lo que sucede es lo siguiente:

1. Como dB = inv $[e_B, \Phi_{nB}]$, entonces $e_B \times d_B = k\Phi_{nB} + 1$
2. Por tanto $[M^{eB}]^{dB} \bmod n_B = M^{k\Phi nB + 1} \bmod n_B = M \times M^{k\,\Phi nB} \bmod n_B$
3. Por el Teorema de Euler tenemos que $M^{k\,\Phi n} \bmod n = 1$
4. Luego $M \times M^{k\,\Phi nB} \bmod n_B = M \times 1 = M$ y se recupera el mensaje

Firmando y comprobando la firma con RSA

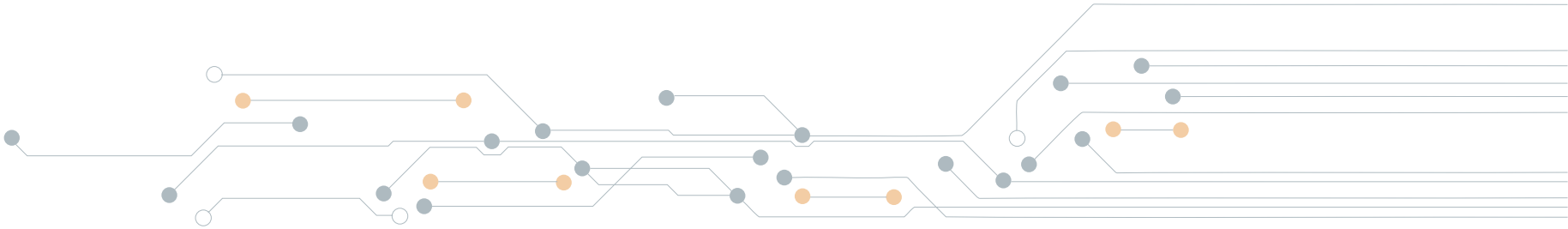
Si Alicia desea enviar un mensaje M firmado a Bernardo, la siguiente operación con su clave privada d_A: $M^{dA} \bmod n_A = C$, enviando en el criptograma C su firma.

Dado que Bernardo conocerá la clave pública de Alicia, n_A y e_A (y todos los que la conozcan) podrá realizar la siguiente operación: $C^{eA} \bmod n_A = M$.

Observa que $C^{eA} \bmod n_A = [M^{dA} \bmod n_A]^{eA} \bmod n_A = [M^{dA}]^{eA} \bmod n_A$.

Como en la operación $[M^{dA}]^{eA} \bmod n_A$ los exponentes se anulan entre sí entregando el valor 1 como ya hemos visto, Bernardo recupera el mensaje M firmado por Alicia dentro del cuerpo n_A.

Como ya se ha comentado, en los sistemas de cifra asimétricos como es el caso de RSA, en la práctica sólo se cifrarán números y además valores relativamente pequeños: sólo centenas de bits. Por tanto, una firma digital se realizará sobre un número, resultado de aplicar una función hash a un mensaje, ahora sí un archivo ya de cualquier tamaño y tipo.



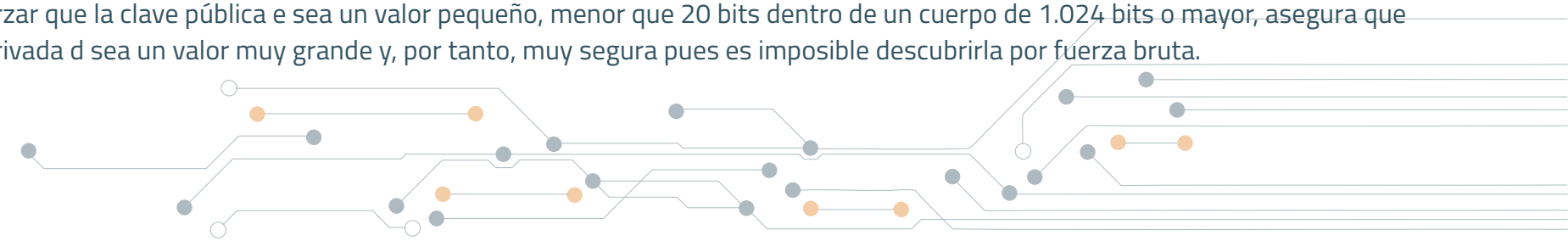
Los pasos a realizar en la firma digital con RSA a través de un hash serán:

1. Alicia obtiene la función hash (un número) del archivo M que desea firmar.
2. Ese número debe representar al archivo de manera única, como si fuese una huella dactilar de él.
3. Sobre ese número o hash h(M), Alicia realizará la firma usando su clave privada d_A.
4. Alicia calcula $h(M)^{dA} \bmod n_A = s$ y le envía a Bernardo la dupla mensaje y firma (M, s).
5. Bernardo recibe (M', s) pues no se sabe si recibe el mismo mensaje M o un mensaje distinto o falso M'.
6. Bernardo realiza la siguiente operación con la clave pública de Alicia: $se^A \bmod n_A = [h(M)^{dA} \bmod n_A]^{eA} \bmod n_A = h(M)$.
7. Es decir, recupera el valor h(M) que usó Alicia para su firma en el momento de emisión.
8. Bernardo calcula la función hash h(M') del mensaje M' recibido, que puede ser el verdadero mensaje M u otro distinto M'.
9. Si el hash calculado en recepción coincide con el hash recuperado y firmado en emisión, se acepta la firma como correcta; en caso contrario se rechaza la firma.

Tamaños de los parámetros en RSA

Hemos comentado que se usa la misma clave pública e para todo el mundo, el número 4 de Fermat. Los motivos para usar un valor de clave pública relativamente pequeña comparada con el módulo n sigue el siguiente razonamiento:

1. Como la trampa Φ_n será igual a (p - 1)*(q - 1) y los primos p y q tienen al menos 512 bits, el tamaño de Φ_n será aproximadamente igual al de n, sólo un poco menor, es decir este caso 1.024 bits.
2. Como la clave pública e y la clave privada d son inversas en el cuerpo Φ_n, es decir d = inv [e, Φ_n], entonces se cumple la siguiente relación: e x d mod Φ_n = 1.
3. Para que se cumpla esa igualdad, el producto de e x d debe salir del cuerpo Φ_n al menos una vez para que la operación en ese módulo nos devuelva el valor 1.
4. Es decir, se cumplirá que e x d = k x Φ_n + 1, con k = 1, 2, 3, 4,...
5. Para que ese producto salga al menos una vez del cuerpo Φ_n, es decir con k = 1, como el valor de e es muy pequeño, de tan sólo 17 bits, entonces d tendrá que ser muy grande, en torno al tamaño de Φ_n.
6. En la práctica ese valor de k = 1, o un valor bajo de k, será muy poco probable y, por tanto, podemos esperar una clave privada d muy cercana o igual en bits al valor de n, como así sucede en la práctica.
7. Esto significa que será computacionalmente imposible adivinar el valor de la clave privada d, puesto que encontrar un número dentro de un cuerpo de 1.024 bits significa un tiempo de cómputo totalmente inabordable, como valor medio $2^{1.023}$ intentos.
8. Luego, forzar que la clave pública e sea un valor pequeño, menor que 20 bits dentro de un cuerpo de 1.024 bits o mayor, asegura que la clave privada d sea un valor muy grande y, por tanto, muy segura pues es imposible descubrirla por fuerza bruta.



El número 4 de Fermat como clave pública

Aunque la única condición que debe cumplir la clave pública e para que pueda existir la clave privada inversa d es que exista el inverso de e en Φ_n, es decir d = inv [e, Φ_n], y por tanto es necesario que mcd [e, Φ_n] = 1, dado que 1 < e < Φ_n entonces podríamos elegir cualquier número dentro de este rango que cumpla con esa condición de primalidad. No obstante, ya hemos visto que es recomendable usar un valor pequeño como clave pública. De hecho, valores típicos suelen ser los números 3, 17 y 65.537 puesto que, además, tienen sólo dos bits iguales a 1 en su representación binaria, que como veremos en este apartado tendrá importancia en la eficiencia del cómputo.

Descartando el número 3 porque se han descrito ataques posibles al sistema, la representación binaria de los números 17 y 65.537 será respectivamente 10001 y 10000000000000001. El número cuatro de Fermat F4 tiene un conjunto de características interesantes:

1. 65.537 es un número primo.
2. $65.537 = 2^{2^{\wedge}4} + 1 = 2^{16} + 1$
3. 65.537 es un número de 17 bits, relativamente pequeño.
4. 65.537 = 10000000000000001 en binario, con una gran cadena de ceros.
5. Esta propiedad de tener muchos ceros permitirá una operación de cifra mucho más rápida.

Si el exponente de una cifra RSA, en este caso la clave pública del receptor e_R, tiene muchos ceros como es el caso de F_4, entonces la operación se realiza de forma muy rápida por el algoritmo de exponenciación rápida que hemos visto. Por lo tanto, para el exponente de clave pública e = F_4 = 10000000000000001, en el primer paso del algoritmo no se hace operación alguna, porque el resultado es la misma base A con la que se inicia el cálculo, luego tendríamos 15 operaciones de elevar al cuadrado, que requieren muy poco tiempo de máquina, y sólo una multiplicación al final que requiere algo más de tiempo de cómputo.

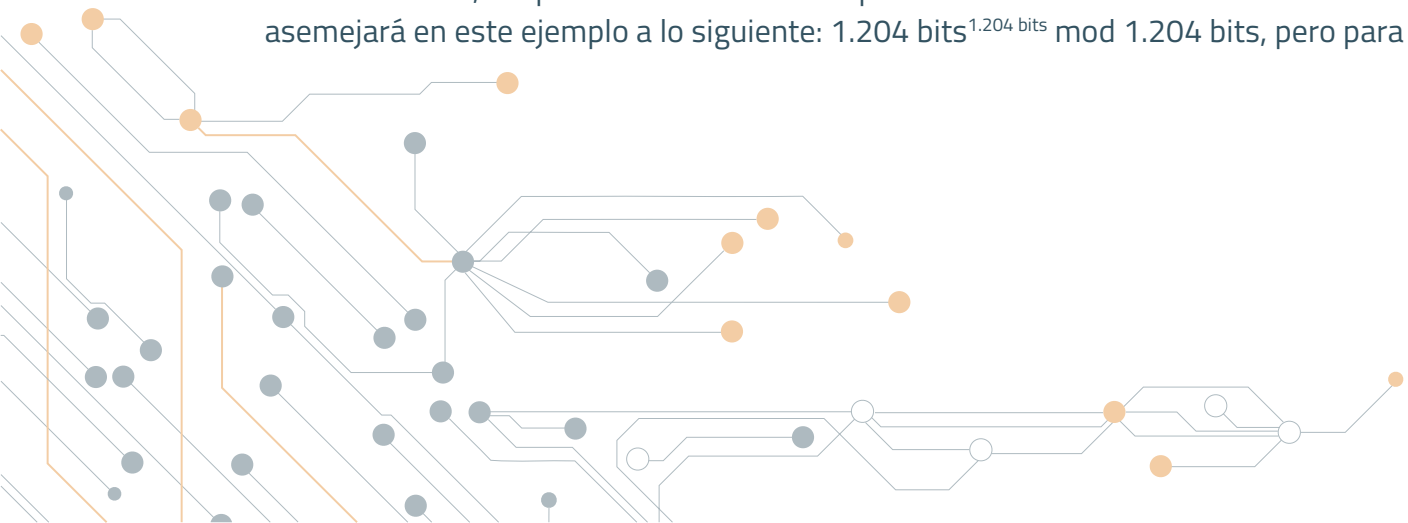
Como conclusión final, el número F4 es apropiado como clave pública universal porque tiene muchos ceros y es un valor relativamente pequeño comparado con el módulo n, que será siempre superior a 1.024 bits. Esto trae consigo las siguientes consecuencias:

1. Forzar a que la clave privada d tenga un tamaño similar al módulo n y, por tanto, sea segura ante un ataque por fuerza bruta.
2. Acelerar la operación de exponenciación para el intercambio de clave al tener sólo dos valores binarios en 1 y los demás en 0.
3. Evitar ataques que podrían producirse si e es un número demasiado pequeño, como sería el caso de e = 3.

Recuerda que el número 65.537 es la clave pública e por defecto de todas las claves RSA comerciales y que podemos ver en certificados digitales; lo que será distinto y propio en cada clave, son los primos p y q, y por lo tanto el módulo n.

También se puede llegar a la siguiente conclusión interesante para una clave de un servidor de 1.204 bits, aunque quizás no sea parte de las condiciones de diseño.

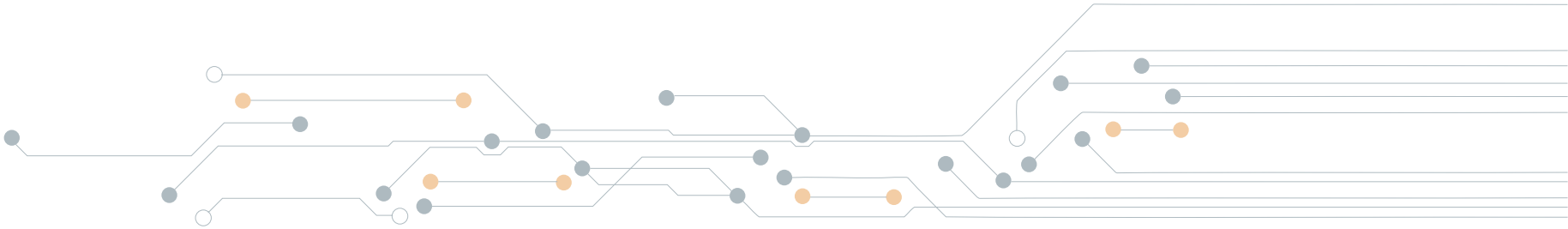
1. La operación que debe hacer el cliente para enviar la clave de sesión simétrica por ejemplo de 128 bits a un servidor, una vez conocida su clave pública mediante un certificado digital, es muy rápida. La operación será $128\ bits^{17\ bits}$ mod 1.024 bits y dicho exponente de 17 bits tiene 15 bits en cero.
2. Por el contrario, la operación de descifrado que deberá realizar el servidor una vez recibido el criptograma será muy pesada pues se asemejará en este ejemplo a lo siguiente: $1.204\ bits^{1.204\ bits}$ mod 1.204 bits, pero para ello se usará el teorema chino del resto.



Cifrando mensajes con RSA

Puesto que los sistemas de cifra con clave pública tienen una velocidad de cifra en torno a los cientos de Kilobytes por segundo, muy lenta si la comparamos con la velocidad de los sistemas de cifra con clave secreta que está en los cientos de Megabytes por segundo, mil veces más rápidos, se usarán preferentemente para cifrar números de unas pocas centenas de bits. Por lo tanto, aunque podemos codificar el mensaje M con la codificación que se nos ocurra (código ASCII decimal por ejemplo) para transformar esa información a un conjunto a números y usar luego RSA para cifrar esa cadena de valores, en la práctica no tiene sentido hacerlo por esa velocidad tan baja.

No obstante, si deseamos cifrar con RSA mensajes de texto, es menester que previamente codifiquemos el texto a cifrar, por ejemplo según los valores en decimal de tres dígitos de los caracteres en código ASCII extendido. Como se ha comentado, pueden existir infinitas maneras de codificar el texto en claro: dando un peso o valor a cada carácter según un alfabeto dado, dando peso a un conjunto de caracteres, modificando estos pesos, etc.



Para la siguiente clave RSA, en donde p = 5.923, q = 6.689, n = 39.618.947 (26 bits), e = 31, d = 5.110.495, vamos a cifrar el mensaje de texto de 18 caracteres M = CALCULANDO BLOQUES.

1. Como n tiene un tamaño de 26 bits, vamos a cifrar bloques de 3 bytes, 24 bits.
2. Si el mensaje es M = CALCULANDO BLOQUES, cifraremos estos 6 bloques de 24 bits cada uno: CAL CUL AND O_B LOQ UES.
3. Buscamos en una Tabla ASCII los valores binarios de los caracteres y pasamos su valor a decimal usando la calculadora de Windows.
4. Los 6 bloques serán:

   a. CAL = 01000011 01000001 01001100 = M_1 = 4.407.628

   b. CUL = 01000011 01010101 01001100 = M^2 = 4.412.748

   c. AND = 01000001 01001110 01000100 = M_3 = 4.279.876

   d. O_B = 01001111 00100000 01000010 = M_4 = 5.185.602

   e. LOQ = 01001100 01001111 01010001 = M_5 = 5.001.041

   f. UES = 01010101 01000101 01010011 = M_6 = 5.588.307

5. Cifrados y descifrado de los bloques:

Cifrado: C1: $4.407.628^{31} \bmod 39.618.947 = 6.734.280$ Descifrado: M_1: $6.734.280^{5.110.495} \bmod 39.618.947 = 4.407.628$

Cifrado: C2: $4.412.748^{31} \bmod 39.618.947 = 21.898.080$ Descifrado: M_2: $21.898.080^{5.110.495} \bmod 39.618.947 = 4.412.748$

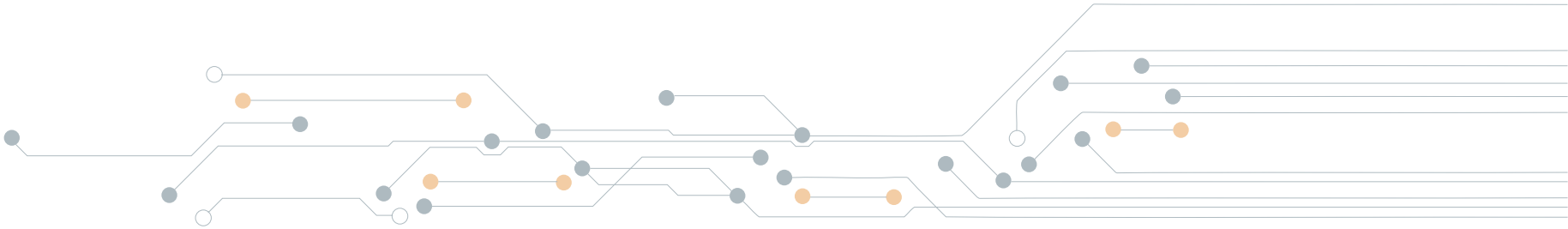
Cifrado: C3: $4.279.876^{31} \bmod 39.618.947 = 33.215.194$ Descifrado: M_3: $33.215.194^{5.110.495} \bmod 39.618.947 = 4.279.876$

Cifrado: C4: $5.185.602^{31} \bmod 39.618.947 = 30.956.224$ Descifrado: M_4: $30.956.224^{5.110.495} \bmod 39.618.947 = 5.185.602$

Cifrado: C5: $5.001.041^{31} \bmod 39.618.947 = 8.150.418$ Descifrado: M_5: $8.150.418^{5.110.495} \bmod 39.618.947 = 5.001.041$

Cifrado: C6: $5.588.307^{31} \bmod 39.618.947 = 22.796.302$ Descifrado: M_6: $22.796.302^{5.110.495} \bmod 39.618.947 = 5.588.307$

Como los valores recuperados en el descifrado son los mismos que los bloques a cifrar, hemos recuperado el mensaje M = CALCULANDO BLOQUES. Simplemente pasamos a binario el número recuperado en el descifrado y vamos leyendo ahora los bits de derecha a izquierda y formando bloques de 8 bits, un byte, y convirtiéndolo a carácter ASCII. En la última lectura, si es necesario añadimos ceros a la izquierda hasta formar 8 bits.



Claves privadas y públicas parejas

Cuando generamos una clave RSA, usamos como trampa el indicador de Euler ϕ_n para calcular la clave privada d a partir del conocimiento de la clave pública e. Puesto que mcd [e, ϕ_n] = 1, se asegura que el inverso d existe y que, además, es el único inverso de la clave pública e en ese cuerpo ϕ_n.

No obstante, como es lógico, la cifra se hace posteriormente en el cuerpo público n para que todo el mundo pueda utilizarlo. Y en dicho cuerpo n ya no se cumple que el único inverso de la clave pública e sea la clave privada d. Hay al menos otro valor diferente a la clave privada d que cumple las mismas funciones y que, por lo tanto, permite descifrar algo enviado de forma secreta que se ha cifrado con la clave pública, o bien firmar digitalmente con esa otra clave privada. A estas claves se les llama Claves Privadas Parejas o de forma abreviada CPP.

Esto es algo en principio inesperado porque normalmente siempre se ha dicho que un sistema de cifra asimétrico, como lo es RSA, tiene una única clave pública y, por lo tanto, también una única clave privada. Esto no será verdadero. Vamos a explicarlo con un sencillo ejemplo, que puede comprobarse fácilmente usando por ejemplo la calculadora de Windows.

Sea una clave RSA con p = 37, q = 41, n = 1.517, ϕ_n = 1.440, e = 13, d = 997. Vamos a cifrar el valor 1.001 con la clave pública e = 13 y luego descifrar el criptograma con la clave privada d = 997.

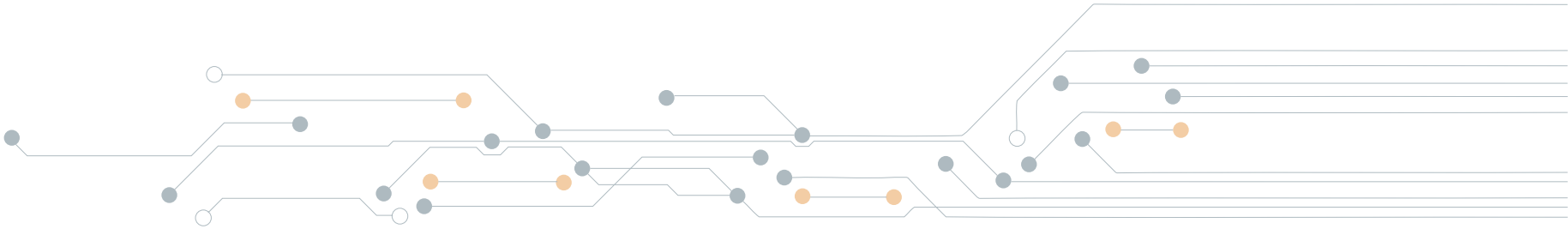
- Cifrado con e = 13: 1.001^{13} mod 1.517 = 1.088
- Descifrado con d = 997: 1.088^{997} mod 1.517 = 1.001. Se ha recuperado el secreto.

Pero si usamos los números 277, 637 y 1.357 como si fuesen la clave privada d, obtenemos lo siguiente:

- Descifrado con d' = 277: 1.088^{277} mod 1.517 = 1.001
- Descifrado con d' = 637: 1.088^{637} mod 1.517 = 1.001
- Descifrado con d' = 1.357: $1.088^{1.357}$ mod 1.517 = 1.001

Es decir, también se ha recuperado el texto en claro o secreto.

Lo que ha sucedido en el ejemplo anterior es que en el cuerpo de cifra n = 1.517 con una clave pública e = 13, los valores 277, 637 y 1.357 son claves privadas parejas (CPP) denominadas d' y que cumplen la misma función que la clave privada d = 997.



Toda clave RSA tendrá como mínimo una clave privada pareja. La cantidad de claves privadas parejas dependerá fuertemente de los primos p y q, siendo las ecuaciones que nos entregan este valor de CPP las siguientes:

Si $\gamma = mcm [(p-1), (q-1)]$ y además $d_\gamma = e_{-1} \bmod \gamma = inv (e, \gamma)$, entonces la clave privada d tendrá λ claves parejas di de la forma:

$$d_i = d_\gamma + i\gamma \quad con\ 1 < d_i < n \quad con \quad i = 0, 1, \ldots \lambda, \text{ siendo } \lambda = \lfloor (n - d_\gamma)/\gamma \rfloor$$

Por ejemplo, para la clave anterior con p = 37, q = 41, n = 1.517, ϕ_n = 1.440, e = 13, d = 997, tenemos:

$\gamma = mcm [(p-1), (q-1)] = mcm (36, 40) = 360$

Luego: $d\gamma = inv (13, 360) = 277$, así $di = d_i + i\gamma = 277 + i*360$ (i = 0, 1, 2, 3)

Es decir, di = 277, 637, 997, 1.357.

Además $\lambda = \lfloor (n - d_\gamma)/\gamma \rfloor = \lfloor (1.517 - 277)/360) \rfloor = 3{,}44 = 3$ CPP.

Al igual que existe una cantidad λ de claves privadas parejas, existirá un número similar λ de claves públicas parejas. En este caso las ecuaciones serán:

Si $\gamma = mcm [(p-1), (q-1)]$ y además $e_\gamma = d^{-1} \bmod_\gamma = inv (d,\gamma)$, entonces la clave pública e tendrá λ claves parejas ei de la forma:

$ei = e_\gamma + i\,\gamma$ con $1 < e_i < n$ con $i = 0, 1, \ldots \lambda$, siendo $\lambda = \lfloor (n - e_\gamma)/\gamma \rfloor$

Por ejemplo, para la misma clave anterior con p = 37, q = 41, n = 1.517, $\phi_n = 1.440$, e = 13, d = 997, tendremos ahora:

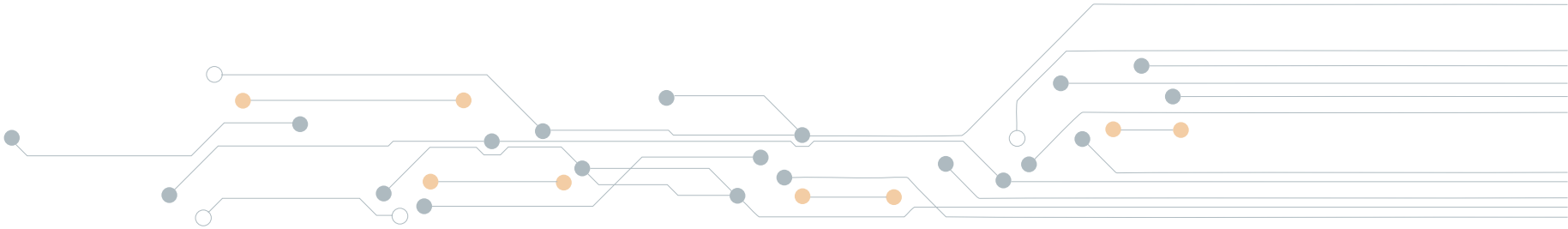
$\gamma = mcm [(p-1), (q-1)] = mcm (36, 40) = 360$

Luego: $e_\gamma = inv (997, 360) = inv (277, 360) = 13$, así $e_i = e_\gamma + i_\gamma = 13 + i*360$ (i = 0, 1, 2, 3, 4)

Es decir di = 13, 373, 733, 1.093, 1.453

Además $\lambda = \lfloor (n - e_\gamma)/\gamma \rfloor = \lfloor (1.517 - 13)/360) \rfloor = 4,17 = 4$ CPúbP

A primera vista, no resulta agradable el conocimiento de estas claves parejas en RSA y que realizan la misma función que una clave privada o una clave pública, más aún si se tiene en mente lo que se dice habitualmente de este sistema de cifra asimétrico: que existe una única clave pública (e, n) y una única clave privada (d), y lo que se cifra con una de ellas sólo se descifra con la otra. De hecho, todos los certificados digitales X.509 que trabajan con RSA tienen este tipo de claves.



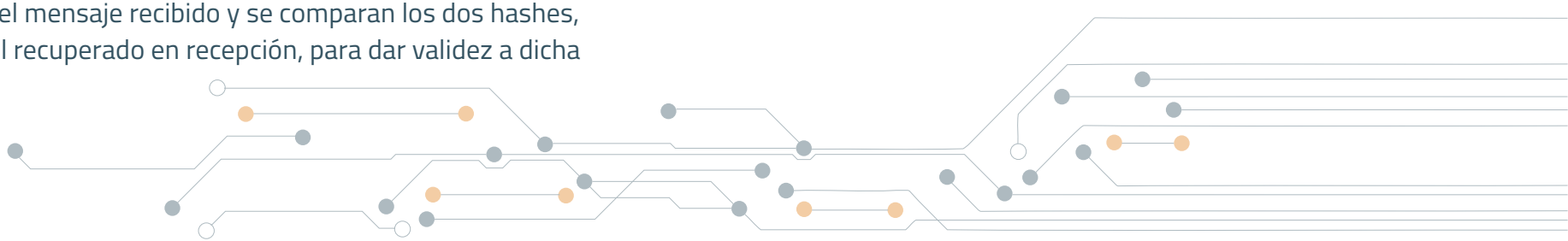
Hemos visto además que el cálculo de estas claves privadas parejas es muy sencillo, que están separadas por un valor constante y que incluso para claves con valores reales de miles de bits se obtienen sus valores muy rápidamente porque su cálculo es elemental.

La verdad es que no hay de qué alarmarse puesto que debido a los valores estándar que se usan de p, q y e en la generación de claves reales, aunque en ciertos casos existan muchas CPP, sus valores serán todos muy próximos al valor del módulo n y, por lo tanto, será imposible adivinarlos o intentar romperlos por fuerza bruta.

¿Qué pasará ahora con las claves públicas parejas? La situación en este caso es muy similar, las claves públicas parejas tendrán valores muy altos. Además, la clave es por definición pública y no tiene ningún secreto. No obstante, existe un escenario de firma digital como el que se comenta a continuación que nos puede dejar ciertas dudas. Supongamos que un usuario firma el hash de un mensaje con su clave privada, lo que envía a destino junto con el mensaje original pues no es necesario agregarle confidencialidad. En destino se descifrará el criptograma con la clave pública del emisor, recuperando por lo tanto el hash firmado en emisión. Se calcula el hash del mensaje recibido y se comparan los dos hashes, el de emisión y el recuperado en recepción, para dar validez a dicha firma.

Hasta aquí todo es correcto pero, ¿qué pasaría ahora si mediante algún artilugio alguien puede comprobar la firma usando una clave pública pareja distinta a la clave pública e conocida por todo el mundo? Esto podría interpretarse como una muestra de debilidad del sistema, en tanto nadie conoce la existencia de estas claves públicas parejas y consecuentemente podrían, incluso, dudar de las bondades del algoritmo. Pero será igual de difícil adivinar una clave pública pareja al ser todas (excepto el valor de e) muy grandes y cercanas al módulo n.

Para minimizar las claves parejas y reducirlas a la unidad, habrá que usar primos seguros. Se dice que p es un primo seguro si p = 2*q +1 siendo q primo. Por ejemplo, el 83 es un primo seguro porque 83 = 2*41 + 1, donde 41 es un primo. No obstante, aunque no se usen primos seguros y la clave en cuestión presente varias CPP, se concluye que las claves privadas y públicas parejas no son una vulnerabilidad del sistema RSA, es tan sólo una curiosidad. Por esto mismo, programas tan populares como OpenSSL no tienen en cuenta esto y no usan primos seguros al generar claves RSA.



Números no cifrables

Puesto que la operación de cifrado RSA de un número N es N^{clave} mod n, donde clave puede ser la clave pública de destino o la clave privada de emisión, según el tipo de cifra que se desee realizar, y siendo los elementos N a cifrar todos los valores del cuerpo n, que van desde 0 hasta n-1, existirán algunos números Ni que, aunque se cifren, irán en claro. Como es evidente, el valor de la clave en ese exponente puede ser cualquier número entre 2 y n-2 que cumpla los requisitos que correspondan a dicha clave.

Dos casos obvios de números que se cifran y van en claro serán el 0 y 1 puesto que:

0^{clave} mod n = 0

1^{clave} mod n = 1

Otro valor no tan obvio pero que puede demostrarse matemáticamente que se transmite en claro es el número n-1 puesto que:

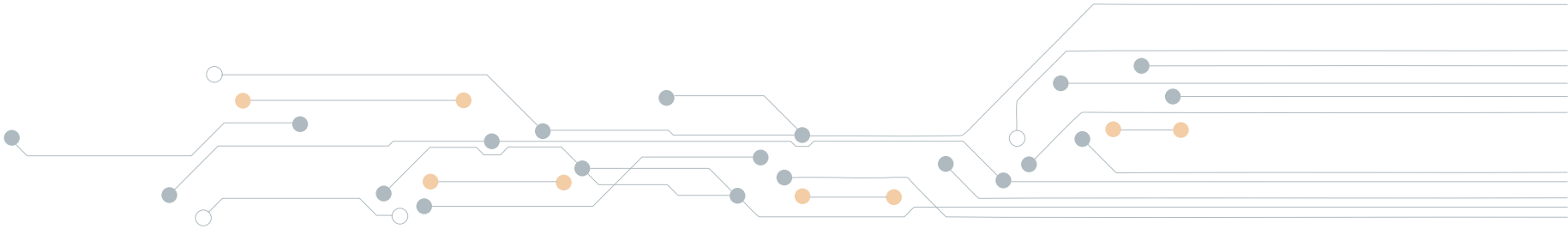
$n-1^{clave}$ mod n = n-1.

Por ejemplo para la clave RSA con n = 323 y e = 11, se obtiene:

0^{11} mod 323 = 0

1^{11} mod 323 = 1

322^{11} mod 323 = 322



Además de esos tres valores, en RSA existirán siempre como mínimo otros 6 valores que irán en claro. Por tanto, cualquier clave RSA tendrá como mínimo 9 números no cifrables.

En el ejemplo anterior, los otros seis valores que se transmiten en claro son:

$18^{11} \bmod 323 = 18$

$152^{11} \bmod 323 = 152$

$153^{11} \bmod 323 = 153$

$170^{11} \bmod 323 = 170$

$171^{11} \bmod 323 = 171$

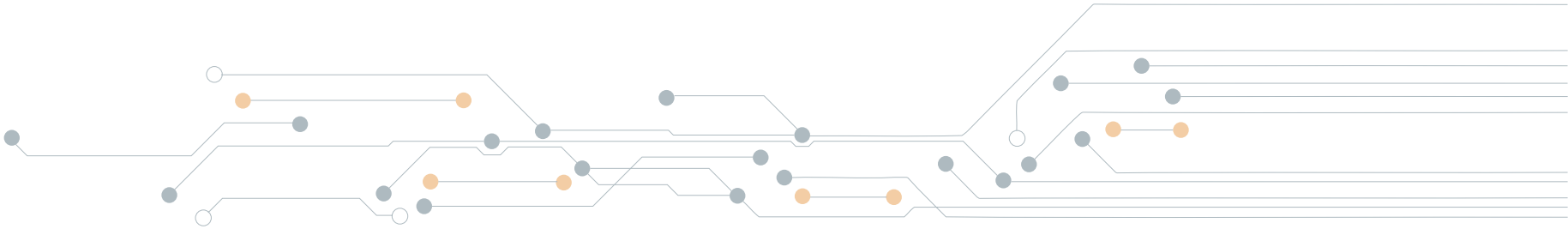
$305^{11} \bmod 323 = 305$

A diferencia de las claves privadas y públicas parejas, cuyos valores se obtenían de forma inmediata mediante una ecuación en la que existía una separación constante entre dichas claves, en este caso el cálculo de los números no cifrables NNC es más laborioso porque habrá que hacer un ataque por fuerza bruta en el espacio de los primos p y q, para comprobar qué valores de X nos entregan las siguientes igualdades:

$X^e \bmod p = X \quad$ para $1 < X < p-1$

$X^e \bmod q = X \quad$ para $1 < X < q-1$

Como es de esperar, para claves reales de 1.024 o más bits, resulta computacionalmente imposible realizar esos cálculos dentro de los



primos p y q de 512 bits cada uno, pues habrá que hacer 2512 operaciones en p y otras tantas en q. Por lo tanto, para claves reales, excepto los valores 0, 1 y n-1, no será posible encontrar los demás números no cifrables.

A continuación se indican las ecuaciones que intervienen en el cálculo de estos números no cifrables NNC. La única dificultad de cálculo se encuentra en las dos últimas ecuaciones, que significan aplicar fuerza bruta con todos los valores de N candidatos a ser número no cifrable, siendo $1 < N < p-1$ para el primo p y $1 < N < q-1$ para el primo q.

La cantidad de números no cifrables dentro de un cuerpo n será:

$$\sigma_n = [1 + mcd\ (e-1, p-1)][1 + mcd\ (e-1, q-1)]$$

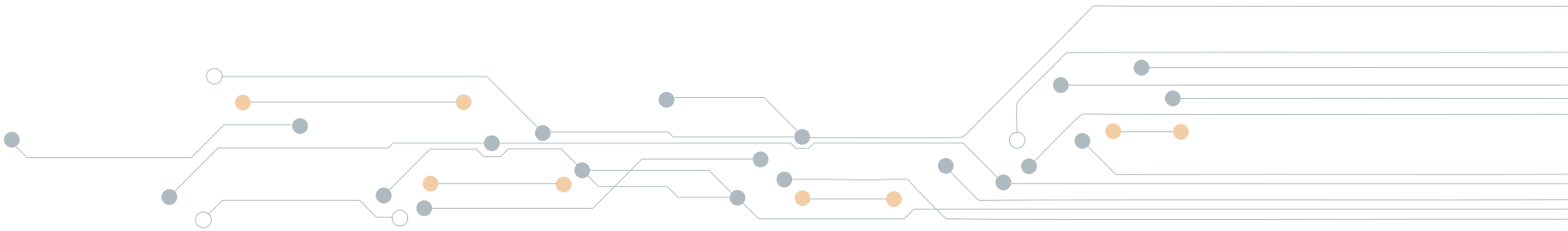
Y los valores de esos números no cifrables serán:

$$N = [q\{inv\ (q, p)\}N_p + p\{inv\ (p, q)\}N_q] \bmod n$$

con: N_p las soluciones de $N^e \bmod p = N$

N_q las soluciones de $N^e \bmod q = N$

Sea p = 13; q = 17; n = p*q = 221; e = 7; d = inv (7, 192) = 55. Vamos a encontrar los números no cifrables de esta clave RSA:



La cantidad de números no cifrables σ_n de la clave será:

$\sigma_n = [1 + mcd (e-1, p-1)][1 + mcd (e-1, q-1)]$

$\sigma_{221} = [1 + mcd (6, 12)][1 + mcd (6, 16)] = (1+6)(1+2) = 21$

Los números no cifrables serán:

NNC = [q{inv (q, p)}Np + p{inv (p, q)}Nq] mod n

NNC = [17{inv (17, 13)}Np + 13{inv (13, 17)}Nq] mod 221

inv (17, 13) = inv (4, 13) = 10

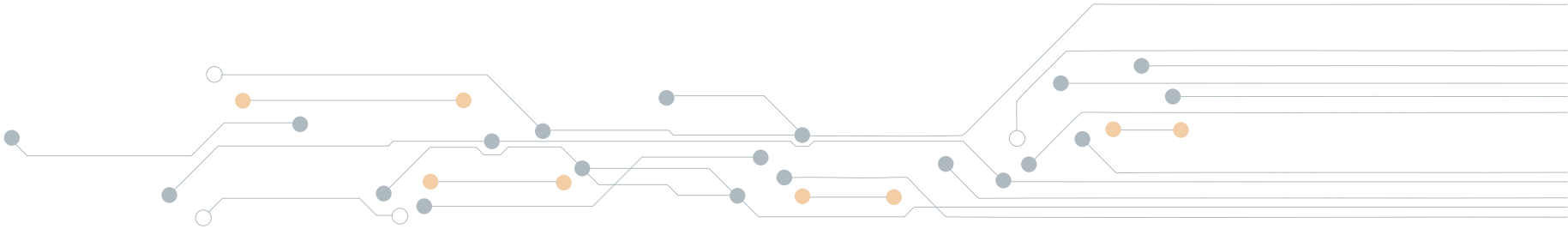
inv (13, 17) = 4

NNC = [{17*10}Np + {13*4}Nq] mod 221

NNC = [170*Np + 52*Nq] mod 221

Por fuerza bruta encontramos las soluciones de $Np = N^7 \bmod 13 = N$ para $1 < N < 12$: Np = {0, 1, 3, 4, 9, 10, 12}

Por fuerza bruta encontramos las soluciones de $Nq = N^7 \bmod 17 = N$ para $1 < N < 16$: Nq = {0, 1, 16}



Luego:

NNC = [170*{0, 1, 3, 4, 9, 10, 12} + 52*{0, 1, 16}] mod 221

NNC = [{0, 170, 510, 680, 1.530, 1.700, 2.040} +{0, 52, 832}] mod 221

NNC = [0+0, 0+52, 0+832, 170+0, 170+52, 170+832, ..., 2.040+832] mod 221

NNC = [0,52,832,170,222,1.002,510,562,1.342,680,732,1.512,1.530,1.582,2.362,1.700,1.752,2.531,2.040,2.092,2.872] mod 221

Reduciendo cada uno de esos 21 números módulo 221 obtenemos:

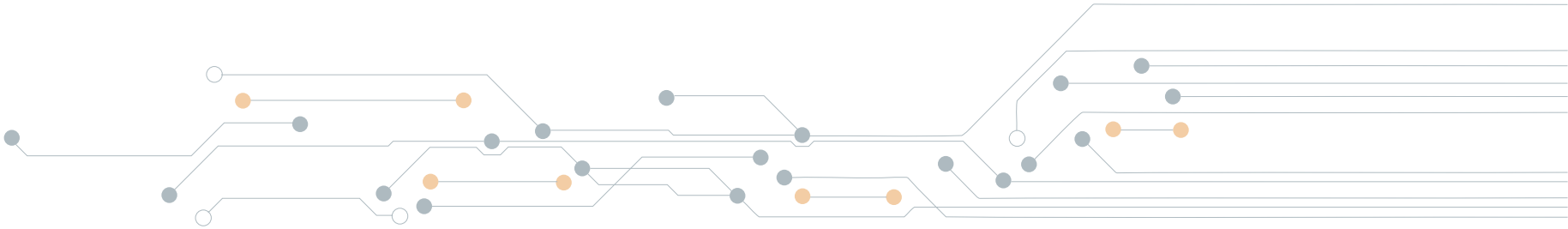
NNC = [0, 52, 169, 170, 1, 118, 68, 120, 16, 17, 69, 186, 204, 35, 152, 153, 205, 101, 51, 103, 220]

Y ordenando de menor a mayor:

NNC = [0, 1, 16, 17, 35, 51, 52, 68, 69, 101, 103, 118, 120, 152, 153, 169, 170, 186, 204, 205, 220]

De los resultados del ejercicio anterior podemos observar dos características interesantes de estos números no cifrables en RSA.

1. Que aparecen varios números consecutivos: (0, 1); (16, 17); (51, 52); (68, 69); (152, 153); (169, 170); (204, 205), algo que se repite por lo general.
2. Que excepto el valor 0, los valores de los extremos siempre sumarán el valor del módulo n, en este caso:
1+220 = 16+205 = 17+204 = 35+186 = 51+170 = 52+169 = 68+169 = 69+153 = 101+120 = 103+118 = 221 = n.



A pesar de este comportamiento tan peculiar, ninguna de estas características debería interpretarse, al menos en principio, como una vulnerabilidad.

Para minimizar la cantidad de números no cifrables y reducirlos a su valor mínimo 9, también habrá que usar primos seguros.

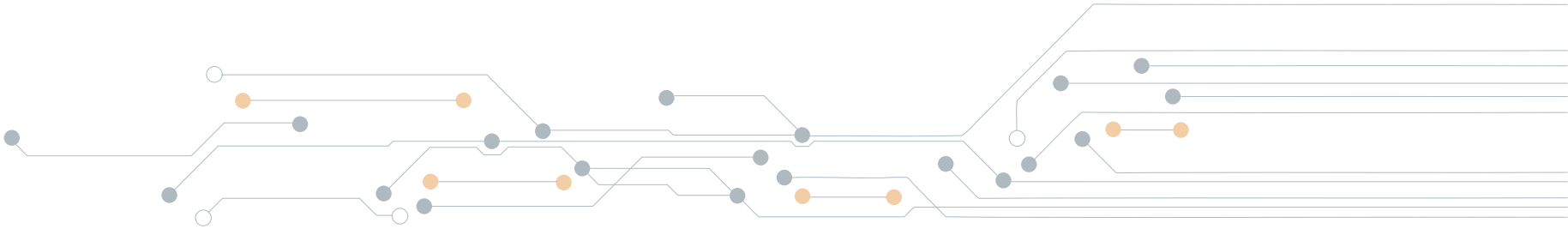
Así, una clave RSA que tena una única clave privada pareja y nueve números no cifrables se conocerá como clave óptima.

Una vez conocidas las condiciones de diseño para que una clave RSA tenga la menor cantidad posible de números no cifrables, la pregunta que nos acecha es ¿cuál será la cantidad mayor de estos NNC?

El peor de los escenarios será cuando:

- mcd (e - 1, p - 1) = p – 1
- mcd (e - 1, q - 1) = q – 1

En estas condiciones σ_n = [1 + mcd (e - 1, p - 1)] [1 + mcd (e - 1, q - 1)] = [1 + (p -1)] [1 + (q - 1)] = p*q = n. Así, todos los elementos del cuerpo n = p*q irán en claro. Esto nos indica que una vez elegidos p y q, hay que tener especial cuidado en la elección de la clave pública e.



Una clave RSA con p = 13, q = 17, Φ_n = 192, n = 221 y e = 11 será óptima al tener 9 números no cifrables en el cuerpo n = 221. Pero si hubiésemos elegido como clave pública el valor e = 49, igual de válido que 11, observamos ahora lo siguiente:

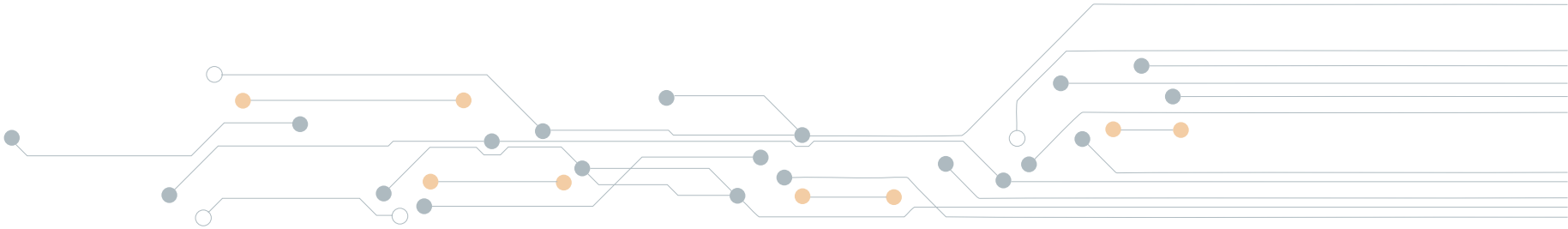
- mcd (e - 1, p - 1) = mcd (48, 12) = 12
- mcd (e - 1, q - 1) = mcd (48, 16) = 16

Por lo tanto, σ_{221} = [1 + 12][1 + 16] = 13*17 = 221 = n. Es decir, cualquier número que se cifre irá en claro.

¿Por qué ha pasado esto? En el ejemplo anterior, observamos que e = 49 = 192/4 +1 = Φ_n/4 + 1. En general, si elegimos un valor de clave pública e = Φ_n/k + 1 para valores de k igual a 2, 3, 4, ... siempre pequeños, la cantidad de NNC puede ser muy alto, incluso todo el cuerpo como acaba de suceder para e = 49 y que se va a repetir en esta clave RSA para e = 97.

Si σ(n) = n, como todo el cuerpo de cifra irá en claro, entonces el valor de la clave pública se repetirá en la clave privada o en una clave privada pareja. Si llega a cumplirse que e = Φ_n/2 + 1, entonces σ(n) = n, será no cifrable cualquier valor de n y además la clave privada d tendrá el mismo valor que la clave pública e. Por ejemplo, la clave con p = 199, q = 239, n = 47.561, Φ_n = 47.124, e = Φ_n/2 + 1 = 23.563, tiene todos sus elementos no cifrables y la clave privada d = 23.563, igual que la pública.

En condiciones reales en que la clave pública e es el número 4 de Fermat, como n y Φ_n serán números de al menos 1.024 bits, entonces en el supuesto caso de que se cumpliese la relación e = Φ(n)/k + 1 el valor de k sería tan grande que la cantidad de números no cifrables nunca llegará a estos extremos.



El caso que más nos preocupa es que un número secreto que cifremos con RSA para un intercambio de clave, al final vaya en claro. Pero los números que cifraremos en la práctica pueden ser de estos dos tipos:

- Una clave de sesión para su uso en un algoritmo simétrico y que estará entre los 128 y 256 bits, en lo que se conoce como intercambio de clave con RSA usando la clave pública del receptor.
- Un número resultado de aplicar a un documento una función hash de 160 o 256 bits, en lo que se conoce como firma digital RSA usando la clave privada del emisor.

En ambos casos son números de tan sólo unas centenas de bits dentro de un cuerpo de cifra que actualmente es de 1.024 o 2.048 bits. Por lo tanto, la posibilidad de que dentro de un cuerpo n de 1.024 bits un conjunto de números secretos caiga precisamente en la zona de centenas de bits, es decir, sea un número con una gran cantidad de ceros a la izquierda y sólo cien o doscientos bits significativos, es extremadamente pequeña. Pero, así y todo, si aceptamos que hay varios números de ese conjunto no cifrable de la clave RSA que tienen sólo una o dos centenas de bits significativos, la probabilidad ahora de que alguno de ellos sea precisamente el número secreto de 128 bits que hemos elegido para ese intercambio de clave K es de $1/2^{128}$, un valor completamente despreciable.

Como conclusión, vemos también que para valores de diseño estándar de una clave RSA por sobre los mil bits, tampoco será un motivo de preocupación la existencia de una gran cantidad de números no cifrables, Es decir, tampoco estamos frente a una vulnerabilidad.

Ataques a RSA

La seguridad del algoritmo RSA puede atacarse de tres maneras:

- Ataque por la factorización entera del cuerpo de cifra.
- Ataque por la paradoja del cumpleaños.
- Ataque por cifrado cíclico.

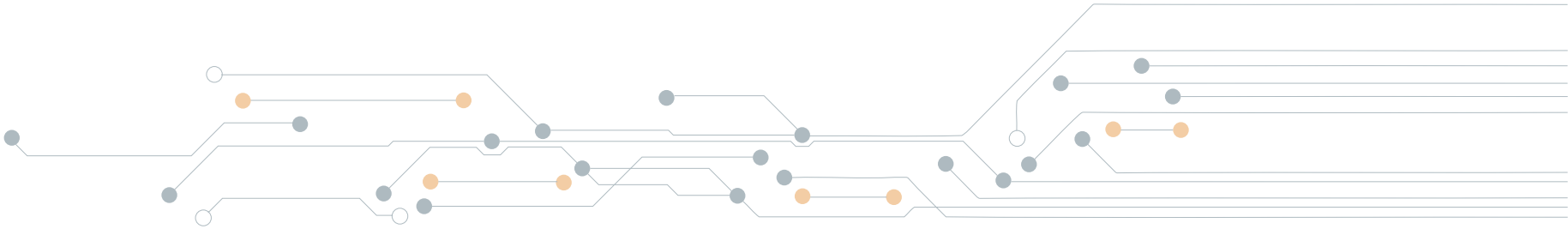
Los ataques por factorización y paradoja del cumpleaños permitirían romper RSA al encontrar la clave privada, no aplicando fuerza bruta.

Al factorizar el módulo n y conocer así los primos p y q, se podrá calcular la clave privada d simplemente usando el algoritmo extendido de Euclides con la clave pública e y el indicador de Euler Φ_n.

En el caso del ataque mediante la paradoja del cumpleaños, se encontrará directamente el valor de la clave privada. Lo interesante de este ataque es que permite paralelizar la acción y aplicar así el principio de "divide y vencerás", igual como sucedía con los algoritmos de cifra simétrica.

Por su parte, un ataque mediante el cifrado cíclico, permitiría encontrar el mensaje secreto cifrado con RSA, sin necesidad de conocer la clave privada del destinatario o víctima, pero a diferencia de los otros dos no se romperá el sistema de cifra.

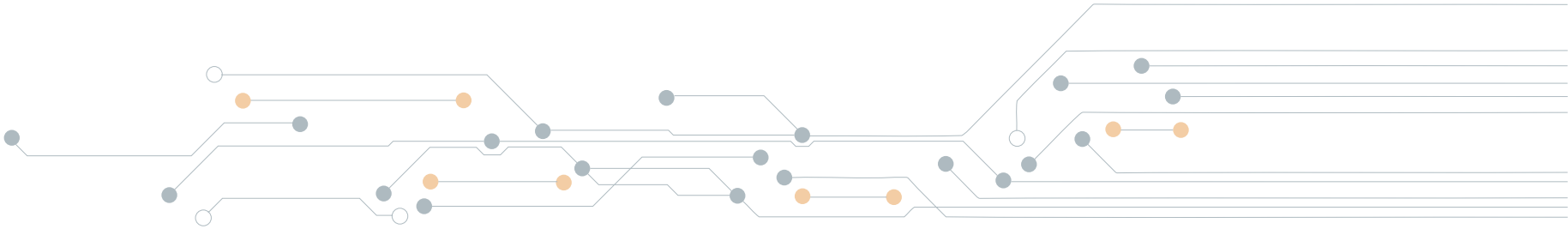
Hay que tener en cuenta que para las claves RSA actuales, con un cuerpo de cifra de al menos 1.024 bits, ninguno de estos tres ataques son viables.



Factorización

Los algoritmos de factorización pueden dividirse en dos grupos, los denominados de propósito general y los de propósito específico. Independientemente de ello, entre los más conocidos se encuentran:

1. Método de factorización directa o criba de Eratóstenes
2. Método de Fermat
3. Método de Euler
4. Método de Dixon
5. Método de Williams p+1
6. Método de Pollar rho
7. Método de Pollar p-1
8. Método de las fracciones continuas
9. Método de las curvas elípticas
10. Método de la criba numérica



El mejor algoritmo conocido a la fecha, el de la criba numérica o *General Number Field Sieve* GNFS, tiene asociada una complejidad representada en la siguiente expresión para un número de b bits:

$$O\left(\exp\left(\left(\tfrac{64}{9}b\right)^{\frac{1}{3}}(\log b)^{\frac{2}{3}}\right)\right)$$

Donde el factor b que representa en bits el tamaño del número a factorizar, se encuentra como un elemento del exponente y por ello el carácter exponencial de este problema.

El problema de la factorización entera es uno de esos problemas en matemáticas clasificados como abiertos; es decir, siempre puede aparecer un nuevo algoritmo que mejore las prestaciones de los anteriores o bien que presente ciertas ventajas frente a aquellos en casos especiales. Lo que es cierto es que a la fecha nadie ha logrado quitarle esa característica de problema NP o comportamiento exponencial.

Desde 1991 RSA Laboratories propone una serie de desafíos de factorización de diferentes módulos RSA con premios en metálico, pero en el año 2007 da por terminado ese desafío, poco después de que en el año 2005 se factorizasen números de 193 dígitos (640 bits) y 200 dígitos (663 bits). Ya fuera del concurso con estos premios, el 12 diciembre de 2009 un equipo de seis instituciones de investigación lideradas por Thorsten Kleinjung logran factorizar el mayor número RSA hasta el momento: 768 bits.

Paradoja del cumpleaños

En el año 1981 Ralph Merkle y Martin Hellman proponen un método para atacar al algoritmo DES, *Data Encryption Standard*, en su modalidad de doble cifrado con texto en claro y criptograma conocidos, basado en la paradoja del cumpleaños y que deriva finalmente en lo que conocemos como ataque por encuentro a medio camino, *meet in the middle*, que no hace aconsejable este tipo de cifrado doble.

Este mismo principio nos permitirá encontrar colisiones en las cifras realizadas con RSA. El ataque tiene la particularidad de que al prosperar tras observarse una colisión, nos entregará la clave privada de la víctima, una clave privada pareja, y en muy pocos casos un falso positivo.

El ataque sigue estos pasos

- Se conoce el módulo n y la clave pública e de la víctima.
- Se elige un valor o número N de ataque aleatorio cualquiera, siendo recomendable que sea el número 2.
- Se divide el número del módulo n en dos mitades, una izquierda que denominaremos i y otra derecha que denominaremos j.
- La zona izquierda realizará cifrados del valor N para exponentes i en el intervalo $0 \leq i \leq n/2$ en módulo n.
- La zona derecha realizará cifrados del valor N para exponentes j en el intervalo $n/2 + 1 \leq j \leq n-1$ en módulo n.
- Para $i = i+1$ y $j = j+1$ calcula $N^i \bmod n$ y $N^j \bmod n$.
- Se repite esta operación hasta que un resultado de N_j colisiona con el primer resultado de N_i o bien un resultado de N_i colisiona con el primer resultado de N_j.
- Encontrada la colisión, el algoritmo llega a su fin y puede encontrarse la clave buscada mediante la resolución de un conjunto de ecuaciones.

Sea p = 7; q = 17, n = 119, e = 11. Aunque la clave privada sea d = 35 (y una clave privada pareja 83), el atacante sólo conoce los valores públicos de la víctima n = 119 y e = 11. El atacante usará N = 2 y los valores iniciales de los contadores serán i = 0 y j = 59.

i = 0 2^0 mod 119 = 1	j = 59 2^{59} mod 119 = 25
i = 1 2^1 mod 119 = 2	j = 60 2^{60} mod 119 = 50
i = 2 2^2 mod 119 = 4	j = 60 2^{61} mod 119 = 100
i = 3 2^3 mod 119 = 8	j = 62 2^{62} mod 119 = 81
i = 4 2^4 mod 119 = 16	j = 63 2^{63} mod 119 = 43
i = 5 2^5 mod 119 = 32	j = 64 2^{64} mod 119 = 86
i = 6 2^6 mod 119 = 64	j = 65 2^{65} mod 119 = 53
i = 7 2^7 mod 119 = 9	j = 66 2^{66} mod 119 = 106
i = 8 2^8 mod 119 = 18	j = 67 2^{67} mod 119 = 93
i = 9 2^9 mod 119 = 36	j = 68 2^{68} mod 119 = 67
i = 10 2^{10} mod 119 = 72	j = 69 2^{69} mod 119 = 15
i = 11 2^{11} mod 119 = 25	j = 70 2^{70} mod 119 = 30

Se observa que colisionan el último cifrado en i con el primer cifrado en j. Como la coincidencia del resultado 25 se encuentra para i = 11 y para j = 59, el atacante calcula:

w = (11-59) / mcd (11, |11-59|) = -48 / mcd (11, 48) = - 48

Entonces deberán existir valores s, t de forma que se cumpla lo siguiente:

w*s + e*t = 1

-48*s + 11*t = 1

Las posibles soluciones a las ecuaciones son:

w*s mod e = 1

e*t mod w = 1

Por lo tanto:

-48*s mod 11 = 1, por lo tanto s = inv (-48, 11) = inv (7, 11) = 8

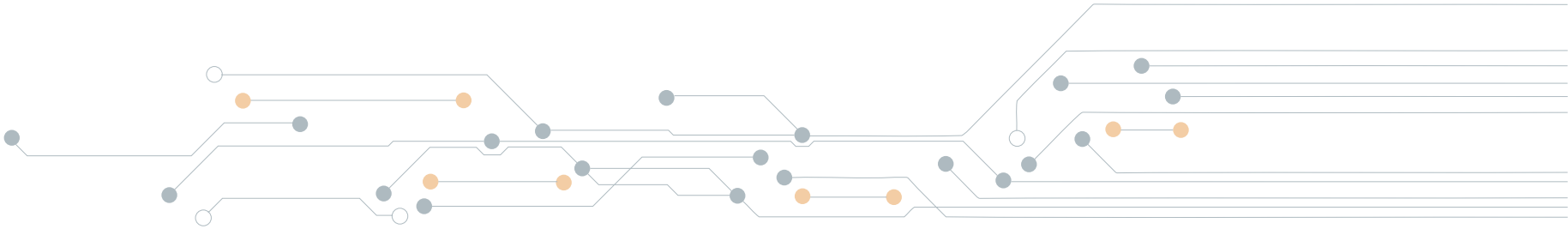
11*t mod 48 = 1, por lo tanto t = inv (11, 48) = 35

El valor 35 es la clave privada buscada.

Observa que a igual resultado se llega, pero con una colisión distinta, ahora el último cifrado en j colisiona con el primer cifrado en i, si los valores iniciales de los contadores son i = 0 y j = 47. Además, la cantidad de pasos realizados en el ataque han sido sólo dos. Más aún, si hubiésemos elegido j = 48, habríamos dado con la solución en el primer intento.

i = 0 2^0 mod 119 = **1**	j = 47 2^{47} mod 119 = 60
i = 1 2^1 mod 119 = 2	j = 148 2^{48} mod 119 = **1**

Como la coincidencia del resultado 4 se encuentra para i = 2 y para j = 50, el atacante calcula w = (2-50) / mcd (11, |2-50|) = -48 / mcd (11, 48) = - 48 y las ecuaciones son las mismas.



Cifrado cíclico

Se supone que un mensaje secreto cifrado con la clave pública del destino, típica operación en el intercambio de clave, sólo podía descifrarse y recuperar ese secreto usando la clave privada de ese destino, es decir la clave inversa de la usada en el proceso de cifra, o bien con alguna clave privada pareja.

No obstante, también será posible romper ese secreto usando solamente las claves públicas de la víctima, algo que estaría en contra de toda la lógica de los sistemas de cifra asimétrica o de clave pública. El problema que se nos presenta es que entonces cualquiera que conozca las claves públicas usadas en el proceso de intercambio de clave, podría en teoría recuperar el secreto.

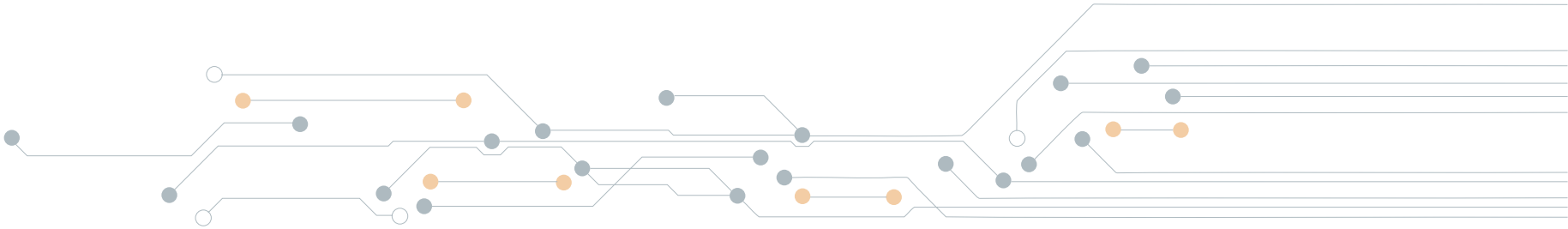
Se trata de lo siguiente. Como C = Ne mod n, con N un valor secreto, vamos a realizar cifrados sucesivos de los criptogramas Ci resultantes con la misma clave pública e. Si en uno de estos cifrados obtenemos nuevamente el cifrado C original con el que se ha iniciado el ataque, resulta obvio que el valor del paso anterior será el secreto N buscado. Esto se debe a que en RSA se generan anillos, con longitudes distintas y en los que se encuentran un conjunto de números de ese módulo, sin repetirse. Los números de todos esos anillos forman el cuerpo de cifra o módulo n.

Por tanto, conocido o capturado el criptograma C, realizaremos los siguientes cifrados:

$Ci = C^{ei-1} \bmod n$; para i = 1, 2, ..., con $C_0 = C$

Si en el cifrado iésimo se encuentra el criptograma C inicial, entonces el cifrado anterior (i-1) será el número secreto, como se muestra en el siguiente ejercicio. Sea p = 37, q = 61, n = 2.257, e = 7 (d = 1.543). El valor secreto es N = 50 que se cifra como 507 mod 2.257, dando como criptograma C = 1.475. Atacando por cifrado cíclico, se obtiene:

- **1.475**^{7} mod 2.257 = 1.758
- 1.7587 mod 2.257 = 2.207
- 2.2077 mod 2.257 = 782
- 7827 mod 2.257 = 499
- 4997 mod 2.257 = 50 (el atacante todavía no conoce el secreto)
- 507 mod 2.257 = **1.475** (el atacante conoce el secreto 50)



4. El algoritmo de Elgamal

En 1985 el investigador egipcio Taher Elgamal propone sendos algoritmos de cifra y firma digital que llevan su nombre. Una de sus características es que se envían dos valores dentro del cuerpo de cifra, un primo muy grande. Si bien el algoritmo es seguro, no logra desbancar a RSA como estándar de cifra, aunque una variación de su algoritmo de firma conocido como DSS es actualmente un estándar de firma digital por el NIST.

Como el algoritmo de Elgamal se basa en el problema del logaritmo discreto, cada usuario elegirá como cuerpo de cifra un número primo p grande y una raíz primitiva a, ambos valores públicos. Hecho esto, se elige como clave privada un número aleatorio l dentro de p, y se calcula la clave pública como $\beta = \alpha^{\lambda} \bmod p$.

Al igual que en Diffie y Hellman, la seguridad del sistema reside en que para descubrir la clave privada l, conocidos los valores públicos p, α y β, el atacante deberá enfrentarse al problema del logaritmo discreto en un primo grande, algo computacionalmente intratable para valores cercanos a los mil bits con la capacidad de cómputo actual.

Para enviar un número N con confidencialidad de Alicia a Bernardo, se procede de la siguiente manera:

Emisión (Cifrado de Alicia):

- Genera un número aleatorio ν de sesión y calcula $\alpha_B{}^{\nu} \bmod p_B$.
- Con la clave pública de Bernardo β_B calcula $N*\beta_B\nu \bmod p_B$
- Envía a Bernardo el par de números $\{\alpha_B{}^{\nu} \bmod p_B, N*\beta_B{}^{\nu} \bmod p_B\}$

Recepción (Descifrado de Bernardo):

- Bernardo con su clave privaba b calcula $(\alpha_B{}^{v})b \bmod p_B$
- Como $\beta_B = \alpha_B{}^{b} \bmod p_B$, entonces $N*\beta_B{}^{v} \bmod p_B = N*(\alpha_B{}^{b})^{v} \bmod p_B$
- Como $(\alpha_B{}^{v})b \bmod p_B = (\alpha_B{}^{b})^{v} \bmod p^B$ estará permitido hacer esta división:
  $N*(\alpha_B{}^{b})^{v} \bmod p_B / (\alpha_B{}^{v})^{b} \bmod p_B = N$ porque ésta se expresa como:
  $N*(\alpha_B{}^{b})^{v} * inv[(\alpha_B{}^{v})^{b}, p_B] = N$, y se recupera el secreto

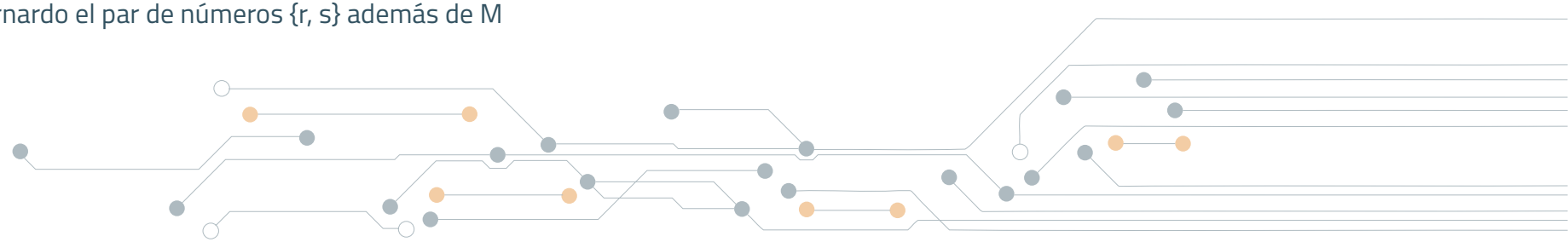
Si lo que se desea es firmar digitalmente un mensaje de Alicia para Bernardo, se procede de la siguiente manera:

Emisión (Firma de Alicia):

- Genera un número h que será primo relativo con $\phi(p_A)$ es decir mcd $\{h, \phi(p_A)\} = 1$
- Calcula inv $\{h, \phi(p_A)\}$
- Calcula $r = \alpha_A{}^{h} \bmod p_A$
- Resuelve la congruencia $M = a*r + h*s \bmod \phi(p_A)$ es decir:

  $s = (M - a*r)*inv[h,\phi(p_A)] \bmod \phi(p_A)$

- Y envía a Bernardo el par de números {r, s} además de M



Recepción (Comprobación de la firma por Bernardo):

- Usando la clave pública de Alicia (p_A, α_A y β_A), Bernardo calculará:

  $r^s \bmod p_A$ y $(\beta_A)^r \bmod p_A$
- Calcula además $k = [(\beta_A)^r * r^s] \bmod p_A$
- Como $r = \alpha_A{}^h \bmod p_A$ y además $\beta_A = \alpha_A{}^a \bmod p_A$, entonces:

  $k = [\alpha_A{}^{ar} * \alpha_A{}^{hs}] \bmod p_A = \alpha_A{}^{(ar + hs)} \bmod p_A$
- Como $M = (a*r + h*s) \bmod \phi(p_A)$ y α_A es una raíz primitiva de p_A, se cumple que:

  Si $k = \alpha_A{}^M \bmod p_A$ entonces la firma es válida

En la práctica lógicamente no se firma un mensaje M sino su función hash h(M).

Telefónica EDUCACIÓN DIGITAL