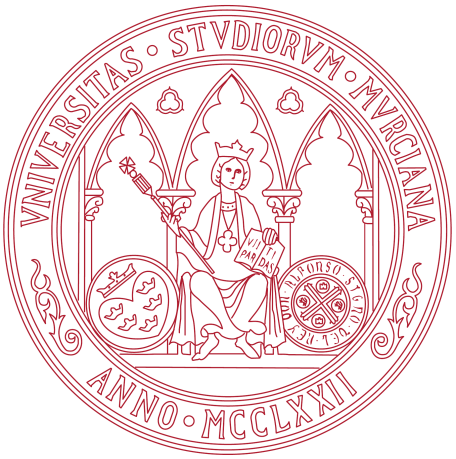
PRUEBAS DE CONOCIMIENTO CERO Y SUS APLICACIONES

JOSÉ LUIS CÁNOVAS SÁNCHEZ

Tutores
LEANDRO MARÍN MUÑOZ
ANTONIO JOSÉ PALLARÉS RUIZ



Facultad de Matemáticas
Universidad de Murcia

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

RESUMEN

TODO

Short summary of the contents in -English- Spanish... a great guide by Kent Beck how to write good abstracts can be found here:

https://plg.uwaterloo.ca/~migod/research/beckOOPSLA.html

ABSTRACT

1500 words of Introduction in English...

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

LISTINGS

1

INTRODUCCIÓN

2

PRELIMINARES

Para poder comprender los resultados de los siguientes capítulos necesitaremos recordar ciertas definiciones y propiedades de álgebra que se cursan durante el grado, e introducir otros preliminares de algoritmia que formalizan el estudio. No incluiremos demostraciones, pues son los conceptos básicos de donde partiremos para desarrollar el resto del trabajo, pero el lector que quiera conocerlas puede consultar las referencias en **TODO: cite** [1].

2.1 PRELIMINARES DE ÁLGEBRA

GRUPOS Y ANILLOS

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

Definición 2.1. Una *operación binaria* en un conjunto A es una aplicación $* : AxA \to A$.

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 2.2. 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 2.3. 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*.

ARITMÉTICA EN $\mathbb{Z}$

Definición 2.4. 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 2.5. *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 2.6 (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 2.7 (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.

CONGRUENCIAS

Definición 2.8. 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 2.9. *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 2.10. *El conjunto $\mathbb{Z}_n^*$ de los elementos invertibles, con el producto, de $\mathbb{Z}_n$, es un grupo abeliano.*

Teorema 2.11. *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 2.12. $\mid \mathbb{Z}_n^* \mid = \varphi(n)$, *donde φ es la función de Euler.*

Teorema 2.13 (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 2.14. *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 2.15 (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 2.16. 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 2.17. *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 2.18. 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 2.19 (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* $\mathbb{Z}_n^* = \{\alpha^i \bmod n \mid 0 \leqslant i \leqslant \varphi(n) - 1\}$.

EL GRUPO SIMÉTRICO S_n: PERMUTACIONES

Definición 2.20. 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 2.21. 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.

2.2 PRELIMINARES DE COMPUTACIÓN

En esta sección introducimos unos preliminares de complejidad computacional que nos permitirán entender de manera básica por qué se utilizan ciertos problemas matemáticos en la seguridad informática. Comencemos con unas definiciones:

Definición 2.22. 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 2.23. 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 2.24. Una *instancia* de un problema es el caso particular de un problema al que se le han dado valores a los parámetros.

Definición 2.25. 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 2.26. Se llama *problema de decisión* a un problema cuya respuesta está en el conjunto $\mathcal{B} = \{\texttt{Verdadero}, \texttt{Falso}\}$.

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 2.27. 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 `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.

NOTACIONES ASINTÓTICAS

Se utilizan para estudiar cómo crece el tiempo de ejecución, la memoria usada o cualquier otro recurso, de un algoritmo dependiendo del tamaño de los datos de entrada. Nos interesa en particular una:

Definición 2.28. 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:

Definición 2.29. 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 $O(f) : t \in O(f)$.

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.*

CLASES DE COMPLEJIDAD

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

Definición 2.30. 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 2.31. El conjunto de problemas de decisión que se pueden resolver en tiempo polinomial se llama clase **P**.

Definición 2.32. Se llama clase de complejidad **NP** al conjunto de problemas de decisión donde una respuesta que sea $\mathtt{Verdadero}$ se puede verificar en tiempo polinomial, dada cierta información extra, que llamaremos *certificado*.

Definición 2.33. 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 2.34 (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 2.35. 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, si $L_1 \leqslant_P L_2$, entonces L_2 es computacionalmente tan difícil como L_1, o de otro modo, L_1 no es más difícil que L_2.

Definición 2.36. 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 2.37 (Teorema de Cook (1971))**.** *El problema de satisfacibilidad booleana es **NPC**.*

Para conocer más sobre el problema y la demostración se puede consultar [4].

O expresado de otra manera:

Teorema 2.38. *Todo problema* $Q \in \boldsymbol{NP}$ *se puede reducir en tiempo polinomial al problema de satisfacibilidad booleana.*
$\forall Q \in \boldsymbol{NP}$, $Q \leqslant_P SAT$.

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.

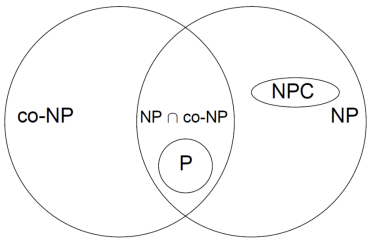


Figura 1: Conjetura de las relaciones entre clases **NP**, **co-NP**, **NPC**, **P**.

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.

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 2.39. 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 2.40. 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 2.41 (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.

2.3 PRELIMINARES DE GRAFOS

La teoría necesaria para comprender las aplicaciones de ZKP con grafos es muy básica, los resultados de esta sección se pueden consultar en [3].

Definición 2.42. 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 2.43 (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 2.44. 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 $| Sym(V) | = n!$.

REPRESENTACIÓN DE GRAFOS

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

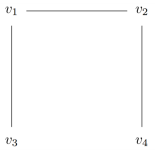
- La *matriz de adyacencia* $M_{n \times n} = (m_{ij})$, que se construye mediante:

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

- La *matriz de incidencia* (vértice-arista) $M_{n \times m} = (m_{ij})$ 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 $M_{m \times 2}$ cuyas filas contienen los índices de los extremos inicial y final de las aristas del grafo.
Ejemplo 2.45. El grafo:



se representaría como:

1	2
1	3
2	4

PROBLEMA DEL ISOMORFISMO DE GRAFOS

3

RESIDUOS CUADRÁTICOS

TODO : Párrafo de introducción al capítulo y referencias

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

3.1 PRIMERAS PROPIEDADES

Definición 3.1. 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 3.2. 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 $0 \notin Q_n$ y $0 \notin \overline{Q_n}$.

Definición 3.3. 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 3.4. *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 3.5. 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 3.6. *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 $x^2 \equiv a \bmod n$.

□

3.2 SÍMBOLO DE LEGENDRE

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

Definición 3.7. 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 3.8 (Criterio de Euler)**.** *Sea p un primo impar. Sea $a \not\equiv 0 \bmod p$. Entonces:*

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

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.

□

Proposición 3.9 (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:

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

□

Teorema 3.10 (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:

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

3.3 SÍMBOLO DE JACOBI

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

Definición 3.11. 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 3.12. 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:

Teorema 3.13. *Propiedades del Símbolo de Jacobi:*

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

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

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

(iii) Las fórmulas del Teorema 3.10 se siguen verificando si p *y* q *son enteros impares positivos, ya no necesitan ser primos.*

3.4 EL PROBLEMA DE RESIDUOSIDAD CUADRÁTICA

Indicar el problema, que es NP y que se puede reducir al de encontrar la factorización de N

4

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.

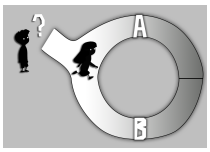
Las pruebas de conocimiento cero surgen a partir de los sistemas de pruebas interactivas, que forman una parte importante de la teoría de complejidad computacional, y pidiendo la propiedad de *conocimiento cero* obtenemos el subconjunto de sistemas interactivos que conforman las pruebas de conocimiento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

Las referencias para este capítulo se pueden encontrar en

4.1 UNA PEQUEÑA HISTORIA

Antes de estudiar formalmente las ZKP, vamos a ver un ejemplo que se publicó originalmente como un cuento sobre la cueva de Alí Babá [5], pero que aquí adaptamos para resumirlo.

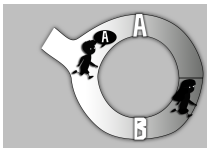
Imaginemos una cueva donde el camino se bifurca y al final de cada pasillo se juntan ambos caminos formando una especie de anillo. En el punto en que se unen dentro de la cueva, hay una puerta con un código secreto que permite abrirla desde ambos lados, para cruzar al otro pasillo.



La cueva [2]. Peggy entra por A o B al azar. Víctor espera fuera.

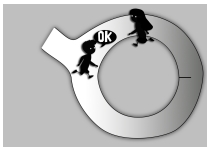
Peggy conoce la clave secreta y quiere **p**robarlo a su amigo Víctor, pero sin tener que revelársela. Peggy y Víctor quedan en la entrada de la cueva con unos *walkie-talkies*, de modo que Víctor esperará fuera y Peggy entrará a la cueva y tomará uno de los pasillos, que llamaremos A y B, sin decirle cuál a Víctor.

Al llegar a la puerta, avisa a Víctor para que entre a la cueva y espere en la bifurcación, donde **V**íctor, para intentar **v**erificar que Peggy conoce la clave, le indicará por qué pasillo quiere que vuelva, el A o el B.



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

Si Peggy realmente conoce la clave, podrá volver a la bifurcación por el pasillo solicitado, abriendo, si es preciso, la puerta. En caso de que no conociera la clave, al entrar tenía una probabilidad del 50 % de adivinar qué pasillo pediría Víctor.



La cueva. Peggy vuelve por el camino pedido.

Víctor no se queda contento con una sola prueba, así que la repiten hasta que se convence. Si lo repitieran, por ejemplo, 20 veces, Peggy tendría solo una probabilidad de 2^{-20}, prácticamente nula, de acertar todas las veces y engañar a Víctor.

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

Más tarde Eva habla con Víctor, que está seguro de que Peggy 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 Peggy puede realizarla con éxito.

4.2 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 4.1. 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 4.2. 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 4.3. *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** (2.32).
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 N compuesto, 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 4.4 (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 *Verdadera*, x es un residuo cuadrático módulo N.

Teorema 4.5. *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 *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 *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*. □

4.3 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 4.6. 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 $\mathtt{Verdadera}$, donde P realmente conoce si es $\mathtt{Verdad}$. Para una instancia cuya prueba falla, o bien es realmente una instancia $\mathtt{Falsa}$ 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 4.7. 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:

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

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

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

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

Definición 4.8. 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 4.9 (Propiedad de conocimiento cero)**.** Un sistema de prueba interactiva para un problema de decisión Q es una *prueba de conocimiento cero perfecta* si el ensamble $\mathtt{Vista}_{P,V}(q, h)$ es idéntico al ensamble generado por un Simulador $S_{V^*}(q, h)$, para cualquier instancia $\mathtt{Verdadera}$ $q \in Q$ y cualquier historial h.

La existencia de un Simulador $S_{V^*}(q, h)$, cuyos ensambles son iguales que los de una Vista generada entre un P y V honestos, implica que de la Vista no se puede obtener ninguna información que V no tuviera ya antes, pues V podía obtener con el Simulador cuantas Vistas quisiera, con el conocimiento que ya tenía.

4.3.1 *Residuos cuadráticos*

Ahora podemos volver al problema QR, que ya vimos en el Teorema 4.5 que su prueba interactiva cumplía *completitud* y *robustez*.

Teorema 4.10. *La prueba interactiva (4.4) 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:

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

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

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 4.4 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 (4.4) que genera las vistas:

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

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 $\alpha = \mid \mathbb{Z}_N^{Q+} \mid$, 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 $\beta = \mid \Omega_u \mid = \mid \Omega_{xu} \mid$. 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 4.4 del problema QR es perfecta de conocimiento cero.

□

4.3.2 *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 $\lvert V_0 \rvert = \lvert V_1 \rvert = n$.
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 4.11. *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 4.12 (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: Sea $t(n)$ un polinomio en n. P y V repiten $t(n)$ veces los siguientes pasos.

1. P calcula el isomorfismo τ entre G_1 y G_0, es decir, $\tau(G_1) = G_0$.
2. $P \to V:\quad h = \pi(G_0)$, donde $\pi \in_R Sym(V_0)$.
3. $V \to P:\quad b \in_R \{0, 1\}$.
4. $P \to V:\ \omega,$ tal que

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

5. 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 $Falsa$, $G_0 \ncong G_1$, 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 *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 $\pi_{i+1} \in_R Sym(V_{b_{i+1}})$ 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 $Vista_{P,V^*}(G_0, G_1, h)$, 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, $Vista_{P,V^*}(G_0, G_1, h) = S_{V^*}(G_0, G_1, h) = (G_0, G_1, h)$, por lo que sus distribuciones de probabilidad son idénticas y los ensambles coinciden.

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

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 $\mid Sym(V_0) \mid = n!$, $P(\Phi_i = \pi) = \frac{1}{n!}$.

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 $\mid V_0 \mid = \mid V_1 \mid = n$, por lo que $\mid Sym(V_0) \mid = \mid Sym(V_1) \mid = n!$, 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.

□

4.3.3 *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 cálculo 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, $\log_g y = u$?

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 4.13. *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 4.14 (Prueba interactiva para DL)**.**
Datos comunes: Una instancia (G, 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*.

□

PRUEBAS DE CONOCIMIENTO CERO ESTADÍSTICAS

PRUEBAS DE CONOCIMIENTO CERO COMPUTACIONALES

5 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.

5.1 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: poner 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.

5.1.1 *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 5.1 (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 4.4, 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.

5.1.2 *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 5.2 (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 $u \equiv \pm y^2 \cdot \prod_{j=1}^{k} v_j^{b_i} \bmod N$.

---

6 IMPLEMENTACIONES

?

Implementaciones de símbolos, raíz discreta, ZKPs, ...

APPENDIX

A

CÓDIGO FUENTE

Listing 1: A floating example (`listings` manual)

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

BIBLIOGRAFÍA

[1] Aberto del Valle y José Asensio. *Apuntes de Grupos y Anillos*. 2013.

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

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

[4] 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.

[5] Louis C.; Berson Thomas A. Quisquater Jean-Jacques; Guillou. "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.

DECLARACIÓN DE ORIGINALIDAD

Yo, José Luis Cánovas Sánchez, autor del TFG PRUEBAS DE CONOCIMIENTO CERO Y SUS APLICACIONES, bajo la tutela de los profesores Leandro Marín Muñoz y Antonio José Pallarés Ruiz, 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.

Murcia, Junio 2017

José Luis Cánovas Sánchez