Universidad de Buenos Aires

Facultades de Ciencias Económicas,

Cs.Exactas y Naturales e Ingeniería

Carrera de Especialización en Seguridad Informática

Trabajo Final de Especialización

Tema
***Zero Knowledge proofs* (*ZKP*)**

Título
***Zero Knowledge proofs* y sus aplicaciones prácticas**

Autor: Ing. Nicolás Axel Bukovits

Tutor: Dr. Pedro Hecht

Año de presentación: 2023

Cohorte del cursante: 2022

Declaración Jurada de origen de los contenidos

Por medio de la presente, el autor manifiesta conocer y aceptar el Reglamento de Tesis vigente y que se hace responsable que la totalidad de los contenidos del presente documento son originales y de su creación exclusiva, o bien pertenecen a terceros u otras fuentes, que han sido adecuadamente referenciados y cuya inclusión no infringe la legislación Nacional e Internacional de Propiedad Intelectual.

FIRMADO

Nicolás Axel Bukovits

DNI 36.930.787

Resumen

En el presente trabajo se aborda el concepto de ZKP (*Zero Knowledge proofs*) y sus principales aplicaciones prácticas. Se explica claramente qué es ZKP, cómo funciona y qué propiedades posee. Inicialmente se presenta la versión interactiva y luego la adaptación no interactiva, señalando en ambos casos sus principales usos. Además de las definiciones teóricas se presentan ejemplos prácticos para comprender mejor el funcionamiento de esta tecnología. En la parte final del trabajo se enuncian ciertas vinculaciones de la criptografía con la teoría de la complejidad computacional que ha generado ZKP, así como también un análisis desde el punto de vista ético y cómo esta tecnología puede utilizarse para garantizar comportamientos éticos, sin sacrificar privacidad.

Palabras clave: ZKP, conocimiento cero, privacidad, seguridad, ética, protocolos, aleatoriedad, criptomonedas, *blockchain.*

Índice de contenidos

1. Nómina de abreviaturas 6
2. Introducción 7
3. Clasificación de ZKP 7
4. Protocolos interactivos 9
5. Propiedades de ZKP 10
   5.1. Completeness 10
   5.2. Soundness 11
   5.3. Zero-knowledge 12
6. Abstracciones de ZKP simples 14
7. Ejemplo de ZKP de un problema computacional 16
8. Estructura general de los protocolos de conocimiento cero 17
9. Protocolos de autenticación 18
   9.1. Fiat-Shamir 18
   9.2. Guillou-Quisquater 22
   9.3. Schnorr 23
   9.4. Análisis de seguridad de protocolos de autenticación ZKP 25
10. Protocolos ZKP basados en álgebra no conmutativa 27
   10.1. Configuración inicial 32
   10.2. Elección de par de claves del probador 32
   10.3. Elección de par de claves del verificador 33
   10.4. Paso 1 del protocolo 33
   10.5. Paso 2 del protocolo 34
   10.6. Paso 3 del protocolo 34

10.7. Verificación de la respuesta 35
11. Pruebas de conocimiento no interactivas 35
11.1. Protocolos interactivos a pruebas no interactivas 37
12. Uso de ZKP para firma digital 39
12.1. Firma de Schnorr 39
13. Blockchain, criptomonedas y ZKP 40
14. Voto electrónico 45
15. ZKP y teoría de la complejidad 46
16. Análisis desde el punto de vista ético 55
17. Conclusiones 60
18. Glosario 61
19. Bibliografía 64

1. Nómina de abreviaturas

3SAT: *Three literal propositional satisfiability problem*

BPP: *Bounded-error Probabilistic Polynomial time*

co-AM: *Complement* Arthur-Merlin

co-NP: *Complement Nondeterministic polynomial time*

GQ: Guillou-Quisquater

GSDP: *Generalized Symmetric Decomposition Problem*

IP: *Interactive Polynomial Time*

LPN: *Learning parity with noise*

LWE: *Learning with errors*

ML-SAG: *Multilayer Linkable Spontaneous Anonymous Group*

MPC: *Multi Party Computation*

NP: *Nondeterministic polynomial time*

P: *Polynomial time*

PAPA: *Privacy, Accuracy, Property and Accessibility*

PH: *Polynomial Hierarchy*

PSPACE: *Polynomial Space*

PZK: *Perfect Zero-Knowledge*

QAP: *Quadratic Arithmetic Program*

R1CS: *Rank 1 Constraint System*

RGPD: Reglamento General de Protección de Datos de la Unión Europea

RSA: *Rivest Shamir and Adleman*

TTP: *Third Trusted Party*

ZK-SNARK: *Zero-Knowledge Succinct Non-Interactive Argument of Knowledge*

ZK-STARK*: Zero-Knowledge Scalable Transparent Argument of Knowledge*

ZKP: *Zero knowledge proof*

2. Introducción

El concepto de ZKP (*Zero Knowledge proofs*) fue introducido por primera vez en un artículo publicado por Shafi Goldwasser, Silvio Micali y Charles Rackoff en la década de 1980 [1]. En ese artículo, los autores presentan las bases teóricas para la rama del estudio de la complejidad computacional del conocimiento en las demostraciones o pruebas y, en particular, sus aplicaciones en pruebas interactivas.

Antes de introducir una definición de ZKP es importante recordar algunos conceptos: Una afirmación es simplemente una cadena finita de símbolos de un alfabeto dado, válida bajo ciertas reglas sintácticas. Una prueba consiste usualmente en otra cadena finita de símbolos que explica bajo ciertas reglas semánticas de un lenguaje si la afirmación es falsa o verdadera. Un lenguaje es un conjunto de cadenas sobre un alfabeto determinado.

Una prueba de conocimiento cero o ZKP es una prueba que únicamente permite determinar si una determinada afirmación es cierta o no, sin revelar ninguna información adicional acerca de la afirmación en sí, es decir, no hay “fuga” de información. En cierto sentido, se puede pensar en una prueba de conocimiento cero como una respuesta obtenida de un oráculo confiable [2]. El enfoque de ZKP está en cuanta información se transmite cuando se presenta una prueba. En criptografía generalmente se asume al mencionar ZKP que las pruebas de conocimiento cero son de conocimiento (*zero-knowledge proof of knowledge*), que es un caso particular en donde la afirmación consiste sólo en el hecho de conocer o poseer cierta información secreta.

3. Clasificación de ZKP

Las pruebas de conocimiento cero se pueden clasificar principalmente en dos grandes ramas: interactivas y no interactivas. En ambos casos, hay dos entidades (el verificador y el probador) y el objetivo del probador es poder dar una prueba de que una afirmación es cierta que el verificador pueda validar.

En las pruebas no interactivas no existe ningún intercambio ni interacción entre el probador y el verificador. Es simplemente un sistema en donde el probador presenta una prueba que en cualquier momento el verificador puede analizar junto con la afirmación a verificar y decidir si la prueba es válida o no. Es decir, se puede realizar de manera *offline*, no necesita que el verificador y el probador compartan un canal de comunicación que utilicen en algún momento para validar una prueba. Los requerimientos para este tipo de pruebas son más débiles que en los protocolos interactivos, tal como se analiza en la sección “Pruebas de conocimiento no interactivas”.

En cambio, en las pruebas interactivas hay un intercambio de mensajes entre el probador y el verificador. La mayoría de la literatura se refiere a este tipo de pruebas. Se trata de un protocolo en donde el probador trata de “convencer” fuertemente al verificador que una afirmación es cierta. Se utiliza el término “convencer” porque estos tipos de pruebas son probabilísticos. El concepto matemático de prueba ya no es absoluto como el presentado en la introducción, sino que depende de un juego interactivo, por eso también se los denomina protocolos. La prueba no es un objeto estático, escrito en un lenguaje que puede ser verificado en cualquier momento por cualquier verificador. Es decir, el probador y el verificador son formalmente máquinas de Turing probabilísticas y además se asume que el verificador está acotado polinomialmente. No se asume por lo general una limitación computacional en el probador y además un probador honesto debe tener cierta ventaja computacional por sobre el verificador. Caso contrario, un verificador podría simular la prueba por su cuenta. Por lo tanto, se asume que el probador no está acotado computacionalmente o si bien está acotado polinomialmente es capaz de presentar un testigo de la clase de complejidad NP de la pertenencia de una afirmación a cierto lenguaje (esta última asunción es la que más se usa en la práctica para utilidad de la criptografía, mientras que la primera asunción es más útil para el estudio de estos protocolos en el ámbito de la rama de la complejidad teórica) [3]. Debido a su naturaleza probabilística, las pruebas interactivas permiten probar afirmaciones que con un método no interactivo no sería posible, por lo tanto, tienen mayor poder expresivo.

4. Protocolos interactivos

Antes de dar una definición precisa de ZKP es importante definir qué son los protocolos interactivos. Inicialmente el estudio de ZKP se basó sólo en sistemas de pruebas interactivas y posteriormente se adaptó su versión a sistemas de pruebas que no requieren interacción.

Las siguientes definiciones están extraídas de [1]: Vamos a definir una máquina de Turing interactiva como una máquina de Turing clásica pero que posee una cinta de sólo lectura, una cinta de trabajo, una cinta aleatoria, una cinta de sólo lectura de comunicación y una cinta de sólo escritura de comunicación. La cinta aleatoria contiene una secuencia de bits infinita y aleatoria que puede ser leída de izquierda a derecha. Se dice que la máquina lanza una moneda cuando lee un bit de esta cinta. Un protocolo interactivo está formado por un par ordenado de Máquinas de Turing interactivas P y V tal que comparten la misma cinta de input, la cinta de escritura de comunicación de P es la cinta de lectura de comunicación de V y viceversa. En la definición clásica la máquina de Turing P no está acotada computacionalmente mientras que la máquina V está acotada polinomialmente. El protocolo consiste en turnos en donde en cada uno sólo una máquina está activa a la vez. Cuando una máquina está en estado activo realiza cierto cómputo con su cinta de input, de trabajo y aleatoria y finalmente escribe en su cinta de sólo escritura de comunicación. Cuando esto ocurre, pasa al estado inactivo y la otra máquina automáticamente pasa al estado activo repitiendo los mismos pasos. El cómputo se puede detener en cualquier momento si alguna de las máquinas no envía ningún mensaje en su estado de activa. La máquina V acepta o rechaza el input escribiendo *accept* o *reject* en su salida y finalizando el protocolo.

La anterior definición es una caracterización formal del concepto de protocolo interactivo utilizando el modelo de cómputo de Máquinas de Turing. En la siguiente imagen se puede visualizar el comportamiento descripto para dos máquinas denominadas A y B:

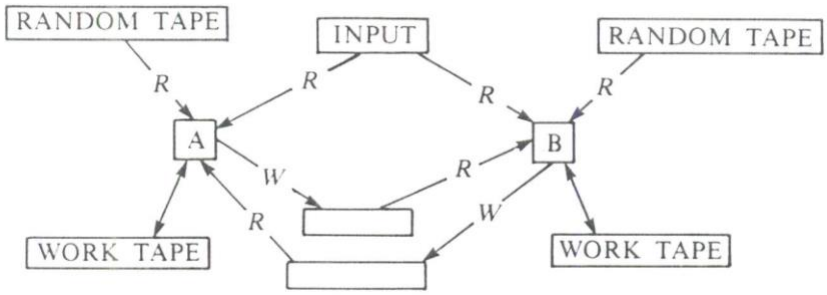

Figura 1 Modelo formal de protocolo interactivo [1]. R representa la lectura y W la escritura.

5. Propiedades de ZKP

En esta sección se analizan las propiedades que deben cumplir los sistemas de pruebas generales y en particular los interactivos, para luego evidenciar que distingue a los sistemas de pruebas de conocimiento cero. Usualmente se requiere que los métodos para verificar una afirmación sean computacionalmente eficientes para el verificador para que sea útil en la práctica. Para el caso de los protocolos interactivos la necesidad radica también en que la cantidad de mensajes intercambiados esté acotada polinomialmente. Además de este requerimiento, formalmente, ZKP debe satisfacer tres propiedades esenciales: *completeness*, *soundness*, y *zero-knowledge*.

5.1. Completeness

Una prueba es completa, si dada la prueba válida, el algoritmo de verificación tiene como salida el valor verdadero, es decir, la prueba es aceptada [4]. Esta es la definición clásica para los sistemas de prueba generales. Para el caso de las pruebas (protocolos) interactivos, la propiedad de completa se cumple si dados un probador honesto y un verificador honesto el protocolo termina satisfactoriamente con una probabilidad muy alta, es decir, el verificador acepta la prueba. La definición de “probabilidad muy alta” depende de la aplicación particular, pero por lo general implica que la probabilidad de falla no es significativa en la práctica [2]. Esta propiedad está relacionada con garantizar que una prueba (protocolo) funcione adecuadamente dado un probador honesto.

Formalmente la propiedad significa que dado $x \in L$, con *L* un determinado lenguaje que identifica las afirmaciones que son verdaderas y *x* una cadena de texto, entonces existe un algoritmo polinomial *V* que se llama de verificación y un certificado π tal que $V(x, \pi) = 1$, significando 1 que es aceptado. [3]

Para el caso de los sistemas de pruebas interactivos, significa que dado $x \in L$ entonces *V* acepta en *(P,V)(x),* siendo *V* el algoritmo polinomial de verificación, *P* el algoritmo que corresponde al probador y *(P,V)* el protocolo interactivo entre ambos (que se asume que intercambia una cantidad de mensajes polinomial), con una alta probabilidad. Esta definición se puede formalizar aún más usando el modelo de máquinas de Turing presentado anteriormente: Sean *P* y *V* dos máquinas de Turing interactivas, la propiedad de *completeness* se cumple si para cada *k*, dado $x \in L$ como input al protocolo interactivo *(P,V)*, *V* se detiene y acepta con probabilidad al menos de $1 - |x|^{-k}$, tomando como probabilidades los lanzamientos de monedas realizados por las máquinas *P* y *V*.

5.2. Soundness

Se dice que una prueba cumple con la propiedad de *soundness* si sucede que la prueba no es válida entonces el algoritmo de verificación tiene como salida el valor falso, es decir, la prueba es rechazada [4]. Para el caso de los protocolos interactivos, la propiedad se cumple si dada una afirmación falsa, un probador deshonesto no puede convencer a un verificador honesto de que es verdadera, excepto con una muy baja probabilidad. Nuevamente, la noción de baja probabilidad depende de la aplicación en particular. Esta propiedad está relacionada con la protección frente a probadores deshonestos.

Formalmente significa que si $x \notin L$ entonces para todo $\pi, V(x, \pi) = 0$, significando 0 que es rechazado.

Para el caso de las pruebas interactivas significa que si $x \notin L$ entonces para todo $P*$, *V* acepta en $(P*, V)(x)$ con una muy baja probabilidad. Nuevamente, esta definición

se puede formalizar más con el modelo visto de protocolo: Sea *V* una máquina de Turing interactiva, la propiedad de *soundness* se cumple si para cada *k*, dado $x \notin L$ y para toda máquina de Turing interactiva *P**, al tomar como input el protocolo interactivo *(P*,V), V* se detiene y acepta con probabilidad a lo sumo $|x|^{-k}$, tomando como probabilidades los lanzamientos de monedas realizados por las máquinas *P* y *V*. Si el protocolo se repite varias veces la probabilidad anterior se puede reducir a $2^{-|x|}$. Intuitivamente esta definición nos dice que el verificador *V* ni siquiera debe confiar o conocer al algoritmo *P* ya que basta con confiar en su propia aleatoriedad [1].

5.3. Zero-knowledge

Informalmente esta propiedad establece que un verificador deshonesto no obtiene ninguna información adicional además de que una determinada afirmación es verdadera o no. Formalmente, quiere decir que existe un algoritmo de tiempo polinómico (simulador) que puede producir a partir de la entrada de la afirmación a probar, pero sin interactuar con el probador real, transcripciones que son indistinguibles de las resultantes de la interacción con el probador real [5]. Es decir, esta propiedad está relacionada con la protección ante verificadores deshonestos, que desean obtener información adicional de la afirmación en sí.

Siguiendo el modelo de Máquina de Turing interactivo, formalmente esta propiedad establece que para toda máquina polinomial *V'* la distribución de símbolos que *V'* puede ver en sus cintas cuando interactúa con *P* con un input $x \in L$ es indistinguible de la distribución de cualquier cómputo que pueda realizar con *x* en tiempo polinomial. Como se detalla más adelante, para las pruebas de conocimiento cero no interactivas, al eliminar la interacción ya no hay que preocuparse por la "vista" que pueda obtener un verificador *V'* deshonesto, por lo que se relaja la condición y se puede eliminar el cuantificador ∀ *V'*.

Las propiedades de *completeness* y *soundness* son requeridas para cualquier sistema de pruebas general, ya sea interactivo o no. Cuando además se cumple con la

tercera propiedad (*zero-knowledge*), el sistema de pruebas se llama de conocimiento cero [6]. Es decir, la esencia de ZKP se basa en cuánta información se revela al presentar una prueba. Como se señala en [7] los sistemas de pruebas de conocimiento cero (en realidad los protocolos de conocimiento cero) tienen tres grandes características que las diferencian de los sistemas de prueba generales: interacción, aleatorización secreta y dificultad computacional (que el probador utiliza para sus pruebas).

Es importante también remarcar otras características y realizar algunas aclaraciones más sobre las propiedades de ZKP. En particular, para el caso de las pruebas interactivas, la prueba que una entidad determinada A (probador) presenta a otra entidad B (verificador) sólo es válida para B, y no provee ninguna garantía a otra entidad C externa que pueda incluso ver todo el intercambio de mensajes durante la ejecución del protocolo. Es decir, son pruebas negables a terceros, propiedad que no se cumple si se elimina la interacción. La propiedad de *zero-knowledge* puede ser perfecta, estadística (también denominada casi perfecta) o computacional. Se dice que es computacional cuando un observador restringido a pruebas probabilísticas de tiempo polinomial no puede distinguir transcripciones reales de simuladas. Cuando además la probabilidad de las distribuciones de las transcripciones es idéntica (y no se limita computacionalmente las estrategias del verificador), se dice que es *zero-knowledge* perfecto [2]. Si las distribuciones no son exactamente iguales, pero están estadísticamente cerca (por ejemplo en distancia de variación), se dice que la propiedad de conocimiento cero es estadística. Por último, ninguna de las propiedades mencionadas garantiza la seguridad. Para que ZKP sea seguro el adversario debería tener que resolver un problema computacionalmente difícil. Es decir, las pruebas (protocolos) ZKP se basan, como la mayoría de la criptografía asimétrica de clave pública, en la asunción de que ciertos problemas son intratables (factorización, residuos cuadráticos, logaritmo discreto, etc.).

6. Abstracciones de ZKP simples

El siguiente esquema de ZKP fue extraído de [6] y representa un ejemplo no relacionado a la criptografía de una prueba interactiva que ayuda a comprender mejor el funcionamiento de estos protocolos.

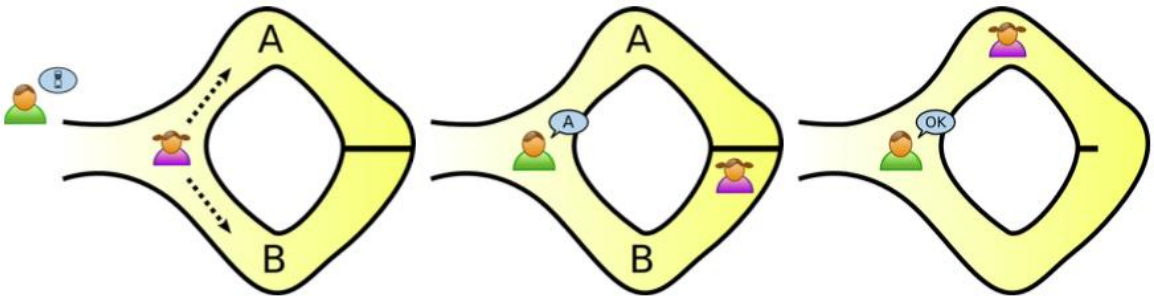

Figura 2 - Cueva mágica de Ali Baba [6]

Como se puede observar en la Figura 2 el esquema consiste en una cueva que tiene dos caminos A y B. Ambos caminos están conectados, pero para poder pasar de uno a otro es necesario atravesar una puerta que solo se abre si se dicen unas palabras secretas. Alice (probador) afirma conocer el secreto y quiere probarle a Bob (verificador) que esto es cierto, pero sin revelarle cuáles son las palabras secretas. Entonces ambos ejecutan los siguientes pasos: Inicialmente Bob espera afuera de la cueva mientras Alice entra. Alice decide tomar un camino de forma aleatoria (A o B) sin que Bob conozca cuál de ellos eligió. Luego Bob entra a la cueva y se acerca hasta el punto de la bifurcación de los caminos y le dice a Alice que regrese por alguno de ellos (A o B) también de forma aleatoria. Si Alice había elegido el mismo camino que Bob le solicita por el que vuelva, entonces Alice simplemente tiene que regresar por el mismo. Sin embargo, si Alice había elegido el camino A y Bob solicita que vuelva por el camino B, o viceversa, la única forma que tiene Alice de hacerlo es a través de la puerta, para la cual necesita conocer el secreto para poder pasar. Es decir, si Alice no conoce el secreto la única forma que tiene de cumplir con lo requerido por Bob es cuando el camino que eligió coincide con el que le solicita Bob, y esto ocurre con una probabilidad de ½. Si el mismo procedimiento descripto se repite muchas veces, la probabilidad que los caminos coincidan en todas

las pruebas se convierte en despreciable y por lo tanto Bob puede estar lo suficientemente seguro que Alice dice la verdad si logra salir en todas las pruebas. Se debe señalar un detalle importante que es que Bob no conoce el camino que eligió Alice (cuyo conocimiento haría que solo una prueba sea necesaria ya que entonces le puede solicitar que salga por el otro camino, forzando a que use el secreto), ya que de ser así podría seguirla para conocer el secreto, por lo que la aleatorización permite reducir la probabilidad que Bob realice lo que se conoce como *eavesdrop* (escucha a escondidas).

Otro ejemplo para comprender el funcionamiento de estos protocolos es el que se menciona en [8] sobre dos bolas y una persona daltónica, es decir que no puede distinguir entre el color rojo y el verde. En este ejemplo, Bob es daltónico y cumple el rol de verificador. Alice posee dos bolas, una de color rojo y otra de color verde, idénticas en forma y tamaño. Alice (probador) quiere probarle a Bob que las bolas son de distinto color (sin revelarle cuál es de cada color). Para ello, le solicita a Bob que ponga ambas bolas detrás de su espalda para que ella no las pueda ver y le pide que seleccione al azar una de ellas y la revele. Bob repite ese proceso, siempre eligiendo mostrar una bola al azar y en cada vez le pregunta a Alice si cambió de bola. Bob sabe la respuesta correcta ya que él eligió cambiar o no la bola de forma aleatoria, pero Alice no conoce su elección, por lo que sólo tiene ½ de probabilidades de acertar si las bolas son del mismo color. Si efectivamente las bolas son de color distinto Alice va a poder siempre responder al desafío de Bob. Es decir, este procedimiento cumple con las propiedades de *completeness* y *soundness* si se repite un gran número de veces y además no brinda información adicional a Bob que nunca llega a deducir de qué color es cada bola ni cómo hacer para descubrirlo, cumpliendo con la propiedad de *zero-knowledge*. Por lo tanto, es un ejemplo sencillo de prueba de conocimiento cero.

7. Ejemplo de ZKP de un problema computacional

A modo de introducir un ejemplo más complejo y aplicado a un problema computacional, se presenta una prueba de conocimiento cero para el problema del isomorfismo de grafos, presentada en [9]. Dos grafos $G(V,E)\ y\ H(V,F)$ se dicen isomorfos si y solo sí existe una permutación $\pi \in S_{|V|}$ del grupo simétrico de $|V|$ elementos, tal que $(u,v) \in E \iff (\pi(u),\pi(\mathrm{v})) \in F$. Se denota entonces que $H = \pi G$. El lenguaje L de los grafos isomorfos está compuesto por todos los pares de grafos isomorfos entre sí, es decir, $\{(G,H) : \exists\, \pi\, , H = \pi G\}$.

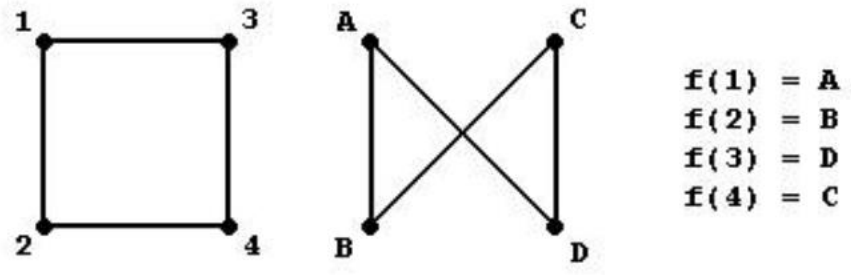

Figura 3 Dos grafos isomorfos. f es la función que aplica la permutación [6]

El protocolo de conocimiento cero para este problema se describe a continuación. La entrada del protocolo consiste en dos grafos $G(V,E)\ y\ H(V,F).\pi \in S_{|V|}$ es el isomorfismo entre ellos y se ejecutan m veces (siendo m la cardinalidad del conjunto E, que tiene que ser la misma cardinalidad que F, ya que de lo contrario no podrían ser isomorfos) los siguientes pasos:

El probador genera un grafo G' que es una copia isomórfica aleatoria de G, mediante la aplicación de una permutación al azar π'. Luego envía este grafo al verificador.

El verificador al azar decide entre solicitarle al probador que le muestre que G y G' son isomorfos o que H y G' son isomorfos.

En la última instancia, el probador debe enviar π' si el verificador le solicita que demuestre que G y G' son isomorfos o debe enviarle $\pi' \cdot \pi$ en caso contrario. El

verificador para decidir si aceptar o no la prueba debe verificar que la permutación recibida en el paso anterior es un isomorfismo entre los grafos que solicitó. Si no lo es, rechaza la prueba, en caso contrario continua con la siguiente ejecución de la ronda.

El protocolo descripto consiste en una prueba de conocimiento cero ya que cumple con las propiedades de *completeness*, *soundness* y *zero-knowledge*. La primera propiedad se cumple ya que si efectivamente G y H son isomorfos, entonces G' va a ser isomorfo tanto con G como con H. Por lo tanto, un probador y un verificador honestos van a completar todas las rondas con éxito con probabilidad 1. Para ver la segunda propiedad hay que considerar entonces que G y H no son isomorfos. Por lo tanto, la copia isomórfica de G' sólo va a ser isomorfa con G o con H, pero no con ambas. Entonces un probador deshonesto solo tiene ½ de probabilidad de hacer trampa en cada ronda. Por último, para ver que el protocolo no brinda a un verificador ninguna información adicional que no sea capaz de computar por su cuenta, hay que analizar qué información se transmite. La única información que revela el probador son las permutaciones π' y $\pi' \cdot \pi$, pero dado que π' es seleccionado aleatoriamente en cada ronda, un simulador que simplemente compute una copia isomorfa de G o de H o de ambas es computacionalmente indistinguible de la interacción con el probador. Para cumplir formalmente con la definición de *zero-knowledge*, se debería probar que la información que se revela no es de ninguna utilidad para cualquier estrategia de un verificador, es decir, cualquier vista que puede obtener un verificador de la interacción. En [9] se demuestra más en detalle que incluso para cualquier verificador deshonesto que intente seleccionar una particular secuencia de solicitudes en el paso 2 para obtener alguna información adicional sobre los grafos, esta estrategia no es efectiva.

8. Estructura general de los protocolos de conocimiento cero

Por lo general, los protocolos de conocimiento cero tienen una estructura formada por 3 pasos:

$$A \rightarrow B: \quad witness$$
$$A \leftarrow B: \quad challenge$$
$$A \rightarrow B: \quad response$$

Figura 4 Esquema de un protocolo de conocimiento cero [2]

En la Figura 4, A tiene el rol de probador y B el de verificador. A provee un testigo (*witness*) a B, que computa a partir de un valor aleatorio secreto. Esto permite que cada una de las ejecuciones del protocolo sea distinta debido a la aleatorización y además establece una serie de preguntas que el probador (y sólo un probador honesto) va a ser posible de responder para "probar" su conocimiento. B luego desafía (envía un *challenge*) a A, seleccionando una de estas preguntas. Finalmente, A envía su respuesta al desafío y B verifica que la misma sea correcta, repitiendo el proceso completo varias veces hasta que la posibilidad de que A esté haciendo trampa sea tan baja cómo B desea. Tanto el testigo como la respuesta que envía A a B, no brindan ninguna información adicional sobre el secreto que conoce A, más allá de la aserción del conocimiento de este.

9. Protocolos de autenticación

La principal aplicación práctica de los protocolos interactivos de conocimiento cero es la autenticación. ZKP permite demostrar que uno es quien dice ser, sin necesidad de revelar el secreto que prueba la identidad, convirtiendo a esta forma de autenticación en una de las más fuertes. Se analizan 3 protocolos clásicos en particular: Fiat-Shamir, Guillou-Quisquater y Schnorr.

9.1. Fiat-Shamir

Amos Fiat y Adi Shamir propusieron un protocolo ZKP de autenticación que está basado en el hecho de que computar las raíces cuadradas y factorizar un módulo son computacionalmente equivalentes (ambos problemas son intratables).

El protocolo consiste en la ejecución de *t* rondas de 3 pasos cada una con la siguiente forma:

$$
\begin{array}{lll}
A \to B: & x = r^2 \bmod n & (1) \\
A \leftarrow B: & e \in \{0,1\} & (2) \\
A \to B: & y = r \cdot s^e \bmod n & (3)
\end{array}
$$

Figura 5 Ronda de Fiat-Shamir básico [2]

En el ejemplo ilustrado A es el probador que tiene conocimiento de un secreto *s* y se lo quiere probar a B, el verificador. El funcionamiento consiste en los siguientes pasos:

Existe un paso inicial, de configuración, que se realiza una sola vez y que consiste en la elección de un número $n = p.q$ por parte de una autoridad de confianza *T* (*p* y *q* son dos números primos grandes, como en el caso de la configuración de RSA). El *n* es público pero los números *p* y *q* se mantienen secretos. A genera un par de claves pública y privada, siendo su clave privada *s* un número que elige coprimo con *n* ($\in Z_n^*$) y su clave pública $v = s^2 \ (mod\ n)$, la cual registra con *T*.

Luego de la configuración inicial se repiten varias rondas de 3 pasos como las descriptas en la imagen. En el primer paso de cada ronda A elige un número aleatorio *r*, calcula $x = r^2 \ (mod\ n)$ y se lo envía a B (1). En el segundo paso, B elige de manera aleatoria un 1 o un 0 y se lo envía a A (2). En el tercer paso, si A recibió un 0 de B le envía *r*, y en caso contrario (si recibió un 1) le envía el resultado de calcular $r.s\ mod\ n$.

Una vez finalizada la ronda, B procede a determinar si fue satisfactoria. Para ello verifica que $y^2 = x\ .v^e \ (mod\ n)$, siendo *y* el cálculo recibido en el paso 3 de la ronda, *e* la elección de B en el paso 2 y *v* la clave pública de A registrada en la configuración inicial. B solo va a aceptar la prueba si todas las rondas son satisfactorias.

A continuación, se demuestra que el protocolo presentado cumple con las propiedades de una prueba de conocimiento cero: la propiedad de *completeness*, es decir, la capacidad de que un probador honesto demuestre el conocimiento ante un verificador honesto se cumple ya que si A efectivamente conoce *s*:

$$y^2 = r^2\ (s^e)^2 = x\ (s^2)^e = x.v^e\ (mod\ n)$$

Es decir, un probador honesto A siempre va a poder responder y completar satisfactoriamente todas las iteraciones de rondas, ya sea con el valor *r* o con $r.s\ mod\ n$ y el verificador lo va a poder aceptar con probabilidad 1.

La propiedad de *soundness* también se cumple ya que si A fuera un impostor, un probador deshonesto que no conoce *s,* no podría prepararse para responder ambas solicitudes de B del paso 2. Si A espera que B responda con un 1, solo podría responder en el paso 3 con la respuesta $y = r$, habiendo seleccionado $x = \frac{r^2}{v}$ en el primer paso. Si B en cambio hubiera enviado un 0, A necesita calcular una raíz cuadrada de *x mod n*, lo cual si el *n* es lo suficientemente grande es impracticable. Si A espera que B responda con un 0, podría responder el paso 3 con la respuesta $y = r$, habiendo seleccionado $x = r^2$ en el primer paso con un *r* arbitrario. Pero si B hubiera enviado un 1, A también necesitaría calcular una raíz cuadrada de *x mod n*. Lo que es seguro es que no puede prepararse para ambos escenarios al mismo tiempo, porque de ser así significaría que puede calcular *s,* la clave secreta, ya que si puede preparar $r_0 = x$ y $r_1 = s.x$ puede obtener $s = r_1/r_0$. De aquí surge la importancia de que A no sepa de antemano qué valor va a seleccionar B en el paso 2, que sea completamente aleatorio, ya que en caso contrario un probador A deshonesto podría prepararse. Del hecho que el paso 1 y 2 se haga uso de la aleatoriedad y de que calcular una raíz cuadrada de un módulo es una tarea computacionalmente difícil surge la seguridad del protocolo. Como a lo sumo un probador deshonesto puede responder a una de las solicitudes de B, tiene un ½ de probabilidades de acertar. Si se repiten las rondas un número lo suficientemente grande de veces *t,* la probabilidad resultante de hacer trampa es 2^{-t}.

Por último, veamos si la propiedad de *zero-knowledge* se cumple, es decir, que B no obtiene ninguna información adicional más allá de la prueba de que A conoce el secreto *s*. Cuando A responde con el cálculo $y = r$, esto no brinda ninguna información sobre el secreto *s* a B, y cuando responde con $y = rs\ (mod\ n)$ tampoco porque B no conoce el número aleatorio *r*. Formalmente el par *(x,y)* de información que brinda A en los pasos 1 y 3, puede ser simulado por B, sin necesidad de interactuar con A,

seleccionado un valor *z* aleatorio y luego definiendo $w = z^2$. El par *(z,w)* tiene una distribución de probabilidad indistinguible de *(x,y).*

Un ejemplo ilustrativo de funcionamiento de dos rondas del protocolo con números artificialmente pequeños es el siguiente [6]:

Alice es el probador y Bob es el verificador. En el paso de inicialización se seleccionan como parámetros $p = 5$, $q = 7$ y $n = pq = 35$. El número n es publicado. Alice escoge de manera secreta $s = 16$, que es un número coprimo con n. Luego publica el número $v = s^2 \ (mod\ n) = 11$.

Para demostrar que Alice conoce el número s sin revelarlo (y suponiendo que con una probabilidad menor o igual a ¼ de hacer trampa por parte de Alice es suficiente para Bob) se ejecutan los siguientes pasos:

1. Alice selecciona de manera aleatoria un número $r = 10$. Calcula $x = r^2 \ (mod\ n) = 10^2 \ (mod\ 35) = 30$ y envía el resultado a Bob.
2. Bob de manera aleatoria elige $e = 0$ y se lo envía a Alice.
3. Alice al recibir $e = 0$, computa $y = r = 10$ y se lo envía a Bob.
4. Bob verifica que $y^2 = 10^2 \equiv 30 \ (mod\ 35)$.
5. Alice selecciona de manera aleatoria un número $r = 20$. Calcula $x = r^2 \ (mod\ n) = 20^2 \ (mod\ 35) = 15$ y envía el resultado a Bob.
6. Bob de manera aleatoria elige $e = 1$ y se lo envía a Alice.
7. Alice al recibir $e = 1$, computa $y = sr \ (mod\ n) = 16.20 \ (mod\ 35) = 5$ y se lo envía a Bob.
8. Bob verifica que $y^2 = 25 \equiv 15.11 \ (mod\ 35)$.

Al ser ambas rondas satisfactorias, Bob acepta la prueba.

Figura 6 Ejemplo de dos rondas del protocolo de autenticación de Fiat-Shamir [6]

9.2. Guillou-Quisquater

El protocolo de Guillou-Quisquater (GQ) es otro protocolo de conocimiento cero de autenticación que está basado en el de Fiat-Shamir. Su estructura es también de 3 pasos y consiste en lo siguiente:

A		B
(n, x)		(n, y)
$r \in_R \mathbb{Z}_n^*$		
$t = r^e \pmod n$	$\xrightarrow{t}$	
	$\xleftarrow{c}$	$c \in_R \{0, \ldots, e-1\}$
$s \equiv rx^c \pmod n$	$\xrightarrow{s}$	
		$s^e \stackrel{?}{\equiv} ty^c \pmod n$
		$(accept \text{ or } reject)$

Figura 7 Ronda de GQ [4]

En este protocolo de autenticación A es poseedor de una clave privada *x* y una clave pública $y = x^e \ (mod\ n)$ y quiere demostrarle a B que conoce la clave privada *x* sin revelarla. En el primer paso de la ronda A computa $t = r^e \ (mod\ n)$ a partir de un valor aleatorio $r \in Z_n^*$. Este valor que tiene la función de testigo es enviado a B, quien en el paso 2 selecciona también de forma aleatoria un número comprendido entre 0 y $e - 1$ y se lo envía a A. En el último paso A computa $s = rx^c \ (mod\ n)$ y se lo envía a B como respuesta. Para determinar si la respuesta fue satisfactoria B tiene que verificar que $s^e = ty^c \ (mod\ n)$. Si en todas las rondas esto se verifica B acepta la prueba, sino la rechaza.

Es sencillo ver que el protocolo cumple con la propiedad de *completeness* ya que $s^e = (rx^c)^e = r^e(x^e)^c = ty^c \ (mod\ n)$. Analizando la propiedad de *soundness* podemos ver que un probador deshonesto se puede preparar para responder algún *challenge* particular *c´*, ya que si selecciona aleatoriamente un *s* y usa $t = s^e y^{-c'} \ (mod\ n)$ como testigo en el primer paso, puede responder correctamente *s* en el último paso y va a ser aceptado, pero solo si efectivamente B seleccionó *c´* en el paso 2. Se puede demostrar que no se puede preparar para más de un *challenge* [4]: Supongamos que se puede preparar para c_1 y c_2 distintos. Se puede afirmar entonces que ${s_1}^e = ty^{c_1} \ (mod\ n)$ y ${s_2}^e =$

$ty^{c_2}\ (mod\ n)$. Si se dividen ambos términos se obtiene lo siguiente: $(s_1/s_2)^e = y^{c_1-c_2} = (x^{c_1-c_2})^e (mod\ n)\ y\ \frac{s_1}{s_2} = x^{c_1-c_2}\ (mod\ n)$. Luego se podría utilizar el algoritmo de Euclides extendido para obtener *u* y *v* tales que $u(c_1 - c_2) + v.e = 1$ (debido a que el máximo común divisor entre e y $c_1 - c_2$ es 1 por ser e un número primo). Una vez obtenidos estos valores es fácil obtener el secreto *x* de la siguiente manera: $(s_1/s_2)^u y^v = x^{u(c_1-c_2)}\ x^{v.e} = x\ (mod\ n)$. Es decir, se podrán extraer las e raíces sin necesidad de conocer el número de orden del grupo, lo que se considera que no es posible en la práctica, entonces no sería posible que se prepare para dos o más *challenges*. La probabilidad de que acierte en el *challenge* que va a elegir B es por lo tanto $\frac{1}{e}$.

La principal diferencia que tiene este algoritmo con el de Fiat-Shamir es que permite disminuir la cantidad de veces que es necesario repetir el protocolo para obtener una probabilidad de falla baja (*soundness*), lo que resulta en una reducción de la cantidad de mensajes intercambiados entre el probador y el verificador. Fiat-Shamir está basado en la dificultad de computar la raíz cuadrada en un grupo finito de orden desconocido, con una probabilidad de hacer trampa para un probador deshonesto de ½ en cada ronda, mientras que Guillou-Quisquater se basa en la dificultad de computar *e* raíces en un grupo finito, con una probabilidad de hacer trampa de $\frac{1}{e}$ por ronda.

9.3. Schnorr

A diferencia de los anteriores dos protocolos de autenticación, Schnorr se basa en la dificultad de resolver el problema del logaritmo discreto. Tiene también una estructura de 3 pasos y es la siguiente:

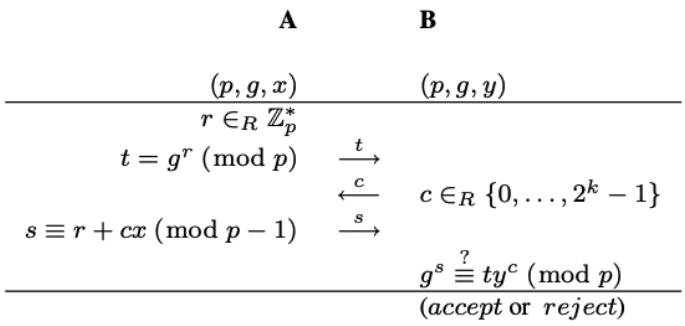

Figura 8 Ronda de Schnorr [4]

En la configuración inicial de este protocolo, una entidad de confianza que ejerce el rol de centro de autenticación de claves publica un número primo *p* grande y un generador *g* de Z_p^*. A, el probador, recibe por parte de esta entidad una clave privada *x*, una clave pública $y = g^x \ (mod\ p)$ y un certificado.

Los pasos de cada ronda del protocolo son: inicialmente A selecciona un número aleatorio $r \ \in Z_p^*$ y le envía a B $g^r \ (mod\ p)$. En el segundo paso B elige un número al azar *c* entre $0 \ y \ 2^k - 1$. El valor de *k* representa un parámetro de seguridad. El número *c* es recibido por A, quien en el tercer paso computa $s = r + c.x \ (mod\ p - 1)$ y se lo envía a B. Finalmente B debe aceptar o no la respuesta, verificando que $g^s = ty^c \ (mod\ p)$.

La propiedad de *completeness* se cumple ya que $g^s = g^r g^{cx} = ty^c \ (mod\ p)$, aceptando con una probabilidad de 1 un verificador honesto frente a un probador honesto. Para demostrar que también cumple con la propiedad de *soundness* vamos a recurrir a la misma técnica que en los otros protocolos, es decir probar por el absurdo. Supongamos que un probador deshonesto (que no conoce el valor de la clave privada *x*) se puede preparar para dos *challenges* distintos c_1 y c_2. Entonces se obtiene por funcionamiento del protocolo que $g^{s_1} = ty^{c_1} \ (mod\ p)$ y $g^{s_2} = ty^{c_2} \ (mod\ p)$. Dividiendo ambos se obtiene que $g^{s_1 - s_2} = y^{c_1 - c_2} \ (mod\ p) = g^{x\,(c_1 - c_2)} \ (mod\ p)$ y por lo tanto que $x = (s_1 - s_2).(c_1 - c_2)^{-1} \ (mod\ p - 1)$. Es decir, sería posible calcular el logaritmo discreto, pero este problema se asume intratable (para valores adecuados de *p* y *g*), por lo tanto es un absurdo y un probador deshonesto solo tiene una probabilidad de hacer trampa igual a 2^{-k}.

La propiedad de *zero-knowledge* de este protocolo no se puede demostrar en el sentido estricto, ya que no se conoce cómo generar un simulador para los valores de *t* y

s que envía el probador [4]. Una ventaja que ofrece este protocolo es que permite el pre-computo de los valores de *t*, permitiendo que un probador pueda calcularlos de manera *offline* para usarlos posteriormente en una ejecución del protocolo.

9.4. Análisis de seguridad de protocolos de autenticación ZKP

En esta sección se estudian qué propiedades y debilidades presentan los esquemas de autenticación ZKP. Una de las primeras consideraciones que se pueden hacer es que se puede diferenciar entre protocolos “basados en” ZKP o ZKP “puros”. La diferencia radica en que ciertos protocolos no cumplen formalmente con la propiedad de *soundness* o *zero-knowledge* en la práctica, principalmente debido a métodos de implementación y selección de parámetros inadecuada o que prioriza la eficiencia sobre la seguridad. Por ese motivo, es mejor considerarlos basados en ZKP ya que no se puede probar que garantizan las propiedades requeridas por ZKP en un sentido estricto, aunque usan varios de los conceptos de estos protocolos.

Unos de los ataques más conocidos a los métodos de autenticación es el de *replay*. Este consiste en reutilizar una comunicación previa exitosa entre dos partes, perteneciente a una ejecución de un protocolo de autenticación, con el fin de suplantar la identidad de la parte que se autenticó mediante el reenvío de ciertos mensajes intercambiados. Como se menciona en [10], muchos protocolos de autenticación son vulnerables en algunos escenarios al ataque de *replay,* entre ellos Kerberos, un popular protocolo del género TTP (*Third Trusted Party*). ZKP es resistente a este ataque, ya que cada ejecución del protocolo es distinta debido a la aleatorización utilizada y no brinda ninguna información a un tercero escuchando la conversación. De hecho, ZKP (para el caso de la versión interactiva) posee la propiedad de ser negable a terceras partes que no participan de la prueba debido a su naturaleza de cero conocimiento y que se utilicen *challenges* distintos en la comunicación. Para que un probador pueda negar una prueba de conocimiento cero a un tercero solo necesita argumentar que los mensajes que envío no fueron emitidos por él, ya que pueden ser simulados por un simulador de manera indistinguible (propiedad de *zero-knowledge*). Esta propiedad no se conserva, por

diseño, en ZKP no interactivo ya que el probador no puede negar una prueba una vez que la hace pública.

Debido a la naturaleza de cero conocimiento, la seguridad de los protocolos de autenticación de ZKP no se degrada con el uso, evita que un tercero pueda obtener alguna información escuchando una comunicación (*eavesdropping*) y además es resistente al ataque de *chosen-text*. Este último ataque generalmente se utiliza en los protocolos de esquemas de autenticación de *challenge-response*, en donde se escoge un *challenge* específico para extraer información acerca del secreto del probador. ZKP evita que un verificador deshonesto sea capaz de obtener alguna información escogiendo *challenges*, ya que siempre la respuesta del probador contiene aleatoriedad. Un problema relacionado con el ataque de *chosen-text* es el del uso de la misma *key* para más de un propósito. En tal escenario es posible que un verificador deshonesto pueda obtener un efecto no esperado para el probador, como la firma digital de un archivo mediante la simple ejecución de un protocolo *challenge-response*. En [2] se ilustra los pasos de este posible ataque: dado un esquema de autenticación en donde B envía a A un número aleatorio encriptado con la clave pública de A y A debe responder con el número desencriptado, al reutilizar la misma *key* RSA tanto para autenticación como para firma digital, si B desafía a A con un número $r_B = h(x)$ siendo $h(x)$ la aplicación de una función de hash a un mensaje *x,* la respuesta de A va a proveer a B de la firma digital RSA del mensaje *x*, desconocido por A. ZKP también evita estos efectos colaterales.

Otro popular ataque es el conocido *man-in-the-middle*, donde un atacante intercepta una comunicación, estableciéndose como intermediario y pudiendo leer y modificar todos los mensajes que se envían dos partes. ZKP es inmune a este ataque debido a que el secreto nunca se envía por el canal de comunicación. También protege contra el posible ataque de *impersonation* (hacerse pasar por otro) que es posible realizar por parte del verificador cuando se usan esquemas débiles de autenticación mediante la utilización de *passwords*. En ZKP nunca se revela el secreto ni ninguna información referida a este al verificador.

Un aspecto también muy interesante para evaluar es la protección que ofrece ZKP frente al *phishing*. Este tipo de ataque perteneciente a la rama de la llamada ingeniería social es muy popular en la actualidad y consiste básicamente en engañar a un usuario

para que revele sus credenciales, utilizando diferentes técnicas como mails fraudulentos, clones maliciosos de sitios web completos, entre otras. En [11] y [10] se proponen dos protocolos de autenticación ZKP que ofrecen protección completa frente al *phishing*. Ambos protocolos permiten que un usuario se autentique en un servicio sin necesidad de tener que revelar su contraseña en cada sitio web, representando una gran ventaja frente a otros esquemas utilizados en la actualidad.

En conclusión, se puede evidenciar que utilizar protocolos de autenticación ZKP representa una de las opciones más seguras en comparación a esquemas tradicionales (autenticación débil mediante contraseñas y autenticación fuerte de *challenge-response*) debido a que es resistente a muchos ataques conocidos. Las posibles debilidades de los protocolos ZKP por lo general se deben a deficiencias en las implementaciones, ya que las bases en las que se sustentan son muy sólidas y robustas. No obstante, el auge de la computación cuántica puede poner en riesgo la seguridad de estos protocolos, si es que esta permite la resolución de problemas hoy intratables en la práctica (como la factorización y logaritmo discreto) sobre los que se basan muchos protocolos ZKP. Es por eso que existen propuestas de protocolos ZKP basados en primitivas algebraicas no conmutativas como se detallan en la próxima sección.

10. Protocolos ZKP basados en álgebra no conmutativa

Lo interesante de la utilización de primitivas basadas en álgebra no conmutativa es que no se conocen aún ataques basados en algoritmos cuánticos o algoritmos sub-exponenciales que puedan comprometer la seguridad. Los algoritmos cuánticos conocidos hasta la fecha no parecen tener ventajas de aceleración del cómputo en todos los problemas difíciles computacionales. El algoritmo de Shor permite por ejemplo la resolución eficiente del problema de factorización en un computador cuántico (en teoría permite el cómputo en tiempo polinomial pero todavía no se han desarrollado máquinas con los suficientes *qbits* como para poner en riesgo y poder romper la seguridad de RSA-1024, que es usado en la práctica). No obstante, no se conocen métodos eficientes

ejecutables en una computadora cuántica para la resolución de problemas relacionados con Lattice, también llamados LWE (*Learning with errors*), problemas de decodificación LPN (*Learning parity with noise*) y primitivas de clave simétrica como funciones hash [12]. Es decir, la creencia actual es que hay ciertos problemas que siguen siendo aún difíciles de resolver en computadores cuánticos. A las primitivas criptográficas que se basan en estos problemas se las denomina *quantum-resistant*, resistentes a lo cuántico. En [13] se presenta un protocolo interactivo de autenticación basado en ZKP que utiliza matrices de Hill como estructura. Lo interesante del modelo es que es un protocolo compacto (al usar exclusivamente aritmética de precisión simple puede ser utilizado en plataformas de bajo porte de 8 y 16 bits) y su seguridad criptográfica es de alrededor de 64 bits (que incluso puede ser incrementado). A continuación, se presenta un resumen del funcionamiento del protocolo presentado en [13] por el Dr. Pedro Hecht:

El módulo utilizado es el número primo 251, que es el número primo más grande que puede ser representado en 1 byte, y el orden de las matrices es 8. Es decir, se usa el grupo lineal no conmutativo $GL(8, Z_{251})$, al que se lo denomina M_8. Se utiliza una operación basada en el problema GSDP (*Generalized Symmetric Decomposition problem*) como función de una vía trampa (*one-way trap*). Dicho problema consiste en lo siguiente:

$$Sea\ (x,y) \in G \times G, S \subset G, (m,n) \in Z,\ \ Encontrar\ z \in S\ tal\ que\ y = z^m\ x\ z^n$$

$$G\ es\ un\ grupo\ no\ conmutativo\ y\ S\ es\ un\ subgrupo\ abeliano$$

El único ataque conocido hasta el momento para resolver GSDP es la *fuerza bruta*, es decir la exploración exhaustiva de todo el conjunto posible de soluciones.

La configuración inicial del protocolo consiste en la generación de dos matrices P y G. P es una matriz no singular $\in M_8$ seleccionada al azar y va a ser utilizada como matriz de vectores propios y generador del grupo conmutativo P_8. G $\in M_8$ es una matriz singular, también elegida al azar junto con otros dos enteros aleatorios $(m,n) \in [2,z]$, $z = 64$, cuyo propósito es ser la base para obtener las claves públicas. Toda la

información mencionada hasta aquí es pública. Tanto el probador como el verificador generan su clave privada utilizando el siguiente algoritmo:

1 – Generar n^2 números aleatorios $(mod\ p)$ hasta obtener una matriz de $n \times n$ no singular de autovectores (P) e invertirla (P^{-1}).

2 – Generar al azar n números diferentes a cero y entre sí $(mod\ p)$ obteniendo una matriz diagonal cuadrada de orden n de autovectores.

3 – Generar una matriz A que sea instancia de un subgrupo conmutativo generado por P, usando $A = P\ D\ P^{-1}$.

Repetir los pasos 2 y 3 para obtener nuevos miembros del subgrupo P.

Figura 9 Algoritmo para generar matrices de Hill aleatorias n x n y de módulo p. [13]

Es decir, las claves privadas de ambos participantes son matrices aleatorias de P_8. Una vez computadas mediante la ejecución del algoritmo de la Figura 9 se calculan las correspondientes claves públicas de la siguiente manera: $G_A = A^m\ G\ A^n$, siendo A la clave privada.

El protocolo luego consiste en rondas de 3 pasos cada una. En el primer paso, el probador elige un número aleatorio k en $[2, z]$ diferente a m y n. Luego calcula $S = A^k\ G_B\ A^{-m}$ y se lo envía al verificador (G_B es la clave pública del verificador). En el segundo paso el verificador decide de manera aleatoria entre enviar uno de los dos siguientes desafíos: generar una matriz H aleatoria $\in M_8$ y luego enviar $Q = B^m\ H\ B^n$ (B es la clave privada del verificador) o directamente enviar $Q = B^m\ SG_A\ B^n$. En el último paso el probador debe calcular $R = S^{-m}\ Q\ S^{-n}$ en caso de haber recibido el primer desafío computado a partir de la matriz aleatoria H o $R = A^{-k}\ Q\ A^{-n}$ en caso contrario. Envía R al verificador quien debe finalmente comprobar que $S^m\ R\ S^n = Q$ o que $B^{-m}\ R\ B^{-n} = G_B G$ respectivamente. Las rondas se repiten r veces.

El protocolo es completo porque dados un probador y un verificador honesto siempre termina satisfactoriamente y la prueba es aceptada con probabilidad igual a 1:

Si el verificador eligió el primer desafío: $S^m R S^n = (S^m S^{-m}) Q (S^{-n} S^n) = Q$

Si el verificador eligió el segundo desafío: $B^{-m} R B^{-n} = B^{-m} A^{-k} Q A^{-n} B^{-n} = B^{-m} A^{-k} B^m S G_A B^n A^{-n} B^{-n} =$

$= B^{-m} A^{-k} B^m A^k \; G_B \; A^{-m} \; A^m \; G \; A^n \; B^n \; A^{-n} \; B^{-n} =$

$= (B^{-m} B^m)(A^{-k} A^k) \; G_B \; (A^{-m} \; A^m) \; G \; (A^n A^{-n})(B^n B^{-n}) =$

$= G_B \; G$

El protocolo cumple también con la propiedad de *soundness*. Si un probador deshonesto no conoce la clave privada lo único que puede hacer es intentar adivinarla o resolver el problema GSDP. Al ser GSDP un problema intratable en la práctica, decide inventar una clave aleatoria ${}^*A \in P_8$. Si el verificador solicita el primer desafío, puede responder siempre con cualquier testigo ${}^*S$ ya que $*S^m R *S^n = (*S^{m*} S^{-m)} \; Q \; (*S^{-n*} S^n) = Q$ y el verificador lo va a aceptar. En cambio, si el probador solicita el segundo desafío, no va funcionar la verificación:

$$B^{-m} R B^{-n} = B^{-m} *A^{-k} Q \; *A^{-n} B^{-n} = B^{-m} *A^{-k} B^m *S G_A \; B^n *A^{-n} B^{-n} =$$

$$= B^{-m} \; *A^{-k} \; B^m \; *A^k \; G_B \; *A^{-m} \; A^m \; G \; A^n \; B^n \; *A^{-n} \; B^{-n} =$$

$$= (B^{-m} B^m)(*A^{-k} \; *A^k) \; G_B \; (*A^{-m} \; A^m \; G \; A^n \; *A^{-n})(B^n B^{-n}) =$$

$$= G_B \; (*A^{-m} \; A^m \; G \; A^n \; *A^{-n}) <> G_B \; G$$

Como se puede observar por lo remarcado en rojo, la intervención de la clave pública impide que la utilización de una clave falsa ${}^*A$ inventada funcione. Como consecuencia, la probabilidad de aceptar una prueba que es falsa en cada ronda es ½ y al repetirse las rondas r veces, la probabilidad de fraude total es $\frac{1}{2}^r$.

Por último, se presenta un simulador de la interacción del verificador con un probador, a modo de demostrar que el protocolo también cumple con la propiedad de *zero-knowledge*.

1 ***Generar testigo***

Generar bit aleatorio b ε_R [0, 1]

*Si b=0 → Generar testigo al azar S=*S ε_R M_8*

Si b=1 → Generar testigo $S=G_B\, G\, G_A^{-1}$

2 ***Recibir desafío***

*Si b=0 → Generar al azar *Q ε_R M_8*

Si b=1 → $Q=B^m S G_A B^n = B^m G_B G (G_A^{-1} G_A) B^n = B^m G_B G B^n$

3 ***Devolver respuesta***

Si b=0 → $*R = *S^{-m}\, *Q\, *S^{-n}$

Si b=1 → R=Q

4 ***Verificar y registrar sesión***

Si b=0 → $*S^m\, *R\, *\, S^n = *Q$

Si b=1 → $B^{-m} R B^{-n} = (B^{-m} B^m)\, G_B G (B^n B^{-n}) = G_B G$

Añadir (S, Q, b, R) al registro de sesión.

5 ***Iterar***

Repetir a voluntad 1. a 4. r-veces.

Figura 10 Simulador del protocolo. b = 0 representa el primer desafío y b = 1 representa el segundo desafío que puede enviar el verificador.ε_R representa la pertenencia y que fue seleccionado al azar [13]

Por lo expuesto en la Figura 10 y del hecho que ninguna información es revelada de lo intercambiado en la interacción por el probador ($S = A^k\, G_B\, A^{-m}$ y $R = A^{-k}\, Q\, A^{-n}$) se puede concluir que el protocolo es *zero-knowledge*.

En los siguientes párrafos se ilustra un ejemplo numérico del funcionamiento del protocolo definido por el Dr. Pedro Hecht, presentado en [14]:

10.1. Configuración inicial

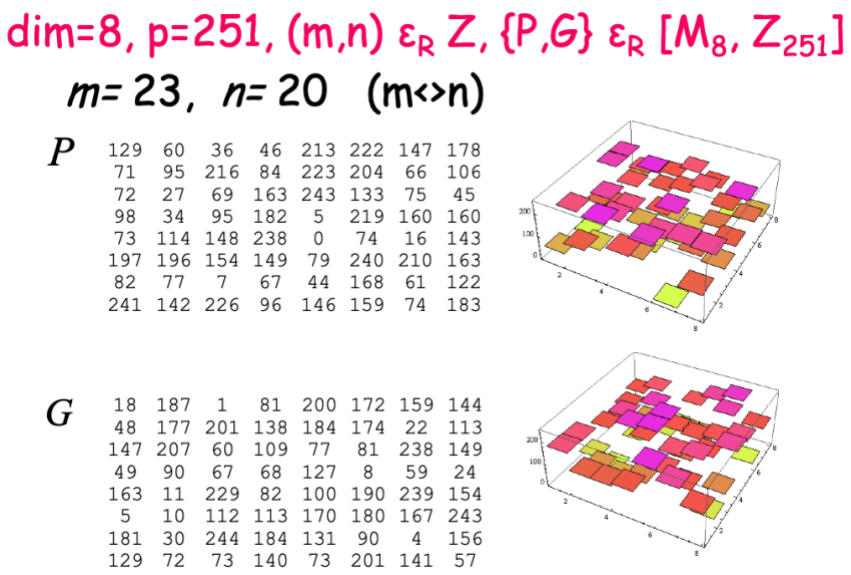

Figura 11 Selección de parámetros iniciales. P y G son las dos matrices aleatorias y m y n dos números enteros aleatorios menores a 64 [14]

10.2. Elección de par de claves del probador

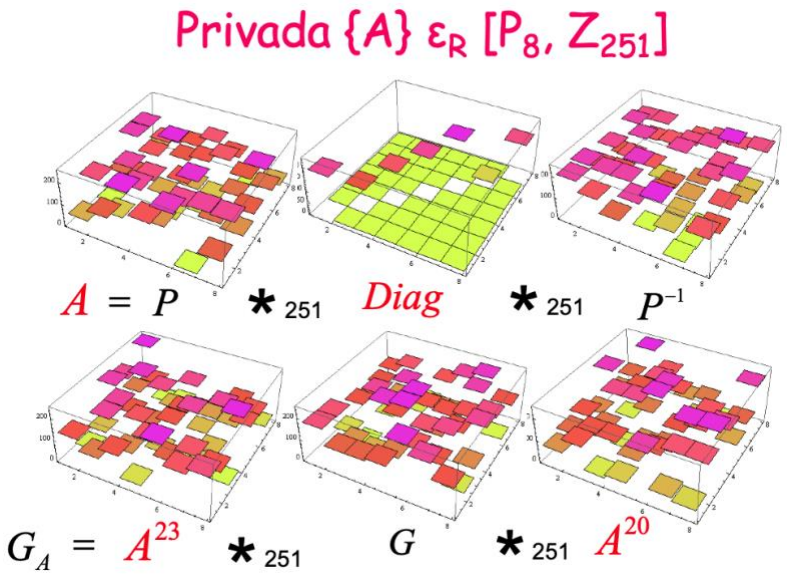

Figura 12 Claves pública y privada del probador. A es la privada y G_A es la pública [14]

10.3. Elección de par de claves del verificador

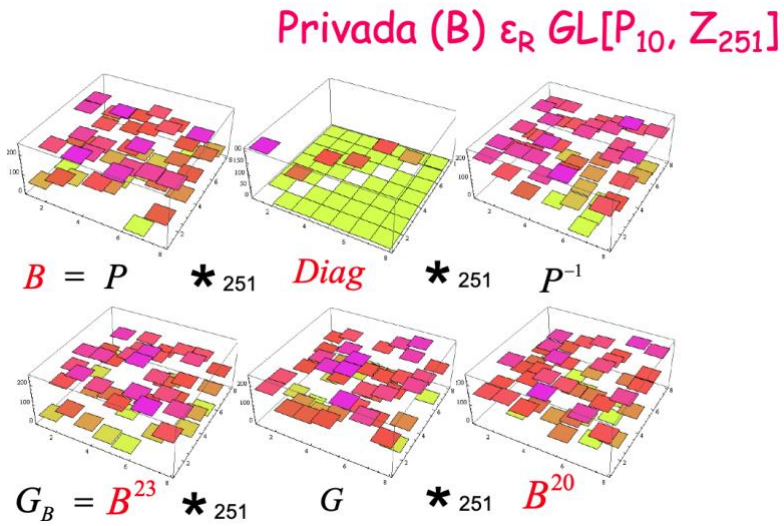

Figura 13 Claves pública y privada del verificador. B es la privada y G_B es la pública [14]

10.4. Paso 1 del protocolo

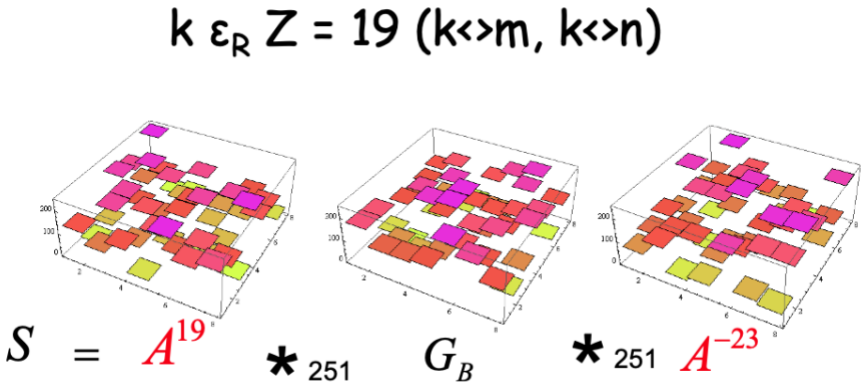

Figura 14 El probador envía el cómputo S al verificador [14]

10.5. Paso 2 del protocolo

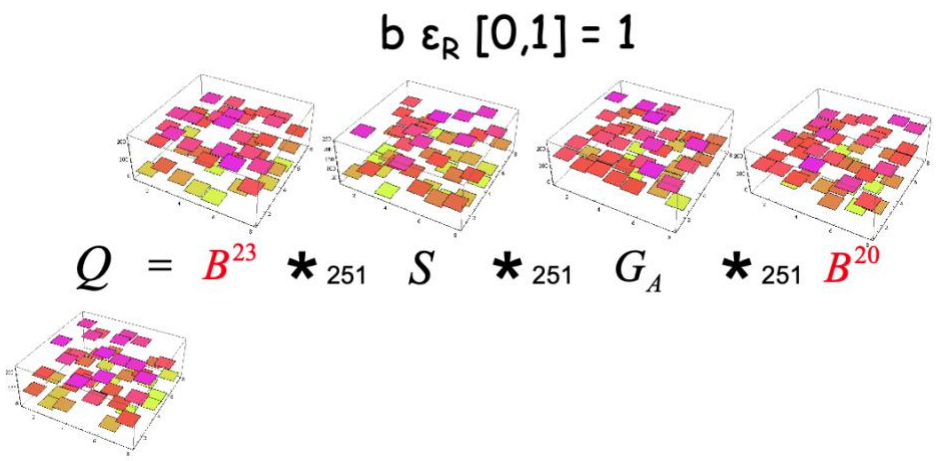

Figura 15 Se muestra el envío del challenge b = 1 calculador por el verificador. El cómputo Q es enviado al probador [14]

10.6. Paso 3 del protocolo

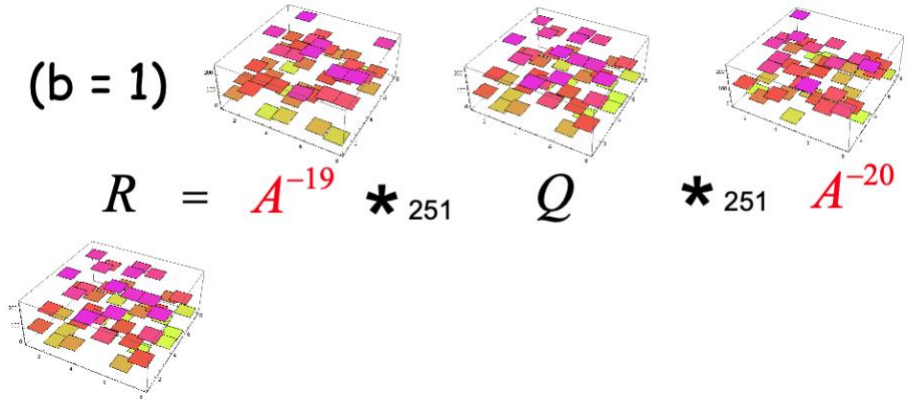

Figura 16 Respuesta del probador cuando el challenge solicitado es el b=1 [14]

10.7. Verificación de la respuesta

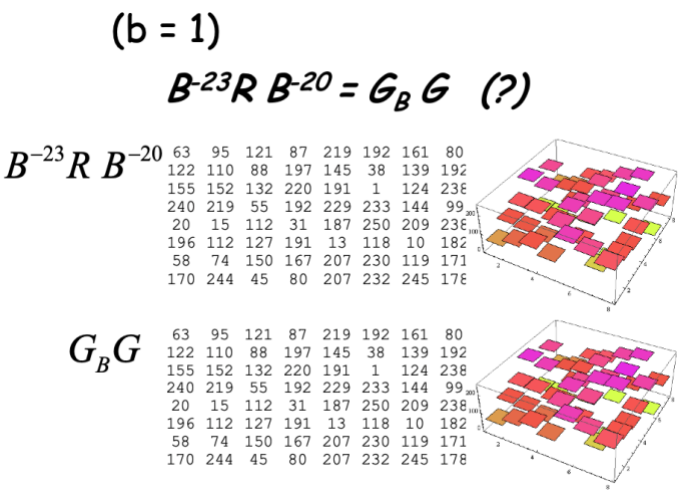

Figura 17 Verificación de la respuesta por parte del verificador cuando b=1 [14]

11. Pruebas de conocimiento no interactivas

Poco después de la presentación del artículo original de Shafi Goldwasser, Silvio Micali y Charles Rackoff sobre los protocolos interactivos de conocimiento cero, se publica en el ACM en 1988 otro artículo, en el cual también participó Micali, denominado “Non-Interactive Zero-Knowledge and Its Applications” [7]. En este artículo, los autores demuestran que la interacción requerida en las pruebas de conocimiento cero puede ser eliminada si el probador y el verificador comparten una corta cadena de texto aleatoria. El resultado da origen a las pruebas de conocimiento cero no interactivas y en particular al primer criptosistema de clave pública que se demostró ser seguro frente al ataque criptoanalítico de cifrado elegido (*chosen ciphertext*), el cual representa el ataque de mayor grado de eficacia. Las pruebas de conocimiento cero no interactivas, además de eliminar la necesidad de la interacción, eliminan también la necesidad del secreto utilizado en la aleatorización de los protocolos, resultando en que la característica esencial de las pruebas de conocimiento cero es la dificultad computacional por sí sola de ciertos problemas.

Hay varios aspectos para remarcar sobre las pruebas de conocimiento cero no interactivas. En primer lugar, el requerimiento de compartir una cadena de texto entre el probador y el verificador es mucho más débil que el requerimiento de la interacción, ya que en este el probador y el verificador pueden mediante la interacción construir de manera conjunta una cadena aleatoria que compartan, pero lo inverso no es posible [7]. En segundo lugar, el requerimiento de compartir una cadena de texto es incluso más débil que cada parte tenga que utilizar la aleatorización en cada instancia de un protocolo, ya que en este el probador no puede conocer a futuro que números aleatorios van a ser elegidos, pero al compartir una cadena, en la propuesta hecha por Micali, Blum y Feldman, el probador puede conocer por adelantado todas las elecciones que va a hacer un verificador. Por último y relacionado con el concepto anterior, la propiedad de *zero-knowledge* de las pruebas no interactivas no depende en el secreto o la impredecibilidad de la cadena de texto compartida sino en la aleatoriedad de su formación y se puede demostrar que aunque su aleatoriedad no sea buena, las propiedades de *completeness* y *soundness* no se ven afectadas (un verificador no podría aceptar una afirmación falsa), sólo su propiedad de *zero knowledge* [7]. Este resultado es importante porque si se utilizan métodos algorítmicos para generar números aleatorios, la calidad de estos nunca es perfecta sino pseudoaleatoria.

Dentro de las pruebas de conocimiento cero no interactivas se pueden identificar principalmente dos grandes tipos: ZK-SNARK y ZK-STARK [15]. ZK-SNARK es el acrónimo de *Zero-Knowledge Succinct Non-Interactive Argument of Knowledge* y posee las siguientes características: *zero-knowledge* (la misma propiedad que los protocolos interactivos, un verificador sólo obtiene conocimiento de que cierta afirmación es cierta o no y nada más), *succinct* (el tamaño de la prueba es pequeño y puede ser verificado rápidamente por el verificador), non-Interactive (no hay varias rondas de intercambio de mensajes a diferencia del caso de los protocolos), *argument* (la prueba cumple con el requerimiento de *soundness,* con una muy baja probabilidad de aceptar una prueba falsa). Formalmente *succinct* significa que el tiempo de verificación escala poli-logarítmicamente en $|w|$ siendo w el testigo. ZK-STARK es el acrónimo de *Zero-Knowledge Scalable Transparent Argument of Knowledge*. La diferencia con el anterior son las propiedades de *scalable* y *transparent*. *Scalable* significa que el tiempo necesario

para generar la prueba crece cuasi linealmente en $|w|$ y que al mismo tiempo el tiempo de verificación crece poli-logarítmicamente en $|w|$. *Transparent* significa que todos los mensajes del verificador son lanzamientos de moneda aleatorios públicos, en el sentido de que los parámetros para la configuración inicial son generados aleatoriamente, públicamente y son verificables, otorgando más transparencia al proceso.

Los ZK-STARKs son considerados más confiables debido a su transparencia al no requerir una configuración inicial que dependa de la confianza en otros participantes. No obstante, las pruebas generadas por estos son un poco más grandes que los ZK-SNARKs y requieren por lo tanto de mayor tiempo de verificación.

11.1. Protocolos interactivos a pruebas no interactivas

En el esquema visto que presentan en general los protocolos interactivos de conocimiento cero (Figura 4) se puede ver que el verificador cumple más bien un rol pasivo y que participa una única vez en la comunicación de cada ronda. No obstante, dicha participación es importante porque, como se ha visto, de la aleatoriedad de su elección depende la posibilidad o no de que un probador deshonesto pueda probarle algo que no es cierto. Amos Fiat y Adi Shamir propusieron un procedimiento, al que se denomina heurística de Fiat y Shamir y que permite transformar cualquier protocolo interactivo en una prueba no-interactiva. La idea de este procedimiento es que el probador también se encargue de lo que hace el verificador en la versión interactiva. Formalmente el rol del verificador se considera equivalente al de una función aleatoria, por lo que puede ser reemplazado por una función de hash. El esquema original fue presentado por Fiat y Shamir en [16] y consiste en lo siguiente:

Firmar el mensaje

1. A elige de manera aleatoria $r_1, \dots, r_t \in [0, n)$ y calcula $x_i = r_i^2 (mod\ n)$.
2. A computa $f(m, x_1, \dots, x_t)$ y utiliza los primeros $k.t$ bits como valores $e_{i,j}$ $(1 \leq i \leq t, 1 \leq j \leq k)$.
3. A computa $y_i = r_i \prod_{e_{i,j}=1} s_j (mod\ n)$ para $i = 1, \dots, t$
4. A envía I,m, la matriz $e_{i,j}$y todos los y_i al verificador.

Verificación de la firma

1. B computa $v_j = f(I, j)$ para todo $j = 1, \dots, k$.
2. B computa $z_i = y_i^2 \prod_{e_{i,j}=1} v_j (mod\ n)$ para $i = 1, \dots, t$.
3. B verifica que los primeros $k.t$ bits de $f(m, z_1, \dots, z_t)$ son $e_{i,j}$.

Figura 18 Esquema de firma digital basado en Fiat-Shamir [16]

Como se observa, el sistema propuesto es una adaptación no interactiva de la versión de autenticación de Fiat-Shamir donde se utilizan los residuos cuadráticos como problema computacional. La función f es una función segura de hash, k es un entero que representa un parámetro de seguridad, v_j son residuos cuadráticos módulo n, s_j son las raíces cuadradas más pequeñas de v_j^{-1} $(mod\ n)$, I es una cadena de texto con toda la información de identificación y $e_{i,j}$ representa el *challenge* aleatorio.

La heurística se puede resumir en que el probador debe realizar todos los siguientes pasos: generar un mensaje aleatorio de compromiso, tomar eso como entrada para calcular un desafío de manera similar a lo que haría un verificador, y por último calcular la respuesta el desafío y publicarla. Lo interesante de este proceso es que si el protocolo ZKP interactivo del que se parte es de identificación (como en el caso visto de la propuesta original de Fiat-Shamir), la versión no interactiva resultante puede utilizarse como firma digital, por lo tanto, otra de las aplicaciones prácticas de ZKP es su utilidad como firma digital.

12. Uso de ZKP para firma digital

Como se ha mencionado, es posible adaptar los protocolos vistos de autenticación de identidad basados en ZKP para usarlos como esquemas de firma digital. El *challenge e* que envía el verificador hacia el probador se puede modificar por la aplicación de una función de hash al *witness x* concatenado a un mensaje *m* que se quiere firmar, por ejemplo $e = h(x||m)$. De esta forma el mensaje *m* es firmado digitalmente con el par *(x,y)* siendo *y* la respuesta del probador, actuando *h* como el verificador. La firma digital se puede validar siguiendo los pasos del protocolo del ZKP en cuestión. A continuación, se muestra como ejemplo una adaptación del protocolo de autenticación de Schnorr a la versión no interactiva.

12.1. Firma de Schnorr

Usando la heurística de Fiat-Shamir, la versión no interactiva del protocolo de Shcnorr puede ser utilizada como firma digital y es conocida como Firma de Schnorr, que forma la base de otros sistemas de firma digitales más complejos utilizados en la práctica. El funcionamiento es el siguiente (adaptación de [17]):

Probador

1. Generar un número aleatorio $\alpha \in_R Z_p$ y calcular $t = g^{\alpha} \ (mod\ p)$.
2. Computar el desafío usando una función de hash y un mensaje m que se quiere firmar, $c = H(m, t)$.
3. Definir la respuesta $r = \alpha - c * k$.
4. Publicar el par (c, r).

Verificación

1. Calcular el desafío $c' = H(m, g^r \ (mod\ p) + c * K)$.
2. Si $c = c'$ la firma es satisfactoria.

Figura 19 Firma de Schnorr adaptada de [17]

En el esquema superior, m es el mensaje a firmar, g es el generador utilizado y k y K son las claves privada y pública respectivamente del probador que firma el mensaje.

El sistema funciona de manera correcta porque:

$$g^r \ (mod\ p) = g^{\alpha - c*k} \ (mod\ p)$$
$$g^r \ (mod\ p) = t - c * K$$
$$t = \ g^r \ (mod\ p) + c * K$$
$$H(m, t) = \ H(m, g^r \ (mod\ p) + c * K)$$
$$c = \ c'$$

13. Blockchain, criptomonedas y ZKP

Uno de las más grandes e importantes aplicaciones de la ZKP en la práctica es en las tecnologías de *blockchain* y criptomonedas. Se utilizan principalmente las versiones no interactivas de pruebas de conocimiento cero, generalmente para lograr la privacidad de transacciones. En el presente trabajo se analizan dos casos en particular: Zcash y Monero. Zcash es una criptodivisa que está basada en el código de Bitcoin, otra popular criptodivisa. Su principal característica que lo diferencia de Bitcoin es la privacidad, la cual es resultado de la utilización de una forma especial de ZKP, los ya presentados zk-SNARKs. De hecho, Zcash es considerada como una de las primeras aplicaciones generalizadas de zk-SNARKs y permite que todas las transacciones públicas de la *blockchain* de Zcash estén encriptadas, pero a su vez puedan ser validadas bajo el consenso de la red. Monero es otra criptodivisa muy enfocada en la privacidad de las transacciones. Monero firma las transacciones con una técnica llamada *multisignature*, donde se necesita de varios participantes para firmar y se oculta la identidad de los firmantes.

Monero utiliza *ring signatures* para lograr confidencialidad en sus transacciones. Las *ring signatures* o firmas de anillo en español, consisten en un anillo compuesto de un conjunto de claves públicas, en la cual una de ellas corresponde al firmante y el resto no están relacionadas, y una firma, generada con ese anillo de claves y que cualquiera que la verifica no puede determinar qué miembro de ese conjunto es el firmante. En [17]

se presenta en profundidad los detalles técnicos del funcionamiento de esta criptomoneda. Como se menciona en dicha publicación, el sistema de firmas del protocolo de Monero debe cumplir con tres principales propiedades: *signer ambiguity*, *linkability* y *unforgeability*. La primera propiedad establece la ambigüedad de una firma, en el sentido que se pueda probar que el firmante pertenece a un grupo, pero sin revelar qué miembro de este es. Esto se utiliza para ofuscar los orígenes de los fondos en cada transacción. El segundo permite que dos mensajes diferentes que fueron firmados por la misma clave privada estén relacionados. Esto previene el *double-spending*, es decir, que no se pueda gastar más de una vez la misma moneda. El último establece que ningún atacante puede falsificar una firma, previniendo el robo de fondos de Monero por aquellos que no poseen la clave privada apropiada. El modelo que utiliza Monero para cumplir con estas propiedades está basado en ML-SAG (*Multilayer Linkable Spontaneous Anonymous Group*). Es un sistema de firmas basadas en la firma de Schnorr antes descripta y consiste brevemente en lo siguiente: Se tiene un conjunto de $n \,.\, m$ claves. $R = \{K_{i,j}\}$ $(claves\ p\acute{u}blicas)$, $i \in \{1,2,\dots,n\}$ y $j \in \{1,2,\dots,m\}$. Se conocen las m claves privadas $\{k_{\pi,j}\}$ correspondientes al subconjunto $\{K_{\pi,j}\}$ para algún índice $i = \pi$. Se asume la existencia de dos funciones de hash H_n y H_p, que mapean a enteros en el rango $0 - l$ y a puntos de una curva elíptica respectivamente, siendo l el módulo utilizado. M es el mensaje por firmar y G es el generador utilizado. La notación $\alpha_1 G$ significa G^{α_1}. El firmante realiza los siguientes pasos:

1. Calcular $K'_j = k_{\pi,j} H_p(K_{\pi,j}),\ \forall j \in \{1,2,\dots,m\}$
2. Generar números aleatorios $\alpha_j \in_R Z_l$ y $r_{i,j} \in_R Z_l, \forall i \in \{1,2,\dots,n\}$ y $j \in \{1,2,\dots,m\}$ $(excepto\ i = \pi)$
3. Calcular $c_{\pi+1} = H_n(M, [\alpha_1 G], [\alpha_1 H_p(K_{\pi,1})], \dots, [\alpha_m G], [\alpha_m H_p(K_{\pi,m})])$
4. Para $i = \pi+1, \pi+2, \dots, n, 1, 2, \dots, \pi-1$ calcular, reemplazando $n+1$ por 1:

$$c_{i+1} = H_n(M, [r_{i,1}G + c_i K_{i,1}], [r_{i,1}H_p(K_{i,1}) + c_i K'_1], \dots, [r_{i,m}G + c_i K_{i,m}], [r_{i,m}H_p(K_{i,m}) + c_i K'_m])$$

5. Definir $r_{\pi,j} = \alpha_j - c_\pi k_{\pi,j} (mod\ l)$

   La firma resultante es: $\sigma(M) = (c_1, r_{1,1}, \dots, r_{1,m}, \dots, r_{n,1}, \dots, r_{n,m})$ con las imágenes de claves $(K'_1, \dots, K'_m)$.

Figura 20 Construcción de una firma en un sistema ML-SAG [17]

Para verificar la firma del mensaje hay que realizar los siguientes pasos:

1. $\forall j \in \{1, \dots, m\}$ verificar si $lK'_j = 0$
2. Para $i = 1, \dots, n$ calcular, reemplazando $n+1$ por 1:

$$c'_{i+1} = H_n(M, [r_{i,1}G + c_i K_{i,1}], [r_{i,1}H_p(K_{i,1}) + c_i K'_1], \dots, [r_{i,m}G + c_i K_{i,m}], [r_{i,m}H_p(K_{i,m}) + c_i K'_m])$$

3. Si $c_1 = c'_1$ entonces la firma es válida.

Figura 21 Verificación de firma en un sistema ML-SAG [17]

El proceso de validación de la firma es correcto debido a que si i es distinto de π entonces los c'_{i+1} se calculan de la misma forma que en la construcción de la firma. Para el caso de que i sea igual a π, $r_{\pi,j} = \alpha_j - c_\pi k_{\pi,j} (mod\ l)$ cierra el ciclo, por lo que $r_{\pi,j}G + c_\pi K_{\pi,j} = (\alpha_j - c_\pi k_{\pi,j})G + c_\pi K_{\pi,j} = \alpha_j G$ y $r_{\pi,j}H_p(K_{\pi,j}) + c_\pi K'_j = (\alpha_j - c_\pi k_{\pi,j})H_p(K_{\pi,j}) + c_\pi K'_j = \alpha_j H_p(K_{\pi,j})$. Es decir $c'_{\pi+1} = c_{\pi+1}$. Si alguna clave privada $k_{\pi,j}$

es utilizada nuevamente para hacer cualquier otra firma, K_j' que es entregado con la firma permite revelarlo.

Monero además utiliza los conceptos de ZKP en otro aspecto crucial de su privacidad: el ocultamiento de los montos transferidos (*amount hiding*). La mayoría de las criptodivisas comunican esta información en texto plano. Monero en cambio utiliza *commitment schemes* y *range proofs* para ocultar esta información. Un *commitment scheme* es una primitiva criptográfica que consiste en un protocolo interactivo de dos etapas entre dos partes denominadas emisor y receptor. Ambas partes son consideradas polinomialmente probabilísticas. La primera etapa corresponde al *commit* de un mensaje *m* y que sería al equivalente de cerrar una caja con un valor adentro. La segunda etapa consiste en el *reveal* del mensaje *m* al receptor, correspondiente al revelar lo que hay en la caja. Formalmente un *commitment scheme* tiene que cumplir con dos propiedades: *hiding* y *binding*. *Hiding* significa que un receptor deshonesto no puede obtener ninguna información del mensaje *m*, durante la etapa de *commit*. *Binding* significa que un emisor deshonesto no pueda revelar dos mensajes distintos después de la etapa de *commit*, es decir que no pueda seleccionar un mensaje *m* en la etapa de *commit* y luego revelar otro mensaje *m´* en la etapa de *reveal*. En particular los montos de las transacciones en Monero son almacenados como *Pedersen commitments*. Estos son *commitments* que además poseen la propiedad adicional de ser aditivamente homomórficos (*additively homomorphic*). Esta es una propiedad muy interesante en criptografía que establece que lo siguiente es válido: Si $C(a)$ y $C(b)$ son dos *commitments* para los valores a y b, entonces $C(a+b) = C(a) + C(b)$. Esto es muy útil y es lo que usa Monero para demostrar que las entradas igualan a las salidas, sin necesidad de revelar los montos en cuestión. No obstante, existe un problema con esta adición que es si por ejemplo se tiene $C(a), C(b), C(c)$ y $C(d)$ y se quiere demostrar que $(a + b) - (c + d) = 0$, esa ecuación puede incluso ser cumplida si uno de los valores es negativo. Para contrarrestar este problema, Monero hace uso de *range proofs* para probar que cada monto está en un rango determinado (en particular de 0 a $2^{64} - 1$). *Range proofs* son un tipo de pruebas de conocimiento cero que permite probar que un número está en un determinado rango sin necesidad de revelar ese número. Existen diferentes métodos para construir estas pruebas. En [18] se hace un estudio de estos en donde se identifican principalmente

cuatro grandes familias: *square decomposition, signature-based, multi-base decomposition* y Bulletproofs. Las técnicas de *square decomposition* consisten en descomponer el secreto en una suma de raíces cuadradas. *Signature-based* consiste en probar en *zero-knowledge* que se conoce la firma de un secreto, previamente habiendo firmado todos los elementos del intervalo al que se desea probar que pertenece el secreto, por lo que si el probador le demuestra al verificador que conoce una firma que pertenece al conjunto, quiere decir que el secreto pertenece al intervalo [18]. *Multi-base decomposition* es una estrategia un poco más compleja en donde el secreto se descompone en su representación en bits, y luego el probador prueba que cada bit pertenece al intervalo 0-1 mediante aritmética booleana y que la representación es correcta. Esto último se puede hacer por ejemplo con los *commitment schemes* homomórficos vistos. Finalmente, el último de todos (y el más complejo) es Bulletproofs y es en el que se basa Monero. A diferencia de los anteriores, no requiere de un *trusted setup* (inicialización segura). También descompone el secreto en su representación en bits, pero para generar las pruebas utiliza relaciones de productos internos que resultan en pruebas mucho más pequeñas en tamaño. El lector interesado en conocer en profundidad el detalle de funcionamiento de este tipo de pruebas, puede consultarlo en [19], el paper donde los autores creadores de Bulletproofs explican su propuesta.

El sitio web oficial de Zcash [20] posee información sobre el funcionamiento de zk-SNARKs en este protocolo específico. Resumidamente, el proceso consta de diferentes etapas:

Computation → Arithmetic Circuit → R1CS → QAP → zk-SNARK

Figura 22 Etapas de construcción de zk-SNARK en Zcash [20]

Lo primero que hay que realizar es transformar la función de validación de transacciones en una representación matemática. La representación matemática es el *arithmetic circuit* o circuito aritmético, que consiste en una serie de pasos u operaciones pequeñas básicas, similar a las operaciones lógicas de un circuito booleano, pero en este caso son adiciones, substracciones y multiplicaciones. Una vez que se obtuvo este circuito se genera un R1CS (*Rank 1 Constraint System*) que es un sistema para verificar que el circuito está haciendo lo esperado. Al ser muchos caminos del circuito los que se

deben verificar, se realiza un siguiente paso más que es el de QAP (*Quadratic Arithmetic Program*), que es una técnica que lo que hace es concentrar todas estas verificaciones en una única equivalente. Además, la verificación que hay que realizar es entre polinomios y no números. Los polinomios son grandes pero solo es necesario verificar que dos de ellos son equivalentes en un punto aleatorio elegido (debido a que si los polinomios no concuerdan, la verificación falla en casi todos los puntos). Esta es la forma de obtener una prueba no interactiva corta. Para que además sea de cero conocimiento, se utilizan “corrimientos aleatorios” de los polinomios originales.

Cuando se hace un pago en Zcash, se publica un *commitment* que básicamente consiste en un hash de a quien se le envió el dinero, el monto enviado, un número *rho* que es único a cada pago y un número aleatorio. Cuando una transacción es utilizada, el que envía el dinero utiliza una clave para publicar un *nullifier*, que es un hash del número secreto *rho* de un *commitment* que no ha sido utilizado. Adicionalmente presenta una prueba de conocimiento cero demostrando que están autorizados a gastarla. Se verifica que este hash no esté presente en el conjunto de *nullifiers* que traza las transacciones gastadas que es mantenida por cada nodo en la *blockchain*.

En conclusión, se puede evidenciar que ZKP se usa fuertemente en muchos aspectos de las criptomonedas que están enfocadas en la privacidad. Monero y Zcash no son las únicas que lo utilizan, muchas otras criptodivisas también han adoptado e incorporado esta tecnología en sus protocolos, entre otras Ethereum [15].

14. Voto electrónico

Las pruebas de conocimiento cero también tienen su utilidad para implementar sistemas seguros de votos electrónicos, como se puede observar en las propuestas [21] y [22] , donde se hacen uso de esta tecnología.

En [21] se presenta un protocolo de voto electrónico basado en las firmas grupales. Las firmas grupales son similares a las firmas de anillo ya vistas pero difieren en que debe existir una autoridad que emita identidades para los miembros del grupo,

autorizar a otros nuevos miembros a formar parte del grupo e incluso revocar miembros. La utilización de firmas grupales permite que una persona pueda demostrar que forma parte de un grupo sin necesidad de revelar su identidad ya que, al ser votos anónimos, se debe garantizar que no se entreguen votos falsos ni que una persona vote más de una vez. La propuesta de [22] consiste en la utilización de firmas digitales y pruebas de conocimiento cero que permiten demostrar que se conoce la clave secreta sin necesidad de revelarla.

15. ZKP y teoría de la complejidad

Además de tener aplicación en la criptografía, ZKP generó una fuerte conexión entre esta y el estudio de la teoría de la complejidad computacional hasta el punto de ser un objeto de estudio en sí. En esta sección se analizan algunos resultados y teoremas que han sido demostrados en relación con clases de complejidad computacional y protocolos de conocimiento cero.

En principio, trivialmente, todos los lenguajes que pertenecen a la clase de complejidad BPP (*Bounded-error Probabilistic Polynomial time*) tienen protocolos de conocimiento cero, en particular, perfectos. La clase BPP agrupa todos los lenguajes para los cuales existe un algoritmo probabilístico de tiempo polinomial que permite computar la pertenencia de una entrada a los mismos con un bajo margen de error. Hay evidencia para suponer que P=BPP (P es la clase de problemas para los cuales hay algoritmos eficientes polinomiales para resolverlos) pero esto no ha sido demostrado aún. En estos protocolos de conocimiento cero, el probador no tiene que realizar nada, el verificador simplemente puede comprobar computando por su cuenta si una determinada prueba es válida o no, por lo que se cumplen las propiedades de *completeness*, *soundness* y *zero-knowledge* ya que al no haber intercambio de mensajes, una transcripción vacía es trivialmente fácil de simular.

Lo interesante de la propuesta de [1] es la introducción de pruebas de conocimiento cero no triviales en particular para el lenguaje de los residuos cuadráticos y de los no residuos cuadráticos. Dicho lenguaje se encuentra en la clase $NP \cap \text{co-}NP$.

Informalmente, el protocolo presentado para el lenguaje $QR = \{(x,y)|\ y\ es\ un\ residuo\ cuadrático\ módulo\ x\}$ es el siguiente: Sea $m = |x|$, el protocolo se repite m veces. El probador envía en el primer paso al verificador un residuo cuadrático aleatorio módulo x, u. El verificador responde enviando un bit aleatorio. Si el bit es cero entonces el probador envía una raíz cuadrada aleatoria de u módulo x; caso contrario envía una raíz cuadrada aleatoria de $(u.y)\ módulo\ x$. Llamemos w al número que envía el probador en este último paso. Finalmente, el verificador debe asegurarse que $w^2\ mod\ x = u$ (si había enviado el bit 0) o que $w^2\ mod\ x = (u.y)\ mod\ x$ (si el bit enviado había sido un 1). El protocolo presentado representa una prueba de conocimiento cero para el lenguaje QR como se demuestra en [1].

Se puede demostrar que se puede construir una prueba de conocimiento cero para cualquier lenguaje en NP. Para ello solo es necesario presentar una prueba de conocimiento cero para un problema que sea NP-Completo ya que por definición significa que cualquier problema en NP se puede reducir en tiempo polinomial a este. Es decir, ZKP también provee una nueva caracterización de la clase NP. La clase NP representa los problemas para los cuales en las instancias cuya respuesta es un "sí" (recordemos que se usa la versión de decisión de los problemas para el estudio de la complejidad) se puede presentar un certificado que puede ser verificado en tiempo polinomial. Es decir, es la clase de lenguajes que tienen un sistema de pruebas clásico que cumple con las propiedades vistas de *completeness* y *soundness*. Hay fuerte evidencia para suponer que $P \neq NP$ pero esto no ha sido demostrado aún. Básicamente la respuesta a esta incógnita no resuelta aún nos dice si hay alguna ventaja computacional en tener una prueba de un determinado teorema o si un verificador puede computar en el mismo tiempo la validez o no de un teorema sin una prueba o certificado.

En [6] se presenta la prueba de conocimiento cero para el problema del 3-coloreo de un grafo. Este es un problema NP-Completo y consiste en lo siguiente: Sea $G(V,E)$ un grafo, con V el conjunto de vértices de este y E el conjunto de aristas, se dice que G es 3-coloreable si existe un mapeo $\Phi : V \rightarrow \{1,2,3\}$ llamado coloreo tal que para cada par de vértices adyacentes poseen diferentes colores. Es decir, para cada $(u,v) \in E$, $\Phi(u) \neq \Phi(v)$. Se puede notar que el 3-coloreo del grafo genera una partición del conjunto de

vértices en 3 conjuntos independientes. El lenguaje L del 3-coloreo representa el conjunto de grafos que cumplen con esta definición.

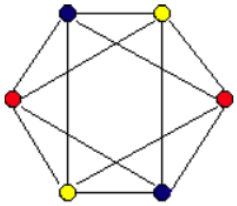

Figura 23 Ejemplo de un 3-coloreo de un grafo [6]

El protocolo de conocimiento cero para este lenguaje consiste en lo siguiente:

Como entrada se tiene un grafo $G(V, E)$. En la primera ronda del protocolo, el probador tiene que asignar de manera aleatoria un color distinto a cada uno de los 3 conjuntos independientes del grafo G, generados a partir de un 3-coloreo apropiado. Luego colorea el grafo G con estos colores elegidos y guarda de manera segura en *n* "cajas" cerradas con una llave estos colores identificando cada caja con el número correspondiente de vértice (*n* representa la cantidad total de vértices). Envía estas cajas sin la llave al verificador. El verificador, en el segundo paso de la ronda, debe elegir de manera aleatoria una arista $e \in E$ del grafo G y se la envía como *challenge* al probador. El probador en el último paso revela los colores de las cajas correspondientes a los vértices $(u, v) = e \in E$, enviándole las llaves correspondientes para poder abrirlas. El verificador para poder aceptar la prueba debe corroborar que las cajas de los vértices *u* y *v* contengan efectivamente colores distintos. Si los colores no son distintos rechaza la prueba, caso contrario sigue la ejecución de otra ronda del protocolo. La cantidad de ejecuciones del protocolo es por lo general m^2 [6], siendo *m* la cantidad de aristas del grafo. El protocolo es completo porque si el probador honesto conoce efectivamente un 3-coloreo apropiado del grafo, entonces cualquier par de vértices u y v, correspondientes a una arista de G van a tener un color diferente, por lo tanto, todas las pruebas van a ser satisfactorias y un verificador honesto va a aceptarlas con probabilidad igual a 1. Si el probador no conoce efectivamente un 3-coloreo apropiado del grafo, es decir si existe al menos alguna arista del grafo que no está coloreada correctamente, entonces la probabilidad de rechazar la prueba es al menos $1/m$ en cada ronda. Por último, para

demostrar que el protocolo no brinda ningún conocimiento sobre el coloreo del grafo, hay que analizar la salida de los mensajes que brinda el probador en una interacción del protocolo. Si llamamos $\pi \in S_3$ a la permutación del grupo simétrico de 3 elementos, el probador en cada ronda solo revela la siguiente información: $\pi(\Phi(u))\ y\ \pi(\Phi(v))$ para dos vértices u y v unidos por una arista de G. Como la permutación fue seleccionada al azar, es fácil construir un simulador cuya salida sea computacionalmente indistinguible de la interacción con el probador real: simplemente el simulador que tiene como salida $(i,j)\ i,j \in \{1,2,3\}\ i \neq j$. De aquí surge la importancia que la permutación sea elegida de manera independiente y aleatoria en cada ronda, ya que en caso contrario si el protocolo se ejecuta m^2 veces es posible para un verificador deshonesto llegar a conocer el coloreo Φ del grafo.

En [23] se demuestra que también se puede construir una prueba de conocimiento cero para el problema del ciclo Hamiltoniano en un grafo, que también es un problema NP-Completo. La diferencia que posee con el protocolo para el problema del 3-coloreo es que el protocolo propuesto por Manuel Blum es un poco más eficiente en términos de mensajes intercambiados entre el probador y el verificador. Las *k* rondas del protocolo consisten en lo siguiente:

Inicialmente los *n* vértices del grafo G son numerados $N_1, N_2, \dots, N_n$. El probador en secreto encripta los vértices en cajas $B_1, B_2, \dots, B_n$, haciendo un mapeo 1-1 al azar de cada vértice en una caja, de manera que cada una de las $n!$ permutaciones posibles de vértices en las cajas sea equiprobable. Luego prepara B_{ij} cajas para cualquier par de cajas posibles (B_i, B_j), colocando un 1 dentro de ellas si el vértice en la caja B_i es adyacente al vértice en la caja B_j y un 0 en caso contrario. Encripta también estas cajas y envía las $n + \binom{n}{2}$ cajas al verificador. En el segundo paso el verificador elige al azar una de las siguientes opciones: le pide al probador que desencripte todas las cajas o le pide que desencripte exactamente *n* cajas $B_{ij}, B_{jk}, \dots, B_{l'i}$ que contienen el ciclo Hamiltoniano. En el último paso el probador desencripta las cajas que le solicita el verificador. El verificador acepta si cuando pide todas las cajas estas contienen una descripción de G o si cuando pide las cajas del ciclo estas tienen todas un 1 y representan

un ciclo Hamiltoniano en cualquier grafo que es representado por las cajas (el ciclo Hamiltoniano del grafo G no es revelado ya que las cajas B_i no son abiertas).

El protocolo propuesto por Manuel Blum es completo ya que si un probador honesto efectivamente conoce un ciclo Hamiltoniano en el grafo, va a poder siempre responder a cualquiera de las solicitudes que le haga un verificador honesto, por lo cual la prueba se acepta con probabilidad igual a 1. También cumple con la propiedad de *soundness* porque si el probador no conoce un ciclo Hamiltoniano puede prepararse o bien para la solicitud de devolver una representación del grafo G o para la solicitud de algún ciclo Hamiltoniano de algún grafo arbitrario *G´* pero no para ambas, porque si pudiera prepararse para ambas significa que conoce un ciclo Hamiltoniano del grafo G. Por lo tanto tiene ½ de probabilidad de hacer trampa y al repetirse el protocolo *k* veces, esta probabilidad es 2^{-k}. Con respecto a la propiedad de cero conocimiento, cuando el probador revela el contenido de todas las cajas, esto no aporta ninguna información adicional al verificador, ya que es simplemente una representación del grafo G. Cuando revela un ciclo Hamiltoniano, este representa un ciclo arbitrario de longitud *n* debido a que cada permutación de los vértices en las cajas es equiprobable y porque en cada ronda un ciclo Hamiltoniano particular es seleccionado y siempre se muestra ese ciclo si se lo solicita. Mas aún, el verificador puede simular toda la interacción con el probador: puede simular al probador eligiendo arbitrariamente encriptar el grafo G o un ciclo arbitrario de longitud *n*. Envía eso a sí mismo. Suponiendo que tiene a disposición un algoritmo probabilístico eficiente (polinomial) para extraer información útil de la conversación con el probador, puede utilizarlo para decidir si solicitar el grafo o el ciclo (en su simulación de verificador). De todos modos, tiene un ½ de probabilidades de que este algoritmo seleccione una opción que el mismo (en su visión de probador) pueda responder. Si no es así, vuelve al algoritmo a su estado inicial e inicia una nueva ronda. Como conclusión, el verificador simulando tanto al probador como así mismo, en unas 2 pasadas esperadas por cada ronda, obtiene el mismo beneficio del algoritmo sin la ayuda del probador, por lo que en tiempo polinomial lo que puede hacer el verificador por su cuenta no es menos que lo que puede hacer en la interacción con el probador.

Es muy importante remarcar que ambas pruebas de conocimiento cero presentadas (para el 3-coloreo y Ciclo Hamiltoniano) utilizan los ya presentados

commitment schemes. Recordemos que estos tienen dos propiedades esenciales: *hiding* y *binding*. Estas dos propiedades tienen su versión tanto estadística como computacional, similar a lo visto de la propiedad de cero conocimiento. Se sabe que es imposible obtener simultáneamente *hiding* y *binding* estadístico, pero si se permite que alguna de las dos propiedades sea computacional, se asume que los *commitment schemes* existen [3]. Los *commitment schemes* pueden ser construidos a partir de funciones *one-way*, es decir funciones fáciles de computar, pero muy difíciles de invertir. Por los protocolos presentados para los problemas NP-Completo y lo explicado acerca de los *commitment schemes,* se puede enunciar el siguiente teorema:

Teorema: Si las funciones *one-way* (de una sola vía) existen, entonces todo $L \in NP$ tiene una prueba interactiva de conocimiento cero computacional.

Esta es una condición suficiente para la existencia de pruebas de conocimiento cero para lenguajes en NP y representa además un caso de uso positivo de la clase computacional NP-Completo (generalmente se usan estos problemas para demostrar la dificultad de resolución de otros problemas). Formalmente la demostración del teorema es la siguiente [6]:

Sea $L \in NP$ un lenguaje y t la composición de la reducción polinomial computable e invertible de L al problema $3SAT$ y de la reducción estándar de $3SAT$ al 3-coloreo (que sabemos que existe por ser 3-coloreo NP-Completo). Entonces $x \in L \leftrightarrow t(x)$ es 3-coloreable. El protocolo de conocimiento cero para L es entonces el siguiente:

La entrada es x, y tanto el probador como el verificador computan $G \leftarrow t(x)$. El probador luego utiliza un protocolo interactivo de conocimiento cero para probar que G es 3-coloreable. El verificador actúa según el resultado de este sub-protocolo. Es importante remarcar que para que la prueba interactiva sea considerada *zero-knowledge*, t tiene que ser invertible, es decir, $t^{-1}\big(t(x)\big) = x$. Esto se debe a que el verificador V recibe $t(x)$ y x, por lo que para cumplir con la forma estándar de *zero-knowledge* se utiliza otro verificador V^* que computa x a partir de $t(x)$ (esto se puede hacer porque t es polinomialmente invertible).

Un detalle importante del teorema presentado es que garantiza la existencia de pruebas de conocimiento cero computacionales, no perfectas. Una pregunta interesante es si se pueden construir pruebas de conocimiento cero perfectas para todo problema en NP. Resulta que esto fue demostrado que es un muy poco probable en [24]. La demostración consta de dos teoremas. El primero demuestra que para cualquier lenguaje que tiene una prueba de conocimiento cero perfecta, su complemento posee un protocolo interactivo de una sola ronda. Es decir, $PZK \subseteq co - AM$, donde PZK es la clase de lenguajes que tienen pruebas de conocimiento perfectas y AM es la clase de lenguajes reconocibles por los juegos de Arthur-Merlin de una sola ronda, en donde un probador interactúa una sola vez con un verificador y presenta evidencia estadística de la pertenencia de una cadena a un determinado lenguaje. El otro teorema establece que si todo lenguaje en $co - NP$ posee una prueba interactiva corta, entonces la clase de jerarquía polinomial (PH) colapsa al segundo nivel. El resultado más importante que se demuestra en [24] es el siguiente corolario:

Corolario: Si cualquier lenguaje NP-Completo posee un protocolo de conocimiento cero perfecto, entonces la clase de complejidad de jerarquía polinomial (PH) colapsa al segundo nivel

Como conclusión, se cree que es muy poco probable que existan pruebas de conocimiento cero perfectas para todo lenguaje en NP. El corolario se deduce de los teoremas mencionados, cuyas demostraciones se pueden consultar en [24]. No obstante, se ha probado que ciertos lenguajes en NP poseen pruebas de conocimiento cero perfectas, como ser el isomorfismo entre grafos.

Otra desventaja de la prueba de la existencia de ZKP para todo lenguaje en NP es la presunción de la existencia de funciones *one-way*. Muchos teóricos buscaron una manera de eliminar la necesidad de cualquier presunción y se llegó a la conclusión que esta puede ser eliminada si se permite la interacción con dos probadores y el verificador, en donde ambos probadores se ponen de acuerdo en una estrategia para convencer al verificador de la validez de una afirmación, intercambiando una cantidad polinomial de mensajes con el verificador, y sin comunicarse entre ellos durante la ejecución de la prueba [25]. En este modelo, todos los lenguajes en NP poseen un protocolo de

conocimiento cero perfecto, sin necesidad de hacer ninguna presunción. En otras palabras, en [25] se demuestra el siguiente teorema:

Teorema: Todo $L \in NP$ tiene una prueba interactiva de conocimiento cero perfecta con dos probadores.

Además de dicho teorema los autores presentan varios resultados importantes, como por ejemplo, que el nuevo modelo propuesto con dos probadores no tiene mayor poder expresivo que el de un solo probador. A modo de explicación del teorema, se presentan las ideas principales del modelo (el lector interesado puede ver la demostración completa en [25]):

Ambos probadores deben compartir una cinta infinita $R = \{r_1, r_2, \dots, r_k, ..\}$ $r_i \in \{0,1,2\}$, es decir una lista infinita de números comprendidos en ese rango. Luego se toma como base la prueba anteriormente presentada del ciclo Hamiltoniano de [23]. La idea es que el primer probador ejecute el paso 1 del protocolo, que consistía en computar una permutación de los vértices del grafo y encriptar en varias cajas la información de si los vértices son adyacentes o no. El segundo probador debe ejecutar el paso 3 del protocolo, que consiste en desencriptar lo solicitado por el verificador. La diferencia fundamental es la función de encriptación utilizada. No es una función basada en la existencia de una función de una sola vía, sino un algoritmo basado en *commitments* que tiene el siguiente esquema:

Sea $\sigma_0, \sigma_1 : \{0,1,2\} \rightarrow \{0,1,2\}$ tal que:

- $\forall\, i, \sigma_0(i) = i$.
- $\sigma_1(0) = 0, \sigma_1(1) = 2\ y\ \sigma_1(2) = 1$.

Para hacer el *commit* del bit k, m_k se debe hacer lo siguiente:

- V elige al azar un 1 o un 0 y se lo envía al probador 1 ($c_k \in \{0,1\}$)
- El probador 1 calcula $E(c_k, m_k) = \sigma_{c_k}(r_k) + m_k\ (mod\ 3)$, siendo r_k el valor leido de la cinta R que comparte con el probador 2. Envía $E(c_k, m_k)$ a V.

Para hacer el *reveal* del bit m_k se debe hacer lo siguiente:

- V envía k al probador 2.
- El probador envía a V la cadena r_k.
- V computa $\sigma_{c_k}(r_k)$ y establece m_k como $(E(c_k, m_k) - \sigma_{c_k}(r_k))\ (mod\ 3)$

Figura 24 Algoritmo de commitment de prueba de conocimiento cero con dos probadores [25]

El probador 2 no conoce $E(c_k, m_k)$ ni c_k. Conoce al programa del probador 1 por lo que la probabilidad que revele un mensaje diferente al elegido por el probador 1 es menor a ½ . El verificador conoce k, $E(c_k, m_k)$ y c_k, pero eso no le otorga ningun ventaja en adivinar el valor m_k. De estas dos observaciones y de la demostracion ya presentada de la prueba para el ciclo Hamiltoniano del modelo original, surge directamente la demostracion de que la propuesta constituye una prueba interactiva de dos probadores. La parte de la demostración de que cumple con la propiedad de *zero-knowledge* perfecto puede ser consultada en [25].

Como cierre de esta sección, podemos mencionar a la clase de complejidad IP (*Interactive Polynomial Time*) que corresponde a la clase de lenguajes para los cuales existe una prueba interactiva. Se ha demostrado que IP = PSPACE, donde esta última abarca todos los problemas que pueden ser resueltos por una máquina de Turing en un

espacio polinomial. IP es una clase más grande que NP y que además lo contiene, es decir hay problemas para los cuales existe una prueba interactiva pero no se puede construir una prueba en un sistema clásico por lo tanto no están en NP. El problema más conocido de este tipo es el del no-isomorfismo de grafos que pertenece a la clase de complejidad co-NP (es decir que el complemento del problema, en este caso, el isomorfismo, está en la clase NP) para el cual se demostró que hay una prueba de conocimiento cero. También se puede demostrar que todos los lenguajes de IP tienen una prueba de conocimiento cero. En [26] se demuestra el siguiente teorema:

Teorema: Asumiendo que los esquemas probabilisticos de encriptación existen, entonces todo $L \in IP$, admite una prueba de conocimiento cero interactiva.

Este resultado es muy interesante, porque básicamente nos afirma que cualquier afirmación que es demostrable también es demostrable en *zero-knowledge*.

16. Análisis desde el punto de vista ético

Además de ser interesante por la cantidad de aplicaciones y por el nivel de seguridad que ofrece ZKP, su función de protección de privacidad es una de sus características más importantes. La sociedad actual demanda cada vez mayores niveles de privacidad y control en el uso de datos personales. El Reglamento General de Protección de Datos de la Unión Europea (RGPD) y la Ley 25.326 de Protección de datos personales en Argentina (actualmente siendo revisada) son dos ejemplos de este fenómeno. ZKP representa una tecnología que puede aportar mucho en estos aspectos.

Existen varias herramientas para evaluar desde una perspectiva ética un asunto relacionado con las tecnologías de la información. En este trabajo se utilizan los

principios PAPA (*Privacy, Accuracy, Property and Accessibility*) que fueron introducidos por Richard O. Mason en 1986 en el artículo [27].

El primer principio de privacidad es el que permite determinar qué información es compartida y cual se mantiene en secreto. Está relacionado con la libertad que tiene un individuo de decidir qué revelar y qué mantener a salvaguardas. El principio *Accuracy*, precisión en español, está relacionado con la autenticidad, fidelidad y exactitud de la información y a quien se le otorga la responsabilidad de dichas propiedades de la información, por ejemplo, frente a posible errores o falsedad de esta. El principio de propiedad determina quién es el dueño de la información, los derechos de propiedad intelectual y los canales a través de los cuales es posible intercambiarla. El último principio de accesibilidad está relacionado con el otorgamiento de permisos de acceso a la información, en qué condiciones se comparte la información y con qué garantías.

En [28] se presenta una tabla con un resumen de un análisis ético realizado a las tecnologías de ZKP:

Area	Critical Question	Zero Knowledge Proofs (ZKPs)	Accountability Insights	Ethics Insights
Privacy	What data must people reveal about themselves to others?	Prover: Data to be validated. Verifier: None.	Prover reveals slightly more information.	No personal data is exchanged or revealed.
Accuracy	Who is responsible for the reliability, authenticity, and accuracy of data?	Prover: full responsibility. Verifier: no responsibility.	Prover is responsible for data accuracy.Verifier is responsible only for protocol.	Prover is ethically accountable for data errors as long as the verifier has a legitimate protocol.
Property	Who owns the data and "conduits through which information passes"?	Prover: full ownership of data. Verifier: decentralized ownership of conduits.	Prover has ownership of data but not conduits. Verifier does not have ownership of data nor conduits.	For most applications, prover retains full property rights over data. Data conduits are decentralized.
Accessibility	What data does a person or organization have a right to obtain, with what protection, and under what conditions?	Prover: Full right to access and transmit personal information. Verifier: No right to access or transmit personal information.	Prover has greater access to retrieve and transmit information. Verifiers are very limited.	Accessibility is greatest for the prover.
ETHICS ANALYSIS CONCLUSIONS: ZKPs have ethical benefits and responsibilities that are conferred almost entirely to the Prover. ZKPs give the Prover the most control over their own privacy and personal data as well as the responsibility to ensure that that data is accurate.				

Table 1: PAPA ethics framework analysis of Zero Knowledge Proofs.

Figura 25 Principios PAPA aplicados a ZKP [28]

En relación con la privacidad, en la Figura 25 se detalla que ZKP brinda una protección prácticamente absoluta, ya que ninguna información personal tiene que ser revelada ni es intercambiada, característica que es resultado de la propiedad de *zero-knowledge*. Esto es muy importante porque permite por ejemplo lograr identificación, controles de validaciones de transacciones ante una autoridad, entre otras, sin necesidad

de tener que revelar datos personales o saldos respectivamente. Con respecto al segundo principio de precisión, el probador es el que tiene la responsabilidad absoluta por mantener los datos correctos, sin errores y está relacionado con el principio de propiedad, ya que al ser dueño de los datos, tiene el control total de los mismos. El verificador no obtiene ninguna información que el probador no desee compartir, es decir, la accesibilidad es controlada principalmente por el probador.

Otra aplicación importante de ZKP es que puede garantizar el comportamiento ético ya que una prueba o evidencia "honesta" puede ser usada para promover una conducta genuina y preservar la privacidad al mismo tiempo [29]. Debido a la propiedad de *soundness* de estos protocolos, se garantiza que un usuario debe actuar honestamente para poder presentar una prueba válida y a su vez, la propiedad de *zero-knowledge* garantiza que la privacidad del usuario no es comprometida. Esta aplicación proviene del trabajo realizado en [30] en donde se introduce por primera vez el concepto de *secure computation*, donde varios participantes ejecutan un protocolo para computar de manera conjunta una función de sus entradas privadas, de manera tal que ninguno pueda obtener alguna información además de la salida del protocolo. Específicamente lo que se demuestra es que dos partes A y B pueden generar un número entero aleatorio $N = p.q$ mediante la interacción de manera tal que los factores p y q se mantienen en secreto de A y B individualmente, pero pueden ser recuperados si lo desean de manera conjunta. A este método de cómputo también se lo conoce popularmente como MPC (*Multi-Party Computation*).

Otro uso muy interesante de la tecnología ZKP es su potencial aplicabilidad para el desarmamiento nuclear. Investigadores de la Universidad de Princeton publicaron un artículo donde presentan un protocolo físico ZKP de comparación de objetos aplicable para la verificación de armas nucleares. En [31] los investigadores demuestran una técnica no electrónica de radiografía diferencial de neutrones rápidos que utiliza detectores de emulsión sobrecalentada y que puede determinar que dos objetos son idénticos sin revelar su geometría o composición. Esto permite que no sea necesario compartir información secreta sobre el diseño de las armas y además puede tener muchos usos en otros ámbitos donde sea necesario comparar propiedades físicas de objetos sin necesidad de revelar detalles de estos, como por ejemplo análisis forense de

ADN. En la publicación [31] se detalla el protocolo experimental que fue propuesto para el análisis de ojivas nucleares en particular. El protocolo es del tipo interactivo en el cual participan un huésped (que posee las armas) y un inspector (que desea verificar las armas). Se requiere de la existencia de por lo menos una ojiva de referencia que puede ser propuesta por designación del inspector. Mediante la aplicación de pruebas radiográficas es posible confirmar si objetos prácticamente idénticos al de referencia son genuinos, sin que el inspector obtenga algún conocimiento sobre su geometría o composición. El protocolo consiste en varias etapas: inicialización, preparación de las precargas e inspección. Es importante remarcar algunas cuestiones: el inspector no está presente en la etapa de preparación de las precargas, el único objeto que el inspector considera genuino es el que es seleccionado como de referencia, ambas partes (huésped e inspector) tienen que haber acordado los procedimientos de cadena de custodia, monitorización del perímetro y cualquier otra medida de seguridad física apropiada para las pruebas, y finalmente el uso de técnicas no electrónicas elimina el riesgo de *tampering* por parte del huésped una vez que las precargas fueron elegidas y de *snooping* por parte del inspector para intentar descubrir los patrones de precarga. En el siguiente párrafo se enuncian muy resumidamente los pasos de cada una de las etapas.

En la primera etapa de inicialización el huésped debe producir una colección de detectores, midiendo y registrando sus curvas de eficiencia de neutrones. El inspector selecciona un subconjunto de estos detectores para usar en la inspección y puede quedarse con los restantes para confirmar sus curvas de calibración. Además, el huésped captura un gran número de radiografías en varios objetos en m diferentes ángulos y l diferentes estados de energía, para minimizar cualquier error sistemático en la etapa de precargas. En la etapa de preparación de precargas, el huésped prepara n conjuntos precargados de detectores $p_{i,j}^{k}, k = 1 \dots n$, en el cual cada uno corresponde a la imagen complemento de los objetos aclamados ser idénticos al de referencia, en un particular ángulo $\theta_i, i = 1 \dots m$ y una particular energía $E_j, j = 1 \dots l$. En la fase de inspección el inspector elige una orientación θ_i de los objetos y una energía E_j del rayo de neutrones. El huésped le muestra al inspector los n conjuntos precargados de detectores $p_{i,j}^{k}, k = 1 \dots n$ y el inspector selecciona una asignación aleatoria de los

conjuntos de detectores al objeto de referencia y los $n-1$ objetos inspeccionados. El "complemento" implementado por el huésped se define como la precarga que dará lugar a que la fluencia total registrada en cada detector sea la misma que si no hubiera habido ningún artículo presente cuando el artículo inspeccionado es el mismo que el artículo de referencia; a esto se lo denomina cuenta total o N_{max}. Los objetos son expuestos a la fuente de neutrones y se registran las radiografías en los detectores. Ambas partes deben monitorear la fuente de fluencia con su propio artefacto, para que el inspector esté seguro de que una medición se está llevando a cabo y que la fluencia acordada de N_{max} ha sido utilizada. Luego de la inspección de cada objeto se cuenta el número de burbujas macroscópicas en cada detector. Ambas partes calculan el resultado de la prueba acordada para la hipótesis de que la distribución de los recuentos es n variables aleatorias de Poisson independientes, cada una con una expectativa N_{max}. Algunos detectores son seleccionados por el inspector para remover las burbujas y prepararlos para reusarlos, re-exponiéndolos a la fuente para verificar su funcionalidad y calibración. La etapa de inspección se repite cuantas veces considere necesario el inspector para aumentar la propiedad de *soundness* del experimento. La radiografía propuesta por los autores de este experimento físico con un perfil plano de N_{max} y ruido de Poisson posee la propiedad de *zero-knowledge*. La demostración formal de esta propiedad es compleja y se prueba en detalle en [31].

17. Conclusiones

ZKP posee varios casos de uso prácticos, como ser autenticación, firma digital, voto electrónico, *blockchain* y criptomonedas, garantía de comportamientos éticos sin sacrificar privacidad, entre otros que han sido detallados en este trabajo. Incluso ha generado una fuerte conexión entre la criptografía y el estudio de la complejidad computacional. Algunas de estas aplicaciones prácticas son cruciales en el ámbito de la seguridad informática como ser la autenticación, un proceso que se encuentra en casi cualquier interacción con un sistema informático. La característica más atractiva de la tecnología de ZKP es el nivel de privacidad que brinda en los protocolos o pruebas que se basan en ella. La sociedad actual demanda cada vez mayor privacidad y control de sus datos personales, y esta tecnología puede brindar esas características sin tener que sacrificar la propiedad de seguridad. Su creciente uso y aplicación en *blockchain* y criptomonedas es notable, pudiendo mencionar varios ejemplos de utilización reales, como Monero y Zcash. Por todos estos motivos se cree que ZKP va a tener una participación muy importante en la criptografía de los próximos años, con mayor utilización en productos de seguridad. Como líneas futuras, es probable que haya un aumento de propuestas de uso de ZKP para generar aplicaciones de identidad auto-soberana (*Self-Sovereign Identity*) y una mayor adopción de estrategias de MPC (Multi-Party Computation).

18. Glosario

Chosen ciphertext: Ataque criptoanalítico considerado de gran eficacia en donde el atacante tiene a disposición el motor de desencripción y puede generar un texto plano a partir de cualquier texto cifrado que elija.

Chosen text: ataque generalmente utilizado en esquemas de autenticación *challenge-response* que consiste en la utilización de desafíos específicos para obtener información acerca del secreto que posee una de las partes o para lograr un efecto secundario como la firma de un archivo.

Clase (computación): conjunto de problemas computacionales que poseen la misma complejidad computacional.

Commitment scheme: primitiva criptográfica que tiene dos fases: una de *commit* donde se elige un valor que no se puede cambiar más adelante y una de *reveal* donde se muestra el valor previamente elegido.

Eavesdropping: consiste en escuchar conversaciones o comunicaciones privadas a escondidas, en secreto, sin la autorización de las partes y sin que estas lo sepan.

Grafo: Par de conjuntos V y E, donde V representa un conjunto de nodos o vértices y E es un subconjunto del conjunto de pares no ordenados de elementos distintos de V. A los elementos de E se los denomina ejes o aristas.

Grupo (matemática): Estructura algebraica formada por un conjunto y una operación interna que cumple con las propiedades de clausura, asociatividad, existencia de elemento neutro y en el cual todo elemento posee inverso.

Impersonation: ataque que consiste en hacerse pasar por otro, robar su identidad, gracias a la posesión del secreto u objeto que identifica a una parte.

Intratable (computación): problema computacional que no puede ser resuelto en la práctica. Las causas pueden ser porque el problema requiere una cantidad de memoria exponencial en su salida (por ejemplo, la generación de todos los subconjuntos posibles de un conjunto extenso en tamaño), porque no se conocen algoritmos eficientes para resolverlo (factorización de números) o porque no es decidible (problema de *halting*).

Kerberos: protocolo de autenticación de esquema cliente-servidor que permite la autenticación mutua en una red.

Man in the middle: ataque informático que consiste en interceptar los mensajes que dos partes intercambian por un canal de comunicación, pudiendo leer, modificar, borrar e insertar mensajes sin que ninguna de las partes conozca que el enlace ha sido interceptado.

Máquina de Turing: Modelo matemático de cómputo propuesto por Alan Turing consistente en una máquina abstracta que manipula símbolos en una cinta en base a ciertas instrucciones, capaz de ejecutar cualquier algoritmo.

One-way (una sola vía): función que es computacionalmente fácil de calcular pero muy difícil de invertir.

Oráculo: máquina abstracta utilizada dentro del contexto del estudio de teoría de la computabilidad y complejidad que es capaz de resolver un problema computacional (puede ser de decisión o incluso una función no computable) en una única operación.

Phishing: ataque de ingeniería social cuyo objetivo es la obtención de credenciales de acceso, mediante técnicas de engaño y manipulación.

qbit: unidad básica de información de un computador cuántico que puede tener el valor 1 o 0 y también una superposición de ambos.

Replay: ataque informático consistente en el reenvío de mensajes previamente transmitidos, generalmente pertenecientes a la ejecución de un protocolo de autenticación, cuyo objetivo es lograr la suplantación de identidad de alguna de las partes.

Snooping (fisgonear): hacer todo lo necesario para poder escuchar una conversación u obtener información secreta que no se está autorizado a poseer.

Tampering (manipulación): modificación con una intención maliciosa de algún dispositivo o sistema informático de manera no autorizada.

Third Trusted Party: entidad externa que facilita la interacción entre dos partes que confían en ella.

Trusted setup (inicialización segura/confiable): etapa de configuración inicial de un protocolo donde se establecen parámetros y claves, cuya adecuada selección por parte de entidades confiables es crucial para el correcto funcionamiento y seguridad de este.

19. Bibliografía

[1] S. Goldwasser, S. Micali y C. Rackoff, «The knowledge complexity of interactive proof systems,» *SIAM Journal on Computing,* vol. 18, pp. 186-208, 1989.

[2] A. Menezes, P. v. Oorschot y S. Vanstone, Handbook of Applied Cryptography, CRC Press, 1996.

[3] S. Vadhan, «The Complexity of Zero Knowledge,» Harvard University, Cambridge, 2007.

[4] R. Oppliger, Contemporary Cryptography, Norwood: Artech House, 2005.

[5] B. Ewanick, «Zero Knowledge Proofs,» de *Comp 4109*, 2011.

[6] A. Mohr, «A Survey of Zero-Knowledge Proofs with Applications to Cryptography,» Southern Illinois University, Carbondale.

[7] S. Micali, M. Blum y P. Feldman, «Non-Interactive Zero-Knowledge and Its Applications,» *ACM,* 1988.

[8] Wikipedia, «Zero-knowledge proof,» [En línea]. Available: https://en.wikipedia.org/wiki/Zero-knowledge_proof. [Último acceso: Diciembre 2022].

[9] O. Goldreich, S. Micali y A. Wigderson, «Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems,» *Association for Computing Machinery,* vol. 38, nº 1, pp. 691-729, 1991.

[10] S. Grzonkowski y P. M. Corcoran, «Security Analysis of Authentication Protocols for Next-Generation Mobile and CE Cloud Services,» College of Engineering & Informatics, Galway, 2011.

[11] P. Knickerbocker, «Combating phishing through zero-knowledge authentication,» Department of Computer and Information Science and the Graduate School of the University of Oregon, 2008.

[12] Y. Yu y X. Xie, «Privacy-preserving computation in the post-quantum era,» *National Science Review,* vol. 8, nº nwab115, 2021.

[13] J. P. Hecht, «A Compact ZKP Authentication Protocol,» *Journal of Cryptology,* 2011.

[14] J. P. Hecht, «Protocolo de autenticación ZKP usando grupos no conmutativos,» de *Congreso Iberoamericano de Seguridad Informática CIBSI*, 2011.

[15] Ethereum, «ethereum.org,» Ethereum, 29 Septiembre 2022. [En línea]. Available: https://ethereum.org/en/zero-knowledge-proofs/. [Último acceso: 1 Octubre 2022].

[16] A. Fiat y A. Shamir, «How To Prove Yourself: Practical Solutions to Identification and Signature Problems,» Department of Applied Mathematics, Israel, 1999.

[17] Koe, K. M. Alonso y S. Noether, Zero to Monero: Second Edition, Monero, 2020.

[18] E. Morais, T. Koens, C. v. Wijk y A. Koren, «A Survey on Zero Knowledge Range Proofs and Applications,» 2018.

[19] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille y G. Maxwel, «Bulletproofs: Short Proofs for Confidential Transactions and More,» Stanford University, University College London, Blockstream, 2017.

[20] Zcash, «What are zk-SNARKs?,» [En línea]. Available: https://z.cash/technology/zksnarks/. [Último acceso: 18 Febrero 2023].

[21] G. J. A.-P. Espuelas, ESTUDIO Y ANÁLISIS DE ESQUEMAS DE VOTACIÓN ELECTRÓNICA MEDIANTE PROTOCOLOS DE FIRMAS DIGITALES ANÓNIMAS, Universidad Autónoma de Madrid, 2017.

[22] A. C. García, Desarrollo de un sistema de votaciones electrónicas verifcables basado en pruebas de conocimiento cero, Unvisidad Autónoma de Madrid, 2021.

[23] M. Blum, «How to Prove a Theorem So No One Else Can Claim It,» International Congress of Mathematicians, Berkeley, 1986.

[24] L. Fortnow, «The Complexity of Perfect Zero-Knowledge,» MIT Math Dept., Cambridge, 1999.

[25] M. Ben-Or, S. Goldwasser, J. Kilian y A. Wigderson, «Multi-Prover Interactive Proofs: How to Remove Intractability Assumptions,» *ACM,* 1988.

[26] S. Goldwasser, «Everything Provable is Provable in Zero-Knowledge,» de *Advances in Cryptology - CRYPT0 '88*, Santa Barbara, 1990.

[27] R. O. Mason, «Four Ethical Issues of the Information Age,» Southern Methodist University, Dallas, 1986.

[28] J. Anastasopoulos, «Smart Contract Research Forum,» Julio 2021. [En línea]. Available: https://www.smartcontractresearch.org/t/discussion-post-zero-knowledge-proofs-an-ethics-perspective/575. [Último acceso: Septiembre 2022].

[29] J. Hasan, «Overview and Applications of Zero Knowledge Proof (ZKP),» *International Journal of Computer Science and Network,* vol. 8, nº 5, 2019.

[30] A. C.-C. Yao, «How to Generate and Exchange Secrets,» IEEE, New Jersey, 1986.

[31] S.´. Philippe, R. Goldston, A. Glaser y F. d'Errico, «A physical zero-knowledge object-comparison system for nuclear warhead verification,» *NATURE COMMUNICATIONS,* 2016.