UNIVERSITAT JAUME I

Grado en Matemática Computacional

Estancia en Prácticas y Proyecto Final de Grado

Criptografía de clave simétrica y de clave asimétrica

Autor:
Christian Bea Monreal

Supervisor:
Miguel Chover Selles
Tutor académico:
Alejandro Miralles Montolio

Fecha de lectura: 27 de julio de 2020
Curso académico 2019/2020

Resumen

En este trabajo se presenta la memoria de la estancia en prácticas y el trabajo de final de grado de la carrera de Matemática Computacional de la Universitat Jaume I de Castellón.

La estancia en prácticas se desarrolla principalmente en torno a la realidad virtual, una tecnología en auge y con mucha proyección de futuro.

La memoria del trabajo de final de grado trata acerca de la criptografía. Dada su importancia a lo largo de la historia, es un buen tema para mostrar los conceptos básicos de la misma junto con su base matemática para poder entender su funcionamiento. Se trata en primer lugar los fundamentos de la aritmética modular, lo que nos permite acercarnos a problemas matemáticos de gran utilidad en criptografía como pueden ser la generación de números aleatorios o la factorización de enteros. Esta base nos permitirá introducir los dos tipos principales de cifrado: el simétrico y el asimétrico. Se trabajará con estos dos tipos de cifrado y, por último, se dará una pincelada al futuro de la criptografía.

Palabras clave

Aritmética modular, Criptografía, Cifrado, Clave.

Keywords

Modular arithmetic, Cryptography, Encryption, Key.

Índice general

Parte I

Estancia en prácticas

Capítulo 1

Estancia en prácticas

1.1. Entidad

La estancia en prácticas se ha realizado entre los meses de octubre de 2019 y abril de 2020 en el Instituto de Nuevas Tecnologías de la Imagen (INIT) [1] situado dentro de la Universitat Jaume I (UJI) [2] de Castellón de la Plana, en concreto se han hecho en el Centro de Visualización Interactiva (CEVI) [3].

Los temas que se tratan en este centro son principalmente la visualización interactiva (juegos por ordenador, creación automática de contenidos 3D y visualización realista) y la realidad virtual y aumentada (dispositivos y técnicas de interacción, de visualización y de seguimiento del usuario).

1.2. Objetivos del proyecto formativo

Definición 1.1. *En informática, una librería es un conjunto de implementaciones, en un lenguaje de programación determinado, que ofrece una interfaz para una funcionalidad concreta.*

El objetivo del proyecto ha sido el estudio de librerías para la creación de juegos multijugador con el fin de desarrollar la parte multijugador de algún juego (ya creado) en realidad virtual. Las formas mediante las que se suele aprender a usar librerías es tanto mediante tutoriales que se pueden encontrar en internet como por la documentación oficial de las mismas.

En caso de finalizar esta tarea antes de completar la cantidad de horas que se debían dedicar a la estancia se buscaría alguna otra tarea a desarrollar.

1.3. Tecnologías empleadas

Durante el desarrollo de la estancia en prácticas se han utilizado las siguientes tecnologías:

- Motor de juegos, Unity [4], además de diversos manuales oficiales.
- Herramienta para la comunicación entre servidores, Photon [5].
- Tecnología de realidad virtual, VIVE [6].
- Lenguaje de programación C#.

1.4. Explicación detallada del proyecto realizado

Para la tarea de realizar la parte multijugador de un juego en realidad virtual se necesitan conocimientos de programación, de motores de juegos, de comunicación entre servidores (o computadoras) y de realidad virtual. Por todo ello, supone una buena parte del tiempo la dedicación a la preparación previa al desarrollo del proyecto. Teniendo en cuenta que tengo experiencia previa con la programación (durante la carrera y por mi cuenta) y también lo básico con el motor de juegos Unity (por cuenta propia), el tiempo dedicado a la preparación previa fue relativamente corto. Además ya disponía de algún conocimiento sobre el multijugador en este motor a través de lo llamado Unet, que es una herramienta incorporada en el mismo Unity para el desarrollo multijugador, pero surgió un problema. Tratando de retomar el conocimiento sobre el funcionamiento de esta herramienta a través de la web de Unity, me di cuenta que esta funcionalidad iba a desaparecer del motor, de forma que antes de 2022 iba a dejar de ser servible.

Tras el percance y comentarlo con el supervisor, debido a que desarrollarlo con esta herramienta iba a suponer que dejara de ser servible en poco tiempo, había que encontrar y aprender sobre una librería con la misma funcionalidad. Esta herramienta finalmente fue Photon, que permite las cosas necesarias como recrear un lobby, es decir, una escena del juego donde "se reúnen" jugadores, de forma que alguno puede crear una sala que será visible para otros jugadores y que puedan unirse a ella para jugar juntos a una aplicación creada con Unity.

Tras aprender lo suficiente sobre Photon el objetivo es comprender el código del juego original (que se trata de un juego tipo Air Hockey contra una inteligencia artificial y en realidad virtual, desarrollado por un alumno de la carrera de Diseño y Desarrollo de Videojuegos), que como tiene que ver con la realidad virtual, hace más costosa la comprensión del código. Supone además que se tenga que realizar un pequeño tutorial sobre las nociones básicas de realidad virtual para Unity. Una vez comprendido, se procede a hacer más entendible el código (de una copia del juego, para no modificar el original), incluyendo comentario y reordenando algunas partes.

Siguiendo un tutorial para la creación de un lobby y junto con la ayuda de algún código de demostración de Photon, además de teniendo que adaptar algunas cosas de la realidad virtual, se consigue adaptar el juego, de modo que dos jugadores puedan conectarse a un lobby, unirse a

la misma sala y jugar uno contra el otro. Aún sin tener montado el sistema de realidad virtual se puede probar el juego ya que en Unity se puede adaptar el juego en realidad virtual para verlo por la pantalla del ordenador y que el ratón actúe como mando.

El siguiente paso que se realizó fue aprender a montar el sistema de realidad virtual, uno de los de HTC VIVE. Conlleva unas horas preparar el ordenador (aunque ya estaba prácticamente preparado debido a que ya lo habían utilizado anteriormente para la realidad virtual) y realizar todo el montaje de cables y demás, al menos la primera vez. Existen algunos tutoriales para poder realizar todo el proceso.

Sin embargo, el problema surge a la hora de la sincronización de los objetos, tanto de lo que mueve cada jugador como del objeto que es golpeado en el juego, ya que para este juego es necesaria mucha sincronización, pero la herramienta Photon, al menos la versión gratuita, no permite la que se necesita, de forma que tras bastante tiempo buscando soluciones se decide junto con el supervisor dejarlo como está y continuar con otra cosa.

Al supervisor se le presentó retomar un proyecto y esa fue mi siguiente tarea. Con anterioridad habían desarrollado en realidad virtual una escena simple, una habitación con una silla en el centro. El objetivo de los que pidieron que se realizara esta escena era hacer un pequeño estudio sobre algunas sensaciones de la gente que usaba las gafas de realidad virtual. Mi trabajo fue, a partir de esta escena, crear una aplicación móvil lo más similar posible a la escena, que permitiera girar la cámara alrededor de la silla, además de poder acercar la cámara y alejarla, haciendo una pequeña simulación de si tuvieras las gafas de realidad virtual, para repetir el estudio y comparar los dos resultados. La dificultad principal fue aprender a como realizar aplicaciones móvil para Unity, pero a través de los manuales de la web de Unity fue sencillo.

Esta última tarea fue finalizada justo antes de la suspensión de las clases presenciales, así como prácticamente todo lo relacionado con la universidad, a partir del 16 de marzo debido a la crisis sanitaria provocada por el Coronavirus COVID-19 [7]. Me quedaban pocas horas por realizar, por lo que tras hablar con el supervisor mi tarea fue documentar algunos tutoriales sobre realidad virtual para así ahorrar faena a próximos estudiantes en prácticas u otras personas. Los documentes que se realizaron fueron sobre como preparar el ordenador y Unity para raealidad virtual, la forma de conectar los dispositivos de realidad virtual y como crear un menú interactivo para usarlo en realidad virtual.

1.5. Conclusiones

El desarrollo de la realidad virtual parece ser un gran avance, aún en proceso de mejora, tanto en la industria de los videojuegos como en otros muchos ámbitos. De esta forma, haber trabajado en ello, aunque no sea mucho, proporciona calidad a tus conocimientos, teniendo una buena base por si se diera el caso que tuviera que trabajar en esta tecnología con mucha proyección de futuro. Además he podido aprender bastante más de lo que sabía en cuanto al desarrollo de juegos y también en programación, al haber tratado un caso práctico y real, lo que no suele ocurrir durante la carrera.

Parte II

Memoria TFG

Capítulo 2

Motivación y objetivos

El término criptografía proviene del griego, concretamente de las palabras "kryptós"(secreto) y "graphé"(escritura). La criptología es la ciencia que se dedica al estudio de la criptografía, es decir, de los mensajes que una vez procesados de cierta manera no son legibles por cualquiera. En la escritura secreta se necesita un criptosistema, que es un conjunto de algoritmos para realizar un servicio concreto. Generalmente son tres partes las necesarias en un criptosistema: la generación de la clave, el cifrado y el descifrado (conceptos que se verán en el apartado 5). También es importante estudiar las debilidades de los criptosistemas, que es a lo que se dedica el criptoanálisis. Se puede usar tanto con el fin de romper la seguridad de los mismos para conocer secretos y descifrar mensajes o, por otro lado, tratando de averiguar estos fallos para poder mejorar la seguridad de las personas que los utilicen.

Se pretende en este documento adentrarse en la criptografía debido a la importancia de la misma desde hace algunos miles de años: desde el cifrado César que usaba el militar romano Julio César en el siglo I a.C. para comunicarse con sus generales hasta el cifrado de mensajes durante la segunda guerra mundial de los nazis, derrotados en parte por conseguir ser descifrados por Alan Turing. Casi cualquier civilización ha querido proteger una información que considere de relevancia y que no quiere que cualquiera obtenga. Su importancia, por tanto, procede principalmente de cumplir una de las características más importantes de la seguridad, la confidencialidad, ya que un agente externo no podrá acceder a cierta información. Además, con el uso moderno de la criptografía también se permite obtener autenticidad en el envío y recepción de mensajes, pudiendo saber con certeza la identidad del emisor.

Capítulo 3

Aritmética modular

La aritmética es una rama de las matemáticas que estudia los números junto con sus operaciones básicas, es decir: la suma, la resta, la multiplicación y la división. Más precisamente, lo que se estudia es las estructuras numéricas y las propiedades de las operaciones elementales, de forma que finalmente se llega a construir la teoría de números.

3.1. El conjunto de restos

Al dividir un entero A entre otro $B > 1$, por el algoritmo de división, se obtiene un cociente C y un resto $R \geq 0$ que cumplen que $A = BC + R$ con $R < B$ (estas condiciones se dan en cada una de las operaciones de módulo que aparecen durante todo el texto). Si se da el caso que solamente interesa el resto de la operación hay que saber que existe un operador que puede ser utilizado, llamado módulo (*mod*). En este caso: $A \ mod \ B = R$. Gracias a este operador entra en juego la aritmética modular, también conocida como la aritmética del reloj.

Propiedad 3.1. *$A \ mod \ B = (A + KB) \ mod \ B$ con K un entero cualquiera.*

Demostración. Sea $A = BC + R$.
Entonces se tiene que $A + KB = (BC + R) + KB = B(C + K) + R$.
Por tanto $A \ mod \ B = (A + KB) \ mod \ B = R$. □

Propiedad 3.2. $(A + B) \ mod \ C = (A \ mod \ C + B \ mod \ C) \ mod \ C$.

Demostración. Sea $A = CQ_1 + R_1$, con $0 \leq R_1 < C$ y Q_1 enteros $\Rightarrow A \ mod \ C = R_1$
Sea $B = CQ_2 + R_2$, con $0 \leq R_2 < C$ y Q_2 enteros $\Rightarrow B \ mod \ C = R_2$

Aplicando la propiedad 3.1 en la última igualdad de la primera línea:
∘ $(A + B) \ mod \ C = (C(Q_1 + Q_2) + R_1 + R_2) \ mod \ C = (R_1 + R_2) \ mod \ C$.
∘ $(A \ mod \ C + B \ mod \ C) \ mod \ C = (R_1 + R_2) \ mod \ C$. □

Propiedad 3.3. $(A \cdot B)\ mod\ C = (A\ mod\ C \cdot B\ mod\ C)\ mod\ C$.

Demostración. Sea $A = CQ_1 + R_1$, con $0 \leq R_1 < C$ y Q_1 enteros $\Rightarrow A\ mod\ C = R_1$
Sea $B = CQ_2 + R_2$, con $0 \leq R_2 < C$ y Q_2 enteros $\Rightarrow B\ mod\ C = R_2$

Aplicando la propiedad 3.1 en la última igualdad de la primera línea:
$\circ$ $(A \cdot B)\ mod\ C = (C(CQ_1Q_2 + Q_1R_2 + Q_2R_1) + R_1 \cdot R_2)\ mod\ C = (R_1 \cdot R_2)\ mod\ C$.
$\circ$ $(A\ mod\ C \cdot B\ mod\ C)\ mod\ C = (R_1 \cdot R_2)\ mod\ C$. □

Ejemplo. *Con los números* $A = 10$, $B = 7$, $C = 5$.

Se tiene que $10 = 7 \cdot 1 + 3 \Rightarrow 10\ mod\ 7 = 3$.

$(10 + 7)\ mod\ 5 = 17\ mod\ 5 = 2$ *ya que* $17 = 5 \cdot 3 + 2$.
Por otro lado: $(10\ mod\ 5 + 7\ mod\ 5)\ mod\ 5 = (0 + 2)\ mod\ 5 = 2$.

También se tiene: $(10 + 2 \cdot 7)\ mod\ 7 = 24\ mod\ 7 = 3 = 10\ mod\ 7$ *ya que* $24 = 7 \cdot 3 + 3$.

Dados tres enteros A, B y C, si se obtiene el mismo resultado al aplicar $mod\ C$ a A y a B se dice que A es congruente con B módulo C y se denota $A \equiv B\ (mod\ C)$. Más concretamente se puede hablar de una relación binaria, la relación de congruencia módulo C. También se suele utilizar la siguiente definición para denotar las congruencias:

Definición 3.1. $A \equiv B\ (mod\ C)$ *si y solo si* $C \mid A - B$.

Propiedad 3.4. *Sea* $A \equiv B\ (mod\ C)$ *y* D *es un entero, entonces:*

I) $A + D \equiv B + D\ (mod\ C)$

II) $AD \equiv BD\ (mod\ C)$

Demostración. Por la definición 3.1 se tiene que: $C \mid A - B$.

I) Entonces: $C \mid (A + D) - (B + D) \Rightarrow A + D \equiv B + D\ (mod\ C)$.

II) Entonces: $C \mid (A - B)D \Rightarrow C \mid AD - BD \Rightarrow AD \equiv BD\ (mod\ C)$.

□

Propiedad 3.5. *Sea* $A \equiv B\ (mod\ N)$ *y* $C \equiv D\ (mod\ N)$, *entonces:*

I) $A + C \equiv B + D\ (mod\ N)$

II) $AC \equiv BD\ (mod\ N)$

Demostración. Se tiene que existen K_1, K_2 tales que $A = B + K_1N$, $C = D + K_2N$.

I) Entonces: $A + C = B + D + (K_1 + K_2)N \Rightarrow A + C \equiv B + D\ (mod\ N)$.

II) Entonces: $AC = BD + (BK_2 + DK_1 + K_1K_2N)N \Rightarrow AC \equiv BD\ (mod\ N)$.

□

Propiedad 3.6. *$A \equiv A + KB \ (mod\ B)$ con K un entero cualquiera.*

Demostración. Basta aplicar la propiedad 3.1 junto con la definición de congruencia. Además también se observa que: $KB \equiv 0 \ (mod\ B)$ con K un entero cualquiera. □

Ejemplo. *Se tiene que* $10\ mod\ 6 = 16\ mod\ 6 = 4$. *Se considera otro entero,* 3.
Entonces: $10 + 3\ mod\ 6 = 13\ mod\ 6 = 1$ *y* $16 + 3\ mod\ 6 = 19\ mod\ 6 = 1$.
También: $10 \cdot 3\ mod\ 6 = 30\ mod\ 6 = 0$ *y* $16 \cdot 3\ mod\ 6 = 48\ mod\ 6 = 0$.

También se tiene $8\ mod\ 6 = 20\ mod\ 6 = 2$.
Entonces: $10 + 8\ mod\ 6 = 18\ mod\ 6 = 0$ *y* $16 + 20\ mod\ 6 = 36\ mod\ 6 = 0$.
También: $10 \cdot 8\ mod\ 6 = 80\ mod\ 6 = 2$ *y* $16 \cdot 20\ mod\ 6 = 320\ mod\ 6 = 2$.

Además de los conceptos de módulo y congruencia se precisa de otros para poder definir el conjutno de restos.

Definición 3.2. *Sea la relación binaria R definida sobre un conjunto K no vacío. Se dice que R es una relación de equivalencia sobre K si cumple tres propiedades:*

- *Reflexividad:* $xRx \quad \forall x \in K$.
- *Simetría: Si* xRy, *entonces* $yRx \quad \forall x, y \in K$.
- *Transitividad: Si* $xRy \wedge yRz$, *entonces* $xRz \quad \forall x, y, z \in K$.

Definición 3.3. *Dado un conjunto K y una relación de equivalencia R sobre K, se llama clase de equivalencia de un elemento k en K al conjunto $\bar{k} = [k] := \{j \in K \ / \ jRk\}$.*

Definición 3.4. *Dado un conjunto K y una relación de equivalencia R sobre K, se llama conjunto cociente al conjunto de las clases de equivalencia de los elementos de k. Se denota K/R.*

Proposición 3.1. *La relación de congruencia módulo C es una relación de equivalencia.*

Demostración. Se aplica la definición 3.1 varias veces.
Reflexividad: Es obvio que $C \mid A - A$, por lo que $A \equiv A \ (mod\ C)$.
Simetría: $A \equiv B \ (mod\ C) \Rightarrow C \mid A - B \Rightarrow C \mid B - A \Rightarrow B \equiv A \ (mod\ C)$.
Transitividad: Si se tiene que $A \equiv B \ (mod\ C)$ y que $B \equiv D \ (mod\ C)$, entonces $A\ mod\ C = B\ mod\ C$ y también $B\ mod\ C = D\ mod\ C \Rightarrow A\ mod\ C = D\ mod\ C \Rightarrow A \equiv D \ (mod\ C)$. □

Por tanto, por la proposición anterior, se puede definir el conjunto cociente de la relación de congruencia módulo C sobre un conjunto. Concretamente se está trabajando con los números enteros, así que se podrá construir el conjunto cociente sobre $\mathbb{Z}$, al que llamamos conjunto de restos y lo denotamos $\mathbb{Z}_C$. Para poder construirlo se necesita saber las clases de equivalencia que, en este caso, la de un elemento $A \in \mathbb{Z}$ es el conjunto de enteros tales que su resto al dividirlos

por C es el mismo que el de A. Es decir, tener $A \equiv B \ (mod\ C)$ es equivalente a expresar que sus clases de equivalencia son la misma: $[A]_C = [B]_C$. Se escribirá $[A] = [B]$ siempre que no haya lugar a error con el módulo que se está tratando.

Proposición 3.2. *Si A mod $B = R$, entonces $[A] = [R]$ en $\mathbb{Z}_B$.*

Demostración. Se tiene que $A = KB + R$ para algún K entero $\Rightarrow KB = A - R \Rightarrow B \mid A - R$. Por tanto, aplicando la definición 3.1 se obtiene que $A \equiv R \ (mod\ B)$, lo que se ha comentado que es equivalente a expresar $[A]_B = [R]_B$. □

Teniendo en cuenta la proposición anterior y que al dividir un entero por C el conjunto de los posibles restos son los números menores que C, se observa que el conjunto de restos $\mathbb{Z}_C$ es un conjunto finito con C elementos: $\mathbb{Z}_C = \{[0], [1], \ldots, [C-1]\}$.

Ejemplo. *Si consideramos el conjunto $\mathbb{Z}_8$, por la proposición 3.2, el conjunto estará formado por las clases de solamente los restos que se pueden obtener al dividir por 8, que son del 0 al 7.* $\mathbb{Z}_8 = \{[0], [1], [2], [3], [4], [5], [6], [7]\}$.

3.1.1. Aritmética del conjunto de restos

Es posible definir para los elementos del conjunto de restos la suma y la multiplicación, que son las operaciones binarias + y $\cdot$ que están definidas de $\mathbb{Z}_C \times \mathbb{Z}_C$ en $\mathbb{Z}_C$ definidas como $[A] + [B] = [A + B]$ y $[A] \cdot [B] = [A \cdot B]$ respectivamente.

Proposición 3.3. *Algunas propiedades de las operaciones suma y multiplicación definidas son las siguientes:*

I) *Son operaciones conmutativas y asociativas.*

II) *El producto es distributivo respecto de la suma.*

III) *Tienen elemento neutro. El $[0]$ es el de la suma y el $[1]$ el de la multiplicación.*

Demostración.

I) Conmutativa: $[A] + [B] = [A + B] = [B + A] = [B] + [A]$.
Asociativa: $[A]+([B]+[C]) = [A]+[B+C] = [A+(B+C)] = [(A+B)+C] = [A+B]+[C] = = ([A] + [B]) + [C]$.
Análogo para la multiplicación.

II) $[A] \cdot ([B] + [C]) = [A] \cdot [B + C] = [A(B + C)] = [AB + AC] = [AB] + [AC] = = ([A] \cdot [B]) + ([A] \cdot [C])$.

III) $[A] + [0] = [A + 0] = [A]$ y $[A] \cdot [1] = [A \cdot 1] = [A]$.

□

En este caso es posible definir el elemento opuesto. Está claro que $[A]+[-A] = [A+(-A)] = [0]$ en $\mathbb{Z}_C$ para cualquier C, por lo que el elemento opuesto de $[A]$ es $[-A]$, que se denota como $-[A]$.

Ejemplo. *El elemento opuesto del* $[5]$ *en* $\mathbb{Z}_6$ *es el* $[-5]$*, pero se ve que* $-5 = 6(-1)+1$*, y por la proposición 3.2,* $[-5] = [1]$*. En efecto se observa que* $[5]+[1] = [6] = [0]$*.*

También es posible que exista elemento inverso. El elemento inverso de $[A]$ en $\mathbb{Z}_C$ es el elemento $[B]$, que se denota como $[A]^{-1}$, tal que $[A] \cdot [B] = [1]$. Al elemento $[B]$ también se le denomina el inverso multiplicativo módulo C de $[A]$. Se podría demostrar que este elemento no siempre existe y, en caso de existir, es único. A priori, la forma de encontrar el inverso multiplicativo de $[A]$ en $\mathbb{Z}_C$ es realizar el cálculo $[A] \cdot [B]$ para todo $B \in [0, C-1]$ y eligiendo, si existiera, el que cumple $[A] \cdot [B] = [1]$.

Ejemplo. *Se busca el inverso multiplicativo de* $[3]$ *en* $\mathbb{Z}_7$*.*
$[1] \cdot [3] = [3]$, $[2] \cdot [3] = [6]$, $[3] \cdot [3] = [9] = [2]$, $[4] \cdot [3] = [12] = [5]$, $[5] \cdot [3] = [15] = [1]$.
Es decir, el inverso multiplicativo de $[3]$ *en* $\mathbb{Z}_7$ *es* $[5]$*.*

Para realizar la resta, $-$, de dos elementos entra en juego la suma y el elemento opuesto:
$[A] - [B] = [A] + [-B]$

La división, :, es análoga pero con la multiplicación y el elemento inverso:
$[A] : [B] = [A] \cdot [B]^{-1}$

Ejemplo. *Teniendo en cuenta los ejemplos anteriores:*
En $\mathbb{Z}_6$, $[3] - [5] = [3] + [1] = [4]$.
En $\mathbb{Z}_7$, $[2] : [3] = [2] \cdot [5] = [10] = [3]$.

3.2. Máximo común divisor. Algoritmo de Euclides

3.2.1. Máximo común divisor

Definición 3.5. *El máximo común divisor (MCD) de dos números enteros a y b es el divisor más grande que tienen en común los dos números. Es decir,* $MCD(a, b) = d$ *si:*

- $d \mid a$, $d \mid b$.
- *Si existe d' tal que* $d' \mid a$, $d' \mid b$, *entonces* $d' \mid d$.

Propiedades.

I) *El máximo común divisor siempre existe y es único.*

II) $MCD(a, b) = MCD(b, a) \quad \forall a, b \in \mathbb{Z}$.

III) $MCD(a,0) = MCD(0,a) = a \quad \forall a \in \mathbb{Z}$.

IV) $MCD(a,1) = MCD(1,a) = 1 \quad \forall a \in \mathbb{Z}$.

V) $MCD(a,b) = 1$ *si y solo si los números son primos entre sí (es decir, si no tienen ningún factor primo común).*

La forma habitual enseñada para calcular el MCD es descomponer los dos números en sus factores primos, de forma que la multiplicación de los factores comunes elevados a la menor potencia resulta ser el MCD.

Ejemplo. *La descomposición en factores primos del* 234 *es:* $2 \cdot 3^2 \cdot 13$.
La descomposición en factores primos del 378 *es:* $2 \cdot 3^3 \cdot 3$.
Por tanto: $MCD(234, 378) = 2 \cdot 3^2 = 18$.

Este método es útil para números pequeños, siendo problemático lo costoso que es la descomposición en factores primos de números grandes.

3.2.2. Algoritmo de Euclides

Con este algoritmo se soluciona el problema del coste a la hora de calcular el MCD. Lo calcula a partir del algoritmo de división y de la siguiente proposición.

Proposición 3.4. *Sean dos enteros a y b ($a \geq b$), sea r el resto de dividir a por b. Entonces* $MCD(a,b) = MCD(b,r)$.

Demostración. Se definen $d := MCD(a,b)$ y $s := MCD(b,r)$. Se quiere ver entonces que $d = s$.
Por una parte:
Como d es divisor de a y de b: $a = a_1 d$, $b = b_1 d$.
Por el algoritmo de división: $a = bc + r$, $0 \leq r < b \Rightarrow r = a - bc = (a_1 - b_1 c)d \Rightarrow d \mid r$.
Por la definición 3.5, como $d \mid b$, $d \mid r$ y $s = MCD(b,r)$, entonces $d \mid s$.

Por otra parte:
Como s es divisor de b y de r: $b = b_2 s$, $r = r_2 s$.
Como $a = bc + r \Rightarrow a = (b_2 c + r_2)s \Rightarrow s \mid a$.
Por la definición 3.5, como $s \mid a$, $s \mid b$ y $d = MCD(a,b)$, entonces $s \mid d$.

Se tiene que $d \mid s$ y que $s \mid d \Rightarrow d = s$. □

La idea del algoritmo es aplicar la proposición anterior hasta llegar al caso trivial visto en las propiedades $MCD(t,0) = t$. Por tanto, se realiza el algoritmo de división repetidamente hasta que se obtenga un resto de 0, momento en el que se obtiene que el MCD es el divisor de esa división con resto 0.

Pseudocódigo en A.1 de Anexo A.

Ejemplo. *Se quiere calcular* $MCD(378, 234)$.
$378\ mod\ 234 = 144 \rightarrow 234\ mod\ 144 = 90 \rightarrow 144\ mod\ 90 = 54 \rightarrow 90\ mod\ 54 = 36 \rightarrow$
$\rightarrow 54\ mod\ 36 = 18 \rightarrow 36\ mod\ 18 = 0$
Por la proposición 3.4: $MCD(378, 234) = MCD(234, 144) = MCD(144, 90) = MCD(90, 54) =$
$= MCD(54, 36) = MCD(36, 18) = MCD(18, 0) = 18$

3.3. Algoritmo extendido de Euclides

La versión extendida del anterior algoritmo, además de devolver el MCD de dos enteros a y b, también devuelve los coeficientes enteros x e y que hacen una combinación lineal de a y b que da como resultado su MCD. La igualdad se conoce como la identidad de Bézout: $xa + yb = MCD(a, b)$. Además, $MCD(a, b)$ es el elemento positivo más pequeño de la forma $xa + yb$.

Pseudocódigo en A.2 de Anexo A.

Ejemplo. *Se quiere encontrar* $MCD(44, 16)$ *y la identidad de Bézout correspondiente.*
Se tiene $44 = 2 \cdot 16 + 12$ *y entonces:* $1 = 2 \cdot 0 + 1$ *y* $0 = 2 \cdot 1 - 2$.
Ahora e tiene $16 = 1 \cdot 12 + 4$ *y entonces:* $0 = 1 \cdot 1 + (-1)$ *y* $1 = 1 \cdot (-2) + 3$.
Ahora se tiene $12 = 3 \cdot \boldsymbol{4} + \boldsymbol{0}$ *y entonces:* $1 = 3 \cdot \boldsymbol{(-1)} + 4$ *y* $-2 = 3 \cdot \boldsymbol{3} + (-11)$.
Como el resto principal ya es 0*, se ha llegado a que:*
$MCD(44, 16) = 4$, $x = -1$ *e* $y = 3 \Rightarrow (-1) \cdot 44 + 3 \cdot 16 = 4 = MCD(44, 16)$.

3.3.1. Cálculo del inverso multiplicativo modular

Teorema 3.1. *Sea* p *un entero positivo. Un elemento* $[a] \in \mathbb{Z}_p$ *tiene inverso multiplicativo en* $\mathbb{Z}_p$ *si y solo si* $MCD(a, p) = 1$.

Este teorema confirma lo ya comentado de que no siempre existe el inverso multiplicativo. Cuando se cumple tal condición se encuentran con el algoritmo los enteros x e y tales que $xa + yp = 1$. Entonces $xa - 1 = (-y)p$, lo que implica que $p \mid xa - 1$, y que por la definición 3.1: $xa \equiv 1\ (mod\ p)$.

Es decir, si se aplica el algoritmo extendido de Euclides a dos elementos primos entre sí, a y p $(a > p)$, el inverso multiplicativo de $[a]$ en $\mathbb{Z}_p$ es la clase de equivalencia del coeficiente x obtenido.

3.4. Exponenciación modular rápida

Para números de muchos dígitos es bastante costoso realizar la operación de exponenciación modular. Elevar un número a otro es bastante útil como se verá en los diferentes apartados, de forma que se necesita algún algoritmo que realice este trabaja rápido.

La idea básica del algoritmo, para calcular $b^e \ mod \ m$ (e es un número de n bits), es la siguiente:

1. Se convierte el exponente a notación binaria:

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

2. Entonces se puede escribir:

$$b^e = b^{\left(\sum_{i=0}^{n-1} a_i 2^i\right)} = \prod_{i=0}^{n-1} \left(b^{2^i}\right)^{a_i}$$

3. Se aplica el operador módulo, haciendo los cálculos teniendo en cuenta la propiedad 3.3:

$$\prod_{i=0}^{n-1} \left(b^{2^i}\right)^{a_i} \quad mod \ m$$

Pseudocódigo en A.3 de Anexo A.

Ejemplo. *Se quiere calcular* $7^{142} \ mod \ 23$.

$142 = 2^1 + 2^2 + 2^3 + 2^7 = 2 + 4 + 8 + 128 = 10001110_2$.
Por tanto $7^{142} \ mod \ 23 = 7^{2+4+8+128} \ mod \ 23 = (7^2 \cdot 7^4 \cdot 7^8 \cdot 7^{128}) \ mod \ 23$.
Por la propiedad 3.3: $7^{142} \ mod \ 23 = (7^2 \ mod \ 23 \cdot 7^4 \ mod \ 23 \cdot 7^8 \ mod \ 23 \cdot 7^{128} \ mod \ 23) \ mod \ 23$.

Entonces se realizan esos cálculos.
$7^1 \ mod \ 23 = 7$
$\boldsymbol{7^2 \ mod \ 23} = (7^1 \cdot 7^1) \ mod \ 23 = (7^1 \ mod \ 23 \cdot 7^1 \ mod \ 23) \ mod \ 23 = (7 \cdot 7) \ mod \ 23 =$
$= 49 \ mod \ 23 = \boldsymbol{3}$
$\boldsymbol{7^4 \ mod \ 23} = (7^2 \cdot 7^2) \ mod \ 23 = (7^2 \ mod \ 23 \cdot 7^2 \ mod \ 23) \ mod \ 23 = (3 \cdot 3) \ mod \ 23 =$
$= 9 \ mod \ 23 = \boldsymbol{9}$
$\boldsymbol{7^8 \ mod \ 23} = (7^4 \cdot 7^4) \ mod \ 23 = (7^4 \ mod \ 23 \cdot 7^4 \ mod \ 23) \ mod \ 23 = (9 \cdot 9) \ mod \ 23 =$
$= 81 \ mod \ 23 = \boldsymbol{12}$
$7^{16} \ mod \ 23 = (7^8 \cdot 7^8) \ mod \ 23 = (7^8 \ mod \ 23 \cdot 7^8 \ mod \ 23) \ mod \ 23 = (12 \cdot 12) \ mod \ 23 =$
$= 144 \ mod \ 23 = 6$
$7^{32} \ mod \ 23 = (7^{16} \cdot 7^{16}) \ mod \ 23 = (7^{16} \ mod \ 23 \cdot 7^{16} \ mod \ 23) \ mod \ 23 = (6 \cdot 6) \ mod \ 23 =$
$= 36 \ mod \ 23 = 13$

$7^{64}\ mod\ 23 = (7^{32} \cdot 7^{32})\ mod\ 23 = (7^{32}\ mod\ 23 \cdot 7^{32}\ mod\ 23)\ mod\ 23 = (13 \cdot 13)\ mod\ 23 =$
$= 169\ mod\ 23 = 8$
$\boldsymbol{7^{128}\ mod\ 23} = (7^{64} \cdot 7^{64})\ mod\ 23 = (7^{64}\ mod\ 23 \cdot 7^{64}\ mod\ 23)\ mod\ 23 = (8 \cdot 8)\ mod\ 23 =$
$= 64\ mod\ 23 = \boldsymbol{18}$

Entonces se tiene que:
$7^{142}\ mod\ 23 = (7^2\ mod\ 23 \cdot 7^4\ mod\ 23 \cdot 7^8\ mod\ 23 \cdot 7^{128}\ mod\ 23)\ mod\ 23 =$
$= (3 \cdot 9 \cdot 12 \cdot 18)\ mod\ 23 = 5832\ mod\ 23 = 13$

3.5. El teorema de Euler. El pequeño teorema de Fermat

Antes de exponer los teoremas se introducen algunas definiciones y resultados previos.

Definición 3.6. *La función Phi (φ) de Euler es la que cuenta el número de enteros positivos menores o iguales al entero positivo n que son coprimos con n.*

Es decir, se trata de la función $\varphi : \mathbb{N} \setminus \{0\} \to \mathbb{N} \setminus \{0\}$, también llamada totalizador o función indicador, tal que $\varphi(n) = Card\{k \in \mathbb{N} \setminus \{0\} : k \leq n \wedge MCD(k,n) = 1\}$. Está claro que si n es un número primo, como todos sus números menores son coprimos con n, etonces $\varphi(n) = n - 1$.

Esta función puede ser expresada en función de los elementos de un conjunto de restos: $\varphi(n) = Card\{[k] \in \mathbb{Z}_n : MCD(k,n) = 1$ (es decir, $\exists [k]^{-1})\}$.

Definición 3.7. *Sea $m > 1$ un natural. Un conjunto de enteros $\{y_1, \ldots, y_m\}$ es un sistema completo de restos módulo m ($SCR(m)$) si se cumple:*

- *Dado $x \in \mathbb{Z}$, $\exists$ un único $y_j \in SCR(m)$ tal que $x \equiv y_j\ (mod\ m)$.*
- *$y_i \not\equiv y_j\ (mod\ m) \quad \forall i, j$ con $i \neq j$.*

Todos los sistemas completos de restos módulo m tienen m elementos.

Definición 3.8. *Sea $m > 1$ un natural. Un conjunto de enteros $\{y_1, \ldots, y_k\}$ es un sistema reducido de restos módulo m ($SRR(m)$) si se cumple:*

- *$MCD(y_i, m) = 1 \quad \forall i$.*
- *$y_i \not\equiv y_j\ (mod\ m) \quad \forall i, j$ con $i \neq j$.*
- *Dado $x \in \mathbb{Z}$ tal que $MCD(x,m) = 1$, $\exists$ un único $y_j \in SRR(m)$ tal que $x \equiv y_j\ (mod\ m)$.*

Todos los sistemas reducidos de restos módulo m tienen la misma cantidad de elementos, aunque no siempre son m elementos.

Lema 3.1. ***Lema de Euclides***
Si $n \mid ab$ *y* $MCD(a,n) = 1$*, entonces* $n \mid b$*.*

Demostración. Como $n \mid ab$, $\exists r$ tal que $nr = ab$.
Por tener que $MCD(a,n) = 1$ y por la identidad de Bézout: $\exists x, y$ tales que $ax + ny = 1$.
Se puede escribir $b(ax + ny) = b \Rightarrow bax + bny = b \Rightarrow nrx + bny = b \Rightarrow n(rx + by) = b \Rightarrow$
$\Rightarrow$ Como $rx + by \in \mathbb{Z}$, entonces se puede afirmar que $n \mid b$. □

Teorema 3.2. *Sean* $a \in \mathbb{Z}$ *y* $n \in \mathbb{Z}^+$*. Si* $MCD(a,n) = 1$ *y* $\{r_1, \ldots, r_n\}$ *es* $SCR(n)$*, entonces* $\{ar_1, \ldots, ar_n\}$ *es* $SCR(n)$*.*

Demostración. Se supone que $\{ar_1, \ldots, ar_n\}$ no es $SCR(n)$, por lo que se puede decir que para algunos i, j con $i \neq j$, $ar_i \equiv ar_j \ (mod\ n)$.
Esto implica, por la definición 3.1, que $n \mid a(r_i - r_j)$.
Como $n \nmid a$, por el lema 3.1 $\Rightarrow n \mid r_i - r_j \Rightarrow r_i \equiv r_j \ (mod\ n)$ #
Como a partir de asumir que $\{ar_1, \ldots, ar_n\}$ no es $SCR(n)$, se ha llegado a la contradicción de que $\{r_1, \ldots, r_n\}$ no es $SCR(n)$, entonces se tiene que $\{ar_1, \ldots, ar_n\}$ es $SCR(n)$. □

Como corolario se obtiene que el teorema es aplicable a un $SRR(n)$.

3.5.1. El teorema de Euler

Teorema 3.3. ***Teorema de Euler***
Sean $a \in \mathbb{Z}$ *y* $n \in \mathbb{Z}^+$ *con* $MCD(a,n) = 1$*, entonces* $a^{\varphi(n)} \equiv 1 \ (mod\ n)$*.*

Demostración. Sean $\{x_1, \ldots, x_{\varphi(n)}\}$ los enteros positivos coprimos con n, se puede observar que este conjunto es un $SRR(n)$.
Como $MCD(a,n) = 1$, aplicando el corolario del teorema 3.2: $\{ax_1, \ldots, ax_{\varphi(n)}\}$ es un $SRR(n)$.
Se puede ver que $ax_1, \ldots, ax_{\varphi(n)}$ es un reordenamiento módulo n de $x_1, \ldots, x_{\varphi(n)}$. Es decir, como conjuntos representan el mismo (aplicando módulo n a los elementos), aunque no necesariamente están ordenados de la misma manera.
Por ello, el producto de los elementos módulo n es congruente:

$$a^{\varphi(n)} \cdot (x_1 \cdot \ldots \cdot x_{\varphi(n)}) \equiv x_1 \cdot \ldots \cdot x_{\varphi(n)} \quad (mod\ \ n)$$

Por la definición 3.1 se tiene que $n \mid (x_1 \cdot \ldots \cdot x_{\varphi(n)})(a^{\varphi(n)} - 1)$.
Como $n \nmid x_1 \cdot \ldots \cdot x_{\varphi(n)}$ ya que $MCD(n, x_i) = 1$ para cada i, por el lema 3.1: $n \mid a^{\varphi(n)} - 1$.
Por último, otra vez por la definición 3.1 se obtiene $a^{\varphi(n)} \equiv 1 \ (mod\ n)$. □

3.5.2. El pequeño teorema de Fermat

Teorema 3.4. ***Pequeño teorema de Fermat***
Sea p un número primo, para cualquier $a \in \mathbb{N}$ se tiene que $a^p \equiv a \ (mod\ p)$.

Demostración.
Caso 1: $p \mid a$
Entonces $p \mid a(a^{p-1} - 1) \Rightarrow p \mid a^p - a \Rightarrow$ Por definición 3.1, $a^p \equiv a \ (mod\ p)$.

Caso 2: $p \nmid a$
Entonces $MCD(a, p) = 1$ y se puede aplicar el teorema de Euler: $a^{\varphi(p)} \equiv 1 \ (mod\ p)$.
Como p es primo, entonces $\varphi(p) = p - 1 \Rightarrow a^{p-1} \equiv 1 \ (mod\ p)$.
Entonces se aplica la propiedad 3.4, que multiplicando por a permite afirmar que:
$a^{p-1} \cdot a \equiv 1 \cdot a \ (mod\ p) \Rightarrow a^p \equiv a \ (mod\ p)$. □

Capítulo 4

Matemáticas en la criptografía

En este apartado se van a presentar algunos conceptos que se utilizan en la criptografía. Todos los nombrados son básicos para la criptografía pero presentan algunos problemas matemáticos, además del gran coste computacional en algunos.

4.1. Generación de números aleatorios

Es una de las tareas más importantes dentro de la criptografía ya que nos permitirá, entre otras cosas, le generación de claves (ver apartado 5.1.1). Un número aleatorio es tal que no se puede saber nada de él antes de que sea generado. En una generación de varios números aleatorios, la generación de uno de ellos no depende de ninguno de los anteriores. Así, el problema radica en que un ordenador no puede generar uno ya que es una máquina que realiza los pasos que le son ordenados, mediante lo que no se puede generar algo que es impredecible.

Distinguimos entre generador de números aleatorios (RNG) y generador de números pseudoaleatorios (PRNG), los que proporcionan una secuencia de números a partir de una entrada, la semilla. La forma de obtención de los números de los RNG es, principalmente, a través de recoger datos de fenómenos físicos impredecibles como la descomposición radiactiva, la variación de corriente eléctrica o las olas del mar. De esta forma, al tratarse de fenómenos impredecibles y que cambian constantemente, el grupo de números que son generados a partir de esa entrada es prácticamente imposible que se repita Lo que sucede con los PRNG es que la sucesión de números es reptible mediante la introducción de la misma entrada. A pesar de esta posibilidad de introducción manual de la semilla también se considera la posibilidad de que la entrada sea el momento del día en milisegundos o un movimiento realizado con el ratón, métodos que se utilizan buscando la ventaja de la velocidad en la obtención de los números.

4.2. Generación de números primos y test de primalidad

Un número primo es un número natural mayor que 1 que solamente tiene como divisores el uno y él mismo. Estos números son importantes en la criptografía ya que se utilizan en la generación de las claves públicas en el método RSA, ya que esta clave es la multiplicación de dos números primos grandes (ver apartado 5.2.4). El problema de esto radica en la dificultad de generar números primos que, además, sean grandes, que son los que se necesitan. Vemos primero la forma de determinar si un número es primo (test de primalidad) y a continuación la forma de generar un número potencialmente primo que, si pasa el test, se le considerará primo.

Nos encontramos ante el problema que para un algoritmo que diga si un número es primo, si tratamos con números muy grandes (de cientos de cifras), es complicado determinar esto con garantías. Las características deseadas en un algoritmo de este tipo son que sirva para cualquier tipo de número (generalidad), que se realice en una cantidad de tiempo razonable y que devuelva la respuesta correcta siempre. Sin embargo, cualquier test de primalidad que ha sido desarrollado hasta el momento deja de cumplir alguna de estas características.

Por ello, los tests de primalidad son prácticamente todos algoritmos probabilísticos ya que, al contrario que los tests deterministas, dada una entrada (en este caso un número grande) no siempre se produce la misma salida y, por tanto, no siempre la salida correcta (en este caso decir si es primo o no). No obstante, el año 2002 fue diseñado el test de primalidad AKS que es determinista, lo que se contrarresta con un mayor tiempo computacional. Distinguimos dos tipos de algoritmos probabilísticos:

- *Monte Carlo*: Dan la respuesta correcta al menos la mitad de las veces. Hay algunos que cuando responden que 'sí' la respuesta es correcta y si responden que 'no' puede ser incorrecta (sesgado en el 'sí'), y otros al contrario (sesgado en el 'no').
- *Las Vegas*: No siempre producen una respuesta, pero cuando la producen es correcta.

Uno de los más famosos test de primalidad es el de Miller-Selfridge-Rabin, que es de tipo Monte Carlo sesgado en el 'no'. Se va a ver en este documento una de las variantes del test, que se aplica a un tipo de números concretos. Las operaciones de exponenciación se realizan tal como se ve en el apartado 3.4 (exponenciación modular rápida).

Pseudocódigo en A.4 de Anexo A.

Por tanto, para generar un número primo grande será necesario generar un número grande, preferentemente que tenga probabilidades altas de ser primo por la forma en la que se genera, para posteriormente realizar un test de primalidad que, en caso de superar, podamos decir que es probablemente primo. Se presenta un ejemplo de generador de número primo, teniendo en cuenta la siguiente definición.

Definición 4.1. *Sean n, $B \in \mathbb{N}$ con n solamente divisible por primos menores o iguales que B, entonces se dice que n es B-suave y B es el límite de suavidad.*

Pseudocódigo en A.5 de Anexo A.

4.3. Factorización de números enteros

Factorizar un número entero trata en descomponerlo en multiplicación de números primos. Si generamos un número como multiplicación de dos números primos grandes, la descomposición de este número resultará de encontrar estos dos números primos, lo que es un proceso computacional bastante costoso. Por tanto, la principal seguridad del método RSA se basa en este gran coste, ya que se necesitará descomponer la clave pública en sus factores primos (ver apartado 5.2.4).

Uno de los algoritmos más conocidos con este fin es el método $p-1$ de Pollard, en el que se asume que el número a descomponer es compuesto. La salida del método es un factor de n o nos dice que se ha producido un fallo al intentar encontrar un factor. Se presenta un algoritmo simplificado del método.

Pseudocódigo en A.6 de Anexo A.

4.4. Logaritmo discreto

Dados dos enteros c, m y un primo p, se llama el problema del logaritmo discreto (DLP) a tratar de encontrar el único entero no negativo $e \leq p-2$ tal que $c = m^e \ mod \ p$. Tal como se ha visto en el apartado 3.4 existe un algoritmo que permite un rápido cálculo de la exponenciación modular pero, en cambio, lo que es costoso es el cálculo de este número e, que se denota $e = log_m(c) \ mod \ p$. El problema del logaritmo discreto es bastante útil en la criptografía por su elevado coste computacional con una buena elección de p, por ejemplo en el protocolo de intercambio de claves Diffie-Hellman (ver apartado 5.2.5). Actualmente lo que se considera una buena elección de p es uno con más de 308 cifras y, además, que $p-1$ tenga al menos un factor primo grande.

Un algoritmo que permite resolver el problema es el Baby-Step Giant-Step (o algoritmo de Shanks), que además es genérico ya que funciona para cualquier grupo finito. Veamos un algoritmo más sencillo, que sirve para el caso específico de un grupo cíclico.

Pseudocódigo en A.7 de Anexo A.

Ejemplo. *Se quiere calcular* $log_5(71) \ mod \ 167$.

Entonces $s := \lfloor \sqrt{167} \rfloor = 12$.

Baby Step:

Se calculan los pares $(j, \ 5^j \cdot 71 \ mod \ 167)$ *para* $j = 0, 1, \ldots, 12-1$*:*

$(0, 71)$, $(1, 21)$, $(2, 105)$, $(3, 24)$, $(4, 120)$, $(5, 99)$, $(6, 161)$, $(7, 137)$, $(8, 17)$, $(9, 85)$, $(10, 91)$, $(11, 121)$

Ordenados de forma ascendente por la segunda componente queda:
$(8, 17)$, $(1, 21)$, $(3, 24)$, $(0, 71)$, $(9, 85)$,$(10, 91)$, $(5, 99)$, $(2, 105)$, $(4, 120)$, $(11, 121)$, $(7, 137)$, $(6, 161)$

Giant Step:

Se calculan los pares $(5^{i \cdot 12} \ mod \ 167, i)$ *para* $i = 1, 2, \ldots, 9$*:*
$(152, 1)$, $(58, 2)$, $(132, 3)$, $(24, 4)$, $(141, 5)$, $(56, 6)$, $(162, 7)$, $(75, 8)$, $(44, 9)$, $(8, 10)$, $(47, 11)$, $(130, 12)$

Ordenados de forma ascendente por la primera componente queda:
$(8, 10)$, $(24, 4)$, $(44, 9)$, $(47, 11)$, $(56, 6)$, $(58, 2)$, $(75, 8)$, $(130, 12)$, $(132, 3)$, $(141, 5)$, $(152, 1)$, $(162, 7)$

Entonces se observa que $5^3 \cdot 71 \equiv 5^{4 \cdot 12} \equiv 24 \ (mod \ 167)$.
Por lo que $i = 4$, $j = 3 \Rightarrow x = 4 \cdot 12 - 3 = 45$.

Se concluye que $log_5(71) \equiv 45 \ (mod \ 167)$ *ya que se puede comprobar que* $5^{45} \equiv 71 \ (mod \ 167)$.

Capítulo 5

Cifrado

Cifrar o encriptar unos datos consiste en volver ilegible la información proporcionada por dichos datos. Así, si se quiere que la información vuele a estar disponible de forma legible se tiene que realizar el proceso de descifrado.

Un uso general del cifrado de información es garantizar que información personal, que puede ser enviada a través de distintos servidores, no sea entendible en caso de ser interceptada por un intruso, ya que habrá sido cifrada. Por ejemplo, si trabajas en un proyecto secreto podrás enviar la información cifrada sobre el proyecto a tu socio, de forma que, aunque alguna persona externa se haga con esta información, solamente tendrá texto inservible si no lo descifra. Ahora bien, cuando tu socio recibe la información tiene el mismo problema que un interceptor, por lo que debe saber la forma de descifrar la información.

Tanto el cifrado como el descifrado son algoritmos que realizan tales acciones. Se podría pensar, por tanto, que si un intruso capta la información y además conoce el algoritmo de descifrado, entonces tendrá acceso a la información, pero no es así. Los dos algoritmos precisan de un parámetro, llamado clave, para realizar el trabajo. De esta forma, el intruso necesitaría conocer esta clave para desvelar la información, lo que resta importancia a mantener en secreto el algoritmo utilizado. Es tanta la importancia de las claves que la principal forma de clasificar tipos de algoritmos de cifrado es según la forma de utilizarlas.

5.1. Cifrado simétrico

El cifrado simétrico también es conocido como cifrado de clave privada. En este tipo de cifrado se utiliza la misma clave tanto para cifrar como para descifrar. Este método se puede usar para cifrar información de tu disco duro o incluso para mandarte información con alguien, de forma que todas las partes deben ponerse de acuerdo con la elección de la clave para poder ambos cifrar y descifrar.

5.1.1. Claves

Las claves son lo más importante en cuanto a seguridad de los algoritmos de cifrado, por lo que hay que tratarlas con cuidado. Es más, son la única medida efectiva en la mayoría de los casos ya que no es factible basar la seguridad en los algoritmos. Se podría intentar mantener en secreto el algoritmo, aunque podría ser atacado y descubierto. Además, si no fueras capaz de crear tus propios algoritmos deberías contactar con alguna empresa que los cree y confiar en que no lo revelen. Es más sencillo mantener en secreto simplemente la clave y se podría usar una diferente para cada información.

Generación y elección de claves

En estos algoritmos la clave es un número que generalmente estará expresado en dígitos binarios. Esta clave puede tener el tamaño que sea, aunque será más segura si contiene un mayor número de bits ya que, si alguien intenta realizar un ataque por fuerza bruta deberá probar más cantidad de combinaciones. También es conveniente utilizar una clave distinta para cada información y además renovarla habitualmente. Así, lo mejor es tener claves totalmente aleatorias o, en todo caso, pseudoaleatorias, lo que se puede conseguir mediante un RNG o un PRNG (ver apartado 4.1). Es importante saber que utilizando un PRNG, si un atacante adivina simplemente la semilla utilizada habrá podido obtener también la clave, de forma que la semilla también debe ser lo suficientemente grande. A tener en cuenta que esto es aplicable también para las claves del cifrado asimétrico.

5.1.2. Cifrado de flujo

Primeramente se debe disponer de un generador de bits arbitrarios, llamado generador de flujo de clave. Generalmente la idea es que el texto en claro se cifra bit a bit, en el que cada bit se combina mediante la función XOR con uno de los bits generados. Utilizar la operación OR exclusivo (XOR) es lo habitual debido a que el cifrado y el descifrado son la misma operación.

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Como ventaja de este tipo de cifrado se puede decir que son habitualmente más rápidos que los de bloque, además que suelen utilizar menos código.

5.1.3. Cifrado de bloque

En este caso se cifra por bloques, es decir, se divide el texto en bloques de bytes (generalmente de 8 o 16 bytes) y se cifran por separado. El texto cifrado de un bloque será del mismo tamaño que el original.

Como ventaja de este tipo de cifrados se puede decir que permiten la reutilización de las claves, lo que no es posible en los de flujo debido al funcionamiento que se ha comentado.

Relleno

Una duda lógica es qué sucede si el tamaño del texto no es múltiplo del tamaño de cada bloque, ya que el último bloque no será del tamaño de un bloque. La solución será añadir una serie de bits al final del último bloque hasta completar el tamaño correspondiente. La dificultad es que no sirve concatenar unos bits cualesquiera ya que al descifrar una información, si no conocíamos lo original, no podremos distinguir la información original del relleno.

Una de las soluciones más utilizadas es, si por ejemplo tenemos que rellenar 3 bytes, completamos el bloque en hexadecimal de la forma 030303, es decir, cada byte faltante indica el número de bytes faltantes. Igualmente podría ser que algo que se podría considerar relleno sea casualmente el final de una información original. Esto se soluciona añadiendo un bloque entero de relleno si el tamaño del texto es múltiplo del tamaño de un bloque.

Tipos

Uno de los tipos de cifrado por bloques es el ECB, que es el modo en el que cualquiera puede haber pensado cuando se le explica este tipo de cifrado. Cada bloque es cifrado individualmente y todos con la misma clave. Al ser cada bloque independiente, un error en uno de los bloques solamente afecta a él mismo. Un problema es que dos bloques con la misma información se cifran exactamente iguales, lo que podría llegar a facilitar el trabajo de un atacante.

Un modo que soluciona este problema es el CBC, pero que tiene el problema que un error en un bloque afecta a los siguientes. El primer bloque se cifra mediante la operación XOR del texto en claro con un vector inicial (IV) no secreto y aleatorio. Será este texto cifrado el que opere mediante XOR con el texto en claro del siguiente bloque, y así sucesivamente.

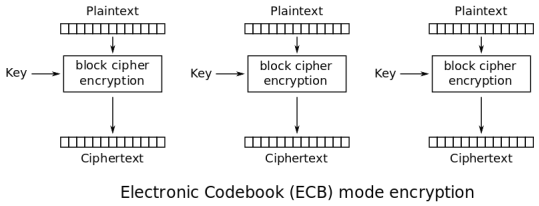

Figura 5.1: Cifrado del método de cifrado por bloques ECB.

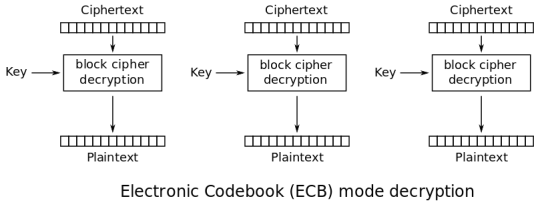

Figura 5.2: Descifrado del método de cifrado por bloques ECB.

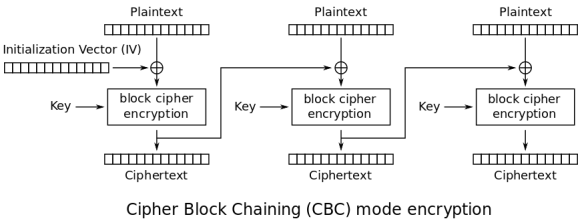

Figura 5.3: Cifrado del método de cifrado por bloques CBC.

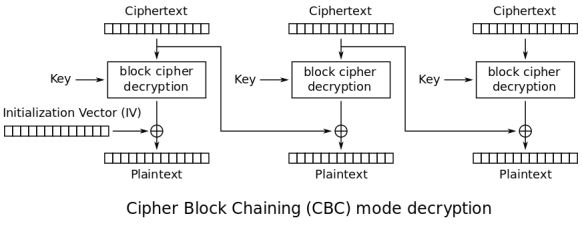

Figura 5.4: Descifrado del método de cifrado por bloques CBC.

5.1.4. Problema de la distribución de claves

Si se necesita compartir información con alguna otra persona y que la información no sea conocida por nadie más no sería recomendable enviarla sin ningún tipo de cifrado. Por tanto, se cifra la información y se envía, de forma que ante cualquier intruso que fuera capaz de interceptar el mensaje se estaría protegido, ya que sería totalmente inentendible. Ahora bien, el destinatario recibe la información pero no sabe la forma de descifrarla. Se podría pensar en enviar también la clave, pero no es apropiado ya que también podría ser interceptada.

Las personas que quieren compartir varios mensajes con información secreta pueden decidir que lo mejor es quedar en persona para generar la misma clave y guardarla en sus ordenadores, lo que parece una solución factible. El problema aparece si se quiere compartir la clave entre una cantidad considerable de personas, todos deberían quedar para compartir la clave o alguien tendría que ir a ver a todos. Quizás también suceda que alguien salga del grupo, por lo que por seguridad se debería cambiar todas las claves. O el caso de que alguien entre, que se tendría que quedar de nuevo para compartir la clave.

Para todos los casos anteriores también cabe la posibilidad de utilizar un tercero de confianza (Third Trusted Party, TTP), que es una persona o compañía que envía las claves a las partes. Este tercero puede pertenecer a la misma empresa que los comunicadores, aunque en caso de abandonarla habría que volver a empezar. Si el tercero es exterior se tendría que confiar, ya que el TTP puede en cualquier momento los mensajes si quisiera. Se puede generar, además, una jerarquía de TTP para que no solamente uno se encargue de repartir las claves.

5.2. Cifrado asimétrico

El cifrado asimétrico también es conocido como cifrado de clave pública o cifrado de dos claves. En este tipo de cifrado se utiliza una clave para el cifrado y otra para el descifrado. Ambas claves están relacionadas pero de una forma demasiado compleja como para encontrar una a partir de la otra si no se conoce concretamente la relación. El cifrado se realiza con lo que se llama clave pública y el descifrado con la clave privada.

5.2.1. Solución a la distribución de claves

La idea de este cifrado es que, tal como indica su nombre, la clave pública no es secreta, de forma que si alguien quiere enviarte información cifrada deberá pedir tu clave pública para cifrar. Cuando te envíe la información, si alguien intercepta el mensaje solamente verá texto sin sentido, y solamente tú con tu clave privada (que sí será secreta) podrás descifrar la información. Con esto se soluciona el problema de distribuir las claves, aunque se debe de tener en cuenta una sutileza, y es que el cifrado asimétrico es más lento debido a su complejidad extra. Como solución, debido a que la clave es más corta que un mensaje, es habitual el cifrado usando lo que se conoce como sobre digital, que consiste en lo siguiente:

1. El emisor genera una clave privada de un uso y cifra el mensaje con ella.
2. El emisor pide al receptor su clave pública y cifra la clave privada anterior con ella.
3. El emisor envía al receptor el mensaje y la clave cifrados.
4. El receptor descifra la clave con la que se descifra el mensaje con su clave privada asociada a la clave pública que ha mandado al emisor.
5. El receptor descifra el mensaje con la clave que acaba de descifrar.

5.2.2. Funciones unidireccionales

Las funciones unidireccionales o funciones de un solo sentido son funciones en las que es fácil calcular la imagen de un elemento cualquiera del dominio pero, dado un elemento de la imagen cualquiera, es computacionalmente inviable calcular un elemento del dominio que al aplicarle la función obtenga este elemento de la imagen. Es decir, si $f : A \to B$ es una función unidireccional, $f(a)$ es fácil de calcular para cualquier $a \in A$ pero, en cambio, dado $b \in B$ cualquiera, será complicado obtener un elemento $a \in A$ tal que $f(a) = b$.

Teóricamente no está probada la existencia de funciones unidireccionales, es solamente una conjetura. Aún así, tal como se ha comentado, se considerará funciones unidireccionales aquellas que tengan una gran complejidad computacional a la hora de obtener la preimagen de un elemento. Se asume su existencia, al menos en la criptografía, debido a que son de gran utilidad e incluso necesarias.

Hash

Las funciones hash o funciones de resumen se aplican a una entrada cualquiera siendo las imágenes de cualquier entrada del mismo tamaño. Por tanto está claro que se pueden producir colisiones, es decir, dadas dos entradas cabe la posibilidad de que produzcan una misma imagen. La imagen de un elemento cualquiera se llama también resumen del elemento.

Las funciones hash criptográficas son aquellas funciones hash que son unidireccionales. Por tanto, será difícil encontrar una preimagen de un resumen, además que también será complicado encontrar dos elementos cuyos resúmenes sean el mismo.

5.2.3. Recuperación de claves

Es posible perder tu clave privada, ya sea por un fallo del disco duro, que se rompa o el motivo que sea. Es por ello que se necesita algún método que permita recuperar una clave. Los métodos de recuperación más habituales se basan en el sobre digital (ver apartado 5.2.1).

Tercero de confianza

Este método consiste en el ya mencionado tercero de confianza (TTP). Esta persona o compañía genera un clave pública con su correspondiente privada y distribuye la clave pública. Cuando se genere la clave que se quiere guardar lo único que hay que hacer es encriptarla con la clave pública del TTP, que la guarda de forma segura. Si en determinado momento se necesita recuperar la clave se puede enviar al TTP la clave cifrada y, mediante su clave privada podrá descifrarla y obtener la clave. Un problema claro del método es que el TTP puede conocer tu clave y además dependes de su disponibilidad, ya que si no está disponible por el motivo que sea, un nuevo TTP tendrá que generar un par de nuevas claves pública y privada y volver a empezar el proceso.

Grupo de fideicomisarios

Este método funciona igual que el anterior pero la clave se divide entre diferentes personas o TTP. Esto evita el problema que una sola persona pueda conocer tu clave pero, en cambio, aumenta la posibilidad de que alguna de las partes no esté disponible en un momento determinado. También se debe tener en cuenta que tiene una mayor dificultad de implementación. Además se hace más sencillo un ataque por fuerza bruta si una de las partes decide intentarlo, ya que tiene un trozo de la clave, por lo que se debería utilizar claves más grandes de lo habitual.

5.2.4. RSA

Desarrollado en 1979, se trata del algoritmo de cifrado asimétrico más utilizado actualmente. Debe su nombre a las iniciales de sus creadores: Ronald Rivest, Adi Shamir y Leonard Adleman. La idea de su funcionamiento es la siguiente:

1. Se generan dos números primos grandes p y q de tamaños similares (ver apartado 4.2).
2. Obtenemos $n = p \cdot q$. Generalmente $2^{1024} \leq n \leq 2^{2048}$, por lo que p y q tendrán que ser elegidos teniendo esto en cuenta.
3. Se elige un número primo pequeño e, habitualmente el 65537.
4. Se obtiene d, que es el número tal que $d \cdot e \equiv 1 \ (mod \ \varphi(n))$. A tener en cuenta que $\varphi(n)$ es la función Phi de Euler (ver definición 3.6), que en este caso, al ser $n = p \cdot q$, entonces $\varphi(n) = \varphi(p) \cdot \varphi(q) = (p-1) \cdot (q-1)$. También se debe tener en cuanta que existe ese d ya que existe cuando $MCD(e, \varphi(n)) = 1$ y, más en concreto, si e es primo siempre existirá. El elemento d se calcula mediante al algoritmo extendido de Euclides (ver apartado 3.3.1).
5. Obtenemos la clave pública (e, n) y la clave privada (d, n).

Ejemplo. *Se escogen los números primos* $p = 281$ *y* $q = 167$.
Entonces $n = p \cdot q = 46927$.
Se elige el primo $e = 65537$.
Por tanto $\varphi(n) = (p-1) \cdot (q-1) = 280 \cdot 166 = 46480$.
Por último, mediante el algoritmo extendido de Euclides se calcula d*, el inverso multiplicativo de* e *módulo* $\varphi(n)$*, es decir, tal que* $d \cdot 65537 \equiv 1 \ (mod\ 46480) \Rightarrow d = 16273$.
Entonces la clave pública será $(e, n) = (65537, 46927)$ *y la privada* $(d, n) = (16273, 46927)$.

Cifrado y descifrado con RSA

Para cifrar un mensaje m se realiza el cálculo $c = m^e \ mod\ n$. A partir de c y con la clave privada se puede descifrar mediante el cálculo m'$= c^d \ mod\ n$. Realmente se necesita comprobar que $m = m$' ya que si no fuera así, al descifrar no se estaría obteniendo el mensaje original. Se prueba de la siguiente manera.

- El mensaje cifrado es $c = m^e \ mod\ n$.
- Entonces al descifrar se tiene m'$= c^d \ mod\ n = (m^e \ mod\ n)^d \ mod\ n = m^{ed} \ mod\ n$.
- Hay que recordar que $ed \equiv 1 \ mod\ \varphi(n) \Rightarrow ed = 1 + \varphi(n) \cdot k$ con k un entero que no es conocido.
- Entonces m' $= m^{1+\varphi(n)\cdot k} \ mod\ n = m \cdot m^{\varphi(n)\cdot k} \ mod\ n$.
- Por el teorema de Euler (ver apartado 3.5.1) se tiene que $m^{\varphi(n)} \equiv 1 \ (mod\ n) \Rightarrow$
  $\Rightarrow m$' $= m \cdot m^{\varphi(n)\cdot k} \ mod\ n = m \cdot 1^k \ mod\ n = m \ mod\ n$.

Seguridad de RSA

Está claro que lo que necesita saber un atacante para poder descifrar es d. Esto puede averiguarlo aplicando el algoritmo extendido de Euclides si sabe cuanto vale $\varphi(n)$, pero no es el caso ya que no conoce ni p ni q. Podría encontrar estos dos valores factorizando n y encontrado así sus dos factores primos, y aquí es donde radica la seguridad, en el problema que existe a la hora de factorizar números lo suficientemente grandes (ver apartado 4.3).

5.2.5. DH

Desarrollado en 1976, se trata de un protocolo que permite adoptar una clave entre dos partes que no tienen contacto entre ellas, es decir, se genera una clave de un algoritmo simétrico para cifrar los mensajes entre dos partes a partir de un par de claves pública y privada en lugar de compartir una clave simétrica utilizando un algoritmo de clave pública El nombre se debe a sus creadores Whitfield Diffie y Martin Hellman. La idea de su funcionamiento es la siguiente:

1. Ambas partes se ponen de acuerdo en un primo p que será el módulo y un generador g públicos.

2. La primera parte elige un número $a < p$ y la segunda un $b < p$ privados.

3. La primera parte calcula $A = g^a \ mod \ p$ y se la envía a la segunda. La segunda parte calcula $B = g^b \ mod \ p$ y se la envía a la primera.

4. La primera parte calcula $(B \ mod \ p)^a = (g^b \ mod \ p)^a = g^{b \cdot a} \ mod \ p$. La segunda parte calcula $(A \ mod \ p)^b = (g^a \ mod \ p)^b = g^{a \cdot b} \ mod \ p = g^{b \cdot a} \ mod \ p$.

Se obtiene que ambos cálculos son iguales, por lo que se puede utilizar como clave compartida.

Ejemplo. *Ambas partes se ponen de acuerdo para elegir* $p = 2003$ *y* $g = 106$.
La parte privada elegida por la primera parte es $a = 381$*, de modo que procede a calcular* $A = 106^{381} \ mod \ 2003 = 1717$*, número que envía a la segunda parte.*
La elegida por la segunda partes es $b = 751$*:* $B = 106^{751} \ mod \ 2003 = 158$*, que lo envía a la primera parte.*
Por tanto, ambas partes disponen de la misma clave una vez realizado el cálculo correspondiente, la primera parte el primero y la segunda el segundo:
$158^{381} \ mod \ 2003 = 1717^{751} \ mod \ 2003 = 1193$.

Seguridad de DH

Un atacante puede interceptar A y B ya que son enviados entre las partes por algún canal, de forma que para obtener la clave compartida, y por tanto descifrar los mensajes que obtenga, solamente neceista conocer el valor a o el valor b (ya que p y g son públicos). Es decir, el trabajo de un atacante sería encontrar $a = log_g(A) \ mod \ p$ o $b = log_g(B) \ mod \ p$. Esta faena es el problema del logaritmo discreto, lo que supone un elevado coste computacional para p lo suficientemente grande (ver apartado 4.4). Lo adecuado sería una elección de p de al menos 1024 bits ($p \geq 2^{1024}$).

5.3. Derivación de claves desde contraseñas

Prácticamente todo el habituado al uso de una computadora sabrá lo que es y habrá utilizado una contraseña. Se trata de un código necesario para que se te permita realizar una función concreta. Esta definición es aplicable también a una clave, de forma que su fin es el mismo con la diferencia fundamental que una clave de posiblemente cientos de bits no seremos capaz de recordarla pero una contraseña de unos pocos caracteres sí.

Si se cifra algún archivo con una simple contraseña será más sencillo un ataque por fuerza bruta que contra una clave. Es habitual, con el fin de evitar esto, que la función de derivación

de clave sea una función hash que asigne a cada entrada algunos cientos de bits, siendo habitualmente usada la función SHA256 que genera 256 bits. Aún así persiste el problema de que un atacante haya podido averiguar la contraseña que, junto con el conocimiento de la función hash rompería el cifrado. Incluso existen tablas, llamadas rainbow tables, que tienen calculados los resúmenes obtenidos mediante SHA256 de posibles contraseñas de hasta 8 caracteres. La solución aceptada universalmente es concatenar una cadena de bits aleatorios, llamada salt, a la contraseña, para posteriormente calcular su resumen, lo que nos proporciona una clave razonable. Este salt puede ser público sin problemas.

Capítulo 6

Protocolos criptográficos

Existen bastantes aplicaciones útiles que derivan de la utilización de algunas de las cosas que se han visto como el protocolo RSA o las funciones hash. Alguna aplicación puede ser sencilla en el sentido de que no es de gran importancia, simplemente resuelve un problema típico como el lanzamiento de una moneda, en el sentido de elegir la opción correcta de dos hechos equiprobables. Por otro lado, se pueden realizar cosas más serias como poder autenticar la autoría de mensajes o detectar modificaciones en archivos, teniendo como fin, por ejemplo, evitar fraudes.

6.1. Lanzar una moneda

Como su nombre indica trata de simular el lanzamiento de una moneda. Dos personas pretenden decidir alguna cosa o simplemente saber, como diversión, quién acierta. La solución al problema pasa por la realización de un método que obtenga de forma equiprobable dos conjuntos distintos de opciones. Lo más habitual es alguno que obtenga un número y la elección consista en ver si el número es par o impar, lo que se puede hacer de varias formas. Una de las dos personas será la que prepare el método y la otra, por tanto, será la que elija si quiere, en este caso, par o impar. Una forma sencilla de hacerlo es la siguiente.

1. A genera un número aleatorio (ver apartado 4.1), del que B debe adivinar si es par o impar.

2. A calcula la imagen de este número a partir de una función hash criptográfica (ver apartado 5.2.2).

3. A envía el resultado a B, que trata de adivinar si es par o impar.

4. A le dice si ha acertado o no y, como comprobación, le manda el número original y le dice la función que ha utilizado.

5. B calcula la imagen del número por la función y comprueba que coincide con lo que había recibido en primer lugar.

La seguridad de que A no haya hecho trampas radica en que a partir de una función hash criptográfica es demasiado complicado (casi imposible en un tiempo razonable) encontrar dos números de los que se obtenga una misma imagen. Si fuera sencillo, A podría obtener una misma imagen de un número par y de otro impar y si B dice par enviar el número impar y al revés.

Este método pertenece a los llamados protocolos de compromiso, ya que A se compromete a no variar el resultado original tratando de engañar a B y, hasta que A no lo desvele B no será capaz de saberlo. Más allá de lanzar una moneda, el protocolo se puede adaptar al lanzamiento de un dado de cualquier número de caras, con la misma probabilidad de obtener cualquiera de ellas.

6.2. Firma digital

El objetivo principal de la firma digital es que se pueda verificar la autenticidad de mensajes o documentes. Además se tiene la propiedad de no repudio, por lo que el firmante no podrá negar haber firmado el documento. Se evita también que un documente firmado sea modificado y se intente, quizás, engañar al receptor.

Para realizar la firma digital de un documento se suele proceder de la siguiente manera.

1. Se obtiene un par de claves de cifrado asimétrico.
2. Se calcula el resumen del documento.
3. Se cifra el resumen con la clave privada.
4. Se envía a la persona que se desee el resumen cifrado junto con el documento original.

La comprobación de la autoría la puede realizar el receptor viendo que al realizar el resumen del documento original y el descifrado del supuesto resumen cifrado ambos coinciden. Como se realiza lo anterior con la clave pública del emisor y la clave privada solamente la tiene él queda claro la propiedad de que él no puede negar haberlo firmado. Por último, ante un cambio en el documento original, su resumen será diferente y, por tanto, no coincidirá al intentar hacer la comprobación. Se tendrá que volver a firmar y enviar el nuevo documento junto con el nuevo resumen cifrado.

6.3. Otros

Existen otros muchos y bastantes variados protocolos criptográficos, como por ejemplo el ya visto de intercambio de claves de Diffie-Hellman (ver apartado 5.2.5).

Hay protocolos para compartir secretos que se reparten entre varias personas, de forma que solamente se puede revelar si se juntan todas las personas. Ahora bien, puede suceder que si son muchas partes, al intentar recuperar el secreto falte alguna. Se puede solucionar estableciendo partes de más, y existen protocolos de forma que cuando se junten solamente se necesite algunas de ellas.

Para una comunicación confidencial por un canal inseguro se puede utilizar el que se llama protocolo de rellenado con paja y aventado, que hace referencia a una técnica de la agricultura. La idea básica es que se añaden bits extra que no tienen nada que ver junto a la comunicación útil, de forma que existirá un mecanismo de autenticación, que solamente conoce el receptor, para que sea capaz de obtener esta información útil.

Capítulo 7

Conclusiones

La criptografía es muy importante a la hora de proteger información privada y sensible. Es por ello que se debe tener en cuenta durante el desarrollo de cualquier algoritmo o protocolo relacionado con la criptografía que una persona no deseada no debe ser capaz de acceder a dicha información. En cambio, en todo protocolo comentado en este documento (y en cualquiera existente) esto no es así. La realidad es que los protocolos basan su seguridad en la necesidad de una gran cantidad de tiempo para la ruptura del mismo, incluso décadas o cientos de años, pero finalmente se podría lograr. Entonces, ¿podemos estar completamente tranquilos con respecto al tema de la seguridad?

Los ordenadores cuánticos basan su funcionamiento en la mecánica cuántica y, sin entrar en detalles, son capaces de realizar una cantidad bastante mayor de operaciones en el mismo tiempo que un ordenador actual. Este aumento de velocidad es suficiente para poder romper la seguridad de los protocolos criptográficos en un tiempo factible para un intruso. Aún así, por el momento se puede estar tranquilo en cuanto a seguridad criptográfica por algunos motivos:

- Existen algunos ordenadores cuánticos pero se estima que el tiempo de su extensión a cualquier usuario es de unos cuantos años, incluso hay expertos que indican que pasarán unos 100 años.
- La comunidad está trabajando en el desarrollo de algoritmos nuevos que no permitan que un ordenador cuántico rompa el algoritmo en un tiempo razonable.

Parte III

Anexo

Anexo A

Pseudocódigos

A.1. Algoritmo de Euclides

Entrada: Dos enteros a y b, $a \geq b$
Salida: $MCD(a, b)$

while $b \neq 0$ **do**
    $r \leftarrow a \ mod \ b$ ▷ resto de la división a/b
    $a \leftarrow b$
    $b \leftarrow r$
end while
return a

A.2. Algoritmo extendido de Euclides

Entrada: Dos enteros a y b, $a \geq b$
Salida: $(MCD(a,b),\ x,\ y)$ tales que $xa + yb = MCD(a,b)$

if $b = 0$ **then**
  return a, 1, 0
end if
$v_0 \leftarrow 1$
$v_1 \leftarrow 0$
$w_0 \leftarrow 0$
$w_1 \leftarrow 1$
while $b \neq 0$ **do**
  $q \leftarrow a//b$ ▷ cociente de la división a/b
  $r \leftarrow a - q * b$
  $v \leftarrow v_0 - q * v_1$
  $w \leftarrow w_0 - q * w_1$
  $a \leftarrow b$
  $b \leftarrow r$
  $v_0 \leftarrow v_1$
  $v_1 \leftarrow v$
  $w_0 \leftarrow w_1$
  $w_1 \leftarrow w$
end while
return a, v_0, w_0

A.3. Exponeciación modular rápida

Entrada: Tres enteros b, e y m
Salida: $res := b^e \ mod \ m$

if $m = 1$ **then**
  return 0
end if
$res \leftarrow 1$
$b \leftarrow b \ mod \ m$ ▷ resto de la división b/m
while $e > 0$ **do**
  if $e \ mod \ 2 = 1$ **then** ▷ si e es impar
    $res \leftarrow (res * b) \ mod \ m$
  end if
  $e \leftarrow e//2$ ▷ cociente de la división $e/2$
  $b \leftarrow (b * b) \ mod \ m$
end while
return res

A.4. Test de primalidad Miller-Selfridge-Rabin

Entrada: $n > 1$ impar tal que $n - 1 = 2^t m$ con $t \in \mathbb{N}$ y m impar
Salida: $True$ si n es probablemente primo o $False$ si n no es primo

$a \leftarrow$ entero cualquiera elegido tal que $2 \leq a \leq n - 2$
$b_0 \leftarrow a^m \ mod \ n$
if $b_0 \equiv \pm 1 \ (mod \ n)$ **then**
    return $True$
end if
$i \leftarrow 1$
while $i < t - 1$ **do**
    $b_i \leftarrow b_{i-1}^2 \ mod \ n$
    if $b_i \equiv -1 \ (mod \ n)$ **then**
        return $True$
    else if $b_i \equiv 1 \ (mod \ n)$ **then**
        return $False$
    end if
    $i \leftarrow i + 1$
end while
$b_i \leftarrow b_{i-1}^2 \ mod \ n$
if $b_i \equiv -1 \ (mod \ n)$ **then**
    return $True$
else
    return $False$
end if

Se dirá que n es un pseudoprimo fuerte en base a si ha sido declarado probablemente primo habiendo elegido a en el paso inicial. Por tanto lo ideal sería probar para varios valores de a, de forma que si todos concluyen que n es probablemente primo se tendrá una mayor certeza de la primalidad de este número.

A.5. Método de generación de números primos grandes

Entrada: b, cantidad de bits deseados para el número primo
B, límite de suavidad (determinado empíricamente)
Salida: n, número primo de b bits

$n \leftarrow$ entero impar de b bits generado aleatoriamente
for los primos menores que B **do**
    if n divisible por el número **then**
        volver a empezar el algoritmo
    end if
end for
$primo \leftarrow$ aplicar el test de primalidad A.4
if $primo = True$ **then**
    return n
else
    volver a empezar el algoritmo
end if

A.6. Método p-1 de Pollard para la factorización de enteros

Entrada: n, entero positivo que se asume compuesto
Salida: Un factor de n o $False$ en caso de no haberse podido encontrar un factor

$B \leftarrow$ se elige un número como límite de suavidad
$a \leftarrow$ entero cualquiera elegido tal que $2 \leq a \leq n-1$
if $1 < MCD(a,n) < n$ **then**
    return $MCD(a,n)$
end if
$j \leftarrow 2$
while $j \leq B$ **do**
    $a \leftarrow a^j \ mod \ n$
    $d \leftarrow MCD(a-1,n)$
    if $1 < d < n$ **then**
        return d
    else if $d = n$ **then**
        se puede volver a empezar el algoritmo cambiando la B y/o la a
         o se puede devolver $False$
    else
        $j \leftarrow j+1$
    end if
end while

Se debe tener en cuenta que el algoritmo debe ejecutarse de forma rápida a la vez que se tiene un porcentaje razonable de éxito en encontrar un factor. Para lo primero B debe ser lo suficientemente pequeño y para lo segundo lo suficientemente grande. Además B no debería estar demasiado cerca de $\sqrt{n}$ para obtener ventajas con respecto a realizar la prueba trivial de descomponer n (probar a dividir n por todos los números primos hasta $\sqrt{n}$).

A.7. Método Baby-Step Giant-Step para el logaritmo discreto

Entrada: m, generador de un grupo cíclico G de orden n
c, elemento cualquiera de G
Salida: En caso de existir: x, que es $log_m(c)\ mod\ n$

- $s \leftarrow \lfloor \sqrt{n} \rfloor$ ▷ $\lfloor x \rfloor$ devuelve el máximo entero no superior a x
- *Baby-Step*: Calcular $(j,\ m^j c\ mod\ n)$ para $j = 0, 1, \ldots, s-1$ y ordenar por la segunda componente en orden ascendente.
- *Giant-Step*: Calcular $(m^{is}\ mod\ n,\ i)$ para $i = 1, 2, \ldots, s$ y ordenar por la primera componente en orden ascendente.
- Buscar si hay i, j tales que $m^j c = m^{is}$.

En ese caso, devolver $log_m(c)\ mod\ n$, que es: $x \equiv is - j\ (mod\ n)$.

Bibliografía

[1] Página web del INIT.
https://www.init.uji.es/

[2] Página web de la UJI.
https://www.uji.es/

[3] Página web del CEVI de la UJI.
https://www.uji.es/serveis/ocit/base/grupsinvestigacio/detall?codi=009

[4] Página web de Unity.
https://unity.com/es

[5] Página web de Photon.
https://www.photonengine.com/en-US/Photon

[6] Página web de VIVE.
https://www.vive.com/eu/

[7] Se suspende actividad presencial hasta nuevo aviso. Noticias FUE-UJI.
https://www.fue.uji.es/noticia/suspende-actividad-presencial-hasta-nuevo-aviso-1982

[8] Tesis de Leonel Sergio Carrasco Pérez (2012), *Factorización de enteros*.
Universidad Autónoma Metropolitana, Ciudad de México, México.
http://mat.izt.uam.mx/mcmai/documentos/tesis/Gen.09-O/Carrasco-LS-Tesis.pdf

[9] TFG de Urko Nalda Gil (2014), *Factorización*.
Universidad de La Rioja, Cantabria, España.
https://biblioteca.unirioja.es/tfe_e/TFE000668.pdf

[10] TFG de Noelia Marí Salvador (2018), *Una propuesta híbrida para el criptoanálisis RSA*.
Universitat Politècnica de València, Valencia, España.
https://riunet.upv.es/bitstream/handle/10251/107779/MAR%C3%8D%20-%20Una%20propuesta%20h%C3%ADbrida%20para%20el%20criptoan%C3%A1lisis%20RSA..pdf?sequence=1&isAllowed=y

[11] TFG de Jennifer Santamaría Fernández (2013), *El logaritmo discreto y sus aplicaciones en Criptografía*.
Universidad de Cantabria, Cantabria, España.
https://repositorio.unican.es/xmlui/bitstream/handle/10902/3101/Jennifer%20Santamaria%20Fernandez.pdf?sequence=1

[12] Pacheco, F., (2014). *Criptografía*.
Buenos Aires, Argentina: Editorial Fox Andina.

[13] Ireland, K. y Rosen, M., (1982). *A Classical Introduction to Modern Number Theory*.
Nueva York, EEUU: Editorial Springer-Verlag.

[14] Mollin, Richard A., (2002). *RSA and Public-Key Cryptography*.
EEUU: Editorial Chapman & Hall.

[15] Burnett, S. y Paine, S., (2001). *RSA Security's Official Guide to Cryptography*.
EEUU: Editorial McGraw-Hill.

[16] Aumasson, J.P., (2017). *Serious Cryptography: A Practical Introduction to Modern Encryption*.
San Francisco, EEUU: Editorial No Starch Press.