CatioCrypto 2025: Zero-Knowledge Proofs

27 Septiembre 2025

Autor Emanuel Nicolás Herrador
Speaker Armando Faz Hernandez

1. Commitments Schemes (Esquemas de comprometimiento)

1.1. Definiciones

Como analogía, tenemos que pensar en que Ana piensa un número y Beto lo tiene que adivinar. Para comprometerse en no cambiar el número luego de la elección de Beto, la idea es colocarlo dentro de una caja con candado y entregárselo a Beto. Luego de la elección de Beto, Ana le otorga la llave para verificar si acertó.

Las propiedades que queremos son:

- Ocultante: Cuando Beto recibe la caja no puede saber el valor hasta que reciba la llave.
- Vinculante: Cuando se abra la caja no se puede mostrar un valor diferente al almacenado

Formalmente, un esquema de comprometimiento no interactivo define los algoritmos:

- Inicializa(1^λ): Retorna parámetros del sistema para un nivel de seguridad λ
- Compromete$(m, r) \to (c, d)$: Toma el mensaje m al que se va a comprometer y una entrada aleatoria r, retorna un compromiso c y un valor de apertura d
- Verifica$(c, m, d) \to \{0, 1\}$: Retorna un bit indicando si la verificación es exitosa

Las propiedades se traducen en:

- Ocultante: El compromiso c no revela información sobre el mensaje m
- Vinculante: Ningún adversario puede generar c, $m \neq m'$ y d, d' tal que ambos acepten Verifica(c, m, d) y Verifica(c, m', d').

Además, estas tienen, a su vez, las siguientes propiedades:

- Incondicional: Aún con poder de cómputo infinito no puede violar la propiedad
- Computacional: A un atacante limitado polinomialmente le es difícil violar la propiedad

Hay 4 posibilidades pero lo más común es ver "Incondicional vinculante y computacional ocultante", o "Computacional vinculante e incondicional ocultante"; mientras que no se cumple que ambas puedan ser incondicionales.

1.2. Ejemplo: lanzar una moneda por teléfono

Alicia y Beto deciden otorgar un objeto a aquél que gane una tirada pero no confían en el otro. El protocolo para el lanzamiento de moneda es:

1. Ana se compromete al valor de un bit aleatorio b_A
2. Ana envía su compromiso C a Beto
3. Beto escoge un bit b_B aleatoriamente y lo envía a Alicia
4. Alicia abre C para que Beto conozca b_A
5. Ambos calculan el resultado $b = b_A \oplus b_B$

Con esto se asegura el lanzamiento de la moneda y el protocolo completo solo queda en agregar que Beto adivine. Aquí b sería el lanzamiento.

Comprometer un bit con RSA Un esquema de compromiso para un bit se puede hacer usando RSA.

- Ana tiene un par de llaves RSA: pública (e, n) y privada (d)
- Ana se compromete a un bit b, escoge $x_b \leftarrow \{0, \ldots, n\}$ aleatoriamente tal que su bit menos significativo es b
- Ana envía $C = x_b^e \pmod{n}$ a Beto

Tiene las siguientes propiedades:

- Ocultante: Notar que Beto no puede saber x_b a menos que pueda romper RSA.
- Vinculante: Una vez que escogió el bit, no lo puede cambiar dada la construcción del compromiso.

Otra forma de hacerlo es usando QR_N (residuos cuadráticos módulo N), el cual únicamente usa $N = pq$ con p, q primos aleatorios. La forma es:

1. Ana elige aleatoriamente un módulo $N = pq$ y envía (N, x) a Beto con
$$x \xleftarrow{\$} \begin{cases} QR_N & \text{si } b = 0 \\ \mathbb{Z}/QR_N & \text{si } b = 1 \end{cases}$$
2. Ana abre el compromiso enviando (p, q)
3. Beto verifica que $N = pq$ y que si x es QR_N entonces retorna $b = 0$, pero sino retorna $b = 1$.

1.3. Otros protocolos

1.3.1. Pedersen

Sea $G = \langle g \rangle$ un grupo cíclico de orden q y sean $g, h \in G$ dos generadores. El compromiso de Pedersen para escalares $m \in \mathbb{Z}/q\mathbb{Z}$ es:

1. Ana obtiene $r \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$ y calcula $c = g^m h^r$
2. Ana abre el compromiso enviando (m, r) a Beto
3. Si beto verifica $c = g^m h^r$ entonces retorna m, sino error.

1.3.2. ElGamal

Sea $G = \langle g \rangle$ un grupo cíclico de orden q y sean $g, h \in G$ dos generadores. El compromiso de ElGamal para elementos $M \in G$ es:

1. Ana obtiene $r \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$ y calcula $c_0 = g^r, c_1 = Mh^r$. El compromiso es (c_0, c_1).
2. Ana abre el compromiso enviando (M, r) a Beto
3. Si beto verifica $c_0 = g^r, c_1 = Mh^r$ entonces retorna M, sino error.

2. Pruebas

La idea es que se desea brindar acceso a un recurso solo a ciertos usuarios, los cuales se pueden identificar con una contraseña. Para ello, deben enviar su contraseña al servidor para verificar su validez pero el problema es que cualquiera escuchando al medio puede saberla. Las pruebas entran para remediar esta situación.

Para esto, el servidor solo necesita saber si el usuario sabe su contraseña. Es decir, debe saber solo un bit y nada más. Para ello se pretende un protocolo donde al final de la interacción entre el usuario y el servidor, este último obtiene un bit que indica si el usuario pudo identificarse exitosamente.

Definición 2.1 (Zero-Knowledge Protocol)**.** Un protocolo es de conocimiento nulo si comunica exactamente el conocimiento previsto y ningún (cero) conocimiento adicional.

Estos protocolos tienen dos partes (prover/probador y verifier/verificador), los cuales vamos a llamar como Perla y como Víctor en los ejemplos. El prover quiere convencer al verifier sobre la verdad de alguna declaración/statement, mientras que el verifier quiere verificar que el statement sea verdadero.

2.1. Construcción de un ZK Protocol

Supongamos que tenemos un criptosistema de llave pública y que el usuario A posee una public key P_A y una private key k_A. El statement es "Perla quiere probar que ella es el usuario A". Con ello, el protocolo es:

1. Víctor escoge un mensaje M aleatoriamente y envía $C = E(M, P_A)$ a Perla
2. Perla descifra C usando k_A y envía el resultado M' a Víctor
3. Víctor acepta la declaración si $M' = M$

Nota. No es Zero-Knowledge porque el atacante puede actuar como el Verifier y hacer que el Prover le descifre mensajes cifrados de los cuales no conoce el texto original. El prover no tiene forma de evitar esta situación con este protocolo.

En este caso, cuando el Verifier envía C, no existen garantías de que conozca también M. Para solucionarlo, se debe agregar que el Prover envíe también el compromiso de M. El esquema final es:

1. Víctor escoge un mensaje M aleatoriamente y envía $C = E(M, P_A)$ a Perla
2. Perla descifra C usando k_A y envía un compromiso c de M' a Víctor
3. Víctor envía M a Perla
4. Perla revisa si $M = M'$ y retorna error sino se cumple. Caso contrario, abre el compromiso enviando (r, M) a Víctor
5. Víctor acepta la declaración si $M' = M$ y si el compromiso para la verificación

Nota. Con esto ya se llega a que es un protocolo de conocimiento nulo.

2.2. Sistema de prueba interactivo

Ahora los protocolos se llevan a cabo como interacciones entre dos máquinas interactivas de Turing. El Prover tiene poder computacional infinito mientras que el Verifier tiene un tiempo de cómputo limitado polinómicamente. El par (P, V) recibe una entrada x y al final de la interacción devuelve si se acepta o rechaza respecto a si pertenece al lenguaje $L \subseteq \{0,1\}^*$ (i.e., si se verifica x o no).

El par (P, V) es un sistema de prueba interactivo para L si satisface:

- Completitud: Si $x \in L$ entonces la probabilidad de que (P, V) rechace x es despreciable en la longitud de x
- Solidez (Soundness): Si $x \notin L$, entonces para cualquier prover P^*, la probabilidad de que (P^*, V) acepte x es despreciable en la longitud de x.

Argumentos interactivos Un argumento interactivo para un lenguaje L es similar a un sistema de prueba interactivo, excepto que:

- El prover tiene tiempo de cómputo limitado polinómicamente y es probabilístico
- En la completitud se requiere que para cada $x \in L$ exista una entrada auxiliar que permita al probador convencer al verificador
- En la solidez, se cambia por cualquier prover que tenga tiempo de cómputo limitado polinomialmente y que sea probabilístico

2.3. Prueba de conocimiento (Proof of Knowledge)

En una prueba de conocimiento el prover afirma saber una pieza de información. Vamos aquí a tener un algoritmo que va a ser el extractor quien interactúa con el prover (posiblemente malicioso) para saber verificar si realmente conoce esta información.

Un ejemplo es "Schnorr Knowledge Proof" donde el statement es "Sé el logaritmo discreto $x = \log_g(y)$, i.e., $y = g^x$".

TODO: Agregar contenido completo del protocolo de Schnorr, pruebas de conocimiento nulo, Fiat-Shamir y firma digital con Schnorr en base a las filminas.