CatioCrypto 2025: Multi-Party Computation

26-27 Septiembre 2025

Autor Emanuel Nicolás Herrador
Speaker Eduardo Soria-Vázquez

1. Introducción

Algunos ejemplos acerca de qué se ocupa la criptografía son:

- *Verifiable Computation*: Demostrar que se ha ejecutado un programa, sin que quien lo verifique tenga que re-ejecutarlo.
- *Zero-Knowledge Proofs*: Similar a VC, pero quien demuestra hacer el cálculo pueden mantener algunos datos ocultos.
- *Secret Sharing*: Fragmentar un secreto entre varias entidades, de tal forma que solo pueda reconstruirse si ciertos subconjuntos de ellas acuerdan hacerlo.
- *Multi-Party Computation* (MPC): Calcular sobre datos "sin verlos", de forma distribuida.
- Otros: privacidad diferencial (DP), recuperación privada de información (PIR), cifrado totalmente homomórfico (FHE), cifrado funcional (FE), time-lock puzzles, blockchaings, criptomonedas, ...

2. Multi-Party Computation

2.1. Protocolo MPC

Un protocolo MPC consta de n entidades que desconfían las unas de las otras y de modo que cada una pueda aportar un valor P_i secreto. La idea/objetivo es calcular un función f aplicada a estos valores en conjunto pero sin mostrarlos a otra de las entidades.

Una forma de resolverlo sin MPC es con una tercera parte de confianza, pero el problema es que no siempre existe. Lo que vuelve difícil a MPC es la presencia de un adversario que participa y corrompe a ciertas entidades para coordinar su ataque:

- Corrupción pasiva: las entidades corruptas hacen lo que deben pero comparten información entre ellas para tratar de aprender más cosas sobre las demás entidades
- Corrupción activa: las entidades corruptas pueden actuar de forma arbitraria para sabotear los objetivos de seguridad

El objetivo es lograr todas las propiedades que lograríamos con una tercera parte de confianza pero sin ella.

2.2. Primitivas

Algunas de las prmitivas usadas por MPC son:

- Garbled Circuits
- Secret Sharing
- Oblivious Transfer
- Homomorphic Encryption

2.2.1. Secret Sharing (compartición de secretos)

Definición 2.1 (Esquema de compartición de secretos)**.** Sea $\mathbb{F}$ un cuerpo finito y $n, t \in \mathbb{Z} : 0 \leq t < n$. Sean las entidades $P_1, \ldots, P_n$ y un compartidor en posesión de un secreto $s \in \mathbb{F}$. Un esquema (n, t)-umbral de compartición de secretos consta de las siguientes fases:

1. Fase de compartición: El Compartidor fragmenta s en n partes $s_1, \ldots, s_n \in \mathbb{F}$ y la entidad P_i recibe su parte $s_i \in \mathbb{F}$
2. Fase de reconstrucción: Al menos $t+1$ entidades envían su parte a quien quieran que reconstruya s.

Y tal que satisface las siguientes propiedades:

- t-privacidad: Cualquier conjunto de a lo sumo t partes no revela ninguna información sobre el secreto $s \in \mathbb{F}$
- $(t+1)$-reconstrucción: Cualquier conjunto de $t+1$ partes determina de forma única el secreto $s \in \mathbb{F}$

Definición 2.2 (Esquema lineal de compartición de secretos)**.** Decimos que el esquema es *lineal* si además satisface que $\forall s, t \in \mathbb{F}$ compartidos como $(s_1, \ldots, s_n), (t_1, \ldots, t_n) \in \mathbb{F}^n$, tenemos que $(s_1 + t_1, \ldots, s_n + t_n)$ es una compartición de $s + t \in \mathbb{F}$.

Nota. Abusaremos notación y nos restringiremos a esquemas de compartición de secretos que sean sobre un cuerpo finito $\mathbb{F}$, umbrales y lineales (sobre $\mathbb{F}$). Además, usaremos la notación definida como $[s] := (s; s_1, \ldots, s_n)$, de modo que $[s] + [t] = [s + t]$.

Compartición de secretos aditiva Sea $F_p := (\mathbb{Z}/p\mathbb{Z}, +, \cdot),\ \mathbb{Z}/p\mathbb{Z} = (0, \ldots, p-1),\ p$ nro. primo, el esquema es:

1. Fase de compartición:
   a) Compartidor escoge de forma uniformemente aleatoria $s_1, \ldots, s_n \in \mathbb{F}_p : s_1 + \cdots + s_n = s$.
   b) Compartidor envía la parte s_i a P_i
2. Fase de reconstrucción:
   a) Cada P_i envía s_i a la entidad que quieren que reconstruya el secreto
   b) Dicha entidad calcula $s = s_1 + \cdots + s_n$

Nota. Notar que el umbral t debe ser $n-1$ para que sea reconstruible teniendo todos los pedacitos de secretos. El hecho de que no sea reconstruible con $n-1$ entidades se reduce al OTP porque si tenemos $s_1, \ldots, s_{n-1} \in \mathbb{F}_p$ elegidos uniformemente aleatorios, entonces $s_n = s - (s_1 + \cdots + s_{n-1})$ es también uniformemente aleatorio.

Las *ventajas* de la compartición de secretos aditiva es que tiene un tamaño óptimo de las partes (cada uno es tan grande como el secreto), por lo cual es muy eficiente a nivel computacional. Sin embargo, algunas *contras* son que el umbral es siempre fijo ($t = n-1$, por lo que no podemos elegir), que no tiene propiedades multiplicativas y que tampoco ofrece resistencia ante adversarios activos.

Compartición de secretos replicada (con 3 entidades) Se puede generalizar con $n > 3$ pero se suele usar de este modo. El esquema es:

1. Fase de compartición:
   a) Compartidor escoge de forma uniformemente aleatoria $s_1, s_2, s_3 \in \mathbb{F}_p : s_1 + s_2 + s_3 = s$.
   b) Compartidor envía las partes s_{i-1}, s_{i+1} a P_i
2. Fase de reconstrucción:
   a) Cada P_i envía s_{i-1} y s_{i+1} a la entidad que quieren que reconstruya el secreto
   b) Dicha entidad calcula $s = s_1 + s_2 + s_3$

Nota. El umbral t es 1 porque es 1-privado y 2-reconstruible.

Las *ventajas* son que:

- Tiene propiedades multiplicativas

- Ante un adversario pasivo, la reconstrucción puede requerir de menos comunicación que la compartición de secretos aditiva (por la redundancia). Como ejemplo, si es alguien interno (de las 3 entidades), entonces solo requiere de un valor, lo cual es una mejora en el coste de comunicación respecto al esquema lineal. Si la entidad es externa, se mantiene.
- Tiene propiedades de detección de errores. Ante un adversario activo, se puede detectar si una entidad se desvía del protocolo. En el caso de que no envíe nada, por la redundancia se salva. Pero si envía datos incorrectos, podemos detectarlo y frenar el cálculo, aunque no reconstruir.

En cuanto a los *inconvenientes*, el principal es que el tamaño de las partes es superior al del secreto y crece rapidísimo en la generalización: $\binom{n}{t}$ elementos de $\mathbb{F}_p$ por entidad. Notar que en el caso $t = 1, n = 3$, el tamaño de las partes es el doble que del secreto.

Compartición de secretos de Shamir (n entidades) Sea $F_p := (\mathbb{Z}/p\mathbb{Z}, +, \cdot)$, $\mathbb{Z}/p\mathbb{Z} = (0, \dots, p-1)$, $p > n$ nro. primo, el esquema es:

1. Fase de compartición:
   a) Compartidor escoge de forma uniformemente aleatoria $f \in \mathbb{F}_p[X]$ de grado t tal que $f(0) = s$
   b) Compartidor envía a cada P_i su parte $f(i) = s_i$
2. Fase de reconstrucción:
   a) Cada P_i envía s_i a la entidad que quieren que reconstruya el secreto
   b) Dicha entidad obtiene $f \in \mathbb{F}_p[X]$ mediante interpolación y recupera el secreto evaluando $f(0) = s$

Nota. El umbral t es el grado del polinomio, dado que con t puntos no somos capaces de determinar el polinomio de grado t con interpolación, mientras que con $t + 1$ sí se puede. Además, esta compartición es *lineal*.

Se puede usar la Matriz de Vandermonde para demostrar linearidad (incluso isomorfismo), t-privacidad y $(t + 1)$-reconstruibilidad. Recordemos que una matriz de Vandermonde es:

$$Van^{u\times v}(a_1, \dots, a_u) = \begin{bmatrix} a_1^0 & \dots & a_1^{v-1} \\ \vdots & \vdots & \vdots \\ a_u^0 & \dots & a_u^{v-1} \end{bmatrix}$$

Tal que si $u = v$ y $\forall i \neq j, a_i \neq a_j$ entonces $Van^{u\times v}(a_1, \dots, a_u)$ es invertible. Además lo son también todas sus submatrices cuadradas. Con ello, se puede ver que sea $f(X) = \sum_{i=0}^{t} f_i \cdot X^i$ el polinomio en $\mathbb{F}_p[X]$ de grado $\leq t$, entonces $(f(a_1), \dots, f(a_{t+1})) = Van^{u\times v}(a_1, \dots, a_{t+1}) \cdot (f_0, \dots, f_t)$.

Las *ventajas* de este esquema de compartición son que el tamaño de cada parte es igual al tamaño del secreto, tiene propiedades multiplicativas y es equivalente a un código de Reed Solomon (por lo que en presencia de un adversario activo se pueden explotar las propiedades de detección y corrección de errores). En cuanto a las *contras*, esta se basa principalmente en que la reconstrucción es algo costosa respecto a los otros dos esquemas vistos, y de que el tamaño del cuerpo tiene que ser mayor que el grado (sino la matriz de Vandermonde dejaría de ser invertible).

2.3. Propiedades y características

Las métricas de eficiencia para los protocolos de MPC son:

- Cantidad de datos que se comunican (complejidad de comunicación). Suele ser el cuello de botella de los protocolos y es lo que se busca bajar.
- Número de rondas de comunicación (complejidad de rondas)
- Complejidad computacional

La combinación de las distintas dimensiones determina la eficiencia que es posible (en ocasiones, tenemos cotas inferiores) o directamente si es posible o imposible hacer MPC con dicha selección. Veamos cada un de las dimensiones que puede tener el protocolo.

Modelo de computación Tenemos distintas formas de representar la computación. Podemos considerar circuitos booleanos (generalmente considerando and y or), circuitos aritméticos (sobre cuerpos o anillos finitos) o memoria de acceso aleatorio (RAM). Cualquier función o programa se puede expresar en cualquiera de los tres modelos.

Fases en el protocolo Podemos considerar las siguientes posibilidades:

- Paradigma "Offline-Online": Existe una fase de preprocesado ("offline") en la cual no es necesario que las entidades hayan decidido sus inputs para la función. La fase "online" comienza cuando las entidades conocen sus inputs, y generalmente esta puede ejecutarse más rápido debido al trabajo avanzado en el preprocesado. Hay dos tipos principales de preprocesados: dependiente o independiente de la función. En el primer caso se asume el conocimiento de la función que se calculará en la fase online mientras que en la segunda no (en realidad se sabe algo pero es una cota).
- Sin preprocesado: existe una única fase, por lo que no se avanza antes de conocer los inputs.

Suposiciones sobre la red Podemos hacer dos tipos de suposiciones sobre la red del MPC:

- Síncrona: Las entidades están de acuerdo en la misma hora actual y se establecen momentos límite para la recepción de mensajes. Existe la noción de "rondas" de comunicación, por lo que si después de t tiempo no llegan los mensajes entonces se supone que no los envió.
- Asíncrona

Proporción de entidades corruptas

- Mayoría deshonesta: Al menos una entidad es honesta, es decir $t < n$
- Mayoría honesta: Al menos una mayoría (estricta) de las entidades son honestas, i.e. $t < \frac{n}{2}$
- Mayoría honesta $\frac{2}{3}$: Al menos dos tercios de las entidades son honestas, i.e., $t < \frac{n}{3}$

Capacidades del adversario Es lo que vimos anteriormente:

- Seguridad pasiva
- Seguridad activa

Movilidad del adversario

- Seguridad estática: El adversario especifica qué entidades van a ser corrompidas antes de comenzar la ejecución del protocolo de MPC.
- Seguridad adaptativa: El adversario puede decidir qué entidades corromper o controlar de forma adaptativa durante la ejecución del protocolo.

Capacidad computacional del adversario

- Seguridad incondicional: La seguridad se basa en teoría de la información y estadística. No se impone ninguna restricción en cuanto a la cantidad de recursos computacionales del adversario. Imposible de obtener cuando la mayoría es deshonesta (es un teorema). Se divide en seguridad estadística (probabilidad despreciable) o perfecta.
- Seguridad computacional: El adversario es modelado como una máquina de Turing probabilística en tiempo polinomial en vez de tener una capacidad limitada. Es decir, la seguridad reposa sobre la hipótesis de que algún problema específico es difícil de resolver en tiempo polinomial.

Garantías sobre el output Hay tres posibilidades:

- Entrega garantizada: las entidades honestas siempre reciben el output
- Justo: si las entidades corruptas reciben el output, entonces también lo reciben las honestas. El adversario puede decidir si todos tienen el output o ninguno.
- Seguridad con aborto: las entidades corruptas pueden recibir el output mientras impiden que las honestas lo hagan. Pueden parar el protocolo cuando quieran. Aquí puede darse la situación en las que las entidades corruptas aprendan el output pero las demás no.

Con ello, es sabido cuáles combinaciones de las dimensiones son factibles o no de realizar.

TODO: Agregar el cuadro de MPC de las filminas

2.4. MPC a partir de compartición de secretos en circuitos aritméticos

Para calcular una función, lo que se hace s:

1. Cada entidad comparte sus inputs secretos correspondientes Sucesivamente calculamos los valores intermedios y outputs del circuito, en forma de secreto compartido. Las operaciones lineales no requieren de interacción si el esquema de compartición de secretos es lineal. Productos u operaciones no lineales requieren (casi siempre) interacción.

2. Reconstruimos los outputs a las entidades que les correspondan dichos resultados.

Compartición de secretos de Shamir: Muliplicación Sean $[x]_t, [y]_t$ dos secretos compartidos mediante Shamir para umbral t, entonces si cada P_i calcula localmente $x_i \cdot y_i$, las entidades obtienen de forma segura el secreto compartido $[x \cdot y]_{2t}$.

Nota. Lo complejo es que si el grado del polinomio supera n entonces ya no es posible realizar la reconstrucción. Por esto tenemos que tener muy presente que el grado de los secretos jamás supere n.

MPC en el estilo de BGW, seguridad pasiva Estamos considerando circuitos aritméticos (modelo de computación), red síncrona, sin preprocesado y con el adversario pasivo y estático, vamos a alcanzar seguridad perfecta con una mayoría honesta ($t < \frac{n}{2}$). La idea general es la vista antes para protocolos de compartición de secretos con multiplicación.

El protocolo de multiplicación considera que sean $[x]_t, [y]_t$ dos valores a multiplicar entonces:

1. Las entidades calculan de forma local $[z]_{2t} = [x \cdot y]_{2t}$. Denotemos las partes del secreto como $(z_1, \ldots, z_n)$.

2. Cada entidad P_i usa el esquema de Shamir para compartir su parte $[z_i]_t$ con el resto de entidades.

3. De forma local, las entidades calculan $[x \cdot y]_t = \sum_{i=1}^{2t+1} \lambda_i \cdot [z_i]_t$, donde λ_i son los coeficientes de interpolación de Lagrange.

Nota. Se impone $t < \frac{n}{2}$ para que el grado del polinomio $[z]_{2t}$ no exceda n dado que sino no tendríamos la suficiente cantidad de puntos para interpolar. El protocolo tampoco es seguro contra un adversario activo.

MPC en el estilo de BGW, seguridad activa Es un protocolo muy parecido al anterior, solo que tenemos un preprocesado (independiente de f, únicamente con cota superior de cantidad de multiplicaciones que hay), adversario activo, seguridad con aborto (podría ser mejor y obtenerse output garantizado), y con un umbral de corrupción $t < \frac{n}{3}$. Las herramientas que se usan son la idea general para protocolos basados en compartición de secretos, compartición de secretos de Shamir y pares de reducción de grado.

En el esquema del protocolo el adversario puede comportarse mal de tres formas distintas:

- En la fase de compartición:
  - No compartir su secreto: no se puede hacer nada
  - Mentir y enviar un input incorrecto: no se puede hacer nada pero a lo sumo se puede crear un circuito que verifique si el input satisface las condiciones correctas (como estar en cierto rango)
  - Enviar un secreto como un polinomio de grado distinto de t: nos vamos a concentrar en esto. Vamos a usar el preprocesado para esto.
- En la fase de cálculo, el inconveniente viene en donde nos encontramos con una puerta de multiplicación (o no lineal) que requiera interacción y donde podamos mentir (o al final con el output). En este caso, como vimos antes, el protocolo de multiplicación no es seguro contra un adversario activo.
- Fase de reconstrucción de los outputs: el mismo, mentir.

Podemos pensar los esquemas de Shamir como algoritmos de detección de errores de Reed Solomon, por lo que podremos ser capaces de identificar el error y la entidad corrupta. Veamos cómo solucionar cada parte:

- Protocolo de repartición de inputs: Consideramos lo siguiente
  - Preprocesado: un secreto correctamente compartido $[r]_t$ donde r es uniformemente aleatorio y únicamente conocido por P_i
  - Inputs: x, el input secreto de P_i que nos queremos asegurar de que se comparte correctamente
  - Outputs: $[x]_t$

Luego, vamos a hacer:

1. P_i hace broadcast del valor $e = x - r$ (un canal de broadcast se asegura de que todas las entidades reciben el mismo mensaje, se puede hacer simplemente con que todas las entidades hagan echo)
2. De forma local, las entidades calculan $[x]_t = e + [r]_t$

- Protocolo de multiplicación (double random share): Consideramos el preprocesado dado por
  - Preprocesado: un par de reducción de grado $([r]_t, [r]_{2t})$ donde r es uniformemente aleatorio
  - Inputs: $[x]_t, [y]_t$ valores a multiplicar
  - Outputs: $[x \cdot y]_t$

  Luego, vamos a hacer:

  1. Las entidades calculan de forma local $[x \cdot y]_{2t}$
  2. También de forma local, calculan $[z]_{2t} = [x \cdot y]_{2t} - [r]_{2t}$
  3. Las entidades reconstruyen públicamente z (canal de broadcast), aplicando detección de errores
  4. De forma local, las entidades calculan $[x \cdot y]_t = z + [r]_t$

  Nota. Para el paso 2, notemos que r es desconocido por todos y se debe usar una sola vez (sin volverlo a repetir). En cuanto al paso 3, podemos ver $[z]_d$ como una palabra en un Código de Reed-Solomon $[n, d+1, n-d]$. Por ello, cuando $t < \frac{n}{3}$ entonces se pueden detectar errores con $d \leq 2t$ y corregir con $d \leq t$.

  La analogía entre Shamir y Reed-Solomon se puede ver como:

Variable	Shamir	Reed-Solomon
n	Cantidad de entidades	Longitud del código
t	Cantidad de corruptos	Cantidad de errores
d	Grado del polinomio	Dimensión - 1
$t \geq n - d$	Inseguro	No hay detección de errores
$t < n - d$	"Seguridad con aborto"	Detección de errores
$2t < n - d$	Entrega garantizada	Existen algoritmos eficientes de corrección de errores

  En la práctica, lo que se trata de hacer es una reducción de jugadores sucesivamente. A diferencia del código de corrección de errores, la interactividad nos permite mayor posibilidad de detección.

Protocolo de multiplicación (compartición aditiva) El protocolo de multiplicación, si tenemos mayoría deshonesta, se puede implementar usando compartición de secretos aditiva. Para ello, digamos que el secreto es nota como $\langle \cdot \rangle$. Luego, consideramos el preprocesado dado por:

- Preprocesado: $\langle a \rangle, \langle b \rangle, \langle c \rangle$ con $c = a \cdot b,\ a, b \in \mathbb{F}$ (desconocidos)
- Input: $\langle x \rangle, \langle y \rangle, \langle \cdot \rangle$ compartición de secretos aditiva

Luego la idea es tomar:

$$x \cdot y = (x + a - a)(y + b - b) = (\varepsilon - a)(\rho - b) \quad = \varepsilon\rho - a\rho - \varepsilon b + ab$$

donde $\epsilon = x + a, \rho = y + b$. Entonces, podemos tomar para la parte del cálculo secreto a: $\varepsilon\rho - \langle a \rangle \rho - \varepsilon \langle b \rangle + \langle c \rangle$ donde ε, ρ pueden publicarse. Con ello, el esquema sería:

1. Calcular $\langle \varepsilon \rangle = \langle x \rangle + \langle a \rangle$ y $\langle \rho \rangle = \langle y \rangle + \langle b \rangle$
2. Reconstruimos ε, ρ
3. Calculamos $\langle x \cdot y \rangle = \varepsilon\rho - \langle a \rangle \rho - \varepsilon \langle b \rangle + \langle c \rangle$ localmente.