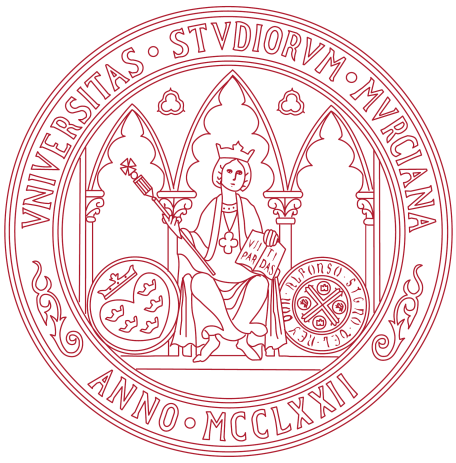
PRUEBAS DE CONOCIMIENTO CERO Y SUS APLICACIONES

JOSÉ LUIS CÁNOVAS SÁNCHEZ

Tutores
ANTONIO JOSÉ PALLARÉS RUIZ
LEANDRO MARÍN MUÑOZ



Facultad de Matemáticas
Universidad de Murcia

José Luis Cánovas Sánchez: *Pruebas de Conocimiento Cero y sus Aplicaciones,*
Julio 2017

DECLARACIÓN DE ORIGINALIDAD

Yo, José Luis Cánovas Sánchez, autor del TFG PRUEBAS DE CONOCIMIENTO CERO Y SUS APLICACIONES, bajo la tutela de los profesores Antonio José Pallarés Ruiz y Leandro Marín Muñoz, declaro que el trabajo que presento es original, en el sentido de que ha puesto el mayor empeño en citar debidamente todas las fuentes utilizadas.[1]

Murcia, Julio 2017

José Luis Cánovas Sánchez

1 En la Secretaría de la Facultad de Matemáticas se ha presentado una copia firmada de esta declaración.

EXTENDED ABSTRACT

Imagine a cave, where the path forks in two passages, and at the end of each one, they join again, with the shape of a ring. In the point the paths meet, there is a magic door, that only opens when someones pronounces the magic work.

Peggy knows the secret word and wants to **p**rove it to her friend, **V**ictor, but without revealing it. Peggy and Victor meet at the entrance of the cave, then Victor awaits while Peggy goes inside the cave, taking one of the passages, that we will name A and B. Victor can't see which way Peggy went.

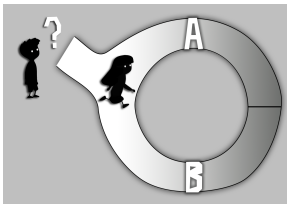
When Peggy arrives at the door, **V**ictor enters the cave, and when he arrive to the fork, stops and yells which path, A or B, he wants Peggy to come back, to **v**erify she knows how to open the door.

If Peggy actually knows the secret, she always can take the requested path, opening the magic door if needed. But if Peggy doesn't know the magic word, she had a chance of 50% to guess correctly what passage Victor was going to ask. That means she had a chance to fool Victor.

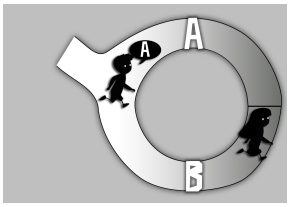
Victor then asks to repeat the experiment. With 20 repetitions, the chances Peggy fools Victor in all of them is only 2^{-20},

Eve, curious about what Victor and Peggy were doing in the cave, **e**avesdrops Victor during the process. The problem is that Eve doesn't know if Peggy and Victor agreed on what paths to choose, because they wanted to prank her for being busybody. Only Victor is confident he is choosing the returning passage randomly.

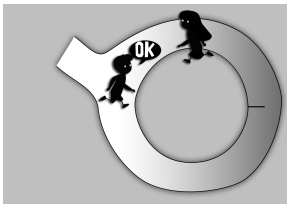
Later, Victor is convinced that the door can be opened and Peggy knows the word, but he can't prove it to Eve because he can't open the door.



The cave [8] . Peggy takes randomly A or B. Victor awaits outside.



The cave. Victor chooses randomly the returning path for Peggy.



The cave. Peggy returns by the requested path.

ÍNDICE GENERAL

1 INTRODUCCIÓN 1

2 PRELIMINARES DE COMPUTACIÓN 3
- 2.1 Notación asintótica 4
- 2.2 Clases de complejidad 4
- 2.3 Algoritmos probabilísticos 7

3 PRELIMINARES DE GRAFOS 9
- 3.1 Teoría de grafos 9
- 3.2 Problemas de decisión basados en grafos 11
  - 3.2.1 Problema del isomorfismo de grafos 11
  - 3.2.2 Problema del camino hamiltoniano 11
  - 3.2.3 Problema de la 3-coloración 12

4 PRELIMINARES DE ÁLGEBRA 13
- 4.1 Grupos y Anillos 13
- 4.2 Aritmética en $\mathbb{Z}$ 14
- 4.3 Congruencias: el anillo $\mathbb{Z}_n$ 15
- 4.4 El grupo simétrico S_n: permutaciones 17
- 4.5 Problema del logaritmo discreto 17

5 RESIDUOS CUADRÁTICOS 19
- 5.1 Primeras propiedades 19
- 5.2 Símbolo de Legendre 20
- 5.3 Símbolo de Jacobi 23
- 5.4 El problema de residuosidad cuadrática 23

6 PRUEBAS DE CONOCIMIENTO CERO 25
- 6.1 Sistemas de Pruebas Interactivas 25
- 6.2 Pruebas de conocimiento cero 27
  - 6.2.1 Residuos cuadráticos 29
  - 6.2.2 Isomorfismo de grafos 32
  - 6.2.3 Logaritmo discreto 35
- 6.3 Otros tipos de pruebas de conocimiento cero 38
- 6.4 Pruebas de conocimiento cero estadísticas 38
- 6.5 Pruebas de conocimiento cero de Verificador Honesto 39
- 6.6 Pruebas de conocimiento cero computacionales 42
  - 6.6.1 Prueba de conocimiento cero para un grafo hamiltoniano 42
  - 6.6.2 Prueba de conocimiento cero para la 3-coloración de un grafo 44
- 6.7 Esquemas de compromiso 44

7 APLICACIONES DE ZKP 47
- 7.1 Firma digital basada en pruebas de conocimiento cero 47
- 7.2 Protocolos de identificación basados en ZKP 48
  - 7.2.1 Protocolo de identificación de Fiat-Shamir 48
  - 7.2.2 Protocolo de identificación de Feige-Fiat-Shamir 49

7.2.3 Protocolo de identificación de Schnorr 50
7.3 Pruebas de conocimiento cero en Identity Mixer 52
7.3.1 Notación para ZKP 52
7.3.2 Combinar diferentes pruebas de conocimiento 53
7.3.3 Firma Camenisch-Lysyanskaya 53
7.3.4 Firma Camenisch-Lysyanskaya aleatorizada 54
7.3.5 Firma de credenciales Idemix 54
7.3.6 Revelación selectiva de atributos Idemix 57
7.4 Pruebas de conocimiento cero en criptomonedas 58
8 IMPLEMENTACIONES 59
8.1 Prueba Interactiva: Problema del Residuo Cuadrático 59

Appendix 63
A CÓDIGO FUENTE 65

BIBLIOGRAFÍA 67

ÍNDICE DE FIGURAS

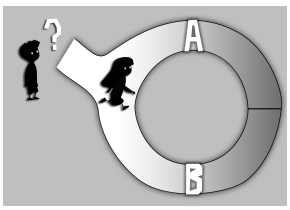
Figura 1 Conjetura de las relaciones entre clases **NP**, **co-NP**, **NPC**, **P**. 6
Figura 2 Idemix: firma de credenciales simplificada. 56

1 INTRODUCCIÓN

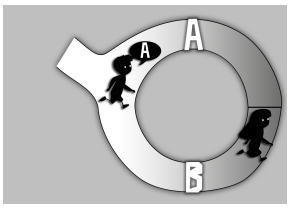
En la actualidad, para demostrar que conocemos un secreto a alguien, utilizamos un medio donde nadie nos espíe y le contamos el secreto directamente a nuestro interlocutor. En el ámbito de la criptografía, cifraríamos el secreto con una clave, simétrica o asimétrica, tal que, sólo quien conozca la clave de descifrado pueda leer nuestro secreto. Esto es la base de las comunicaciones seguras en Internet. Ciframos nuestra contraseña de modo que sólo el servidor de correo pueda leerla, pudiendo iniciar nuestra sesión sin que ningún espía nos la pueda robar. El problema es que hay dos partes que conocen el secreto, dos puntos de ataque.

Existe un área de la criptología que se encarga de abordar este problema, demostrar que conocemos algo, pero sin que de la misma prueba se obtenga más información. Estos métodos se conocen como Pruebas de Conocimiento Cero. Para ilustrar estas pruebas, en 1990 Guillou, Quisquater y Berson publicaron en *How to Explain Zero-Knowledge Protocols to Your Children* [6] una historia sobre cómo Alí Babá demostró poder abrir la cueva sin decir a nadie cuáles eran las palabras mágicas. Aquí presentamos una versión resumida que destaca las propiedades básicas de estos métodos:

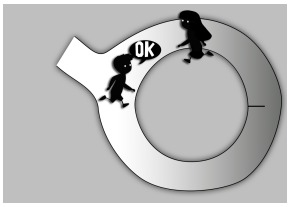
> Imaginemos una cueva, donde el camino principal se bifurca y al final de cada pasillo los caminos se vuelven a encontrar, formando una especie de anillo. En el punto en que se unen, dentro de la cueva, hay una puerta mágica con una palabra secreta, la cual permite abrirla y cruzar al otro lado.
>
> **P**aula conoce la clave secreta y quiere **p**robarlo a su amigo Víctor, pero sin tener que revelársela. Paula y Víctor quedan en la entrada de la cueva con unos *walkie-talkies*, de modo que Víctor esperará fuera y Paula entrará a la cueva y tomará uno de los pasillos, que llamaremos A y B, sin decirle cuál a Víctor.
>
> Al llegar a la puerta, Paula avisa a Víctor para que entre a la cueva y espere en la bifurcación, donde **V**íctor, para intentar **v**erificar que Paula conoce la palabra secreta, le indicará por qué pasillo quiere que vuelva, el A o el B.
>
> Si Paula realmente conoce el secreto, podrá volver a la bifurcación por el pasillo solicitado, abriendo, si es preciso, la puerta. Pero en caso de no conocer la clave, al entrar por uno de los pasillos, tenía una probabilidad del 50 % de adivinar cuál pediría Víctor.
>
> Víctor no se queda contento con una sola prueba, pues Paula podría haber tenido suerte, así que la repiten hasta que se convence. Con unas 20 repeticiones, Paula tendría solo una probabilidad de



La cueva [8] . Paula entra por A o B al azar. Víctor espera fuera.



La cueva. Víctor elige al azar por dónde quiere que regrese Paula.



La cueva. Paula vuelve por el camino pedido.

2^{-20}, prácticamente nula, de acertar todas las veces y engañar así a Víctor.

Eva, curiosa de qué hacían Víctor y Paula en la cueva, **e**spía a Víctor durante todo el proceso. Eva no sabe si Paula y Víctor han acordado previamente qué pasillo pedir por el *walkie-talkie*, y sólo Víctor está seguro de que él los estaba eligiendo al azar y sin previo acuerdo.

Más tarde Eva habla con Víctor, que está seguro de que Paula conoce la clave, y éste querría convencer también a Eva, pero como él no conoce la clave, no puede repetir la prueba a Eva, sólo Paula puede realizarla con éxito.

En el estudio formal de las Pruebas de Conocimiento Cero se utilizan distintos tipos de problemas, y los más utilizados pertenecen a las áreas de **álgebra** y **grafos**. Son problemas difíciles de resolver para quien no conoce alguna información extra de los mismos, que en nuestra fábula serían el problema de abrir la puerta sin conocer la palabra mágica. Los problemas más representativos de las Pruebas de Conocimiento Cero son el del logaritmo discreto, la raíz cuadrada modular o residuos cuadráticos, el isomorfismo de grafos y la 3-coloración de grafos.

Primero deberemos definir qué se entiende por un problema *difícil*, para ello, en el capítulo TODO, se introducirán unos preliminares de computación, que estudian y clasifican los problemas. A continuación, estudiaremos la teoría relacionada con cada problema *difícil* utilizado en las Pruebas de Conocimiento Cero que vamos a ver: en el capítulo TODO, veremos el álgebra relacionada con el logaritmo discreto, en el capítulo TODO, las definiciones y teoremas necesarios para conocer los residuos cuadráticos, y en el capítulo TODO la teoría básica de grafos necesaria para comprender los problemas.

Una vez conocemos la teoría detrás de cada problema, podremos estudiar, en el capítulo TODO, las Pruebas de Conocimiento Cero, definiremos qué propiedades debe cumplir una prueba de este tipo, enunciaremos pruebas que utilicen los problemas anteriores, y demostraremos que cada una de ellas cumple la definición. Finalmente, daremos algunas aplicaciones prácticas equivalentes a las de la criptografía actual en el capítulo TODO.

2

PRELIMINARES DE COMPUTACIÓN

En este capítulo introducimos unos preliminares sobre la complejidad computacional, que nos permitirán entender qué problemas matemáticos, y por qué, se utilizan en criptografía. Comencemos con unas definiciones:

Definición 2.1. Llamamos *problema* a la descripción general de una tarea que depende de unos parámetros. La *definición* de un problema consta de dos partes. La primera da el escenario del problema, describiendo los parámetros necesarios. La segunda parte indica una pregunta de la que se espera una respuesta o solución.

Ejemplo 2.2. Vamos a considerar el problema de multiplicar dos matrices. La definición del problema sería:

> *Nombre:* Problema multiplicación de matrices.
> *Parámetros:* Dos matrices A_1 y A_2.
> *Pregunta:* ¿Cuál es la matriz A tal que $A = A_1 \cdot A_2$?

Definición 2.3. Una *instancia* de un problema es el caso particular de un problema al que se le han dado valores a los parámetros.

Definición 2.4. Un *algoritmo* es una lista de instrucciones que para una instancia produce una respuesta correcta. Se dice que un algoritmo *resuelve* un problema si devuelve respuestas correctas para todas las instancias del problema.

Definición 2.5. Se llama *problema de decisión* a un problema cuya respuesta está en el conjunto $\mathcal{B} = \{\mathtt{Verdadero}, \mathtt{Falso}\}$.

No todos los problemas son de decisión, pero en general es fácil transformarlos en un problema de decisión cambiando la *pregunta*, y los algoritmos que resuelven un problema de decisión suelen también ser fáciles de adaptar para el problema original.

Ejemplo 2.6. El problema del ejemplo anterior se puede transformar a un problema de decisión:

> *Nombre:* Problema multiplicación de matrices.
> *Parámetros:* Tres matrices A, A_1 y A_2.
> *Pregunta:* ¿$A = A_1 \cdot A_2$?

Un algoritmo para solucionarlo puede primero comprobar las dimensiones de las matrices, y si no concuerdan con las de la multiplicación, puede responder $\mathtt{Falso}$ sin más operaciones, pero cuando las dimensiones concuerden, deberá realizar la multiplicación de A_1 por A_2 y luego comparar A con el producto obtenido.

2.1 NOTACIÓN ASINTÓTICA

Para poder estudiar cómo crecen el tiempo de ejecución, la memoria usada, o cualquier otro recurso, de un algoritmo, dependiendo del tamaño de los datos de entrada, utilizamos distintas *notaciones asintóticas*. Nos interesa en particular la siguiente:

Definición 2.7. Sea $f : \mathbb{N} \to \mathbb{R}^+$. Definimos la notación *O grande* u *orden* de f al conjunto de funciones de $\mathbb{N}$ a $\mathbb{R}^+$ acotadas superiormente por un múltiplo positivo de f a partir de cierto valor $n \in \mathbb{N}$:
$O(f) = \{t : \mathbb{N} \to \mathbb{R}^+ \mid \exists c \in \mathbb{R}^+, \exists n_0 \in \mathbb{N} : t(n) \leqslant c \cdot f(n) \quad \forall n \geqslant n_0\}$.

Para estudiar el tiempo de ejecución, $t : \mathbb{N} \to \mathbb{R}^+$ (el tiempo promedio, el caso más desfavorable, ...), de un algoritmo, buscaremos una función f que acote a t lo más de cerca posible. Para ello definimos la relación de orden:

Comparación de órdenes: $O(1) \leqslant O(\ln n) \leqslant O(\sqrt{n}) \leqslant O(n) \leqslant O(n \ln n) \leqslant O(n^c) \leqslant O(n^{\ln n}) \leqslant O(c^n) \leqslant O(n!) \leqslant O(n^n)$, *donde* $c > 1$ *constante.*

Definición 2.8. Sean $f, g : \mathbb{N} \to \mathbb{R}^+$. Diremos que $O(f) \leqslant O(g)$ si $\forall t \in O(f)$ se tiene que $t \in O(g)$, es decir, $O(f) \subseteq O(g)$.

Para acotar t buscaremos el menor $O(f) : t \in O(f)$.

2.2 CLASES DE COMPLEJIDAD

Para estudiar los problemas de decisión los organizamos en clases de complejidad, según sus mejores algoritmos que los solucionan.

Definición 2.9. Se llama algoritmo de *tiempo de cálculo polinomial* al algoritmo cuya función de tiempo del caso más desfavorable es de orden $O(n^k)$, donde n es el tamaño de la entrada y k una constante. Cualquier algoritmo cuyo tiempo de ejecución no puede acotarse de esa manera se llama de *tiempo de cálculo exponencial*.

A veces se denominan *buenos* o *eficientes* a los algoritmos polinomiales, e *ineficientes* a los exponenciales. Sin embargo, hay casos particulares donde no ocurre. Lo más importante al tratar con algoritmos polinomiales es el grado del polinomio, k, que indica su orden de complejidad. Por ejemplo, consideremos un problema cuyo mejor algoritmo tarda n^{100} instrucciones, es de orden polinómico, pero prácticamente intratable a partir de $n = 2$, y por otra parte, un problema que utilice $2^{0,00001n}$ instrucciones podría tratar casos hasta de $n = 10^6$ sin dificultad. Por esto, en criptografía se debe considerar el caso promedio sobre el caso más desfavorable.

Definición 2.10. El conjunto de problemas de decisión que se pueden resolver en tiempo polinomial se llama clase **P**.

Definición 2.11. Se llama clase de complejidad **NP** al conjunto de problemas de decisión donde una respuesta que sea $Verdadero$ se puede verificar en tiempo polinomial, dada cierta información extra, que llamaremos *certificado*.

Definición 2.12. Se llama clase de complejidad **co-NP** al conjunto de problemas de decisión donde una respuesta que sea `False` se puede verificar en tiempo polinomial, dado el correspondiente *certificado*.

Es inmediato que que **P** $\subseteq$ **NP** y **P** $\subseteq$ **co-NP**.

Ejemplo 2.13 (Problema en **NP**)**.** Consideremos el siguiente problema de decisión:

Nombre:	Problema del entero compuesto.
Parámetros:	Un entero positivo n.
Pregunta:	¿Es n compuesto? Es decir, ¿existen enteros $a, b > 1$ tal que $n = ab$?

El problema pertenece a **NP** porque se puede comprobar en tiempo polinomial que n es compuesto si nos dan su *certificado*, un divisor a de n, tal que $1 < a < n$. Basta usar la división euclídea de n entre a y ver que el resto es 0. Sin embargo, aún no se conoce si pertenece a **P**.

Definición 2.14. Sean dos problemas de decisión $L_1, L_2 \in$ **NP**, con correspondientes conjuntos de instancias I_1 e I_2. Sean I_1^+ e I_2^+ los subconjuntos de todas las instancias "`Verdaderas`" de I_1 e I_2, respectivamente. Decimos que L_1 es *reducible en tiempo polinomial* a L_2, $L_1 \leqslant_P L_2$, si existe una función $f : I_1 \Rightarrow I_2$ que cumple:

1. Es ejecutable en tiempo polinomial.
2. $x \in I_1^+ \Leftrightarrow f(x) \in I_2^+$.

De manera más informal, se dice que L_1 se reduce en tiempo polinomial a L_2 si hay un algoritmo que resuelve L_1 utilizando como subrutina un algoritmo que resuelve L_2, y que se ejecuta en total en tiempo polinomial, si lo hace el algoritmo que resuelve L_2.

Si $L_1 \leqslant_P L_2$, entonces se puede entender que L_2 es al menos computacionalmente tan difícil como L_1, o que L_1 no es más difícil que L_2.

Definición 2.15. Un problema de decisión L se dice que es **NP**-completo, o **NPC** si:

(i) $L \in$ **NP**, y

(ii) $L_1 \leqslant_P L \quad \forall L_1 \in$ **NP**.

Los problemas **NPC** son los más difíciles entre los **NP** en el sentido de que son al menos tan difíciles como el resto de problemas en **NP**. Veamos el problema **NPC** más característico:

Nombre:	Problema de satisfacibilidad booleana (SAT).
Parámetros:	Una colección finita C de expresiones booleanas con variables y sin cuantificadores.
Pregunta:	¿Hay alguna asignación de las variables que haga Verdadero a C?

Teorema 2.16 (Teorema de Cook (1971))**.** *El problema de satisfacibilidad booleana es **NPC**.*

Para conocer más sobre el problema y la demostración se puede consultar [10].

O expresado de otra manera:

Teorema 2.17. *Todo problema* $Q \in \boldsymbol{NP}$ *se puede reducir en tiempo polinomial al problema de satisfacibilidad booleana.*

$\forall Q \in \boldsymbol{NP}$, $Q \leqslant_P \text{SAT}$.

A día de hoy conocemos ciertas propiedades sobre las clases de equivalencia, pero quedan muchas cuestiones sin resolver:

- Uno de los siete problemas del milenio, ¿$\mathbf{P} = \mathbf{NP}$?
- ¿$\mathbf{NP} = \mathbf{co\text{-}NP}$?
- ¿$\mathbf{P} = \mathbf{NP} \cap \mathbf{co\text{-}NP}$?

La opinión de los expertos en base a ciertas evidencias es que la respuesta a las tres es *no*, pero hasta que se encuentren demostraciones formales de cada una, quedarán en conjeturas. Y basándose en esas conjeturas es donde se apoya la seguridad informática.

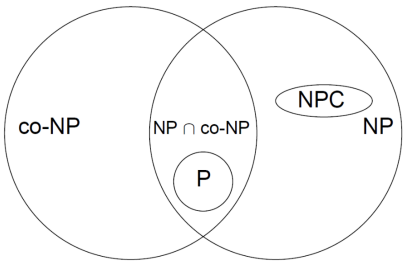


Figura 1: Conjetura de las relaciones entre clases **NP**, **co-NP**, **NPC**, **P**.

Existen tres problemas que se conoce que pertenecen a **NP**, pero se desconoce a día de hoy si pertenecen a **P** o **NPC**: el isomorfismo de grafos, el logaritmo discreto y la factorización de enteros.

En los siguientes capítulos los estudiaremos como base de diferentes pruebas de conocimiento cero.

2.3 ALGORITMOS PROBABILÍSTICOS

Para intentar resolver la infactibilidad computacional de ciertos problemas, surge un modelo alternativo de computación que utiliza métodos probabilísticos. Estos métodos no pueden asegurar cotas superiores absolutas de tiempo, e incluso pueden devolver una respuesta errónea. Sin embargo, dadas unas cotas muy pequeñas de errores, en la práctica, ciertos métodos probabilísticos son más eficientes que los algoritmos conocidos, pues el tiempo de ejecución *esperado* del método, calculado probabilísticamente, es menor que el orden del algoritmo original.

Definición 2.18. Llamamos *algoritmo de Monte Carlo* a un algoritmo probabilístico que resuelve un problema de decisión, pero tiene un error ϵ de equivocarse.

Decimos que es *parcialmente* Verdadero si cuando se le da una instancia Verdadera nunca se equivoca, pero si la instancia es Falsa puede devolver Verdadero con probabilidad ϵ. De la misma manera, decimos que es *parcialmente* Falso si siempre resuelve correctamente instancias Falsas, pero puede cometer un error al resolver instancias Verdaderas.

Definición 2.19. Llamamos *algoritmo de Las Vegas* a un algoritmo probabilístico que resuelve un problema de decisión, pero o bien lo resuelve correctamente, o bien informa de error y termina sin resolver el problema con una probabilidad ϵ.

Ejemplo 2.20 (Test de primalidad)**.** El problema de decisión

Nombre:	Problema de primalidad (PRIM).
Parámetros:	Un entero n.
Pregunta:	¿Es n primo?

es un problema en **NP∩co-NP** (es el contrario del problema del entero compuesto), pero no se conoce ningún algoritmo que lo resuelva en tiempo polinómico, por ello no sabemos aún si pertenece a **P**.

Un algoritmo probabilístico basado en el Pequeño Teorema de Fermat, $a^{n-1} \equiv 1 \bmod n$ si n es primo para cualquier $a \in \mathbb{Z}_n$, puede utilizarse para comprobar si n es primo.

Si n es primo, para todo a que se elija, se cumplirá $a^{n-1} \equiv 1 \bmod n$, por lo que es un algoritmo de Monte Carlo parcialmente Verdadero, para una instancia Verdadera nunca se equivoca.

Sin embargo, si n es compuesto, pueden existir valores a que cumplan la propiedad. Por ejemplo, si n es un *número de Carmichael*, todo a coprimo con n cumple $a^{n-1} \equiv 1 \bmod n$.

Este no es el test de primalidad más fuerte, existen otros mejorados, como el test de primalidad de Miller-Rabin, que asegura que a lo sumo da un falso positivo para $\frac{1}{4}$ de todos los enteros a, $0 < a \leqslant p - 1$. Ejecutando el test un número k de veces, eligiendo aleatoriamente $a \in \mathbb{Z}_n$, obtenemos una probabilidad de error de $\epsilon = \frac{1}{4^k}$, lo que podría darnos en 50 iteraciones un

algoritmo ejecutable en tiempo polinomial, con probabilidad de error, cuando n es compuesto, de 2^{-50}, algo que podemos estar dispuestos a asumir en la práctica.

3

PRELIMINARES DE GRAFOS

TODO : introducción del capítulo La teoría necesaria para comprender las aplicaciones de ZKP con grafos es muy básica, los resultados de esta sección se pueden consultar en [9].

3.1 TEORÍA DE GRAFOS

Definición 3.1. Un *grafo* (simple no dirigido) es un par (V, E) formado por un conjunto finito $V \neq \emptyset$, a cuyos elementos denominaremos *vértices* o *nodos*, y E, un conjunto de pares (no ordenados y formados por distintos vértices) de elementos de V, a los que llamaremos *aristas*.

A los nodos v_1 y v_2 que forman una arista $e = (v_1, v_2)$ se les llama *extremos* de e.

Definición 3.2 (Conceptos)**.**

A dos nodos v_1 y v_2 que forman parte de una misma arista se les llama *adyacentes*.

Se llama *orden* de un grafo a $| V |$.

Se llama *tamaño* de un grafo a $| E |$.

Un grafo con $| V | = 1$ (consecuentemente $| E | = 0$) se llama *trivial*.

Se dice que una arista es *incidente* con un vértice v cuando v es uno de sus extremos.

Definición 3.3. Dos grafos $G_1 = (V_1, E_1)$ y $G_2 = (V_2, E_2)$ se dice que son *isomorfos*, lo cual denotaremos como $G_1 \simeq G_2$, si existe una aplicación biyectiva $\tau : V_1 \to V_2$ tal que $\forall u, v \in V_1$

$$(u, v) \in E_1 \Leftrightarrow (\tau(u), \tau(v)) \in E_2.$$

Tal aplicación recibe el nombre de *isomorfismo*. Si $G_1 = G_2$, llamaremos a τ *automorfismo*.

En un abuso de notación, podemos escribir que $\tau(G_1) = G_2$.

Dados dos grafos $G_1 = (V, E_1)$ y $G_2 = (V, E_2)$ isomorfos, con el mismo conjunto de vértices, $V = \{v_1, \dots v_n\}$, vemos que el isomorfismo entre G_1 y G_2 puede denotarse como una permutación en el índice de sus nodos.

Utilizaremos la notación $Sym(V)$ para denotar los automorfismos de $G = (V, E)$, que será el grupo simétrico S_n de las permutaciones de los vértices, cuando $| V | = n$. Por lo visto en los preliminares de álgebra, tenemos que $| Sym(V) | = n!$.

COLORACIÓN DE GRAFOS

Definición 3.4. Una *coloración* de un grafo $G = (V, E)$ es una aplicación $c : V \to \{1, 2, \ldots, l\}$.

El valor $c(v_i)$ es el *color* correspondiente al nodo v_i.

Definición 3.5. Una *coloración propia* de un grafo es una coloración que hace corresponder colores diferentes a vértices adyacentes:

$\forall (u, v) \in E \to c(u) \neq c(v)$.

Definición 3.6. Llamaremos *número cromático* del grafo G, $\chi(G)$, al mínimo valor de l que permite una coloración propia de G, es decir, al mínimo número de colores necesarios para colorear los vértices de forma que los extremos de cada arista tengan colores distintos.

Definición 3.7. Un *grafo plano* es (V, E), un par de conjuntos finitos llamados *vértices* y *aristas* que satisfacen:

- $V \subset \mathbb{R}^2$.
- Cada arista de E es un arco entre dos vértices distintos.
- Aristas diferentes tienen extremos diferentes.
- El interior de una arista no contiene ni vértices ni puntos de otra arista.

Para cada grafo plano G, el conjunto $\mathbb{R}^2 \setminus (V \cup E)$ es abierto. Sus regiones se llaman *caras*.

Definición 3.8. Un grafo isomorfo a un grafo plano, se denomina *grafo planar*.

Cada representación de un grafo planar como un grafo plano se llama un *embutido*.

Teorema 3.9. *Si* G *es un grafo planar, entonces* $\chi(G) \leqslant 4$.

Este último resultado es el conocido como Teorema de los cuatro colores, que en una versión menos técnica dice:

Teorema 3.10. *Dado cualquier mapa geográfico con regiones continuas, este puede ser coloreado con cuatro colores diferentes, de forma que no queden regiones adyacentes con el mismo color.*

CAMINOS HAMILTONIANOS

TODO

REPRESENTACIÓN DE GRAFOS

TODO: si no implementamos nada de grafos, no hace falta

Dado $G = (V, E)$ con $V = \{v_1, \ldots v_n\}$ y $E = \{e_1, \ldots e_m\}$, podemos representarla por:

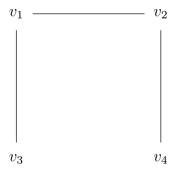
- La *matriz de adyacencia* $M_{nxn} = (m_{ij})$, que se construye mediante:

$$m_{ij} = \begin{cases} 1 & \text{si } (v_i, v_j) \in E \quad (\Leftrightarrow (v_j, v_i) \in E), \\ 0 & \text{en otro caso.} \end{cases}$$

- La *matriz de incidencia* (vértice-arista) $M_{nxm} = (m_{ij})$ con

$$m_{ij} = \begin{cases} 1 & \text{si } v_i \text{ es extremo de } e_j, \\ 0 & \text{en otro caso.} \end{cases}$$

- Una matriz de enteros M_{mx2} cuyas filas contienen los índices de los extremos inicial y final de las aristas del grafo.
  Ejemplo 3.11. El grafo:



se representaría como:

1	2
1	3
2	4

3.2 PROBLEMAS DE DECISIÓN BASADOS EN GRAFOS

A continuación, introducimos los enunciados de los problemas de grafos utilizados en pruebas de conocimiento cero, y referencias donde se estudian sus órdenes de complejidad.

3.2.1 *Problema del isomorfismo de grafos*

Nombre:	Problema GI (*Graph Isomorphism*).
Parámetros:	Dos grafos $G_0 = (V_0, E_0)$ y $G_1 = (V_1, E_1)$, del mismo orden $\mid V_0 \mid = \mid V_1 \mid = n$.
Pregunta:	¿Existe un isomorfismo $\pi : V_0 \to V_1$ tal que una arista $(u, v) \in E_0$ si y solo si $(\pi(u), \pi(v)) \in E_1$?

TODO

3.2.2 *Problema del camino hamiltoniano*

Nombre:	Problema CH.
Parámetros:	Un grafo $G = (V, E)$.
Pregunta:	¿Existe un ciclo en G que recorre pasa por cada vértice en V una única vez?

TODO

3.2.3 *Problema de la 3-coloración*

Nombre:	Problema G3C.
Parámetros:	Un grafo $G = (V, E)$.
Pregunta:	¿Existe una función $\phi : V \to \{1, 2, 3\}$ tal que $\forall (u, v) \in E$, se cumple $\phi(u) \neq \phi(v)$?

TODO

4

PRELIMINARES DE ÁLGEBRA

4.1 GRUPOS Y ANILLOS

Vamos a recordar los conceptos más básicos de la teoría de grupos, empezando con la definición de operación binaria:

Definición 4.1. Una *operación binaria* en un conjunto A es una aplicación $* : A \times A \to A$.

Según una operación binaria cumpla ciertas propiedades, diremos que:

- La operación es *asociativa* si $a * (b * c) = (a * b) * c \quad \forall a, b, c \in A$.
- La operación es *conmutativa* si $a * b = b * a \quad \forall a, b \in A$.
- El elemento $e \in A$ es un *elemento neutro* para $*$ si $a * e = e * a = a \quad \forall a \in A$. Además es único cuando existe neutro.
- Cuando existe el neutro e, el elemento b es el *simétrico* de a si $a * b = b * a = e$.

Podemos ahora dar formalmente la definición de grupo:

Definición 4.2. Un *grupo* es un conjunto G con una operación asociativa, con elemento neutro y con simétrico para cada elemento. Si la operación es además conmutativa, se dice que G es un *grupo abeliano*.

Con la operación suma $+$ habitual, los conjuntos $\mathbb{Z}$, $\mathbb{Q}$ y $\mathbb{R}$ son grupos abelianos aditivos. Dado un conjunto A con una operación multiplicativa $\cdot$, A^* denotará el *conjunto de los elementos invertibles en A*. Así, con la multiplicación usual, $Q^* = Q \setminus \{0\}$, y $\mathbb{Z}^* = \{-1, 1\}$.

Definición 4.3. Un *anillo* es un conjunto A con dos operaciones, $+$ y $\cdot$ (suma y producto), tales que:

- $(A, +)$ es un grupo abeliano aditivo; con neutro 0.
- El producto es asociativo y tiene neutro 1, distinto del neutro aditivo ($1 \neq 0$).
- El producto es distributivo con respecto a la suma, es decir, $\forall a, b, c \in A$ se tiene $a \cdot (b + c) = a \cdot b + a \cdot c$ y $(b + c) \cdot a = b \cdot a + c \cdot a$.

El anillo es *conmutativo* si lo es el producto. Un anillo conmutativo donde todo elemento $\neq 0$ tiene *inverso* (simétrico para el producto), se dice que es un *cuerpo*.

4.2 ARITMÉTICA EN $\mathbb{Z}$

Definición 4.4. Un entero d se dice que es el **máximo común divisor** de dos enteros a y b si es el mayor entero que divide a ambos. Lo denotaremos como $d = mcd(a, b)$.

Euclides describió en su obra *Los Elementos* un método para calcular el mcd de dos enteros, que hoy en día se conoce como *Algoritmo de Euclides*. La propiedad que utiliza el algoritmo para calcular el mcd es la siguiente:

Proposición 4.5. *Sean $a, b \in \mathbb{Z}$. Entonces, para todo $\alpha \in \mathbb{Z}$ se tiene:*

$$mcd(a, b) = mcd(a, b - \alpha a) = mcd(a - \alpha b, b).$$

En particular, cuando $b \neq 0$ y la división entera de a entre b es $a = bq + r$, tenemos que $mcd(a, b) = mcd(b, r)$.

Algoritmo 4.6 (Euclides)**.** Encuentra el mcd de $a, b \in \mathbb{Z}$:

1. Inicializa $r_0 = a$ y $r_1 = b$.
2. Calcula las siguientes divisiones euclídeas

$$\begin{aligned} r_0 &= q_1 r_1 + r_2 \\ r_1 &= q_2 r_2 + r_3 \\ &\dots \\ r_{n-3} &= q_{n-2} r_{n-2} + r_{n-1} \\ r_{n-2} &= q_{n-1} r_{n-1} + r_n \end{aligned}$$

   hasta que se obtenga un $r_n = 0$, con $r_{n-1} \neq 0$.
3. Como $b = r_1 > r_2 > \dots \geqslant 0$ y cada r_i es entero, para $i = 1, 2, \dots$, se obtiene $r_n = 0$ en un número finito de pasos y acaba el algoritmo con $mcd(a, b) = r_{n-1}$.

A partir del *Algoritmo de Euclides* se puede expresar d como una "combinación $\mathbb{Z}$-lineal" de a y b:

$$d = as + bt$$

conocida como la *Identidad de Bézout*.

El algoritmo se conoce como *Algoritmo de Euclides extendido*:

Algoritmo 4.7 (Euclides extendido)**.** Encuentra el $d = mcd$ de $a, b \in \mathbb{Z}$ y valores $s, t \in \mathbb{Z}$ tal que $d = as + bt$.

1. Inicializa $r_0 \leftarrow a, r_1 \leftarrow b, s_0 \leftarrow 1, t_0 \leftarrow 0, s_1 \leftarrow 0, t_1 \leftarrow 1$
   $i \leftarrow 1$
2. Mientras $r_i \neq 0$:
   Calcule la división euclídea $r_{i-1} = q_i r_i + r_{i+1}$
   $s_{i+1} \leftarrow s_{i-1} - q_i s_i$
   $t_{i+1} \leftarrow t_{i-1} - q_i t_i$
   $i \leftarrow i + 1$
3. $d = r_{i-1} \quad s = s_{i-1} \quad t = t_{i-1}$

Observación. Los valores de s y t no tienen por qué ser únicos:
$a(s - kb) + b(t + ka) = as - kba + bt + kba = as + bt = d$

Utilizaremos la Identidad de Bézout para calcular los inversos en aritmética modular más adelante.

4.3 CONGRUENCIAS: EL ANILLO $\mathbb{Z}_n$

Definición 4.8. Sean $a, b, n \in \mathbb{Z}$, $n \neq 0$, diremos que a y b son **congruentes módulo** n, y lo escribiremos $a \equiv b \bmod n$, si la diferencia $a - b$ es múltiplo de n.

Cuando $a \equiv b \bmod n$ decimos que b es un ***residuo*** *de* a *módulo* n.

Proposición 4.9. *La relación de* ***congruencia módulo*** n *es una relación de equivalencia, es decir, es reflexiva, simétrica y transitiva.*

Esto establece una relación de equivalencia en $\mathbb{Z}$, en la que la **clase** de un entero a módulo n es $\overline{a} = \{a + kn\}_{k \in \mathbb{Z}}$. Cuando no exista confusión, escribiremos la clase de equivalencia $\overline{a}$ como a. El correspondiente conjunto cociente, de las *clases de resto módulo* n, es $\mathbb{Z}_n = \{\overline{0}, \overline{1}, \ldots \overline{n-1}\}$, y hereda la suma y producto de $\mathbb{Z}$ convirtiéndose en un anillo conmutativo con neutros $\overline{0}$ para la suma y $\overline{1}$ para el producto.

Proposición 4.10. *El conjunto $\mathbb{Z}_n^*$ de los elementos invertibles, con el producto, de $\mathbb{Z}_n$, es un grupo abeliano.*

Teorema 4.11. *El anillo $\mathbb{Z}_n$ es un cuerpo cuando* n *es un número primo.*

Sea φ la *función de Euler*, tal que a cada entero $n = \pm p_1^{m_1} p_2^{m_2} \cdots p_k^{m_k}$, expresado como producto de primos, le asigna el valor:
$\varphi(n) = p_1^{m_1-1} p_2^{m_2-1} \cdots p_k^{m_k-1}(p_1 - 1)(p_2 - 1) \cdots (p_k - 1)$.
La función de Euler indica el número de enteros entre 1 y $n - 1$ coprimos con n.

Proposición 4.12. $| \mathbb{Z}_n^* | = \varphi(n)$, *donde φ es la función de Euler.*

Teorema 4.13 (Euler). *Si* x *es coprimo con* n*, entonces* $x^{\varphi(n)} \equiv 1 \bmod n$.

Si tenemos ahora dos enteros a y b coprimos, es decir, $\mathrm{mcd}(a, b) = 1$, usando el *algoritmo de Euclides extendido* podemos encontrar r y s de la *Identidad de Bézout* tales que:

$$as + bt = 1$$

Si a esta igualdad le aplicamos módulo b, obtenemos el inverso de a en $\mathbb{Z}_b$:

$$\begin{array}{ccccc} (as + bt) \bmod b & \equiv & as \bmod b & \equiv & 1 \bmod b \\ \overline{as + bt} & = & \overline{as} & = & \overline{1} \end{array}$$

Así hemos demostrado el siguiente resultado:

Proposición 4.14. *Si* $\mathrm{mcd}(a, n) = 1$*, entonces el elemento inverso* a^{-1}*,* $0 < a^{-1} < n$*, existe y* $aa^{-1} \equiv 1 \bmod n$.

Teorema 4.15 (Teorema Chino de los restos)**.**
Supongamos que $n_1, n_2, \ldots, n_k$ *son enteros positivos coprimos dos a dos. Entonces, para enteros dados* $a_1, a_2, \ldots, a_k$*, existe un entero* x *que resuelve el sistema de congruencias simultáneas*

$$\begin{aligned} x &\equiv a_1 \pmod{n_1} \\ x &\equiv a_2 \pmod{n_2} \\ &\vdots \\ x &\equiv a_k \pmod{n_k} \end{aligned}$$

Más aún, todas las soluciones x *de este sistema son congruentes módulo el producto* $N = n_1 n_2 ... n_k$.

Otra interpretación del teorema es: para cada entero positivo con factorización en números primos:
$n = p_1^{r_1} \cdots p_k^{r_k}$
se tiene un isomorfismo entre un anillo y la suma directa de sus potencias primas:
$\mathbf{Z}_n \cong \mathbf{Z}_{p_1^{r_1}} \oplus \cdots \oplus \mathbf{Z}p_k^{r_k}$

Definición 4.16. El *orden* de un elemento $a \in \mathbb{Z}_n^*$, denotado $o(a)$, es el menor entero positivo t tal que $a^t \equiv 1 \bmod n$.

Proposición 4.17. *Si el orden de* $a \in \mathbb{Z}_n^*$ *es* $o(a) = t$*, y ocurre que* $a^2 \equiv 1 \bmod n$*, entonces* $t \mid s$*. En particular,* $t \mid \varphi(n)$.

Definición 4.18. Sea $\alpha \in \mathbb{Z}_n^*$. Si el orden de α es $\varphi(n)$, entonces se dice que α es un *generador* de $\mathbb{Z}_n^*$. Se indica como $\mathbb{Z}_n^* = \langle \alpha \rangle$. Si $\mathbb{Z}_n^*$ tiene un generador, se dice que es *cíclico*.

Proposición 4.19 (Propiedades de los generadores de $\mathbb{Z}_n^*$)**.**

- *$\mathbb{Z}_n^*$ tiene un generador si y solo si $n = 2, 4, p^k$ ó $2p^k$, con p primo y $k \geqslant 1$.*
- *Si α es un generador de $\mathbb{Z}_n^*$, entonces $\mathbb{Z}_n^* = \{\alpha^i \bmod n \mid 0 \leqslant i \leqslant \varphi(n) - 1\}$.*

4.4 EL GRUPO SIMÉTRICO S_n: PERMUTACIONES

Definición 4.20. Denotamos con S_n al *grupo simétrico* de las biyecciones o *permutaciones* del conjunto

$$\mathbb{N}_n = \{1, 2, 3, \ldots n\}$$

con la operación composición $\circ$.

Se dice que una permutación $\sigma \in S_n$ *fija* al elemento $i \in \mathbb{N}_n$ si $\sigma(i) = i$, y en caso contrario lo *mueve*.

La composición siempre es asociativa, y el elemento neutro es la aplicación identidad, que deja fijos a todos los elementos.

Ejemplo 4.21. Una permutación $\sigma \in S_n$ puede describirse poniendo los elementos $1, 2, \ldots, n$ y debajo sus imágenes $\sigma(1), \sigma(2), \ldots, \sigma(n)$. Por ejemplo, en S_6 tenemos la permutación:

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 3 & 6 & 4 & 1 & 5 & 2 \end{pmatrix},$$

que fija al 5, intercambia al 2 con el 6, y con el resto hace un *ciclo* $1 \mapsto 3 \mapsto 4 \mapsto 1$.

Observación. El grupo S_n tiene $n!$ elementos, $|\, S_n \,| = n!$, pues son las posibles formas de ordenar n elementos: n opciones para el primero, $n - 1$ para el segundo, etc.

4.5 PROBLEMA DEL LOGARITMO DISCRETO

TODO Introducir aquí el problema del logaritmo discreto

Nombre:	Problema DL (*Discrete Logarithm*).
Parámetros:	Un grupo cíclico G de orden q primo, donde se supone difícil el problema del logaritmo discreto, un generador g, $G = \langle g \rangle$, y un elemento $y \in G$.
Pregunta:	¿Conoce P el entero $s \in \mathbb{Z}_q$ tal que $g^s = y$, o equivalentemente, $\log_g y = s$?

5

RESIDUOS CUADRÁTICOS

TODO : Párrafo de introducción al capítulo y referencias. Indicar que la teoría de QR no es vista en el grado y se desarrolla más allá de los preliminares de álgebra.

Teoría de símbolos de Lebesgue, ..., residuos cuadráticos, cálculo de raíz discreta?

5.1 PRIMERAS PROPIEDADES

Definición 5.1. Sea $a \in \mathbb{Z}_n^*$. Se dice que a es un *residuo cuadrático* módulo n, o un *cuadrado* módulo n, si existe un $x \in \mathbb{Z}_n^*$ tal que $x^2 \equiv a \bmod n$. Si no existe dicho x, entonces a se llama un *no-residuo cuadrático* módulo n.

Al conjunto de todos los residuos cuadráticos módulo n de $\mathbb{Z}_n^*$ los denotaremos como Q_n o bien como $\mathbb{Z}_n^{Q+}$. Al conjunto de los no-residuos cuadráticos lo denotamos como $\overline{Q_n}$.

Ejemplo 5.2. Si tomamos $n = 4$, los no-residuos cuadráticos son 2 y 3, y el único residuo cuadrático es 1:

$$1^2 \equiv 1 \bmod 4 \qquad 2^2 \equiv 0 \bmod 4 \qquad 3^2 \equiv 1 \bmod 4$$

Observación. Por definición $0 \notin \mathbb{Z}_n^*$, y por tanto $0 \notin Q_n$ ni $0 \notin \overline{Q_n}$.

Definición 5.3. Sea $a \in Q_n$. Si $x \in \mathbb{Z}_n^*$ satisface $x^2 \equiv a \bmod n$, entonces x se llama *raíz cuadrada* módulo n de a.

Proposición 5.4. *Sea p un primo impar. Se cumple que $|Q_p| = \frac{p-1}{2}$ y $|\overline{Q_p}| = \frac{p-1}{2}$, es decir, la mitad de los elementos de $\mathbb{Z}_p^*$ son residuos cuadráticos, y la otra mitad no-residuos cuadráticos.*

Demostración. Sea $\alpha \in \mathbb{Z}_p^*$ un generador de $\mathbb{Z}_p^*$. Un elemento $a \in \mathbb{Z}_p^*$ es un residuo cuadrático módulo p si y solo si $a \equiv \alpha^i \bmod p$ donde i es un entero par. Como p es primo, $\phi(p) = p - 1 = |\mathbb{Z}_p^*|$, que es un entero par, y de ahí se sigue el enunciado, la mitad de los elementos son generados por un i par, la otra mitad por un i impar. □

Ejemplo 5.5. Para $p = 13$ tenemos que $\alpha = 6$ es un generador de $\mathbb{Z}_{13}^*$. Las potencias de α módulo 13 son:

i	1	2	3	4	5	6	7	8	9	10	11	12
$6^i \bmod 13$	6	10	8	9	2	12	7	3	5	4	11	1

Lo que nos da $Q_{13} = \{1, 3, 4, 9, 10, 12\}$ y $\overline{Q_{13}} = \{2, 5, 6, 7, 8, 11\}$.

Proposición 5.6. *Sea n un producto de dos primos impares p y q, $n = pq$. Entonces $a \in \mathbb{Z}_p^*$ es un residuo cuadrático módulo n, $a \in Q_n$ si y solo si $a \in Q_p$ y $a \in Q_q$. Se sigue que $|Q_n| = |Q_p| \cdot |Q_q| = \frac{(p-1)(q-1)}{4}$, y por tanto $|\overline{Q_n}| = |\mathbb{Z}_n^*| - |Q_n| = \frac{3(p-1)(q-1)}{4}$.*

Demostración. Si a es un residuo cuadrático módulo $n = pq$, $a \equiv x^2 \bmod n$, es inmediato que en módulos p y q se cumple $a \equiv x^2 \bmod p$, $a \equiv x^2 \bmod q$.

Si tenemos que a es un residuo cuadrático módulo p, $a \equiv x_p^2 \bmod p$, y también módulo q, $a \equiv x_q^2 \bmod q$, por el Teorema Chino de los Restos existe un x tal que:

$x \equiv x_p \bmod p \quad$ y $\quad x \equiv x_q \bmod q$

De modo que, elevando al cuadrado:

$x^2 \equiv x_p^2 \equiv a \bmod p$

$x^2 \equiv x_q^2 \equiv a \bmod q$

Por lo que $x^2 \equiv a \bmod n$.

□

5.2 SÍMBOLO DE LEGENDRE

Para identificar los residuos cuadráticos disponemos de una herramienta muy útil:

Definición 5.7. Dados un primo impar p y un entero a, se define el *símbolo de Legendre* $\left(\frac{a}{p}\right)$ como

$$\left(\frac{a}{p}\right) = \begin{cases} 0, & \text{si } a \equiv 0 \bmod p \\ 1, & \text{si } a \in Q_p \\ -1, & \text{si } a \in \overline{Q_p} \end{cases}$$

Veamos ahora algunas propiedades del símbolo de Legendre:

Teorema 5.8 (Criterio de Euler)**.** *Sea p un primo impar. Sea $a \not\equiv 0 \bmod p$. Entonces:*

$$\left(\frac{a}{p}\right) \equiv a^{(p-1)/2} \bmod p$$

Demostración. Observemos primero que las raíces de 1 módulo p son 1 y $-1 \bmod p$. También que por el Teorema de Euler, $a^{\Phi(p)} \equiv a^{p-1} \equiv 1 \bmod p$.

De este modo, tenemos que $a^{\frac{p-1}{2}} \equiv \pm 1 \bmod p$.

Ahora demostrar el teorema es equivalente a demostrar que $a^{(p-1)/2} \equiv 1 \bmod p$ si y solo si a es un residuo cuadrático.

Supongamos que a es un residuo cuadrático módulo p. Sea x tal que $x^2 \equiv a \bmod p$. Entonces, $a^{\frac{p-1}{2}} \equiv x^{(p-1)} \equiv 1 \bmod p$, de nuevo por el Teorema de Euler.

Sea ahora $a^{\frac{p-1}{2}} \equiv 1 \bmod p$.

Tomamos g un generador de $\mathbb{Z}_p^*$, de modo que $a \equiv g^r \bmod p$.

Sustituyendo: $g^{r\frac{p-1}{2}} \equiv 1 \bmod p$, y utilizando que g tiene orden $p-1$, nos queda $g^{r\frac{p-1}{2}} \equiv g^{\frac{r}{2}} \equiv 1 \bmod p$, de donde deducimos que necesariamente r es un entero par, $r = 2s$.

Construimos $x \equiv g^s \bmod p$, que cumple: $x^2 \equiv g^{2s} \equiv g^r \equiv a \bmod p$, de modo que a es un residuo cuadrático módulo p.

□

Ejemplo 5.9. Sea $p = 13$ y $a = 5$, como $5^{11} \equiv -1 \bmod 23$, por el criterio de Euler, $\left(\frac{5}{23}\right) = -1$, por lo que 5 es un no-residuo cuadrático de 23.

Proposición 5.10. *Sean p un primo impar, $a, b \in \mathbb{Z}_p$. Si $a \equiv b \bmod p$, entonces $\left(\frac{a}{p}\right) = \left(\frac{b}{p}\right)$.*

Demostración. Si a es residuo cuadrático, entonces existe $x \in \mathbb{Z}_p^*$ tal que $x^2 \equiv a \equiv b \bmod p$, y b es también residuo cuadrático. Análogo en el caso contrario. □

Proposición 5.11 (Propiedad multiplicativa del símbolo de Lebesgue)**.** *Sean a y b enteros coprimos con p, un primo impar. Entonces:*

$$\left(\frac{a}{p}\right)\left(\frac{b}{p}\right) = \left(\frac{ab}{p}\right).$$

En particular, el producto de dos no-residuos cuadráticos es un residuo cuadrático.

Demostración. Utilizando el Criterio de Euler:

$$\left(\frac{a}{p}\right)\left(\frac{b}{p}\right) \equiv a^{(p-1)/2} \cdot b^{(p-1)/2} \bmod p \equiv (ab)^{(p-1)/2} \bmod p \equiv \left(\frac{ab}{p}\right)$$

□

Corolario 5.12. *Sea p primo impar, $a \in \mathbb{Z}_p^*$, entonces $\left(\frac{a^2}{p}\right) = 1$*

Demostración. Como $\left(\frac{a}{p}\right) = \pm 1$, $\left(\frac{a^2}{p}\right) = \left(\frac{a}{p}\right) \cdot \left(\frac{a}{p}\right) = 1$. □

Teorema 5.13 (Ley de reciprocidad cuadrática)**.** *Sean p y q primos impares distintos, se cumple:*

$$\left(\frac{p}{q}\right)\left(\frac{q}{p}\right) = (-1)^{(p-1)(q-1)/4}.$$

O de otro modo:

$$\left(\frac{p}{q}\right) = \begin{cases} \left(\frac{q}{p}\right) & \text{Si} \quad p \equiv 1 \bmod 4 \quad \text{ó} \quad q \equiv 1 \bmod 4 \\ -\left(\frac{q}{p}\right) & \text{Si} \quad p \equiv q \equiv 3 \bmod 4. \end{cases}$$

TODO: la demostración es muy larga, requiere de 2 lemas más, también extensos, y luego sólo usamos la ley para demostrar la equivalente en Símbolo de Jacobi. Indicar que Gauss la demostró de 2 maneras distintas en *Disquisitones Arithmeticae*, que hay una docena de demostraciones, y referencias.

Proposición 5.14 (Primera ley suplementaria)**.** *Sea* p *primo impar. Entonces:*

$$\left(\frac{-1}{p}\right) = (-1)^{(p-1)/2}$$

O de otro modo:

$$\left(\frac{-1}{p}\right) = \begin{cases} 1 & \text{Si} \quad p \equiv 1 \bmod 4 \\ -1 & \text{Si} \quad p \equiv 3 \equiv -1 \bmod 4. \end{cases}$$

Demostración. La primera expresión es inmediata por el criterio de Euler:

$$\left(\frac{-1}{p}\right) = (-1)^{(p-1)/2} \bmod p$$

y nos basta hacer un análisis de casos respecto a la congruencia de $p \bmod 4$.

Como p es primo impar, no será congruente con $\overline{0}$ ni $\overline{2}$.

Si $p \equiv 1 \bmod 4$, podemos escribir $p = 4k+1$ para algún entero k. Entonces,

$$(-1)^{(p-1)/2} = (-1)^{2k} = 1,$$

por lo que $\left(\frac{-1}{p}\right) = 1$.

Y si $p \equiv 3 \bmod 4$, podemos escribir $p = 4k+3$ para algún entero k. Entonces,

$$(-1)^{(p-1)/2} = (-1)^{2k+1} = -1,$$

y tenemos que $\left(\frac{-1}{p}\right) = -1$. □

Proposición 5.15 (Segunda ley suplementaria)**.** *Sea* p *primo impar. Entonces:*

$$\left(\frac{2}{p}\right) = (-1)^{(p^2-1)/8}$$

O de otro modo:

$$\left(\frac{-1}{p}\right) = \begin{cases} 1 & \text{Si} \quad p \equiv \pm 1 \bmod 8 \\ -1 & \text{Si} \quad p \equiv \pm 3 \bmod 8. \end{cases}$$

5.3 SÍMBOLO DE JACOBI

El símbolo de Legendre está definido para módulos un primo impar p. Ahora vamos a ver una generalización del concepto:

Definición 5.16. Sean $a, N \in \mathbb{Z}$, con $N = p_1 p_2 \cdots p_r$, donde los p_i son primos impares, no necesariamente distintos.

Definimos el *Símbolo de Kronecker-Jacobi* $\left(\frac{a}{N}\right)$ como

$$\left(\frac{a}{N}\right) = \prod_{i=1}^{r} \left(\frac{a}{p_i}\right)$$

donde $\left(\frac{a}{p_i}\right)$ es el Símbolo de Legendre.

Observación. A diferencia del Símbolo de Legendre, el Símbolo de Jacobi $\left(\frac{a}{N}\right)$ no indica si a es un residuo cuadrático módulo N. Es cierto que si $a \in Q_N$, su Símbolo de Jacobi será $\left(\frac{a}{N}\right) = 1$, pero el contrario no se cumple.

Ejemplo 5.17. Sea $a = 2$ y $N = 15 = 3 \cdot 5$.

$$\left(\frac{2}{15}\right) = \left(\frac{2}{3}\right)\left(\frac{2}{5}\right) = (-1)\cdot(-1) = 1$$

Ya que $Q_3 = \{1\}$, $\overline{Q_3} = \{2\}$, y $Q_5 = \{1, 4\}$, $\overline{Q_5} = \{2, 3\}$.

Igual que antes, veamos algunas propiedades del Símbolo de Jacobi:
TODO: reescribir las propiedades más ordenadamente y demostrar.

Teorema 5.18. *Propiedades del Símbolo de Jacobi:*

(i) *Si* $a \equiv b \bmod N$, *entonces* $\left(\frac{a}{N}\right) = \left(\frac{b}{N}\right)$

(ii) $\left(\frac{a}{N}\right) = 0$ *si y sólo si* $\mathrm{mcd}(a, N) \neq 1$.

(iii) *Para cada* a, b *y* c *enteros, tenemos:*

$$\left(\frac{ab}{c}\right) = \left(\frac{a}{c}\right)\left(\frac{b}{c}\right), \quad \left(\frac{a}{bc}\right) = \left(\frac{a}{b}\right)\left(\frac{a}{c}\right) \quad \text{si } bc \neq 0.$$

(iv) *Las fórmulas del Teorema 5.13 se siguen verificando si* p *y* q *son enteros impares positivos, ya no necesitan ser primos.*

TODO: algoritmo para calcular símbolos de Jacobi
TODO: raíces cuadradas módulo n

5.4 EL PROBLEMA DE RESIDUOSIDAD CUADRÁTICA

Podemos introducir ahora el problema de decisión QR, donde dado un módulo N, compuesto e impar, decidir si un entero x con símbolo de Jacobi 1 respecto a N, es o no un residuo cuadrático:

Nombre:	Problema de residuosidad cuadrática (QR).
Parámetros:	Un entero compuesto impar N, y el entero $x \in \mathbb{Z}_N^Q$.
Pregunta:	¿Es x un residuo cuadrático, $x \in \mathbb{Z}_N^{Q+}$?

Si N fuera un número primo, el símbolo de Jacobi de x coincidiría con el de Legendre, y la respuesta sería siempre $\mathtt{Verdadero}$.

Si conocieramos la descomposición en primos de N, el algoritmo polinomial que resolvería el problema es calcular el símbolo de Legendre de x respecto de cada factor primo de N, hasta encontrar alguno que valga -1, y responder al problema con $\mathtt{Falso}$, o bien comprobar que todos los símbolos valen 1 y responder $\mathtt{Verdadero}$.

Con este algoritmo que utiliza la factorización de N, tenemos que el problema QR puede **reducirse polinomialmente** (2.14) al problema de factorización de N.

Proposición 5.19. $\mathrm{QR} \leqslant_P$ FACTORIZACIÓN

Si se desconoce la factorización de N, no se conoce a día de hoy ningún algoritmo eficiente para resolver el problema QR aparte del de intentar adivinar la respuesta. Por ejemplo, en el caso de $N = pq$, se tiene una probabilidad de acertar de $\frac{1}{2}$, por $|\mathbb{Z}_N^{Q+}| = |\mathbb{Z}_N^{Q-}| = \frac{(p-1)(q-1)}{4}$ (5.6).

Del mismo modo que no se sabe si **P=NP** (aunque se cree que no), aquí se cree que QR es tan difícil como el problema de factorización, pero no se conoce ninguna demostración aún. Bajo esta suposición, se construyen muchas aplicaciones criptográficas, entre ellas el cifrado de clave pública probabilístico de Goldwasser-Micali, el generador de números pseudo-aleatorios de Blum-Blum-Shub, o una de las pruebas de conocimiento cero más características, y que veremos en el capítulo siguiente.

TODO: el problema de factorización: indicar mejor solución, su orden, etc.

6

PRUEBAS DE CONOCIMIENTO CERO

Las pruebas de conocimiento cero, con siglas ZKP del inglés *Zero-Knowledge Proofs*, permiten demostrar la veracidad de una declaración, sin revelar nada más de ella. En las ZKP intervienen dos partes, el *Prover* y el *Verifier*, o probador y verificador. El prover asegura que una declaración es cierta, y el verifier quiere convencerse de ello a través de una interacción con el prover, de modo que al final de la misma, o bien acaba convencido de que la declaración es cierta, o bien descubre, con una alta probabilidad, que el prover mentía.

Las pruebas de conocimiento cero surgen a partir de los sistemas de pruebas interactivas, que forman una parte importante de la teoría de complejidad computacional, y pidiendo la propiedad de *conocimiento cero* obtenemos el subconjunto de sistemas interactivos que conforman las pruebas de conocimiento cero.

Las referencias para este capítulo se pueden encontrar en

6.1 SISTEMAS DE PRUEBAS INTERACTIVAS

Un *sistema de prueba interactivo* es un concepto de la teoría computacional que modela el intercambio de un número finito de mensajes entre dos partes, el probador P y el verificador V, con el objetivo de que P demuestre a V que una instancia de un problema de decisión es $\mathtt{Verdadera}$. V tiene una capacidad de cómputo limitada, a lo sumo un algoritmo probabilístico de tiempo de cómputo polinomial. P es computacionalmente todopoderoso. Al final del intercambio de mensajes, o bien V acepta que la instancia es $\mathtt{Verdadera}$, o bien la rechaza por ser $\mathtt{Falsa}$.

Definición 6.1. Se dice que un problema de decisión Q, no necesariamente en **NP**, tiene un *sistema de prueba interactivo* si tiene un protocolo de interacción polinomialmente acotado en número de mensajes que cumple:

- *Completitud* Para toda instancia q $\mathtt{Verdadera}$, del problema Q, V acepta q como $\mathtt{Verdadera}$.
- *Robustez* Para cada instancia q $\mathtt{Falsa}$, ningún P, incluso si no sigue el protocolo, puede convencer a V de que q es $\mathtt{Verdadera}$, excepto con una pequeña probabilidad.
- Alternativa:
- *Robustez* Para cada instancia q $\mathtt{Falsa}$, V rechaza la prueba de q con una probabilidad no menor que $\epsilon = 1 - n^{-c}$, para cualquier constante $c > 0$ y donde n es el tamaño de la instancia.

En resumen, si la instancia del problema Q que P quiere demostrar es $\mathtt{Verdadera}$, el protocolo siempre funciona, no hay falsos negativos, pero si la instancia es $\mathtt{Falsa}$, hay una pequeña probabilidad de que V la acepte como $\mathtt{Verdadera}$, pueden haber falsos positivos una probabilidad casi despreciable.

Un P o un V que no siguen el protocolo e intentan romper estas propiedades, los llamaremos un P o V *tramposos*.

Definición 6.2. Denominamos clase de problemas **IP** (Interactivos en tiempo Polinomial) al conjunto de problemas de decisión para los que existe un sistema de prueba interactivo.

Proposición 6.3. ***NP*** $\subset$ ***IP***.

Demostración. Sea Q un problema **NP**. Definimos el siguiente protocolo:

1. P resuelve la instancia del problema gracias a su capacidad de cómputo ilimitada y genera el certificado para V, que existe para cualquier instancia $\mathtt{Verdadera}$ por $Q \in$ **NP** (2.11).
2. V recibe y puede verificar el certificado en tiempo polinomial. Si es válido, V acepta como $\mathtt{Verdadera}$ la instancia. Si no, rechaza la prueba.

El protocolo es completo y robusto, con probabilidad nula de falso positivo, pues si la instancia es $\mathtt{Falsa}$, ningún P, honesto o tramposo, puede generar un certificado que no existe.

□

PRUEBA INTERACTIVA PARA EL PROBLEMA QR

Vamos a ver una prueba interactiva para demostrar que un entero x con Símbolo de Jacobi 1 respecto a n, $x \in \mathbb{Z}_n^Q$, es un residuo cuadrático, $x \in \mathbb{Z}_n^{Q+}$.

Nombre:	Problema QR.
Parámetros:	Un entero compuesto impar N, y el entero $x \in \mathbb{Z}_N^Q$.
Pregunta:	¿Es x un residuo cuadrático, $x \in \mathbb{Z}_N^{Q+}$?

Una instancia $\mathtt{Verdadera}$ del problema es un x residuo cuadrático módulo N. Una prueba interactiva para demostrar que x es residuo cuadrático es la siguiente:

Algoritmo 6.4 (Prueba interactiva para QR)**.**

Datos comunes: Una instancia (x, N) del Problema QR. n es el tamaño de la instancia.

Protocolo: Sea $t(n)$ un polinomio en n. P y V repiten $t(n)$ veces los siguientes pasos.

1. $P \to V: \quad u \in_R \mathbb{Z}_N^{Q+}$ (P elige aleatoriamente u en $\mathbb{Z}_N^{Q+}$).
2. $V \to P: \quad b \in_R \{0, 1\}$.
3. $P \to V:$ w, una raíz cuadrada módulo N aleatoria, de x si $b = 0$, o bien de $x \cdot u$ si $b = 1$.
4. V comprueba si:

$$w^2 \stackrel{?}{\equiv} \begin{cases} u \bmod N, & \text{si } b = 0 \\ xu \bmod N, & \text{si } b = 1. \end{cases}$$

   Si la comparación falla, V termina en rechazo. En caso contrario, vuelve al paso 1.

Tras $t(n)$ rondas, V termina y acepta la instancia como $\mathtt{Verdadera}$, x es un residuo cuadrático módulo N.

Teorema 6.5. *El problema QR tiene un sistema de prueba interactiva.*

Demostración. El protocolo se ejecuta $t(n)$ veces, un número de iteraciones polinomialmente asociado al tamaño n de la entrada, por lo que hay un número finito de mensajes que V, computacionalmente limitado, puede llevar a cabo.

Queda ver que el protocolo anterior es completo y robusto.

La prueba es *completa*, pues para cualquier instancia $\mathtt{Verdadera}$ de QR, $x \in \mathbb{Z}_N^{Q+}$, V acepta la prueba de P. En cada iteración, como P es computacionalmente todopoderoso, puede calcular w, una raíz cuadrada módulo N de x o xu, según el valor de b, ambos en $\mathbb{Z}_N^{Q+}$.

Para una instancia $\mathtt{Falsa}$, $x \in \mathbb{Z}_N^{Q-}$, cuando V envíe $b = 1$, si P sigue el protocolo u será un residuo cuadrático, pero $x \cdot u$ es un no-residuo cuadrático módulo N, por lo que no podrá calcular w por mucho poder computacional ilimitado que tenga. Un P tramposo podría intentar engañar a V en el caso $b = 1$ eligiendo u tal que xu es un residuo cuadrático, pero entonces u es un no-residuo cuadrático y fallaría la prueba si $b = 0$.

Una vez P se compromete con un u, residuo cuadrático o no, la probabilidad de que V lo rechace en esa iteración es 12, según elija $b = 0$ ó 1. Como el protocolo se ejecuta $t(n)$ veces, la probabilidad de que un P tramposo pueda engañara V en todos es de $2^{-t(n)}$. Vemos entonces que el protocolo cumple la propiedad de *robustez*. □

6.2 PRUEBAS DE CONOCIMIENTO CERO

Entre las pruebas interactivas existe un subconjunto que llamamos de *conocimiento cero* si durante el protocolo no se puede inferir información de P,

aparte de la veracidad de la instancia. En particular, aún tras realizar la prueba, y estar V convencido, éste no podría repetirla a otro verificador tomando el lugar de P.

Antes de ver una definición más formal de una prueba de conocimiento cero, necesitamos algunas definiciones previas.

Definición 6.6. Llamamos *Vista* a una transcripción de los mensajes intercambiados entre P y V durante la ejecución de una prueba interactiva.

Pensemos en un protocolo con 3 mensajes por iteración. En la i-ésima ronda, P envía a V el valor A_i como *compromiso*, V responde con el *reto* aleatorio B_i, y P termina enviando la *prueba* C_i. La tupla (A_i, B_i, C_i) son variables aleatorias de los posibles valores que se pueden intercambiar en una iteración. La Vista de la prueba interactiva sería $(A_1, B_1, C_1, A_2, B_2, C_2, \ldots, A_{t(n)}, B_{t(n)}, C_{t(n)})$, la secuencia de los $t(n)$ mensajes intercambiados entre P y V.

Una Vista sólo es de interés para una instancia $Verdadera$, donde P realmente conoce si es $Verdad$. Para una instancia cuya prueba falla, o bien es realmente una instancia $Falsa$ o P no puede probarla, por ser tramposo y no conocer una prueba o *secreto* que le permita pasar la prueba exitosamente con mayor probabilidad que la de los falsos positivos.

Definición 6.7. Llamamos ensamble probabilístico, o *ensemble* en inglés, a una familia numerable de variables aleatorias: $\{X_i\}_{i\in I}$, con I numerable.

Podemos estudiar entonces una Vista como un ensamble probabilístico. Dos Vistas serán iguales cuando las distribuciones de sus variables aleatorias sean idénticas.

Denotaremos con V^* a un verificador tramposo. Éste podría no generar los retos B_i anteriores de manera independiente, podría incluso utilizar información previa de otras Vistas para generar los B_i en un intento de extraer información extra de P. A esta información previa la llamaremos h (historial).

Para una instancia q y un verificador cualquiera V^*, escribimos la Vista de una prueba como:

$$Vista_{P,V^*}(q,h) = (q, h, A_1, B_1, C_1, \ldots, A_{t(n)}, B_{t(n)}, C_{t(n)}).$$

Para un verificador honesto F sería un generador de números aleatorios que no utiliza ninguno de los parámetros de entrada.

El verificador tramposo V^* generará los retos B_i con una función probabilística de tiempo polinomial F tal que

$$B_i = F(q, h, A_1, B_1, C_1, \ldots, A_{i-1}, B_{i-1}, C_{i-1}, A_i).$$

Definición 6.8. Un Simulador $S_{V^*}(q,h)$ es un algoritmo probabilístico de tiempo polinomial, que utiliza toda la información que V^* tiene disponible (el historial h y la función F), para generar una transcripción de una prueba interactiva, para una instancia q del problema Q, sin necesidad de interactuar con P.

Un Simulador se puede ver como un generador de ensambles probabilísticos, las Vistas de una prueba.

Podemos describir, por fin, la tercera propiedad, además de la *completitud* y *robustez*, que debe tener una prueba de conocimiento cero. Se dice también que son ZKP perfectas porque en la práctica se pueden considerar otras definiciones menos restrictivas, las ZKP estadísticas y computacionales.

Definición 6.9 (Propiedad de conocimiento cero)**.**
Un sistema de prueba interactiva (completo y robusto), para un problema de decisión Q, es *perfecta de conocimiento cero* si el ensamble $\mathrm{Vista}_{P,V}(q, h)$ es idéntico al ensamble generado por un Simulador $S_{V^*}(q, h)$, para cualquier instancia $\mathtt{Verdadera}\ q \in Q$ y cualquier historial h.

A primera vista, puede parecernos que nada tiene que ver la existencia de tal Simulador con no poder obtener información de P más allá de si la instancia es $\mathtt{Verdadera}$. Pero es precisamente el hecho de que cualquier máquina M, que **no** conoce la solución, ni un certificado para comprobarla (2.11), pueda generar transcripciones del protocolo idénticas a las que podríamos interceptar a un P y V verídicos. Si M no conoce el secreto, de sus simulaciones nada podemos sacar, y por tanto, tampoco de la interacción entre P y V.

6.2.1 *Residuos cuadráticos*

Ahora podemos volver al problema QR, que ya vimos en el Teorema 6.5 que su prueba interactiva cumplía *completitud* y *robustez*.

Teorema 6.10. *La prueba interactiva (6.4) del problema QR es de conocimiento cero.*

Demostración. Sea (x, N) una instancia $\mathtt{Verdadera}$ del problema QR, $\exists y \in \mathbb{Z}_N$ tal que $y^2 \equiv x \bmod N$. En la i-ésima ronda tenemos las siguientes variables aleatorias:

1. U_i, un residuo cuadrático aleatorio enviado por P en el primer mensaje, $u \in_R \mathbb{Z}_N^{Q+}$.
2. B_i, un bit aleatorio generado por V, $b \in_R \{0, 1\}$.
3. W_i, una *prueba* de P, $w \in_R \Omega_u$ o bien $w \in_R \Omega_{xu}$, según el valor de B_i, es decir, una raíz cuadrada aleatoria módulo N de u o xu, donde Ω_u y Ω_{xu} son el conjunto de raíces cuadradas módulo N de u y xu, respectivamente.

La Vista de una prueba para un verificador V^* cualquiera es:

$$\mathrm{Vista}_{P,V^*}(x, N, h) = (x, N, h, U_1, B_1, W_1, \ldots, U_{t(n)}, B_{t(n)}, W_{t(n)}).$$

Para simplificar la notación, escribiremos la variable aleatoria $V_i = (U_1, B_1, W_1, \ldots, U_i, B_i,$

Para un V honesto, todos los B_i son variables aleatorias independientes, uniformes en $\{0, 1\}$. Para un V tramposo, la función F, probabilística en tiempo polinomial, genera los valores $b_{i+1} = F(x, N, h, v_i, u_{i+1})$, cuando $V_i = v_i$. Unimos el estudio de ambos casos suponiendo, para un V honesto, F como un generador de bits aleatorio, un lanzamiento de moneda.

Ahora que tenemos toda la información accesible a V^* podemos construir un Simulador:

Simulador para el problema QR $S_{V^*}(x, N, h)$.

Datos: (x, N), una instancia Verdadera del problema QR; h, transcripciones de ejecuciones previas del protocolo; v_i, transcripción de la interacción actual (i rondas).

Ejecución: Repetir para $i + 1 \leqslant t(n)$:

1. Elegir $b_{i+1} \in_R \{0, 1\}$
2. Elegir $w_{i+1} \in_R \mathbb{Z}_N^*$
3. **Si** $b_{i+1} = 0$, **entonces** calcular $u_{i+1} \equiv w_{i+1}^2 \bmod N$
   Si no, $u_{i+1} \equiv w_{i+1}^2 \cdot x^{-1} \bmod N$
4. **Si** $b_{i+1} = F(x, N, h, v_i, u_{i+1})$, **entonces** añadir la tupla $(u_{i+1}, b_{i+1}, w_{i+1})$ a la transcripción.
   Si no, volver al paso 1.
5. $i = i + 1$

El Simulador se diferencia del protocolo 6.4 en que, en vez de elegir primero un residuo cuadrático u_{i+1}, elige los valores b_{i+1} y w_{i+1} aleatoriamente, y a partir de ellos calcula u_{i+1}. Entonces, una vez tiene el u_{i+1} necesario para la función F, calcula el bit que V^* hubiera enviado en una interacción real, y comprueba si es el mismo bit b_{i+1} elegido. Aquí es donde vemos que el Simulador es un algoritmo probabilístico en tiempo del tipo Las Vegas (si obtenemos la tupla $i + 1$, será una tupla correcta). La probabilidad de que el bit b_{i+1} sea igual que el obtenido de F es de $1/2$. En promedio, el Simulador necesitará dos rondas por cada tupla $(u_{i+1}, b_{i+1}, w_{i+1},)$, por lo que el tiempo de ejecución esperado es polinomial.

Tenemos una prueba interactiva (6.4) que genera las vistas:

$$\text{Vista}_{P,V^*}(x, N, h) = (x, N, h, U_1, B_1, W_1, \ldots, U_{t(n)}, B_{t(n)}, W_{t(n)}),$$

y un Simulador que genera las transcripciones:

$$S_{V^*}(x, N, h) = (x, N, h, U_1', B_1', W_1', \ldots, U_{t(n)}', B_{t(n)}', W_{t(n)}').$$

Para terminar la demostración, veamos por inducción en i que son iguales, es decir, cumplen la propiedad de *conocimiento cero*.

Para el caso $i = 0$ ambos ensambles son constantes, $(x, N, h) = (x, N, h)$, tenemos la misma instancia e historial.

Suponemos cierto el caso $i-1$, es decir, el ensamble de la Vista:

$$Vista_{P,V^*}(x, N, h) = (x, N, h, U_1, B_1, W_1, \ldots, U_{i-1}, B_{i-1}, W_{i-1}),$$

es igual al del Simulador:

$$S_{V^*}(x, N, h) = (x, N, h, U_1', B_1', W_1', \ldots, U_{i-1}', B_{i-1}', W_{i-1}').$$

Siguiendo el protocolo de la prueba interactiva, se generará la siguiente tupla de la Vista, (U_i, B_i, W_i). La variable U_i se elige al inicio aleatoriamente, por lo que es independiente. La v.a. B_i se calcula con F, por lo que depende de U_i, V_{i-1} y h. W_i depende de ambos. La probabilidad de la tupla nos queda:

$$P(U_i = u, B_i = b, W_i = w) = P(U_i = u) \cdot$$
$$\cdot P(B_i = b \mid V_{i-1} = v, U_i = u, h) \cdot P(W_i = w \mid U_i = u, B_i = b)$$

Sea $\alpha = | \mathbb{Z}_N^{Q+} |$, entonces $P(U_i = u) = \frac{1}{\alpha}$.

Denotamos $P(B_i = b \mid V_{i-1} = v, U_i = u, h) = p_b$, que dependerá de la F utilizada.

Por último, sea $\beta = | \Omega_u | = | \Omega_{xu} |$. Entonces, $P(W_i = w \mid U_i = u, B_i = 0) = \frac{1}{\beta}$, $\forall w \in \Omega_u$, y $P(W_i = w \mid U_i = u, B_i = 1) = \frac{1}{\beta}$, $\forall w \in \Omega_{xu}$.

En total nos queda, $P(U_i = u, B_i = b, W_i = w) = \frac{p_b}{\alpha\beta}$.

Ahora veamos la tupla generada por el Simulador, (U_i', B_i', W_i'). La v.a. U_i' se calcula a partir de B_i' y W_i'. La variable B_i' depende de U_i', V_{i-1} y h por F en el paso 4. Y la v.a. W_i' se elige aleatoriamente.

La probabilidad de la tupla es:

$$P(U_i' = u, B_i' = b, W_i' = w) = P(W_i' = w) \cdot$$
$$\cdot P(B_i' = b \mid V_{i-1} = v, U_i' = u, h) \cdot P(U_i' = u \mid W_i' = w, B_i' = b)$$

Sabemos que $| \mathbb{Z}_N^* | = \alpha \cdot \beta$, por lo que $P(W_i' = w) = \frac{1}{\alpha\beta}$.

La probabilidad

-$w \in \Omega_u \Leftrightarrow b = 0$
-$w \in \Omega_{xu} \Leftrightarrow b = 1$
-W_i', B_i' *indep.*

$$
\begin{aligned}
P(U_i' = u) &= P(U_i' = u, W_i' \in \Omega_u \cup \Omega_{xu}, B_i' \in \{0,1\}) = \\
&= \sum_{w \in \Omega_u} P(U_i' = u, W_i' = w, B_i' = 0) + \\
&+ \sum_{w \in \Omega_{xu}} P(U_i' = u, W_i' = w, B_i' = 1) = \\
&= \sum_{w \in \Omega_u} P(W_i' = w) P(B_i' = 0) + \sum_{w \in \Omega_{xu}} P(W_i' = w) P(B_i' = 1) = \\
&= \beta \cdot \frac{1}{\alpha\beta} \cdot (P(B_i' = 0) + P(B_i' = 1)) = \\
&= \frac{1}{\alpha}
\end{aligned}
$$

indica que U_i' tiene la misma distribución que U_i, de modo que, al calcular $b_i = F(x, N, h, v_{i-1}, u_i)$, se tiene $P(B_i' = b \mid V_{i-1} = v, U_i' = u, h) = p_b$, es decir, B_i' tiene la misma distribución que B_i.

Por construcción, dados w y b, en el simulador u tiene un único valor posible, $u \equiv w^2 x^{-b} \bmod N$, por tanto, la probabilidad de que dada la tupla (u, b, w), U_i' tenga el valor u condicionado a que $B_i' = b$ y que $W_i' = w$, es 1:
$P(U_i' = u \mid W_i' = w, B_i' = b) = 1$

En total, tenemos que $P(U_i' = u, B_i' = b, W_i' = w) = \frac{p_b}{\alpha\beta}$.

Terminamos así la inducción en i y los ensambles de la Vista y el Simulador son idénticos.

Concluimos que la prueba 6.4 del problema QR es perfecta de conocimiento cero.

□

6.2.2 *Isomorfismo de grafos*

Otro problema de decisión del que podemos dar una prueba interactiva de conocimiento cero es el de encontrar un isomorfismo entre grafos:

Nombre:	Problema GI (*Graph Isomorphism*).
Parámetros:	Dos grafos $G_0 = (V_0, E_0)$ y $G_1 = (V_1, E_1)$, del mismo orden $\mid V_0 \mid = \mid V_1 \mid = n$.
Pregunta:	¿Existe un isomorfismo $\pi : V_0 \to V_1$ tal que una arista $(u, v) \in E_0$ si y solo si $(\pi(u), \pi(v)) \in E_1$?

Teorema 6.11. *El problema GI tiene una prueba de conocimiento cero.*

Demostración.
Primero debemos dar un protocolo interactivo entre un P y un V que cumpla completitud y robustez. Después daremos un Simulador para demostrar la propiedad de conocimiento cero.

Algoritmo 6.12 (Prueba interactiva para GI)**.**

Datos comunes: Una instancia ($G_0 = (V_0, E_0)$, $G_1 = (V_1, E_1)$) del Problema GI. $| V_0 | = | V_1 | = n$ es el tamaño de la instancia.

Protocolo: P calcula el isomorfismo τ entre G_1 y G_0, es decir, $\tau(G_1) = G_0$.

Sea $t(n)$ un polinomio en n. P y V repiten $t(n)$ veces los siguientes pasos.

1. $P \to V$: $h = \pi(G_0)$, donde $\pi \in_R Sym(V_0)$.
2. $V \to P$: $b \in_R \{0, 1\}$.
3. $P \to V$: ω, tal que

$$\omega = \begin{cases} \pi & \text{si } b = 0 \\ \pi \circ \tau & \text{si } b = 1. \end{cases}$$

4. V comprueba si:

$$h \stackrel{?}{=} \begin{cases} \omega(G_0) & \text{si } b = 0 \\ \omega(G_1) & \text{si } b = 1, \end{cases}$$

   es decir, si h es isomorfo a G_b por ω.

   Si la comparación falla, V termina en rechazo. En caso contrario, vuelve al paso 1.

Tras $t(n)$ rondas, V termina y acepta la instancia como $Verdadera$, G_0 y G_1 son isomorfos.

Sea (G_0, G_1) una instancia $Verdadera$. Sea cual sea el reto b, P siempre puede devolver el isomorfismo ω, pues G_0 y G_1 son ambos isomorfos a h. Tenemos que el protocolo es *completo*.

Si (G_0, G_1) es $Falsa$, $G_0 \not\cong G_1$, un P tramposo deberá adivinar el reto b antes de calcular h, pues como éste debe ser un isomorfismo de G_0 ó G_1, no podrá serlo de ambos a la vez. La probabilidad de acertar el reto y mandar el isomorfismo correcto es de $1/2$ en cada ronda, por lo que la probabilidad de que un P tramposo engañe a V es de $2^{-t(n)}$. El protocolo es *robusto*.

Sea (G_0, G_1) una instancia $Verdadera$ del problema GI. La Vista entre P y V es el ensamble

$$Vista_{P,V^*}(G_0, G_1, h) = (G_0, G_1, h, H_1, B_1, \Phi_1, \ldots, H_{t(n)}, B_{t(n)}, \Phi_{t(n)}),$$

donde la variable aleatoria H_i representa el grafo isomorfo h_i de la i-ésima ronda, la v.a. B_i el reto b_i de V a P, y Φ_i es el isomorfismo que envía P como

respuesta al final de la ronda. El historial de anteriores transcripciones se representa con h.

Como en el problema QR, V^* podría utilizar un algoritmo probabilístico F, de tiempo polinomial, al calcular los retos b_i, para intentar obtener información de P. F utilizará toda la información accesible a V^* en el momento de enviar el reto.

El Simulador para la prueba interactiva anterior, que utilizará el mismo algoritmo F, es el siguiente:

Simulador para el problema GI $S_{V^*}(G_0, G_1, h)$.

Datos: (G_0, G_1), una instancia $\mathtt{Verdadera}$ del problema GI; h, transcripciones de ejecuciones previas del protocolo; v_i, transcripción de la interacción actual (i rondas).

Ejecución: Repetir para $i + 1 \leqslant t(n)$:

1. Elegir $b_{i+1} \in_R \{0, 1\}$
2. Elegir $\pi_{i+1} \in_R \mathrm{Sym}(V_{b_{i+1}})$ y calcular $h_{i+1} = \pi_{i+1}(G_{b_{i+1}})$.
3. **Si** $b_{i+1} = F(G_0, G_1, h, v_i, h_{i+1})$, **entonces** añadir la tupla $(h_{i+1}, b_{i+1}, \pi_{i+1})$ a la transcripción.
   Si no, volver al paso 1.
4. $i = i + 1$

La probabilidad de que el b_{i+1} elegido coincida con el de la función F es $1/2$, por lo que en promedio se necesitarán dos rondas por tupla. El Simulador es un algoritmo probabilístico que se ejecuta en un tiempo estimado polinomial.

Veamos ahora que el ensamble de la $\mathrm{Vista}_{P,V^*}(G_0, G_1, h)$, es igual al del Simulador, $S_{V^*}(G_0, G_1, h)$. Procedemos por inducción sobre i, el número de rondas.

Para $i = 0$, ambos ensambles son constantes, $\mathrm{Vista}_{P,V^*}(G_0, G_1, h) = S_{V^*}(G_0, G_1, h) = (G_0, G_1, h)$, por lo que sus distribuciones de probabilidad son idénticas y los ensambles coinciden.

Suponemos cierto para $i - 1$ rondas, $P(\mathrm{Vista}_{P,V^*} = v_{i-1}) = P(S_{V^*} = v_{i-1})$.

Siguiendo el protocolo, se generarán las v.a. (H_i, B_i, Φ_i) para la i-ésima ronda. Utilizando el Teorema de la probabilidad compuesta, y observando la dependencia de las variables durante la ejecución del protocolo, calculamos:

$$P(H_i = h, B_i = b, \Phi_i = \pi) =$$
$$P(\Phi_i = \pi) \cdot P(B_i = b \mid \Phi_i = \pi) \cdot P(H_i = h \mid \Phi_i = \pi, B_i = b)$$

El isomorfismo π se elige aleatoriamente entre todas las posibles permutaciones de V_0, como $| Sym(V_0) | = n!$, $P(\Phi_i = \pi) = \frac{1}{n!}$.

La variable aleatoria B_i se calcula con la función F, $B_i = F(h, V_i, H_i)$, por lo que asignamos la probabilidad $P(B_i = b \mid \Phi_i = \pi) = p_b$ dependiente de la F que use V.

Por último $P(H_i = h \mid \Phi_i = \pi, B_i = b) = 1$ por construcción del grafo h por el isomorfismo π, independiente del valor de B_i.

Nos queda en total que $P(H_i = h, B_i = b, \Phi_i = \pi) = \frac{p_b}{n!}$.

Ahora consideramos la tupla de v.a. (H_i', B_i', Φ_i') del Simulador.

La variable Φ_i' se elige aleatoriamente entre $Sym(V_0)$ o $Sym(V_1)$, ambos de mismo orden pues $| V_0 | = | V_1 | = n$, por lo que $| Sym(V_0) | = | Sym(V_1) | = n!$, luego obtenemos $P(\Phi_i' = \pi) = \frac{1}{n!}$.

Como en la Vista, el Simulador utiliza F para calcular el valor b de B_i', así que $P(B_i' = b \mid \Phi_i' = \pi) = p_b$.

Finalmente, h viene determinado por π, de modo que $P(H_i' = h \mid \Phi_i' = \pi, B_i' = b) = 1$.

La probabilidad del Simulador nos queda $P(H_i' = h, B_i' = b, \Phi_i' = \pi) = \frac{p_b}{n!}$, igual que la del ensamble de la Vista.

Concluimos que se cumple la propiedad de *conocimiento cero* y el problema GI tiene una prueba interactiva de conocimiento cero perfecta.

□

6.2.3 *Logaritmo discreto*

Vamos a expresar el problema del logaritmo discreto como un problema de decisión, donde podamos mantener cierta información secreta que no se revele en la prueba de conocimiento cero, y que permita solucionar el problema:

Nombre:	Problema DL (*Discrete Logarithm*).
Parámetros:	Un grupo cíclico G de orden q primo, donde se supone difícil el problema del logaritmo discreto, un generador g, $G = \langle g \rangle$, y un elemento $y \in G$.
Pregunta:	¿Conoce P el entero $s \in \mathbb{Z}_q$ tal que $g^s = y$, o equivalentemente, $\log_g y = s$?

A la prueba interactiva asociada a este problema se les llama *pruebas de conocimiento*. Veremos una prueba de conocimiento cero de conocimiento del secreto s, en inglés llamadas *Zero-Knowledge Proof of Knowledge*. A pesar de la verbosidad del término, se diferencian poco de las ya vistas, como vamos a ver enseguida.

Teorema 6.13. *El problema DL tiene una prueba de conocimiento cero.*

Demostración.

Primero veremos un protocolo interactivo entre P y V para probar el conocimiento de s, y entonces comprobaremos las tres propiedades necesarias, completitud, robustez y conocimiento cero.

Algoritmo 6.14 (Prueba interactiva para DL)**.**

Datos comunes: Una instancia (G, q, g, y) del Problema DL. $n = o(g) = q$ es el tamaño del problema.

Protocolo: Sea $t(n)$ un polinomio en n. P y V repiten $t(n)$ veces los siguientes pasos.

1. P elige aleatoriamente $u \in_R \mathbb{Z}_q^*$.
2. $P \to V: \quad a = g^u$.
3. $V \to P: \quad b \in_R \{0, 1\}$.
4. $P \to V: \quad w = (u + sb) \bmod q$.
5. V comprueba si:

$$g^w \stackrel{?}{=} \begin{cases} a & \text{si } b = 0 \\ a \cdot y & \text{si } b = 1. \end{cases}$$

   Si la comparación falla, V termina en rechazo. En caso contrario, vuelve al paso 1.

Tras $t(n)$ rondas, V termina y acepta la instancia como $Verdadera$, P conoce el logaritmo discreto de $y \in G = \langle g \rangle$.

Nótese que por el propio enunciado del problema, P no ha necesitado de su potencia ilimitada de cálculo.

El protocolo es *completo*, pues si P conoce s, siempre puede calcular un w que pase la comprobación de V en el paso 5.

Es también *robusto*, pues suponiendo un grupo G donde un P tramposo, una máquina limitada a cálculos en tiempo polinomial, no puede calcular fácilmente el secreto s, este P tramposo deberá intentar adivinar el reto b.

Si P^* supone que el reto $b = 0$, seguirá el protocolo, mandando $w = u$, pero fallaría si V manda $b = 1$, al no poder calcular el s.

Si P^* quiere superar un reto $b = 1$, puede elegir u como en el paso 1, enviar $a = g^u \cdot y - 1$, y contestar al reto con $w = u$. Si se equivoca y V envía $b = 0$, necesitaría s para poder enviar el w correcto y fallaría la prueba.

La probabilidad de acertar el reto b es de $1/2$, de modo que la probabilidad de pasar la prueba interactiva con una instancia $\mathtt{Falsa}$ es de $2^{-t(n)}$. Se cumple la propiedad de *robustez*.

Nos queda comprobar la propiedad de *conocimiento cero*.
Como en casos anteriores, suponemos un algoritmo probabilístico polinomial F que utiliza V^* para enviar los retos. Si V fuera honesto, F es un generador de números aleatorios con una distribución uniforme.

Simulador para el problema DL $S_{V^*}(G, q, g, y, h)$.

Datos: (G, q, g, y), una instancia $\mathtt{Verdadera}$ del problema DL; h, transcripciones de ejecuciones previas del protocolo; v_i, transcripción de la interacción actual (i rondas).
Ejecución: Repetir para $i + 1 \leqslant t(n)$:

1. Elegir $b_{i+1} \in_R \{0, 1\}$
2. Elegir $w_{i+1} \in_R \mathbb{Z}_q^*$.
3. **Si** $b_{i+1} = 0$, **entonces** calcular $a_{i+1} = g^w$
   Si no, $a_{i+1} = g^w y^{-1} = g^{w-s}$
4. **Si** $b_{i+1} = F(G, q, g, y, h, v_i, a_{i+1})$, **entonces** añadir la tupla $(a_{i+1}, b_{i+1}, w_{i+1})$ a la transcripción.
   Si no, volver al paso 1.
5. $i = i + 1$

Por la elección aleatoria de b_{i+1} y w_{i+1} el Simulador es un algoritmo probabilístico, y el tiempo de ejecución estimado es de dos iteraciones por ronda simulada, pues la probabilidad de que el b_{i+1} elegido coincida con el indicado por F es de $1/2$.

Por inducción sobre i, el número de rondas realizadas, veamos que los ensambles $Vista_{P,V^*}$ y S_{V^*} son iguales:

$$\begin{aligned} Vista_{P,V^*} &= (G, q, g, y, h, A_1, B_1, W_1, \ldots, A_i, B_i, W_i) \\ S_{V^*} &= (G, q, g, y, h, A_1', B_1', W_1', \ldots, A_i', B_i', W_i'), \end{aligned}$$

donde las v.a. A_i, A_i' representan el testigo a_i de la i-ésima ronda, las v.a. B_i, B_i' el bit del reto, y W_i, W_i' la respuesta de P.

Para $i = 0$, los ensambles $Vista_{P,V^*} = (G, q, g, y, h)$ y $S_{V^*} = (G, q, g, y, h)$ son constantes e iguales.
Suponemos cierto para $i - 1$.
La tupla de las v.a. para la ronda i en la Vista es: (A_i, B_i, W_i).

Su probabilidad se calcula como:

$$P(A_i = a, B_i = b, W_i = w) =$$
$$P(A_i = a) \cdot P(B_i = b \mid A_i = a) \cdot P(W_i = w \mid A_i = a, B_i = b)$$

Por construcción, a depende de la elección de u, elegido de entre $q-1$ posibles valores, $P(A_i = a) = 1/(q-1)$.

V^* utiliza $F(G, q, g, y, a)$ para la elección de b, así que podemos escribir $P(B_i = b \mid A_i = a) = p_b$, dependiente de la F empleada.

Finalmente, en el protocolo w depende para su cálculo de a y b, así que la probabilidad dependiente de ambos valores dados, es 1.

Concluimos que $P(A_i = a, B_i = b, W_i = w) = \dfrac{p_b}{q-1}$.

Observando el orden en que se calculan los valores en el Simulador, la probabilidad de la i-ésima tupla del Simulador es:

$$P(A_i^{'} = a, B_i^{'} = b, W_i^{'} = w) =$$
$$P(W_i^{'} = w) \cdot P(B_i^{'} = b \mid W_i^{'} = w) \cdot P(A_i^{'} = a \mid W_i^{'} = w, B_i^{'} = b)$$

El valor de w_{i+1} se elige en $\mathbb{Z}_q^*$ independientemente de los demás. Así que $P(W_i^{'} = w) = 1/(q-1)$.

Como antes, la elección del bit depende de F. $P(B_i^{'} = b \mid W_i^{'} = w) = p_b$.

Y siguiendo los pasos del Simulador, el valor de a queda unívocamente determinado dados w y b, así que $P(A_i^{'} = a \mid W_i^{'} = w, B_i^{'} = b) = 1$.

Nos queda $P(A_i^{'} = a, B_i^{'} = b, W_i^{'} = w) = \dfrac{p_b}{q-1}$, igual que la Vista, de modo que los ensambles son idénticos y queda demostrado el teorema.

□

6.3 OTROS TIPOS DE PRUEBAS DE CONOCIMIENTO CERO

La propiedad de *conocimiento cero* exige que exista un algoritmo probabilístico Simulador que pueda ejecutarse en tiempo polinomial, y cuyo ensamble sea idéntico al del protocolo. Esto limita mucho el número de pruebas interactivas de conocimiento cero, y por eso se definen condiciones menos restrictivas que en la práctica, con máquinas reales limitadas computacionalmente, pueden funcionar como las ZKP perfectas, e incluso ser más óptimas, al reducir el número de interacciones manteniendo el nivel de robustez.

6.4 PRUEBAS DE CONOCIMIENTO CERO ESTADÍSTICAS

Se denominan así las pruebas interactivas con un Simulador que, en vez de tener ensamble idéntico a la Vista, convergen en n, el parámetro que define el tamaño del problema:

$$\lim_{n\to\infty} \text{Vista}_{P,V^*}(q, h) = \lim_{n\to\infty} S_{V^*}(q, h)$$

para toda instancia $Verdadera\ q \in Q$, donde n es el tamaño de la instancia q, e historial h.

6.5 PRUEBAS DE CONOCIMIENTO CERO DE VERIFICADOR HONESTO

En esta sección veremos un tipo de pruebas de conocimiento cero, utilizadas en la práctica, que derivan de las ZKP perfectas al añadir una condición al Verificador, que cumpla el protocolo indicado.

En las pruebas de conocimiento cero perfectas que hemos visto, los simuladores estudiados consideraban la existencia de una función F que un Verificador tramposo V^* utilizaría para elegir los retos, en un intento de obtener más información de P. Sin embargo, si el conjunto de posibles retos es demasiado grande, el Simulador diseñado perdería la propiedad de ser de *tiempo polinomial*.

Veremos es este apartado una variación de la ZKP perfecta basada en el problema del Logaritmo Discreto (6.2.3), llamado *Protocolo de Identificación de Schnorr*.

Primero han de definirse parámetros comunes conocidos por P y V. Elegimos dos valores primos, p y q tal que $p \equiv 1 \bmod q$, es decir, $q \mid (p-1)$, y el problema del logaritmo discreto es difícil. Sea α un generador del subgrupo de orden q de $\mathbb{Z}_p^*$. Además debemos elegir un valor t que definirá la robustez del protocolo. En resumen tenemos:

- p primo.
- q primo divisor de $(p-1)$.
- $\alpha \in \mathbb{Z}_p^*$ de orden q.
- t un *parámetro de seguridad*, tal que $2^t < q$. La robustez del protocolo, es decir, la probabilidad de que un P^* tramposo engañe a V, será de 2^{-t}.

El secreto de P será el valor a, tal que $0 \leqslant a \leqslant q-1$. La prueba consistirá en demostrar que P conoce dicho valor. P calcula entonces el valor $v = \alpha^{-a} \bmod p$. Éste valor se puede calcular como la inversa $(\alpha^a)^{-1} \bmod p$, o de manera más eficiente como $\alpha^{q-a} \bmod p$.

Tanto P como V conocerán los valores p, q, α, t y v.

Algoritmo 6.15 (Schnorr)**.**
Datos comunes: p, q, α, t y v.
Protocolo: Realizar una vez:

1. P elige aleatoriamente $k \in_R \mathbb{Z}_q$.
2. $P \to V: \quad \gamma = \alpha^k \bmod p$ (el *testigo*).

3. $V \to P$: r aleatorio, tal que $1 \leqslant r \leqslant 2^t < q$ (el *reto*).
4. $P \to V$: $y = k + ar \bmod q$ (la *respuesta*).
5. V comprueba si $\gamma \stackrel{?}{\equiv} \alpha^y v^r \bmod p$

   Si la comparación falla, V termina en rechazo. En caso contrario, acepta la prueba.

Este algoritmo en particular se diseñó con el objetivo de minimizar la cantidad de mensajes intercambiados manteniendo una robustez que lo hiciera seguro. Como veremos, una sola interacción nos dará una probabilidad de 2^{-t} de que un atacante consiga engañar al Verificador.

Pero primero comprobemos la *completitud*. Veamos con las siguientes congruencias que que si P conoce a, siempre podrá responder correctamente al reto:

$$\alpha^y v^r \equiv \alpha^{k+ar} v^r \equiv \alpha^{k+ar} \alpha^{-ar} \equiv \alpha^k \equiv \gamma \bmod p$$

Por tanto, un P y V honestos finalizarán el protocolo en aceptación siempre, la prueba es *completa*.

El *parámetro de seguridad* t define la dificultad de adivinar el reto de V. Si P^* adivinara el valor r aleatorio de V, podría superar la prueba eligiendo un valor y aleatorio y calculando $\gamma = \alpha^y v^r \bmod p$, con lo que enviando primero el y y respondiendo al reto con γ pasaría la prueba. La probabilidad de adivinar correctamente un reto $1 \leqslant r \leqslant 2^t$ elegido aleatoriamente es de $\frac{1}{2^t}$.

Supongamos que P^* puede adivinar el reto de V con una probabilidad mayor a 2^{-t}, entonces P^* conocería para un valor γ de su elección, al menos dos respuestas posibles para dos retos distintos de V, es decir, si antes el P^* tramposo podía precalcular un testigo y una respuesta para un reto de V, ahora puede calcular, para el mismo testigo (pues solo puede enviar uno antes de recibir el reto), P^* conoce la respuesta de al menos dos retos.

Sea γ el testigo para el que P^* conoce los valores r_1, y_1 y r_2, y_2, un par de posibles retos y respuestas válidas, que cumplen

$$\gamma \equiv \alpha^{y_1} v^{r_1} \equiv \alpha^{y_2} v^{r_2} \bmod p$$

Se sigue que

$$\alpha^{y_1 - y_2} \equiv v^{r_2 - r_1} \bmod p$$

Como la *clave pública* v se calculaba como $v \equiv \alpha^{-a} \bmod p$, con a el secreto de P, podemos sustituir:

$$\alpha^{y_1 - y_2} \equiv \alpha^{-a(r_2 - r_1)} \bmod p$$

Como α tiene orden q, podemos trabajar con los exponentes como:

$$y_1 - y_2 \equiv a(r_1 - r_2) \bmod q$$

Por la condición sobre los retos $1 \leqslant r \leqslant 2^t < q$, sabemos que $0 <| r_2 - r_1 |< 2^t < q$, y como q es primo, $mcd(r_2 - r_1, q) = 1$, de modo que existe el inverso $(r_1 - r_2)^{-1} \bmod q$, y el P^* puede calcular el secreto a como:

$$a = (y_1 - y_2)(r_1 - r_2)^{-1} \bmod q$$

A partir de este análisis, vemos que cualquiera que pueda superar la prueba frente a V con una probabilidad mayor a 2^{-t} puede calcular el secreto de P.

Después de ver la *completitud* y *robustez* del protocolo, veamos la propiedad de *conocimiento cero con Verificador Honesto*.

Una transcripción o vista del protocolo es simplemente el ensamble probabilístico con una tupla:

$$Vista_{P,V} = (\gamma, r, y)$$

Calculamos la probabilidad

$$P(\gamma = g, r = b, y = c) = P(\gamma = g)P(r = b \mid \gamma = a)P(y = c \mid \gamma = a, r = b)$$

P calcula γ a partir de un valor k aleatorio entre 0 y $q - 1$, y elevando α, de orden q a dicho valor, de modo que la probabilidad de que γ tome el valor g es igual a la de elegir el k que genera a: $P(\gamma = g) = \frac{1}{q}$.

Bajo el supuesto de Verificador Honesto, $P(r = b \mid \gamma = g) = P(r = b) = \frac{1}{2^t}$, pues V seguirá el protocolo, eligiendo el reto aleatoriamente, sin usar la función F del apartado anterior.

Finalmente, $y = k + ar$ depende del k que genera $\gamma = g$, del secreto a de P y del reto $r = b$ de V. La probabilidad condicionada $P(y = c \mid \gamma = a, r = b) = 1$.

Tenemos que la probabilidad de obtener la Vista (γ, r, y) es de $\frac{1}{q \cdot 2^t}$.

Ahora definimos el Simulador de la prueba, con la restricción de Verificador Honesto:

Algoritmo 6.16.
Datos: p, q, α, t y v.
Protocolo: Realizar una vez:

1. Elegir r aleatorio tal que $1 \leqslant r \leqslant 2^t$.
2. Elegir y aleatorio tal que $0 \leqslant y \leqslant q - 1$.
3. Calcular $\gamma = \alpha^y v^r \bmod p$.

El ensamble del Simulador se puede escribir como el conjunto:

$$S_V = \{(\gamma', r', y') : 1 \leqslant r' \leqslant 2^t, 0 \leqslant y' \leqslant q - 1, \gamma' \equiv \alpha^y v^r \bmod p\}$$

Como γ' depende totalmente de r' e y', $| S_V | = q \cdot 2^t$, es decir, $P(\gamma' = g, r' = b, y' = c) = \frac{1}{q \cdot 2^t}$, la misma que la de la Vista.

Tenemos, por tanto, que el protocolo de Schnorr es una Prueba de Conocimiento Cero con Verificador Honesto.

A día de hoy, para probar que Schnorr es ZKP perfecta, donde un Verificador utiliza la función F que elige los retos no uniformemente entre 1 y 2^t, no se conoce ningún simulador que sea probabilístico en tiempo polinomial, condición indispensable para probar que es perfecta.

Aún así, Schnorr es utilizado ampliamente en la práctica, pues tampoco se conoce ningún ataque a partir de la elección de retos no aleatorios. Para probar que el protocolo es una ZKP perfecta hay que añadir nuevas restricciones, como reducir el espacio de retos, a cambio de incrementar las rondas para mantener el nivel de robustez. Con un bit por reto y t rondas, en la sección anterior demostramos que el protocolo era una ZKP perfecta, pero en la práctica es poco eficiente por la cantidad de mensajes necesarios.

6.6 PRUEBAS DE CONOCIMIENTO CERO COMPUTACIONALES

Decimos que dos ensambles probabilísticos son*indistinguibles polinomialmente* si no existe un *algoritmo probabilístico en tiempo polinomial* que puede distinguir uno de otro. Se puede consultar más sobre indistinguibilidad polinomial en [7], donde se indica que, en particular, es un subconjunto propio de la *indistinguibilidad estadística*, que nos daba las Pruebas de Conocimiento Cero Estadísticas del apartado anterior.

En esta clase de pruebas interactivas, el probador P ya no se considera como una máquina de ilimitado poder computacional, sino que está restringida como la máquina V, a algoritmos eficientes, a lo sumo probabilísticos en tiempo polinomial.

En esta clase de ZKP se encuentran problemas como el de la 3-coloración del grafo (G3C), o el del camino hamiltoniano (HC).

6.6.1 *Prueba de conocimiento cero para un grafo hamiltoniano*

Nombre:	Problema HC.
Parámetros:	Un grafo $G = (V, E)$.
Pregunta:	¿Existe un ciclo en G que recorre pasa por cada vértice en V una única vez?

Blum en "How to Prove a Theorem So No One Else Can Claim It" [1] muestra una prueba de conocimiento cero para este problema bajo la suposición de que el concepto de *caja negra* existe, de modo que P puede ocultar el contenido de la caja hasta que da la llave para abrirla. En la sección 6.7

estudiaremos más en profundidad esta herramienta común a la gran parte de pruebas de conocimiento cero computacionales.

Algoritmo 6.17 (Prueba interactiva para HC)**.**

Datos comunes: Una instancia $G = (V, E)$ del Problema HC. $|\ V\ | = n$ es el tamaño del problema.

P conoce un ciclo hamiltoniano de G.

Protocolo: Sea $t(n)$ un polinomio en n. P y V repiten $t(n)$ veces los siguientes pasos.

1. P oculta G en cajas negras: asocia, de manera uniformemente aleatoria, los vértices $v_1, v_2, \dots v_n$ a las cajas $C_1, C_2, \dots C_n$; además, para cada par de cajas (C_i, C_j) prepara otra caja llamada C_{ij} que guardará un 1 si los vértices ocultos en C_i y C_j son adyacentes, o 0 en caso contrario.
2. $P \to V:$ $\quad n + \binom{n}{2}$ cajas negras: $C_1, C_2, \dots C_n, C_{1,2}, \dots, C_{n-1,n}$
3. $V \to P:$ $\quad b \in_R \{0, 1\}$.
4. Si $b = 0$, P envía las llaves para abrir todas las cajas.

   Si $b = 1$, P abre exactamente n cajas, $C_{ij}, C_{jk}, C_{kl}, \dots C_{l'i}$, que corresponden al ciclo hamiltoniano escondido por los vértices escondidos en las cajas $C_1, C_2, \dots C_n$.
5. Si $b = 0$, V comprueba que el grafo descubierto por las cajas sea G.

   Si $b = 1$, V comprueba que los índices de las cajas $C_{ij}, C_{jk}, \dots C_{l'i}$ forman un ciclo y que todas contienen un 1.

   Si la comprobación falla, V termina en rechazo. En caso contrario, vuelve al paso 1.

Tras $t(n)$ rondas, V termina y acepta la instancia como *Verdadera*, P conoce un camino hamiltoniano en G.

El protocolo es *completo*, pues si P conoce un ciclo hamiltoniano de G, siempre podrá descubrir G o el ciclo cuando V lo pida.

Si un P^* no conoce el ciclo hamiltoniano, puede intentar adivinar cuándo V pedirá descubrir el grafo o el ciclo. En el primer caso, seguirá el protocolo, ocultando G en las cajas negras. En el segundo, P^* sólo debe elegir un ciclo de cajas $C_{ij}, C_{jk}, \dots C_{l'i}$ donde introducirá el valor 1 haya o no una arista uniendo los vértices. Si V pide descubrir el ciclo, P^* revelará un ciclo hamiltoniano, pero no de G.

Si P^* no acierta correctamente qué bit enviará V, en cada ronda hay una probabilidad de $1/2$ de que V pille a P^* en una mentira. Ejecutando el protocolo $t(n)$ veces, la probabilidad de que un P^* tramposo engañe a V es de $2^{-t(n)}$, por tanto, la prueba es *robusta*.

TODO

6.6.2 *Prueba de conocimiento cero para la 3-coloración de un grafo*

Nombre:	Problema G3C.
Parámetros:	Un grafo $G = (V, E)$.
Pregunta:	¿Existe una función $\phi : V \to \{1, 2, 3\}$ tal que $\forall (u, v) \in E$, se cumple $\phi(u) \neq \phi(v)$?

TODO: cambiar lo de abajo por el protocolo y análisis.

Goldreich, micali y Wigdergon demostraron en [TODO] que el problema G3C tiene una prueba de conocimiento cero computacional. Para la prueba interactiva y su demostración se debería introducir previamente la teoría cifrado probabilístico.

La idea de la prueba es que en cada iteración, P elige una permutación aleatoria de la 3-coloración, $\pi \in Sym(\{1, 2, 3\})$, y una clave de cifrado para cada vértice. El *testigo* que enviará a V consistirá en el cifrado probabilístico de la permutación sobre la coloración de cada vértice, ocultando así la 3-coloración. V elegirá como *reto* una arista aleatoria. Como *respuesta*, P enviará la coloración permutada de esa arista, que deberá tener colores distintos en cada extremo, y las claves de cifrado de cada una, de modo que V podrá comprobar si corresponden al *testigo* enviado.

La completitud es inmediata, si P conoce la coloración, siempre podrá dar para (u, v) colores distintos. Si P fuera tramposo, al menos hay una arista a la que P no puede dar colores distintos a sus vértices, y hay una probabilidad de $\frac{1}{|E|}$ de que V le rechace. Con $|E|^2$ iteraciones, P puede conseguir engañar a v con una probabilidad $(1 - \frac{1}{m})^{m^2} \leqslant e^{-m}$, por lo que la prueba es robusta.

La propiedad de conocimiento cero se prueba con un simulador que funciona como los anteriores, primero se elige el *reto* (arista) y la *respuesta* (dos colores), se calcula el *testigo* respecto de ambos y se comprueba si V hubiera elegido dicho *reto*.

Se sabe que G3C pertenece a **NPC** (2.17), por lo que se deduce que todo problema en **NPC** tiene una ZKP computacional.

6.7 ESQUEMAS DE COMPROMISO

En las pruebas de conocimiento cero vistas hasta ahora, el primer mensaje del Probador consistía en un *compromiso* con el Verificador, de modo que en la respuesta al reto no pudiera engañarle, con cierta probabilidad.

Nos interesan en particular sistemas como el del problema de la 3-coloración, o del ciclo hamiltoniano, donde la solución se escondía tras N criptogramas, que podemos ver como cajas negras, que sólo se pueden abrir si se tiene la llave. El Probador, al enviar todas las cajas negras, se compromete, no puede cambiar los valores escondidos, y el Verificador en su reto pide a P que abra ciertas cajas al azar para comprobar que dice la verdad, sin revelar el secreto.

Los *esquemas de compromiso* son una herramienta necesaria en las pruebas de conocimiento cero computacionales de todos los problemas **NPC**, como se indica al final de [1]. Estos esquemas esconden la estructura de una instancia $\mathtt{Verdadera}$, donde P se compromete con su *testigo* antes de conocer el *reto* de V. En la última *respuesta* de P, se revelará una pequeña parte de la estructura de la instancia $\mathtt{Verdadera}$, de la cual V no podrá aprender nada.

Definición 6.18. Un esquema de compromiso de un *bit* es un par de funciones (f, v) de tiempo polinomial. La función

$f : \{0,1\} \times \Upsilon \to \chi$

transforma el bit $b \in \{0,1\}$ con una clave aleatoria $y \in \Upsilon$. El valor $x = f(b,y)$ se llama *blob*. La función de verificación

$v : \chi \times \Upsilon \to \{0,1,\bullet\}$

abre el blob revelando el bit b, o indicando que no es un par (blob,clave) válido.

El par (f, v) debe cumplir la siguientes condiciones:

1. *Vinculación*: Para todo blob $x = f(b,y)$, P no es capaz de encontrar un valor $y' \neq y$ tal que el blob se puede abrir con otro valor, es decir, $v(x,y) \neq v(x,y')$.

2. *Secreto*: Los ensambles $\{f(0,\Upsilon)\}$ y $\{f(1,\Upsilon)\}$ son indistinguibles.

Con la primera propiedad, una vez P se compromete con un bit, al presentar el blob $x = f(b,y)$ a V, no puede cambiarlo, e influye en la robustez del protocolo. La segunda condición, asegura que no se filtra información de los bits que P no proporciona la clave, consiguiendo conocimiento cero.

Podemos clasificar los esquemas de compromiso de bit en dos tipos: vinculación incondicional y secreto incondicional.

ESQUEMAS DE COMPROMISO CON VINCULACIÓN INCONDICIONAL

Partiendo del problema del residuo cuadrático, para $N = pq$, los conjuntos $\mathbb{Z}_N^{Q+}$ y $\mathbb{Z}_N^{Q-}$ son polinomialmente indistinguibles. Podemos utilizar esta propiedad para diseñar un sistema de compromiso de bit.

P elige 2 primos aleatorios suficientemente grandes, p y q, y un no-residuo cuadrático $s \in \mathbb{Z}_N^{Q-}$.

ESQUEMAS DE COMPROMISO CON SECRETO INCONDICIONAL

7

APLICACIONES DE ZKP

? Fiat-Shamir para QR, Schnorr para logaritmo discreto, ... Aplicación de ZKP en los certificados de Idemix. Analizar cómo realizan pruebas de AND, OR, etc.

7.1 FIRMA DIGITAL BASADA EN PRUEBAS DE CONOCIMIENTO CERO

Hemos visto que la interacción habitual en una prueba de conocimiento cero es enviar un *testigo*, un *reto* y una *respuesta* en cada iteración.

P y V	conocen información previa Υ
$P \to V$:	*testigo* u
$V \to P$:	*reto* b
$P \to V$:	*respuesta* $w = \xi(u, b, \Upsilon)$
V verifica	$u = \vartheta(b, w, \Upsilon)$.

La desventaja de estos protocolos es sincronizar a P y V para comunicarse, por la necesidad de que el verificador debe generar el reto para estar seguro de la prueba (por la propiedad de conocimiento cero).

Una técnica para convertir las pruebas en no interactivas es la *heurística de Fiat-Shamir*, que consiste en sustituir el reto por un resumen digital h, *hash* en inglés, del testigo u, de modo que para un P computacionalmente limitado, V puede confiar en que P no ha elegido u tal que con el reto calculado pueda falsear la prueba.

Aprovechando la función del *hash*, se puede convertir el protocolo en un esquema de firma digital de un mensaje m, realizando el *hash* sobre el testigo y el mensaje a la vez (concatenándolos, por ejemplo).

P calcula :	*testigo* u, *respuesta* $w = \xi(u, h, \Upsilon)$, donde $h =$**hash(u | m)**
$P \to V$:	***firma del mensaje*** m: (h, w)
V verifica	$h = hash(\vartheta(h, w, \Upsilon)\|m)$.

Podemos construir así una firma digital, con un solo mensaje, por lo que es válido para cualquier V, y donde P no ha debido compartir ninguna clave o información extra aparte de los parámetros del sistema, Υ.

Observación.

En las pruebas de conocimiento cero perfectas estudiadas, los retos consistían en un bit, por lo que los posibles valores son 0 ó 1 y un ataque a la función de *hash* es muy fácil. Por ello, se utilizan versiones de ZKP donde el conjunto de los posibles valores del reto es suficientemente grande, por

ejemplo, un entero representado en tantos bits como produce la firma digital, usualmente 256 o 512 bits.

Al hacer este cambio, muchas veces perdemos la condición de perfecta en la prueba de conocimiento cero, rebajándola a otro de los tipos vistos, pero que en la práctica es perfectamente aceptable.

7.2 PROTOCOLOS DE IDENTIFICACIÓN BASADOS EN ZKP

Las pruebas de conocimiento cero tienen una gran aplicación en el campo de la seguridad informática, en particular en la **autenticación**. Tras la autenticación se aplicará autorización y control de acceso, por eso es importante un sistema de identificación fiable. Además, de entre las ventajas de las pruebas de conocimiento cero, un sistema de identificación basado en ZKP hereda privacidad, al no revelar información del usuario, y seguridad al no *degradarse* con el uso, es decir, resiste al criptoanálisis por muchos mensajes que se intercepten, y ataques con mensajes elegidos.

TODO: redactar mejor las ventajas.

Estos protocolos de identificación se basan en una prueba de conocimiento cero de un problema Q, donde P (el usuario) tiene un *secreto* que le permite demostrar una instancia $\mathtt{Verdadera}$ de Q al verificador V, y que además conocer dicho secreto le relaciona con una identidad, con la ventaja de no tener que revelar el secreto.

7.2.1 *Protocolo de identificación de Fiat-Shamir*

El protocolo de identificación más característico basado en ZKP y el problema QR es el de Fiat-Shamir.

Como hemos visto, el problema QR es **NP**, de modo que obtener una raíz cuadrada módulo un N compuesto, es computacionalmente inviable, equivalente a factorizar N. Bajo esta suposición, podemos utilizar como información pública un residuo cuadrático módulo N, que llamaremos v, y asociarlo a una identidad, de modo que el usuario que conozca una de sus raíces cuadradas, el secreto s, podrá demostrar que v es un residuo cuadrático por medio de una prueba de conocimiento cero.

Algoritmo 7.1 (Protocolo de identificación Fiat-Shamir)**.**
Configuración de la identidad:

1. La entidad de confianza selecciona y publica $N = pq$, con p y q primos y secretos.
2. Cada usuario P genera un secreto $s \in \mathbb{Z}_N^*$, coprimo con N (si no, se podría obtener la factorización de N y perder la seguridad del protocolo). Calcula $v \equiv s^2 \bmod N$ y lo envía a la entidad de confianza como su clave pública.

Protocolo: Repetir t rondas:

1. P escoge aleatoriamente $r \in_R \mathbb{Z}_N^*$, el *compromiso*.
2. $P \to V$: $u \equiv r^2 \bmod N$, el *testigo*.
3. $V \to P$: $b \in_R \{0, 1\}$, el *reto*.
4. $P \to V$: $w \equiv r \cdot s^b \bmod N$, la *respuesta*.
5. V verifica si $w^2 \equiv u \cdot v^b \bmod N$.

El protocolo de Fiat-Shamir es casi idéntico a la prueba interactiva 6.4, donde aquí (v, N) hace el papel de instancia `Verdadera` del problema QR, y como P, el usuario, es una máquina computacionalmente limitada, en vez de elegir aleatoriamente el residuo cuadrático u y la raíz cuadrada w, parte de las raíces cuadradas, s y r, para poder calcular u, v y w elevando al cuadrado. Como los valores que se envían siguen siendo u, un residuo cuadrático aleatorio, b un bit de reto, y w una raíz cuadrada aleatoria de u o uv, según b, las transcripciones del protocolo de Fiat-Shamir son las mismas que las de la prueba interactiva, que sabemos que es de conocimiento cero perfecta.

El intento de ataque que vimos en la demostración de robustez varía ligeramente para Fiat-Shamir, pues el objetivo es intentar demostrar que v es residuo cuadrático, sin conocer s u otra raíz cuadrada, necesaria para una máquina de cómputo limitado del mundo real.

Un atacante debe adivinar el bit del reto que recibirá, de modo que si cree que $b = 0$ calcula el testigo como en el protocolo, pero si cree que $b = 1$, calcula en el paso 2 el testigo $u \equiv r^2 \cdot v^{-1} \bmod N$, y en 4 la respuesta $w \equiv r \bmod N$. En ambos casos fallaría si no adivina b correctamente, pues para corregir su error debería ser capaz de calcular, respectivamente, una raíz cuadrada de v, que permitiría pasar la prueba siempre, o una raíz cuadrada de $u \equiv r^2 \cdot v^{-1} \bmod N$, computacionalmente inviable pues p y q los guarda como secretos la entidad de verificación.

Como en la prueba interactiva, un atacante tiene una probabilidad de $1/2$ de engañar a V en cada ronda, de modo que la robustez se mantiene con una probabilidad de ataque de 2^{-t}. Si P pasa correctamente las t rondas, V da por válida la prueba. Si falla aunque sea sólo una, rechaza la identificación.

Un detalle importante a tener en cuenta es que, considerando ordenadores reales para el caso práctico, si P no utiliza un buen generador de números aleatorios para sus r del paso 1, un atacante podría adivinar cuándo repetirá el r, mandar $b = 0$ y $b = 1$, y así calcular el secreto s.

7.2.2 *Protocolo de identificación de Feige-Fiat-Shamir*

Una variación del protocolo de Fiat-Shamir para disminuir el número de mensajes intercambiados combinando varios testigos y retos a la vez.

Algoritmo 7.2 (Protocolo de identificación Feige-Fiat-Shamir)**.**
Configuración de la identidad:

1. *Parámetros del sistema*: La entidad de confianza publica el módulo $N = pq$, con p y $q \equiv \mod 4$, primos guardados secretos, de modo que -1 es un no-residuo cuadrático con símbolo de Jacobi 1. También define los enteros k y t que definen la seguridad.

2. *Selección del secreto*: Cada usuario P hace:

   (a) Selecciona aleatoriamente un vector $(s_1, s_2, \ldots, s_k)$, donde cada $s_i \in \mathbb{Z}_N^*$ y $\mathrm{mcd}(s_i, N) = 1$. Además, elige también aleatoriamente k bits $(b_1, b_2, \ldots, b_k)$.

   (b) Calcula $v_i \equiv (-1)^{b_i} \cdot (s_i^2)^{-1} \bmod N$ para cada $i = 1, \ldots k$.

   (c) El usuario se identifica ante la entidad de confianza y envía su *clave pública* $(v_1, \ldots, v_k; N)$, y guarda su *clave privada* $(s_1, \ldots, s_k)$.

Protocolo: Repetir t rondas:

1. P escoge aleatoriamente $r \in_R \mathbb{Z}_N^*$ y un bit $b \in_R \{0, 1\}$.
2. $P \to V$: $u \equiv (-1)^b \cdot r^2 \bmod N$, el *testigo*.
3. $V \to P$: $(b_1, \ldots, b_k)$, con cada $b_i \in_R \{0, 1\}$, el *reto*.
4. $P \to V$: $w \equiv r \cdot \prod_{j=1}^{k} s_j^{b_j} \bmod N$, la *respuesta*.
5. V verifica si $u \equiv \pm w^2 \cdot \prod_{j=1}^{k} v_j^{b_i} \bmod N$.

La probabilidad de que un atacante pueda engañar a V es la de acertar el reto de k bits en cada una de las t rondas, es decir, 2^{-kt}. Como vemos, aumentando el número de retos por ronda, podemos reducir el número de mensajes intercambiados para conseguir una robustez equivalente a Fiat-Shamir.

7.2.3 *Protocolo de identificación de Schnorr*

Este protocolo se basa en el problema del logaritmo discreto

Algoritmo 7.3 (Protocolo de identificación Schnorr)**.**
Configuración de la identidad:

1. *Parámetros del sistema*: La entidad de confianza publica los parámetros (p, q, β) donde p y q son primos tal que $q \mid (p-1)$, y β es un elemento con orden multiplicativo q (por ejemplo, para α un generador $\bmod p$, $\beta = \alpha^{(p-1)/q} \mod p$). Además se escoge un t, $2^t < q$, que definirá el nivel de seguridad.

2. *Selección del secreto*: Cada usuario P:

   (a) Recibe una identidad única I_P.

   (b) Escoge una clave privada a, $0 \leqslant a \leqslant q-1$, y calcula $v = \beta^{-a} \bmod p$.

   (c) La entidad certificadora vincula la identidad I_P y el valor v firmando, con cualquier método de firma $S()$, el certificado $cert_A = (I_P, v, S_T(I_P|v))$.

Protocolo:

1. P escoge aleatoriamente r, $0 \leqslant r \leqslant q-1$, y un calcula $x = \beta^r \bmod p$.
2. $P \to V$: x, el *testigo*, y $cert_P$.
3. $V \to P$: e aleatorio, $1 \leqslant e \leqslant 2^t < q$, el *reto*, y verifica la firma del certificado.
4. $P \to V$: $y = ae + r \bmod q$, la *respuesta*.
5. V verifica si $x = \beta^y v^e \bmod p$.

Un P malicioso tiene una probabilidad de $2^{-}t$ de adivinar el reto, pero esta prueba no es una prueba de conocimiento cero perfecta, pues como vimos en el capítulo anterior, el simulador en tiempo polinomial existe si los retos e tienen un dominio pequeño. Como se menciona en [11], esta prueba es de conocimiento cero con verificador honesto (el que sigue el protocolo eligiendo el reto aleatoriamente), pero eligiendo un reto en un espacio de valores menor logarítmico respecto el parámetro t, y realizando múltiples iteraciones, se puede conseguir la condición de *perfecta*, a cambio de perder eficiencia. También se indica que de [5] se pueden tomar construcciones para transformarlo en una prueba de conocimiento cero manteniendo eficiencia.

7.2.3.1 *Probar el conocimiento de una representación*

Una generalización de Schnorr es utilizar l bases $g_1, \ldots, g_l$ con $g_i \in G =< g >$, un grupo de orden $o(g) = q$ primo. Sea $y \in G$ tal que conocemos los valores $x_1, \ldots, x_l$ que cumplen $y = \prod_{i=1}^{k} g_i^{x_i}$. Una prueba de conocimiento cero de que conocemos la representación de y con respecto a las bases $g_1, \ldots, g_l$ es:

Algoritmo 7.4. P y V conocen y, $g_1, \ldots, g_l$, q y el parámetro del sistema k.

Repetir t rondas:

1. P escoge aleatoriamente l enteros $r_i \in_R \mathbb{Z}$.
2. $P \to V$: $t = \prod_{i=1}^{l} g_i^{r_i}$.

3. $V \rightarrow P$: $c \in_R \{0,1\}^k$, un entero aleatorio de k bits.
4. $P \rightarrow V$: $s_i = r_i - c \cdot x_i \bmod q$, para $i = 1, \ldots, l$.
5. V verifica si $t = y^c \prod_{i=1}^{l} g_i^{s_i}$.

La probabilidad de que un P malicioso adivine los k bits aleatorios del reto c es de 2^{-k}. Este tipo de prueba se utiliza en Idemix, del cual hablaremos en la siguiente sección.

7.3 PRUEBAS DE CONOCIMIENTO CERO EN IDENTITY MIXER

Identity Mixer, o Idemix, es un protocolo desarrollado por IBM[1] para crear certificados digitales basados en atributos, donde se preserva la privacidad. Este protocolo no solo permite la autenticación sin revelar un valor secreto, también permite realizar pruebas sobre los atributos del certificado de un usuario, sin revelar los valores, como por ejemplo, "`año_actual - año_nacimiento` $\geqslant$ `18`".

Los protocolos criptográficos en los que se especifican en Idemix [11] han sido desarrollados durante años por expertos en el área, donde destacan Jan Camenisch y Anna Lysyanskaya con el desarrollo de la firma CL [3] [2] que se utiliza para firmar certificados sin tener que conocer los valores de todos los atributos, y que permite posteriormente al usuario demostrar que posee dicha firma, sin desvelarla. Esto permite poder utilizar múltiples veces un certificado de manera anónima sin que se relacionen los diferentes usos con la misma persona.

7.3.1 *Notación para ZKP*

Como hemos visto, cualquier problema de decisión posee una prueba de conocimiento cero, y de entre esas pruebas, las pruebas de conocimiento de un secreto son las que se utilizan en los certificados. Utilizaremos la siguiente notación de Camenisch y Stadler [4] para denotar una prueba de conocimiento cero de conocimiento de un secreto.

$$\mathsf{ZKPoK}\{(w) : \mathcal{L}(w, x)\}$$

donde w es un testigo, normalmente el secreto, x es información conocida por ambas partes, y $\mathcal{L}(w, x)$ es un predicado que representa una condición sobre w y x. Se puede leer como "conozco un testigo w tal que el predicado $\mathcal{L}(w, x)$ se cumple para w y x". Para abreviar, en vez de ZKPoK (*Zero-Knowledge Proof of Knowledge*), usaremos ZKP o PK.

Por ejemplo,

$$\mathsf{PK}\{(\alpha) : y = g^{\alpha}\}$$

donde $y \in G =< g >$, un grupo de orden q primo, denota el protocolo de identificación de Schnorr.

1 Identity Mixer - https://www.research.ibm.com/labs/zurich/idemix/

7.3.2 *Combinar diferentes pruebas de conocimiento*

Cuando leamos pruebas de conocimiento con varias condiciones, como

$$\mathrm{PK}\{(\alpha_1,\ldots,\alpha_l,\beta_1,\ldots,\beta_{l'}) : y = \prod_{i=1}^{l} g_i^{\alpha_i} \wedge z = \prod_{i=1}^{l'} h_i^{\beta_i}\}$$

con $g_i, h_i, y, z \in G$, será equivalente a realizar de manera paralela e independiente cada protocolo:

$$\mathrm{PK}\{(\alpha_1,\ldots,\alpha_l) : y = \prod_{i=1}^{l} g_i^{\alpha_i}\}$$

y

$$\mathrm{PK}\{(\beta_1,\ldots,\beta_{l'}) : z = \prod_{i=1}^{l'} h_i^{\beta_i}\}$$

En el primer mensaje, P envía los testigos de cada uno de los protocolos. Entonces, V envía un único reto, el mismo para cada protocolo. Finalmente, P responde al reto con la respuesta de cada prueba, y V las verifica.

7.3.3 *Firma Camenisch-Lysyanskaya*

El esquema de firma de Camenisch-Lysyanskaya, o firma CL, es la base de los certificados Idemix.

Sean p' y q' dos primos, tal que $p = 2p' + 1$ y $q = 2q' + 1$ también lo son, llamados *primos seguros*. Los parámetros del sistema, conocidos por el firmante y quien comprueba la firma son: el entero $n = pq$, $Z, S, R \in QR_n$ (grupo multiplicativo de los residuos cuadráticos módulo n). La clave secreta del firmante es (p, q) y el mensaje lo denominaremos m. La firma CL sobre m es la tupla (A, e, v) tal que

$$A \equiv \left(\frac{Z}{S^v R^m}\right)^{1/e} \bmod n$$

donde e, v son aleatorios, e primo y $\frac{1}{e} \cdot e \equiv 1 \bmod \varphi(n)$.

Para comprobar que una firma (A, e, v) es correcta, basta comprobar si

$$Z \stackrel{?}{\equiv} A^e S^v R^m \bmod n.$$

La ventaja de esta firma es que se puede aplicar sobre múltiples mensajes a la vez, que en el caso de un certificado, serán los diferentes atributos a firmar. En este caso los parámetros del sistema se mantienen, $n = pq$, $Z, S \in QR_n$, pero debemos calcular un residuo cuadrático por cada mensaje a firmar, $R_0, \ldots, R_l \in QR_n$. La clave secreta sigue siendo (p, q), y la firma de los mensajes $m_0, m_1, \ldots, m_l$ es (A, e, v) tal que

$$A \equiv \left(\frac{Z}{S^v \prod_{i=0}^{l} R_i^{m_i}} \right)^{1/e} \bmod n$$

Para verificar la firma, comprobamos que

$$Z \stackrel{?}{\equiv} A^e S^v \prod_{i=0}^{l} R_i^{m_i} \bmod n.$$

La firma CL se basa en la *suposición RSA fuerte*, donde tanto el problema RSA y el del logaritmo discreto son difíciles de resolver a la vez, es decir, dado un módulo RSA n, y un valor $u \in \mathbb{Z}_n^*$, es difícil calcular una pareja de valores (e, v), tal que $v^e \equiv u \bmod n$.

Como el firmante conoce los factores primos p y q de n, puede calcular, utilizando el Teorema Chino de los Restos, la firma fácilmente, pero quien no la conoce, se enfrenta al problema anterior para deshacer los cálculos.

7.3.4 *Firma Camenisch-Lysyanskaya aleatorizada*

Dada una firma CL (A, e, v), sobre los mensajes $m_0, m_1, \ldots, m_l$, podemos obtener una firma $(\acute{A}, e, \hat{v})$ distinta de la anterior, excepto por e, y todavía válida. Esta nueva firma la puede generar cualquier usuario, pues no precisa del secreto (p, q) de la firma original.

La nueva firma se calcula a partir de un entero aleatorio r, tal que $\acute{A} = A \cdot S^{-r} \bmod n$ y $\hat{v} = v + er$.

La verificación de $(\acute{A}, e, \hat{v})$ se realiza igual que en la original, ya que:

$$\acute{A}^e S^{\hat{v}} \prod_{i=0}^{l} R_i^{m_i} \equiv A^e S^{-er} S^v S^{er} \prod_{i=0}^{l} R_i^{m_i} \equiv A^e S^v \prod_{i=0}^{l} R_i^{m_i} \equiv Z \bmod n$$

7.3.5 *Firma de credenciales Idemix*

En la expedición del certificado intervienen el Usuario y el Emisor (por ej., una compañía o gobierno). Durante el protocolo de emisión, el Usuario y el Emisor calculan interactivamente una firma CL, es decir, entre los dos generarán la firma CL, porque ninguno tendrá toda la información necesaria para generar la firma por sí mismo. En cada paso, una prueba de conocimiento cero asegurará que cada parte ha seguido el protocolo.

El Emisor conocerá la factorización de n, mientras que el Usuario guardará su clave secreta, m_0 y, opcionalmente, los atributos del certificado, $m_1, \ldots, m_l$. Al final de la firma, el Usuario conocerá una representación, $(m_1, \ldots, m_l)$ de $\frac{Z}{A^e S^v}$ con respecto a la base $(R_0, \ldots, R_l)$, y como ya hemos visto, podrá demostrar con una prueba de conocimiento cero que conoce dicha representación sin revelar ninguno, o solo un subconjunto. Además, el Emisor no conocerá la firma (A, e, v) final, pues v se calculará también de manera conjunta de modo que sólo el Usuario conocerá el valor final.

A continuación presentamos una versión simplificada del protocolo, donde los atributos son conocidos por el Emisor. En otras versiones el Usuario puede enviar pruebas sobre alguna propiedad del atributo, o pseudónimos,

sin revelar el verdadero valor al Emisor. Es importante notar que el número de residuos cuadráticos generados por el emisor, $R_0, \ldots, R_{l'}$ determina el máximo de atributos que puede llevar un certificado.

Algoritmo 7.5 (Emisión de certificado Idemix)**.**
Conocimiento común: $n, Z, S, R_0, \ldots, R_{l'}, m_1, \ldots, m_l$, donde $l' \geqslant l$
Conocimiento del Emisor: p, q tal que $n = pq$.
Conocimiento del Usuario: m_0, su clave secreta.

Protocolo:

1. Ronda 1: Emisor
   1.1. El Emisor escoge un valor aleatorio n_1 y lo envía al Usuario.
2. Ronda 2: Usuario
   2.1. El Usuario elige un valor v' aleatorio (primer paso para calcular el v de la firma).
   2.2. Calcula el valor $U = S^{v'} \cdot R_0^{m_0} \bmod n$.
   2.3. Calcula* la prueba $P_1 := ZKP\{(v', \mu_0) : U \equiv S^{v'} R_0^{\mu_0} \bmod n\}(n_1)$, no interactiva, dependiente de n_1.
   2.4. Envía U, P_1, n_2, donde n_2 es un valor aleatorio elegido por el Usuario.
3. Ronda 3: Emisor firma CL
   3.1. Verifica* la prueba P_1 del Usuario.
   3.2. Elige un valor un e primo y v'' aleatorios (segundo paso para calcular el v de la firma).
   3.3. Calcula el valor $A = \left(\dfrac{Z}{U \cdot S^{v''} \cdot \prod_{i=1}^{l} R_i^{m_i}} \right)^{1/e} \bmod n$.
   3.4. Calcula* $P_2 := ZKP\{(\delta) : A \equiv \left(\dfrac{Z}{U \cdot S^{v''} \cdot \prod_{i=1}^{l} R_i^{m_i}} \right)^{\delta} \bmod n\}(n_2)$, no interactiva, dependiente de n_2.
   3.5. Envía (A, e, v'') y P_2 al Usuario.
4. Ronda 4: Usuario
   4.1. Calcula $v = v' + v''$.
   4.2. Verifica P_2 y la firma CL (A, e, v).

---

*A continuación mostramos las pruebas y verificaciones correspondientes.

Issuer Secret: p, q	Sys. pars. $(m_1, \ldots, m_l)$	**User** Secret: m_0
		Random v' $U := S^{v'} R_0^{m_0} \pmod n$ $PK\{(\nu', \mu_0) : U \equiv S^{\nu'} R_0^{\mu_0} \pmod n\}$
	$\xleftarrow{U, PK}$	
Random v'' and prime e $A := \left(\frac{Z}{US^{v''}\prod_1^l R_i^{m_i}}\right)^{1/e} \pmod n$ $PK\{(\delta) : A \equiv \left(\frac{Z}{US^{v''}\prod_1^l R_i^{m_i}}\right)^{\delta} \pmod n\}$	$\xrightarrow{(A,e,v''),PK}$	$v := v' + v''$ in signature (A, e, v) $Z \stackrel{?}{\equiv} A^e S^v \prod_0^l R_i^{m_i} \pmod n$

Figura 2: Idemix: firma de credenciales simplificada.

La prueba de conocimiento P_1 indica que conocemos un ν', primera parte del ν final de la firma CL, y un m_0, nuestra clave secreta, tales que U se representa como (ν', m_0) en la *base* (S, R_0). Utilizando la heurística de Fiat-Shamir, en vez de pedir varios retos al Emisor para conseguir robustez, utilizamos una función de hash junto a un reto n_1 del emisor, llamado en inglés *nonce*. La prueba que se envía consiste en el reto obtenido con el hash, y las respuestas a dicho reto.

Algoritmo 7.6 ($KP\{(\nu', m_0) : U \equiv S^{\nu'} R_0^{m_0} \bmod n\}(n_1)$)**.**

1. Elegir $\tilde{m}_0$ y $\tilde{v}'$ aleatorios.
2. Calcular $\tilde{U} = S^{\tilde{v}'} \cdot R_0^{\tilde{m}_0} \bmod n$.
3. Reto por heurística Fiat-Shamir: $c = H(U||\tilde{U}||n_1)$.
4. Respuestas al *reto*: $\hat{v}' = \tilde{v}' + c\nu'$, $\hat{m}_0 = \tilde{m}_0 + cm_0$.
5. $P_1 := (c, \hat{v}', \hat{m}_0)$.

Para verificar la prueba anterior, por la heurística de Fiat-Shamir (7.1), debemos *recuperar* el valor de U calculado por el Usuario, y comparar los hashes generados.

Algoritmo 7.7 (Verificar $P_1 := (c, \hat{v}', \hat{m}_0)(n_1)$)**.**

1. Calcular $\hat{U} = U^{-c} \cdot S^{\hat{v}'} R_0^{\hat{m}_0} \bmod n$.
2. Aceptar si $c = H(U||\hat{U}||n_1)$.

En la prueba de conocimiento del Verificador, demostramos que conocemos la inversa del e de la firma CL, pero sin desvelar su valor, que solo el Emisor, conocedor de p y q, puede calcular.

Algoritmo 7.8 ($\mathrm{KP}\{(e^{-1}) : A \equiv \left(\frac{Z}{u \cdot S^{v''} \cdot \prod_{i=1}^{l} R_i^{m_i}}\right)^{e^{-1}} \bmod n\}(n_2)$).

Llamemos $Q := \frac{Z}{u \cdot S^{v''} \cdot \prod_{i=1}^{l} R_i^{m_i}}$.

1. Elegir r aleatorio.
2. Calcular $\tilde{A} = Q^r \bmod n$.
3. Calcular el reto por Fiat-Shamir $c' = H(Q||A||n_2||\tilde{A})$.
4. Calcular la respuesta $s_e = r - c'e^{-1}$.
5. $P_2 := (s_e, c')$.

Como antes, *recuperamos* el valor calculado $\tilde{A}$ y comparamos los hashes.

Algoritmo 7.9 (Verificar $P_2 := (s_e, c')(n_2)$).

1. Calcular $\hat{A} = A^{c'+s_e \cdot e} \equiv A^{c'} Q^{s_e} \bmod n$.
2. Aceptar si $c' = H(Q||A||n_2||\hat{A})$.

7.3.6 *Revelación selectiva de atributos Idemix*

Partiendo de un certificado con la clave secreta m_0, los atributos $m_1, \ldots, m_l$ y la firma CL (A, e, v), vamos a revelar a un Verificador todos nuestros atributos, excepto los dos primeros, m_1 y m_2, y por supuesto, tampoco la clave privada m_0.

Para esto, primero aleatorizaremos la firma CL como vimos en 7.3.4 y realizaremos una prueba no interactiva de que conocemos la representación (m_0, m_1, m_2) (7.2.3.1) de un cierto valor en una cierta base.

Algoritmo 7.10 (Revelación selectiva).

Conocimiento común: $n, Z, S, R_0, \ldots, R_l, m_3, \ldots, m_l$.

Conocimiento del Usuario: m_0, su clave secreta, m_1 y m_2, sus atributos ocultos, (A, e, v) su firma CL.

Protocolo:

1. $V \rightarrow P$: valor aleatorio n_1.
2. Usuario aleatoriza la firma CL:
   2.1. r aleatorio, $(A' := AS^r (\bmod n)$, e, $v' := v - er)$
3. Usuario calcula la prueba de conocimiento no interactiva:
   $\mathrm{PK}\{(\hat{e}, \hat{v}, m_0, m_1, m_2) : \frac{Z}{\prod_3^l R_i^{m_i}} \equiv A'^{\hat{e}} S^{\hat{v}} \prod_{i=0}^{2} R_i^{m_i} \bmod n\}(n_1)$
   3.1. Elige valores aleatorio $\tilde{e}$ y $\tilde{v}'$.
   3.2. Elige valores aleatorios $\tilde{m}_0$, $\tilde{m}_1$ y $\tilde{m}_2$.

3.3. Calcula $\tilde{Z} := (A')^{\tilde{e}} \left(\prod_{i=0}^{3} R_i^{\tilde{m}_i}\right) S^{\tilde{v}'} \bmod n$.

3.4. Genera el reto por Fiat-Shamir $c := Hash(A' \| \tilde{Z} \| n_1)$.

3.5. Calcula:

$$\hat{e} := \tilde{e} + ce$$

$$\hat{v}' := \tilde{v} + cv'$$

$$\hat{m}_i := \tilde{m}_i + cm_i, \text{ para } i = 0, 1, 2.$$

3.6. $P_1 := (c, \hat{e}, \hat{v}', \hat{m}_0, \hat{m}_1, \hat{m}_2)$.

4. $U \to V : A'$ y P_1.

5. Verificador comprueba:

5.1. Calcula $\hat{Z} := \left(\frac{Z}{\prod_{i=3}^{l} R_i^{m_i}}\right)^{-c} (A')^{\hat{e}} \left(\prod_{i=0}^{3} R_i^{\hat{m}_i}\right) S^{\hat{v}'} \bmod n$.

5.2. Acepta si $c = Hash(A' \| \hat{Z} \| n_1)$.

Tras la ejecución del protocolo anterior, el Verificador conoce algunos de nuestros atributos, y una prueba de que el Emisor nos los firmó, junto a nuestra clave secreta, m_0, y otros atributos que hemos ocultado. El Verificador tampoco conoce la firma CL original (A, e, v), sólo A' de la firma aleatorizada, y dos testigos, $\hat{e}$ y $\hat{v}'$ de la misma firma aleatorizada.

Un ejemplo cercano de uso de este protocolo podría ser las encuestas anónimas de asignaturas. Actualmente, si queremos asegurar al alumno que la encuesta es totalmente anónima, sin repercusiones, debemos dejarla *abierta*, pudiendo cualquier persona rellenarla. Si queremos asegurar que sólo los alumnos matriculados puedan acceder a la encuesta, tendrían que iniciar sesión con su correo electrónico, y que confíen en que nadie accederá a sus comentarios.

Con un certificado Idemix, un atributo podría ser "Alumno de la asignatura X", otro sus datos de identificación. La Universidad firmó su certificado, que puede llevar en la tarjeta inteligente, por lo que disponemos de una firma CL. Al iniciar sesión en la encuesta, bastaría con mostrar que se es alumno de la asignatura, y generar la prueba no interactiva de que posee un certificado firmado por la Universidad. Sólo los verdaderos alumnos podrán acceder, y ningún registro puede relacionar una opinión con su autor.

Otras soluciones criptográficas tradicionales sólo se preocupaban de proteger la transmisión, que el mensaje no pudiera ser interceptado. Con las pruebas de conocimiento cero hemos conseguido protegernos ante un interlocutor en quien no confiamos.

8 IMPLEMENTACIONES

En este capítulo mostramos las implementaciones de algunos algoritmos descritos durante el trabajo. Utilizamos para esto un *sistema algebraico computacional* (CAS), de código abierto, llamado `Sage`, construido sobre conocidos paquetes matemáticos como `NumPy`, `Sympy`, `R`, `PARI/GP` o `Maxima`, y que utiliza para programar un lenguaje basado en `Python`.

La ventaja de esta herramienta es la facilidad para manejar estructuras de datos y operaciones que en otros lenguajes sería muy largo de implementar, lo que facilita, además, la lectura del código.

8.1 PRUEBA INTERACTIVA: PROBLEMA DEL RESIDUO CUADRÁTICO

Listing 1: QR

```
s = 0
t = 0

def Setup():
  global s
  p = random_prime(2^100)
  q = random_prime(2^100)
  N = p*q
  s = Zmod(N).random_element()
  x = s^2
  return x,N

def elegir_u(x,N):
  global t
  t = Zmod(N).random_element()
  return t^2

def enviar_prueba(x,N,u,b):
  if b == 0:
    return t
  else:
    return Zmod(N)(t*s)

x,N = Setup()
u = elegir_u(x,N)
w = enviar_prueba(x,N,u,1)

print "Comprobar"
print Zmod(N)(w^2)
print Zmod(N)(u*x)
```

Listing 2: QR2

```
p = 0
q = 0

def Setup():
  global p,q
  p = random_prime(2^100)
  q = random_prime(2^100)
  N = p*q
  s = Zmod(N).random_element()
  x = s^2
  return x,N

def elegir_u(x,N):
  t = Zmod(N).random_element()
  return t^2

def enviar_prueba(x,N,u,b):
  if b == 0:
    z = u
  else:
    z = x*u
  zp = Zmod(p)(z)
  zq = Zmod(q)(z)
  w = crt([ZZ(sqrt(zp)),ZZ(sqrt(zq))],[p,q])
  return w

x,N = Setup()
u = elegir_u(x,N)
w = enviar_prueba(x,N,u,1)

print "Comprobar"
print Zmod(N)(w^2)
print Zmod(N)(u*x)
```

El algoritmo representa...

Durante la etapa de `Setup` se eligen dos primos aleatorios, p y q, con una magnitud de hasta 2^{100}. El módulo del problema QR será $N = pq$, un número compuesto donde resolver una instancia del problema es difícil. Por último, se elige un valor aleatorio s (de **s**ecreto) en $\mathbb{Z}_N$, cuyo cuadrado será el residuo cuadrático x de la instancia actual del problema. Los valores de p, q y s sólo los conocerá P, mientras que la información común a P y V es N y x, la instancia del problema QR.

A continuación, siguiendo el protocolo, P elige su *compromiso* u, un residuo cuadrático de $\mathbb{Z}_N$ al azar. Lo calculamos como x antes, eligiendo un t al azar, y elevando al cuadrado. El valor de t será una de sus raíces cuadradas módulo N.

Por último, `enviar_prueba` recibe como parámetro el *reto* b de V. Según el protocolo, P debe devolver una raíz cuadrada módulo N de u o de $x \cdot u$.

En Listing 1, gracias a que P guarda los valores de s y t, raíces de x y u, respectivamente, le basta con enviar t o $s \cdot t \bmod N$.

En cambio, como vimos en el capítulo de los residuos cuadráticos, la raíz módulo un número primo es fácil de calcular. Por eso, si conocemos la factorización de N, podemos utilizar el Teorema Chino de los Restos para calcular una raíz de $u \cdot x^b \bmod N$ a partir de las raíces módulo p y q. La versión de Listing 2 aplica esta idea, utilizando herramientas de `Sage` muy útiles, cómo la raíz modular, `ZZ(sqrt(zp))` o el Teorema Chino de los Restos, `crt()` (Chinese Remainder Theorem).

La optimización de operaciones utilizando la factorización de N y el TCR es una técnica conocida y utilizada en otros protocolos criptográficos conocidos, como RSA [12], pero el hecho de almacenar p y q como claves privadas, además del secreto s, obliga a tener que aplicar medidas de seguridad a más de un valor, existiendo más puntos de ataque a un usuario.

Listing 3: QT

```
y = 0

# Accion de P. Configuracion de parametros iniciales.

def Setup():
  p = random_prime(2^100)
  q = random_prime(2^100)
  N = p*q
  a = Zmod(p).random_element()
  while a.is_square():
    a = Zmod(p).random_element()
  b = Zmod(q).random_element()
  while b.is_square():
    b = Zmod(q).random_element()
  s = crt([ZZ(a),ZZ(b)],[p,q])
  #print kronecker(s,N)
  return s,N

# Hiding.

def Hiding(s,N,b):
  global y
  y = Zmod(N).random_element()
  while gcd(y,N)!=1:
    y = Zmod(N).random_element()
  x = Zmod(N)(s^b*y^2)
  return x

# Opening.

def v(s,N,x,y):
  if Zmod(N)(x-y^2)==0:
    return 0
  elif Zmod(N)(x-s*y^2)==0:
    return 1
  else:
```

```
    return -1

s,N = Setup()
x = Hiding(s,N,1)
print v(s,N,x,y)
```

APPENDIX

A

CÓDIGO FUENTE

Listing 4: A floating example (`listings` manual)

```
for i:=maxint downto 0 do
begin
{ do nothing }
end;
```

BIBLIOGRAFÍA

[1] Manuel Blum. "How to Prove a Theorem So No One Else Can Claim It". En: (1986).

[2] Jan Camenisch y Anna Lysyanskaya. "An efficient system for non-transferable anonymous credentials with optional anonymity revocation". En: *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2001, págs. 93-118.

[3] Jan Camenisch y Anna Lysyanskaya. "A signature scheme with efficient protocols". En: *International Conference on Security in Communication Networks*. Springer. 2002, págs. 268-289.

[4] Jan Camenisch y Markus Stadler. "Efficient group signature schemes for large groups". En: *Advances in Cryptology—CRYPTO'97* (1997), págs. 410-424.

[5] Ivan Damgard. "Efficient concurrent zero-knowledge in the auxiliary string model". En: *Lecture Notes in Computer Science*. Springer. 2000, págs. 431-444.

[6] Louis C. Guillou Jean-Jacques Quisquater y Thomas A. Berson. "How to Explain Zero-Knowledge Protocols to Your Children". En: *Advances in Cryptology - CRYPTO '89* (1990). Proceedings 435: 628-631. http://pages.cs.wisc.edu/~mkowalcz/628.pdf.

[7] Jennifer Seberry Josef Pieprzyk Thomas Hardjono. *Fundamentals of Computer Security*. Springer, 2003. URL: http://gen.lib.rus.ec/book/index.php?md5=38c388552533bc1e364ee56821313e02.

[8] Adrián Sánchez Martínez. *La cueva.* Imagen.

[9] Alfredo Marín. *Apuntes de Grafos y Optimización Discreta*. 2014.

[10] David S. Johnson Michael R. Garey. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. First Edition. Series of Books in the Mathematical Sciences. W. H. Freeman, 1979. ISBN: 0716710455,9780716710455. URL: http://gen.lib.rus.ec/book/index.php?md5=77F747B4A3CC87AAF94F6A9EA1CBC1CF.

[11] *Specification of the Identity Mixer Cryptographic Library (v2.3.43)*. Inf. téc. IBM Research, ene. de 2013.

[12] William Stallings. *Cryptography and Network Security: Principles and Practice*. 6.ª ed. Pearson Ed. Inc., 2014. ISBN: 978-0-13-335469-0. URL: http://gen.lib.rus.ec/book/index.php?md5=d143a3988f2f36b0de0083f78915e60b.