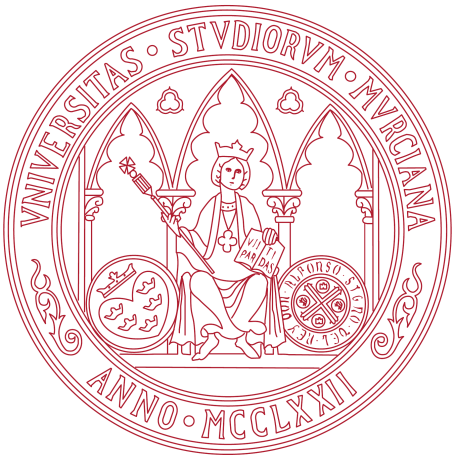
PRUEBAS DE CONOCIMIENTO CERO Y SUS APLICACIONES

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**.

2.1 PRELIMINARES DE ÁLGEBRA

ARITMÉTICA ELEMENTAL

Definición 2.1. 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 $d = \mathrm{mcd}(a, b)$.

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

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

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

Algoritmo 2.3 (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 $b = r_1 > r_2 > ... \geqslant 0$ y cada r_i es entero, para $i = 1, 2, ...$, 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 2.4 (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.

GRUPOS Y ANILLOS

CONGRUENCIAS

Definición 2.5. 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 2.6. *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.

Teorema 2.7. *El anillo* $\mathbb{Z}_n$ *es un cuerpo cuando* n *es un número primo.*

Teorema 2.8 (Euler)**.** *Si* x *es coprimo con*

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

2.2 PRELIMINARES DE COMPUTACIÓN

Teoría de complejidad algorítmica, problema de P NP, problema RSA de factorizar N, problemas de decisión, estadística usada en el estudio de los ZKP (*ensembles*), *probabilistic computations*, ...

La mayor parte está en los primeros capítulos de Fundamentals of Computer Security, y de sus referencias se podrá sacar más detallado.

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 2.10. 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 2.11. 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 2.12. Una *instancia* de un problema es el caso particular de un problema al que se le han dado valores a los parámetros.

Definición 2.13. 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 2.14. 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 2.15. 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 2.16. 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 2.17. 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, $O(f) \subseteq O(g)$.

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.*

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

CLASES DE COMPLEJIDAD

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

Definición 2.18. 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 2.19. El conjunto de problemas de decisión que se pueden resolver en tiempo polinomial se llama clase **P**.

Definición 2.20. 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 2.21. Se llama clase de complejidad **co-NP** al conjunto de problemas de decisión donde una respuesta que sea `Falso` se puede verificar en tiempo polinomial, dado el correspondiente *certificado*.

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

Ejemplo 2.22 (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 2.23. Sean dos problemas de decisión $L_1, L_2 \in \mathbf{NP}$, con correspondientes conjuntos de instancias I_1 e I_2. Sean I_1^+ e I_2^+ los subconjuntos de todas las instancias "Verdadera" 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 2.24. Un problema de decisión L se dice que es **NP**-completo, o **NPC** si:

(i) $L \in \mathbf{NP}$, y

(ii) $L_1 \leqslant_P L \quad \forall L_1 \in \mathbf{NP}$.

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

Para conocer más sobre el problema y la demostración se puede consultar [1].

O expresado de otra manera:

Teorema 2.26. *Todo problema $Q \in$ **NP** se puede reducir en tiempo polinomial al problema de satisfacibilidad booleana.*

$\forall Q \in \mathbf{NP}$, $Q \leqslant_P$ 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, ¿**P** = **NP**?
- ¿**NP** = **co-NP**?
- ¿**P** = **NP** ∩ **co-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.

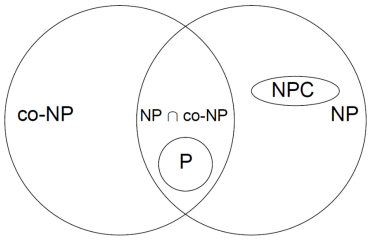


Figura 1: Conjetura de las relaciones entre clases **NP**, **co-NP**, **NPC**, **P**.

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 2.27. 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 2.28. 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 2.29 (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 $\mathtt{Verdadero}$, para una instancia $\mathtt{Verdadera}$ 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 PROBABILIDAD

2.4 PRELIMINARES 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.

El 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 $\overline{Q_n}$.

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

Definición 3.2. 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 3.3. *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 sii $a \equiv \alpha^i \bmod p$ 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. □

Ejemplo 3.4. 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 3.5. *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:
$x \equiv x_p \bmod p$
$x \equiv x_q \bmod q$
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 3.6. Dados un primo impar p y un entero a, se define el *símbolo de Lebesgue* $\left(\frac{a}{p}\right)$ 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 3.7 (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 $a^{\frac{p-1}{2}} \equiv 1$ ó $-1 \bmod p$.

Ahora demostrar el teorema es equivalente a demostrar que $a^{(p-1)/2} \equiv 1 \bmod p$ sii 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 como g tiene orden $p-1$, queda

$g^{\frac{r}{2}} \equiv 1 \bmod p$, 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 3.8 (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 3.9 (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 KRONECKER-JACOBI

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

Definición 3.10. Sean a, $N \in \mathbb{Z}$, con $N = p_1 p_2 \cdots p_r$, donde los p_i son primos, no necesariamente distintos, incluyendo el 2 y el -1 para el signo.

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 para los $p_i > 2$, y para los casos $p = 2$ y $p = -1$ definimos:

$$\left(\frac{a}{2}\right) = \begin{cases} 0, & \text{si } a \text{ es par.} \\ (-1)^{(a^2-1)/8}, & \text{si } a \text{ es impar.} \end{cases}$$

y

$$\left(\frac{a}{2}\right) = \begin{cases} 1, & \text{si } a \geqslant 0 \\ -1, & \text{si } a < 0. \end{cases}$$

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.

Igual que antes, veamos algunas propiedades del Símbolo de Jacobi:

Teorema 3.11. *Propiedades del Símbolo de Jacobi:*

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

(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) Fijado $N > 0$, *el símbolo* $\left(\frac{a}{N}\right)$ *es periódico en* a *con periodo* N *si* $N \not\equiv 2 \bmod 4$, *en otro caso, es periódico con periodo* $4N$.

(iv) Fijado $a \neq 0$, *el símbolo* $\left(\frac{a}{N}\right)$ *es periódico en* N *con periodo* $|a|$ *si* $a \equiv 0$ *ó* $1 \bmod 4$, *en otro caso, es periódico con periodo* $4|a|$.

(v) Las fórmulas del Teorema 3.9 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.

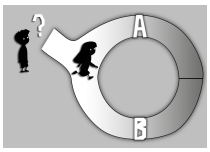
Las pruebas de conocimiento cero surgen a partir de los sistemas de pruebas interactivas, que forman una parte importante de la teoría de complejidad computacional, y pidiendo la propiedad de *conocimiento cero* obtenemos el subconjunto de sistemas interactivos que conforman las pruebas de conocimiento 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 [**ZKPcave:story**], pero que aquí adaptamos para resumirlo.

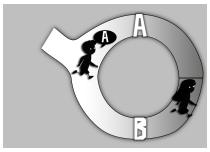
Imaginemos una cueva donde el camino se bifurca y al final de cada pasillo se juntan ambos caminos formando una especie de anillo. En el punto en que se unen dentro de la cueva, hay una puerta con un código secreto que permite abrirla desde ambos lados, para cruzar al otro pasillo.



*La cueva [**ZKPcave:fig**]. Peggy entra por A o B al azar. Víctor espera fuera.*

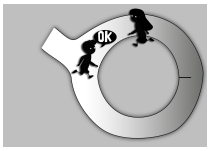
Peggy conoce la clave secreta y quiere **p**robarlo a su amigo Víctor, pero sin tener que revelársela. Peggy y Víctor quedan en la entrada de la cueva con unos *walkie-talkies*, de modo que Víctor esperará fuera y Peggy entrará a la cueva y tomará uno de los pasillos, que llamaremos A y B, sin decirle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

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** (2.20).
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, la instancia es $\mathtt{Falsa}$. 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 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 $\mathtt{Verdadera}$ 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*. □

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.

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:

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

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

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.

□

4.3.1 *Pruebas de conocimiento cero estadísticas*

4.3.2 *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 $w^2 \equiv u \cdot v^b \bmod N$.

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.

Protocolo: Repetir t rondas:

1.

---

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] 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.

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