Tópicos avanzados en criptografía

2 de julio de 2023

Algunas referencias

El área de cómputo criptográfico está en un auge de investigación, y como tal no cuenta con "textbooks centrales". El contenido de este curso está basado a partir de depurar muchos artículos de investigación, charlas, conferencias, etc., y no seguirá la estructura de un libro o recurso en particular. Sin embargo, se pueden identificar los siguientes recursos que pueden ser útiles como complemento individual, pero de nuevo se advierte que muchos de estos no están diseñados para un primer curso y pueden ser bastante difíciles.

Multiparty Computation

- **Daniel Escudero. "An Introduction to Secret-Sharing-Based Secure Multiparty Computation"** (https://eprint.iacr.org/2022/062.pdf).
  Este texto es el más cercano a lo que se enseñará en el curso en cuanto a MPC. La parte I (MPC fundamentals) debería dar una buena idea del área, y la parte II contiene varios de los protocolos que veremos en marzo 6 y 11 (además de otras construcciones adicionales). La parte III incluye el caso llamado "dishonest majority", que sólo tocaremos superficialmente al comienzo, pero puede ser útil para algunos de los temas avanzados hacia la segunda mitad del curso.
- **Daniel Escudero. "A Crash Course on MPC"** (https://medium.com/applied-mpc/a-crash-course-on-mpc-part-0-311fae2ce184).
  Esta serie de ocho blog posts está diseñada para ser muy liviana y dar una idea general, muy práctica, sobre el área. Cada post es relativamente corto y va directo al punto. No es un recurso formal como un libro y esto ayuda a que sea un poco más laxo con la notación, lo cual lo hace más fácil de seguir.
- **Hazay, Carmit, and Yehuda Lindell. Efficient secure two-party protocols: Techniques and constructions. Springer Science & Business Media, 2010.**
  Este libro es útil para las definiciones básicas, y para hacerse una idea del área. Ver en particular las secciones 1.1 y 1.2 para una introducción al area de 2-party computation, y 2.1, 2.2, 2.3 para definiciones formales.
- **Cramer, Ronald, and Ivan Bjerre Damgård. Secure multiparty computation. Cambridge University Press, 2015.**
  Este libro (*una* copia estará disponible en la biblioteca de la escuela) es bueno como una

introducción al área de MPC para más de dos participantes y "honest majority". Ver el capítulo 1 para una introducción, ver capítulo 3 para un protocolo sencillo que sirve de manera de ejemplo (quizá tendrán que ver el capítulo 2 para obtener los preliminares necesarios). El capítulo 4 puede ser un poco avanzado pero si tienen curiosidad es muy bueno para entender en más detalle cómo funciona el formalismo detrás de las demostraciones de seguridad en MPC.

- **Evans, David, Vladimir Kolesnikov, and Mike Rosulek. .ª pragmatic introduction to secure multi-party computation."Foundations and Trends in Privacy and Security 2.2-3 (2018): 70-246.** (https://securecomputation.org/)
  Este libro es mucho más liviano que los dos de arriba, y es bueno también para hacerse una idea del área y conocer algunas construcciones básicas. Es también más moderno, y en general lo recomiendo más que los de arriba. Ver en particular el capítulo 1, 2, y del capítulo 3 las secciones 3.2, 3.3, 3.4, 3.7, 3.8 y 3.9.
- **Awesome MPC** (https://github.com/rdragos/awesome-mpc).
  Este es un repositorio que contiene una lista bastante extensiva de diferentes recursos sobre MPC. Contiene varios de los recursos de arriba y otros libros más, tutoriales, cursos online, algunos tutoriales, software, herramientas, y mucho más. La verdad es un poco más avanzado pero pueden navegar estos recursos para alimentar su curiosidad.
- **Marcel Keller. MP-SPDZ** (https://eprint.iacr.org/2020/521.pdf, https://github.com/data61/MP-SPDZ).
  Hay muchas "frameworks" que permiten, en la *práctica*, implementar programas que se ejecutan de manera segura en MPC. MP-SPDZ es una de ellas y su repositorio, junto con el artículo que describe el programa, pueden ser buenos recursos para aprender un poco sobre MPC en la práctica y cómo se puede usar en contextos reales.
- Course on Secure Computation, by Ashish Choudhury (https://nptel.ac.in/courses/106108229)

Homomorphic Encryption

- **Awesome homomorphic encryption** (https://github.com/jonaschn/awesome-he)
  Lista curada con recursos acerca de homomorphic encryption. Esto incluye textos pero también frameworks y software en general.
- **Homomorphic encryption en Wikipedia** (https://en.wikipedia.org/wiki/Homomorphic_encryption)
  El artículo sobre HE en Wikipedia es bastante accesible y provee una muy buena introducción al tema, además de un muy buen panorama sobre las diferentes construcciones.
- **Homomorphic Encryption, Shai Halevi (IBM Research)** (https://shaih.github.io/pubs/he-chapter.pdf)
  Buen overview sobre homomorphic encryption
- **Armknecht, Frederik, et al. .ª guide to fully homomorphic encryption.Çryptology ePrint Archive (2015).** (https://eprint.iacr.org/2015/1192.pdf)
  Esta es una introducción relativamente liviana al campo de FHE. Se enfoca más en definiciones, aplicaciones y paradigmas generales, que en construcciones específicas, lo cual lo hace muy accesible.

- **Peikert, Chris. .A decade of lattice cryptography."Foundations and Trends in Theoretical Computer Science 10.4 (2016): 283-424.** (https://eprint.iacr.org/2015/939.pdf)
  Este recurso es **muy avanzado**. Sin embargo, es el más completo en cuanto a descripciones detalladas de las costrucciones basadas en látices, las cuales son práctiamente las más promisorias en la práctica. El documento sólo discute FHE al final, en la sección 6.1.
- **Homomorphic encryption standard** (https://homomorphicencryption.org/standard/)
  Este es un documento iniciado por la comunidad para tratar de estandarizar algunas cosas en el campo de homomorphic encryption, en preparación para usos más prácticos. El documento incluye descripciones generales de algunas de las construcciones que se consideran "estándar" en el campo.
- **Van Dijk, Marten, et al. "Fully homomorphic encryption over the integers."2010** (https://eprint.iacr.org/2009/616.pdf)
  Este es un artículo de investigación (así que puede ser un poco avanzado) que contiene la descripción de un sistema de encripción homomorfica relativamente simple. Es posible que una simplificación de esta construcción sea la que se estudie en clase.

Zero-Knowledge Proofs

- **Thaler, Justin. "Proofs, arguments, and zero-knowledge."Foundations and Trends in Privacy and Security 4.24 (2022)** (https://people.cs.georgetown.edu/jthaler/ProofsArgsAndZK.pdf)
  Este es un texto muy completo que incluye gran parte de la literatura que se conoce en cuando a zero-knowledge proofs. Sin embargo, al ser tan completo, es muy denso y tiene mucho contenido.
- **Zero Knowledge Canon** (https://a16zcrypto.com/zero-knowledge-canon/)
  Una lista anotada con referencias y recursos en ZKP.
- **ZKP science** (https://zkp.science/)
  Una lista con varios recursos y referencias sobre ZKP. Incluye referencias tanto a artículos muy introductorios y accesibles, como a artículos de investigación como tal. Ver en particular las referencias en la primera sección: "What is a zero-knowledge proof?".
- **MOOC ZKP course** (https://zk-learning.org/)
  Curso online que está corriendo justo en este momento (varios videos ya están disponibles, otros se van lanzando semanalmente) dictado por figuras prominentes en el área. Tiene como audiencia una población muy diversa y es muy accesible, empezando desde lo más básico. De la descripción del curso: "This class aims to bring together students and experts in academia and industry to explore Zero-Knowledge Proofs (ZKP)." Altamente recomendado.
- **ZKP reference** (https://docs.zkproof.org/reference.pdf)
  Documento cuyo propósito es estandarizar algunas de las construcciones, interfaces, implementaciones, etc. que existen en el campo de ZKP. Esta hecho con una audiencia bastante general en mente y por lo tanto es bastante accesible en su mayor parte. Directamente de la página: "the ZKProof Community Reference document should be used as a reference

for best practice of the field of zero-knowledge proofs, and is a result of a collaboration between dozens of researchers, developers and practitioners."

- **Yuval Ishai's blog posts** (https://zkproof.org/2020/08/12/information-theoretic-proof-systems/)
  A series of two posts from Yuval Ishai giving a high level overview of the Zero-Knowledge Proof field. This is Part 2: https://zkproof.org/2020/10/15/information-theoretic-proof-systems-part-ii/
- **Awesome ZKP** (https://github.com/sCrypt-Inc/awesome-zero-knowledge-proofs)
  Lista de varios recursos relacionados con ZKP. Está muy sesgada a las aplicaciones y uso de ZKP en el contexto de criptomonedas y blockchains, pero aún asi contiene algunas referencias útiles.
- **Damgård, Ivan. "Commitment schemes and zero-knowledge protocols."** (https://pascholl.github.io/download/ComZK08.pdf)
  Notas de clases del profesor Ivan Damgård para uno de sus cursos. Enfatiza mucho (como el título indica) en la construcción de commitment schemes, los cuales son unas primitivas esenciales en muchas construcciones de ZKP, y además provee una introducción general a zero-knowledge proofs junto con algunas construcciones y resultados básicos.
- **Ivan, Damgård. "On Sigma-protocols." Lecture Notes, University of Aarhus, Department for Computer Science (2002).** (https://cs.au.dk/~ivan/Sigma.pdf)
  Otras notas de clase de Ivan Damgård que enfatizan en Sigma Protocols, los cuales son un tipo específico de zero-knowledge proofs.

Contenido del curso

Referencias

[MPC-notes] Daniel Escudero. "An Introduction to Secret-Sharing-Based Secure Multiparty Computation" (https://eprint.iacr.org/2022/062.pdf).

[Yehuda-Carmit] Hazay, C., & Lindell, Y. (2010). Efficient secure two-party protocols: Techniques and constructions. Springer Science & Business Media.

[CDN] Cramer, R., Damgard, I., & Nielsen, J. B. (2012). Secure multiparty computation and secret sharing-an information theoretic appoach. Book draft. (https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=000e258c57da9a9071673f2a54d87b1f190cb129)

1. Introducción a MPC (27 de febrero)

Se introduce la noción de secure multiparty computation, junto con conceptos fundamentales como adversarios, honest/dishonest majority, seguridad pasiva/activa, seguridad computacional/information-theoretic, entre otros. También se presentan qué protocolos son posibles de obtener en diferentes contextos. Finalmente, se presenta el concepto de (linear) secret-sharing, se explica una receta general que usa un esquema de este tipo para obtener MPC.

Video. https://youtu.be/c4p8QXFcavA

Referencias principales. Secciones 1.1, 1.3, y capítulo 2 de [MPC-notes].

Ejercicios

Ejercicio 1.1 (Toda función es un circuito)**.** *En MPC, FHE y ZKP, es cómun considerar el concepto de circuitos aritméticos, que no son más que polinomios multivariados, para modelar cómputos arbitrarios. En la práctica, esta representación suele no ser tan eficiente, pero en teoría toda función puede ser representada de esta forma. Demostrar esto es el propósito de este ejercicio.*

Considere un campo finito $\mathbb{F}$, y sea $f : \mathbb{F}^n \to \mathbb{F}$ cualquier función. Demuestre que existe un polynomio multivariado $F(\mathtt{X}_1, \ldots, \mathtt{X}_n)$ sobre $\mathbb{F}$ tal que $f(x_1, \ldots, x_n) = F(x_1, \ldots, x_n)$ para todo $(x_1, \ldots, x_n) \in \mathbb{F}^n$. En general, ¿cuál es el grado de F? (recordar que un polynomio multivariado es una suma de monomios de la forma $c \cdot \mathtt{X}_1^{a_1} \cdots \mathtt{X}_n^{a_n}$, y el grado es la suma de exponentes $a_1 + \cdots + a_n$ más grande).

Sugerencia: considere primero el resultado para funciones de la forma $f_{\boldsymbol{c}} : \mathbb{F}^n \to \mathbb{F}$ con $f_{\boldsymbol{c}}(\boldsymbol{x}) = 1$ si $\boldsymbol{x} = \boldsymbol{c}$ y $f_{\boldsymbol{c}}(\boldsymbol{x}) = 0$ en otro caso.

Ejercicio 1.2 (Replicated SS para tres participantes)**.** *Considere* replicated secret-sharing *(RSS), el cual es un secret-sharing scheme lineal que se presenta en el ejemplo 2.2 en [MPC-notes]. Describa explicitamente la construcción para el caso de $(n, t) = (3, 1)$. Una vez hecho esto, demuestre la siguiente afirmación: dados dos valores $[\![x]\!]$ y $[\![y]\!]$ que están compartidos usando RSS con $n = 3$ y $t = 1$, cada una de las tres parties P_1, P_2 y P_3 puede computar* localmente *z_1, z_2 y z_3, respectivamente,*

tales que $z_1 + z_2 + z_3 = x \cdot y$. Esto demuestra que, localmente, las parties pueden obtener lo que se conoce como additive shares del producto de x y y.

2. Honest majority con seguridad pasiva (4 de marzo)

Se presenta la construcción de Shamir secret-sharing. Se presenta un protocolo de MPC para $t < n/2$ (honest majority) que involucra $O(n^2)$ mensajes en total por cada multiplicación. Luego, se explica como se puede obtener comunicación lineal $O(n)$, con la ayuda de los llamados "double-sharings". Finalmente se discute parcialmente cómo generar estos double-sharings.

Video. https://youtu.be/8d9raqVW2y4

Referencias principales. Secciones 3.1, 4.1 y 4.2.1 de [MPC-notes].

Ejercicios

Ejercicio 2.1 (Interpolación de polinomios (ver. 1))**.** *Considere el siguiente teorema: dados cualquier $d+1$ pares $(x_0, y_0), \ldots, (x_d, y_d) \in \mathbb{F}^2$ con $x_i \neq x_j$ para cada $i \neq j$, existe un único polynomio $f(\mathtt{X})$ sobre $\mathbb{F}$ de grado a lo sumo d tal que $f(x_i) = y_i$ para cada $i = 0, \ldots, d$. En este y el siguiente ejercicio, demostraremos este teorema de dos maneras diferentes. Para este ejercicio, usaremos el teorema de la división para la unicidad, y polinomios de Lagrange para la existencia.*

1. *Demuestre que existen polinomios $\ell_0(\mathtt{X}), \ldots, \ell_d(\mathtt{X})$ de grado $\leq d$ tales que, para cada $i = 0, \ldots, d$, $\ell_i(x_i) = 1$ y $\ell_i(x_j) = 0$ si $i \neq j$.*
2. *Use estos polinomios (conocidos como polinomios de Lagrange) para definir un polinomio $f(\mathtt{X})$ que satisface $f(x_i) = y_i$ para $i = 0, \ldots, d$. Esto muestra la existencia*
3. *Para la unicidad, demuestre primero el algoritmo de la división: dados polinomios $p(\mathtt{X})$ y $g(\mathtt{X})$, existe únicos polinomios $q(\mathtt{X}), r(\mathtt{X})$ con $\deg r < \deg g$ tales que $p = g \cdot q + r$. Una posible sugerencia es usa inducción.*
4. *Ahora use el algoritmo de la división para mostrar un polinomio de grado a lo sumo d no puede tener más de $d+1$ raíces diferences.*
5. *Finalmente, use este hecho para mostrar que sólo puede existir un polinomio $f(\mathtt{X})$ tal que $f(x_i) = y_i$ para $i = 0, \ldots, d$.*

Ejercicio 2.2 (Interpolación de polinomios (ver. 2))**.** *La segunda prueba del teorema de interpolación usará matrices de Vandermonde. Consideremos los pares $(x_0, y_0), \ldots, (x_d, y_d) \in \mathbb{F}^2$, y considere la matriz V sobre $\mathbb{F}$ de dimensiones $(d+1) \times (d+1)$ cuya entrada i, j (indexando desde 0 hasta d) está dada por x_i^j. Así, si existe un polinomio $f(\mathtt{X}) = c_0 + c_1\mathtt{X} + \cdots + c_d\mathtt{X}^d$ tal que $f(x_i) = y_i$ para $i = 0, \ldots, d$, este polinomio tiene que satisfacer $\boldsymbol{y} = V \cdot \boldsymbol{c}$, donde $\boldsymbol{y} = (y_0, \ldots, y_d)^\intercal$ y $\boldsymbol{c} = (c_0, \ldots, c_d)^\intercal$. Si V es invertible, obtenemos existencia y unicidad inmediatamente: se puede computar un $\boldsymbol{c}$, y sólo un $\boldsymbol{c}$, como $\boldsymbol{c} = V^{-1} \cdot \boldsymbol{y}$.*

Así, para probar el teorema de interpolación, falta probar que V es invertible. Para esto, pruebe lo siguiente: el determinante de V está dado por $\Pi_{i<j}(x_i - x_j)$. Esto implica que V es invertible pues los puntos $x_0, \ldots, x_d$ son diferentes.

Ejercicio 2.3 (Reconstrucción de Shamir secret-sharing)**.** *Recuerde Shamir secret-sharing: dado un secreto $s \in \mathbb{F}$, este se distribuye como $(f(1), \ldots, f(n))$, donde $f(\mathtt{X}) = s + r_1\mathtt{X}^1 + \cdots + r_t\mathtt{X}^t$, donde $r_1, \ldots, r_t \in \mathbb{F}$ son uniformemente aleatorios. Dado un conjunto $A \subseteq \{1, \ldots, n\}$ de tamaño $t+1$, las shares $f(i)$ para $i \in A$ determinan el polinomio $f(\mathtt{X})$ en particular determinan el secreto $f(0) = s$. Demuestre que existen constantes λ_i para $i \in A$ que sólo dependen del conjunto A (en particular, son independientes de s) tales que $s = \sum_{i \in A} \lambda_i \cdot f(i)$. Adicionalmente, dé una fórmula explícita para estas constantes.*

Ejercicio 2.4 (Privacidad de Shamir secret-sharing)**.** *El propósito de este ejercicio es demostrar que Shamir secret-sharing satisface la privacidad del secreto cuando se conocen a lo sumo t shares. Más precisamente, probaremos que la distribución de cualquier conjunto de t shares es uniformemente aleatorio, independientemente de cual es el secreto. Para esto, demuestre lo siguiente: para todo secreto $s \in \mathbb{F}$, y para cualquier conjunto $A \subseteq \{1, \ldots, n\}$ de tamaño t, hay un mapeo bijectivo entre los $(r_1, \ldots, r_t) \in \mathbb{F}^t$ usados para Shamir SS con el polinomio $f(\mathtt{X}) = s + r_1\mathtt{X}^1 + \cdots + r_t\mathtt{X}^t$, y la tupla de t shares $(f(i))_{i \in A} \in \mathbb{F}^t$. ¿Por qué implica esto que las t shares no revelan nada del secreto s?*

3. Generación de double-sharings, y comienzo de seguridad activa (6 de marzo)

Se presenta un método para generar double sharings $([\![r]\!]_t, [\![r]\!]_{2t})$ que involucra una cantidad lineal $O(n)$ de comunicación por cada double sharing generado. También se inicia la discusión en el tema de seguridad activa. Se discuten algunos ataques en los protocolos vistos hasta este momento, se presenta el concepto de "security up to additive attacks", se muestra que varios ataques en el protocolo de comunicación cuadrática se reducen a additive attacks, y se presenta un protocolo (se postpone su analisis) para verificar que una multiplicación no contiene additive attacks.

Video. https://youtu.be/iX6RaL3aak0

Referencias principales. Sección 4.2.2 de [MPC-notes] para la generación de double-sharings, Capítulo 6 para seguridad activa.

Ejercicios

Ejercicio 3.1 (Un dealer corrupto no puede distribuir sharings de grado alto)**.** *En este ejercicio probaremos que lo peor que un adversario puede hacer al distribuir Shamir sharings es cambiar el secreto, pero sin incrementar el grado del polinomio. Considere n parties $P_1, \ldots, P_n$, entre las cuales t son corruptas. Asumamos que $2t + 1 = n$, es decir, hay $n - t = t + 1$ parties honestas.*

Considere una party P_i que tiene que distribuir un secreto s usando Shamir secret-sharing de grado t, esto es, si P_i hiciera esto correctamente, P_i computaría $f(\mathtt{X}) = s + \sum_{j=1}^{t} r_j\mathtt{X}^j$ y enviaría $f(j)$ a cada party P_j. Sin embargo, si P_i es corrupto, no tiene que computar un polinomio $f(\mathtt{X})$, ni enviar $f(j)$ a cada party P_j. En vez de esto, P_i puede enviar otro valor s_j a cada P_j (asumimos que no enviar ningún mensaje corresponde a enviar, digamos, $s_j = 0$), y puede ser que no exista polinomio f de grado $\leq t$ tal que $s_j = f(j)$ para $j = 1, \ldots, t$. En este caso decimos que $(s_1, \ldots, s_n)$ no *es t-consistente (decimos que* sí *es t-consistente si dicho polinomio existe).*

1. *Demuestre que $(s_1, \ldots, s_n)$ se pueden interpretar como Shamir sharings $[\![s + \epsilon]\!]_{n-1}$ de grado $n - 1$, donde ϵ es un valor conocido por el adversario. Esto muestra que el adversario puede incrementar el grado a lo sumo hasta $n - 1$ (y agregar un error aditivo), pero esto no es suficiente: queremos demostrar que el adversario no puede aumentar el grado más allá de t.*
2. *Supongamos sin pérdida de generalidad que las parties corruptas son las últimas t, y que las primeras $n - t = t + 1$ parties son honestas. Demuestre que el adversario conoce $s'_{t+2}, \ldots, s'_n$ y ϵ tales que $(s_1, \ldots, s_{t+1}, s'_{t+2}, \ldots, s'_n)$ son sharings $[\![s + \epsilon]\!]_t$. Esto muestra que, redefiniendo las shares de las parties corruptas $P_{t+2}, \ldots, P_n$, el ataque se reduce a un additive attack con grado aún t, en vez de $n - 1$.*

Vale la pena pensar: ¿qué pasa si $2t + 1 < n$?

Ejercicio 3.2 (BGW es "secure up to additive attacks")**.** *Recordemos el protocolo para multiplicar dos sharings $[\![x]\!]_t$ y $[\![y]\!]_t$, que tiene comunicación cuadrática:*

1. *Las parties multiplican localmente $[\![x \cdot y]\!]_{2t} \leftarrow [\![x]\!]_t \star [\![y]\!]_t$. Denotemos $[\![x \cdot y]\!]_{2t} = (z_1, \ldots, z_n)$. Dado que $n > 2t$, sabemos que existen $\lambda_1, \ldots, \lambda_n \in \mathbb{F}$ tales que $xy = \sum_{i=1}^n \lambda_i z_i$.*
2. *Cada party P_i distribuye sharings $[\![z_i]\!]_t$. Así, las parties obtienen sharings $[\![z_1]\!]_t, \ldots, [\![z_n]\!]_t$.*
3. *Las parties computan localmente $[\![x \cdot y]\!]_t \leftarrow \sum_{i=1}^n \lambda_i [\![z_i]\!]_t$.*

Nos referiremos a este protocolo como el "protocolo BGW" (son las iniciales de sus autores, Ben-Or, Goldwasser y Widgerson).

Tomemos $2t + 1 = n$. Use el ejercicio anterior para demostrar que BGW es secure up to additive attacks, es decir: suponga que el protocolo BGW se ejecuta con un adversario activo*, entonces, existe $\epsilon \in \mathbb{F}$ conocido por el adversario tal que el resultado del protocolo son sharings $[\![xy + \epsilon]\!]_t$.*

Ejercicio 3.3 (DN07 es "secure up to additive attacks")**.** *Considere el protocolo DN07 para multiplicar $[\![x]\!]_t$ y $[\![y]\!]_t$ (este es el protocolo con comunicación lineal, se llama DN07 por sus autores, Damgård y Nielsen). El protocolo asume que las parties tienen un double-sharing $([\![r]\!]_t, [\![r]\!]_{2t})$:*

1. *Las parties multiplican localmente $[\![xy]\!]_{2t} \leftarrow [\![x]\!]_t \star [\![y]\!]_t$*
2. *Las parties restan localmente $[\![d]\!]_{2t} \leftarrow [\![xy]\!]_{2t} - [\![r]\!]_{2t}$*
3. *Escribamos $[\![d]\!]_{2t} = (d_1, \ldots, d_n)$. Cada party P_i envía su share d_i a una party designada, digamos P_1.*
4. *P_1 recibe $(d_1, \ldots, d_n)$. Como $n > 2t$, existen $\lambda_1, \ldots, \lambda_n$ tales que $f(0) = \sum_{i=1}^n \lambda_i f(i)$ para todo polinomio $f(\mathtt{X})$ de grado $\leq 2t$, y en particular P_1 puede computar $d = \sum_{i=1}^n \lambda_i d_i$.*
5. *P_1 envía d a todas las demás parties*
6. *Después de recibir d, todas las parties computan localmente $[\![xy]\!]_t \leftarrow d + [\![r]\!]_t$*

El objetivo de este ejercicio es demostrar que, si el adversario es activo, lo peor que puede suceder es que el resultado del protocolo sea $[\![xy + \epsilon]\!]_t$, donde el adversario conoce ϵ. Como en ejercicios anteriores, esto requiere $2t + 1 = n$.

Para proceder, siga los siguientes pasos:

1. *Suponga que P_1 es honesto. Demuestre que, en el paso 3, lo peor que puede hacer un adversario activo es causar que P_1 reconstruya $d+\delta$ en vez de d, para un valor δ que el adversario conoce. Use esto para mostrar que en este caso el resultado del cómputo es $[\![xy+\delta]\!]_t$. Por lo tanto, la afirmación se cumple tomando $\epsilon = \delta$.*

2. *Ahora supongamos que P_1 es corrupto. Esta party puede atacar no sólo enviando un valor diferente $d+\gamma$ a todas las parties, pero aún más, puede enviar un valor diferente $d+\gamma_i$ a cada otra party P_i. Denotemos por $(z_1,\ldots,z_n)$ las shares finales que las parties producen en el protocolo, que se computan como $z_i = (d+\gamma_i)+r_i$, donde $(r_1,\ldots,r_n) = [\![r]\!]_t$. Demuestre que es posible redefinir las shares de las parties corruptas de tal manera que el vector de shares que se obtiene corresponde a shares de $[\![xy+\epsilon]\!]_t$, para un valor ϵ que el adversario conoce.*

Ejercicio 3.4 (DN07: double sharings)**.** *El ejercicio anterior muestra que el protocolo DN07 es secure up to additive attacks, pero esto es asumiendo que las parties tienen $([\![r]\!]_t, [\![r]\!]_{2t})$. En clase (y en las notas) se presentó un protocolo para obtener estos double-sharings, usando matrices de Vandermonde. Demostrar que, aún cuando se usa este protocolo para generar los double-sharings, la afirmación de arriba sigue siendo cierta: el protocolo DN07 es secure up to additive attacks.*

Hint: *Este ejercicio es sencillo una vez se hace el ejercicio anterior. La clave es mostrar que, en la generación de las double-sharings, siempre es posible redefinir las shares de las parties corruptas para que los double-sharings producidos sean correctos, es decir, sean $([\![r]\!]_t, [\![r]\!]_{2t})$, donde r es uniformemente aleatorio y desconocido para el adversario. En este punto la prueba continúa como en el ejercicio anterior.*

4. Seguridad activa para honest majority (11 de marzo)

se discute un método para verificar que no occurren errores aditivos en las multiplicaciones con honest majority, que resulta ser esencialmente lo único que un adversario activo puede hacer en los protocolos pasivos que hemos discutido. También iniciamos la discusión sobre la seguridad activa en $t < n/3$.

Video. https://youtu.be/juUbjnhVcrA

Referencias principales. Capítulo 6 de [MPC-notes].

Ejercicios

Ejercicio 4.1 (Detección de errores)**.** *Fije valores diferentes $\alpha_1,\ldots,\alpha_n \in \mathbb{F}$. Recuerde que un vector $(x_1,\ldots,x_n) \in \mathbb{F}^n$ se dice d-*consistente*, si existe un polinomio $f(\mathtt{X})$ de grado $\leq d$ tal que $f(\alpha_i) = x_i$ para cada $i = 1,\ldots,n$. Suponga que $(x_1,\ldots,x_n)$ es d-consistente, y considere un vector $(x'_1,\ldots,x'_n) \in \mathbb{F}^n$. Sea $A \subseteq \{1,\ldots,n\}$ el conjunto de índices i tales que $x_i \neq x'_i$, y sea e una cota superior sobre $|A|$. Demuestre que si $e < n-d$, entonces $(x'_1,\ldots,x'_n)$ sólo puede ser d-consistente si $A = \emptyset$, es decir, si $(x'_1,\ldots,x'_n) = (x_1,\ldots,x_n)$.*

En particular, en el contexto de MPC, A serán los índices de shares modificadas por el adversario, y tenemos que $|A| < t$ (así que podemos tomar $e = t$). Entonces:

- *Si $t < n/2$ entonces podemos aplicar detección de errores para polinomios de grado $d = t$, pues $e < n-d$ se traduce en $t < n-t$, lo cual es $t < n/2$.*

- *Si $t < n/3$ entonces podemos aplicar detección de errores para polinomios de grado $d = 2t$, pues $2e < n - d$ se traduce en $2t < n - t$, lo cual es $t < n/3$.*

Ejercicio 4.2 (No se puede hacer detección de errores si $t \geq n/2$)**.** *La idea de este ejercicio es mostrar que para hacer detección de errores es necesario, en términos de la notación del ejercicio anterior, que $e < n - d$. Para esto, suponga que $e = n - d$ (en particular $e \geq n - d$). Demuestre que, dado un vector $(s_1, \ldots, s_n)$ que es d-consistente con un polinomio $f(\mathtt{X})$, para cualquier elección de $\epsilon \in \mathbb{F}$ es posible modificar e entradas de $(s_1, \ldots, s_n)$ de tal manera que el vector resultante es d-consistente con un polinomio $f'(\mathtt{X})$ de grado $\leq d$ tal que $f'(0) = f(0) + \epsilon$.*

En el contexto de MPC, tomando por ejemplo $e = t$ y $d = t$, esto muestra que si $t \geq n/2$, entonces el adversario puede modificar t shares de las parties corruptas de tal manera que el secreto se altera por un valor aditivo de ϵ. ¿cómo puede un adversario explotar esto para pasar el check que vimos en clase para verificar multiplication triples?

Ejercicio 4.3 (Detección de errores y comunicación lineal)**.** *A raíz del primer ejercicio, dado un secreto $[\![s]\!]_d = (s_1, \ldots, s_n)$, si una party recibe shares $(s'_1, \ldots, s'_n)$, donde hay a lo sumo $t < n - d$ shares modificadas, entonces esta party puede hacer detección de errores de tal forma que esta party reconstruye el secreto s, o no reconstruye ningún valor. Sin embargo, la manera en la que todas las parties pueden obtener las shares de $[\![s]\!]_d$ es enviándose todas sus shares entre sí, lo cual es problemático en términos de comunicación pues es cuadrática en n. Para seguridad pasiva arreglamos esto pidiéndole a las parties enviar sus shares a una party designada, P_1, la cual reconstruye el secreto y envía el resultado a las demás parties. Esto resulta en comunicación lineal, pero no funciona para seguridad activa pues, a pesar de que P_1 puede usar detección de errores, nada le garantiza a las parties que el valor que P_1 les envía es la reconstrucción correcta.*

El objetivo de este ejercicio mostrar un protocolo para la reconstrucción, que tenga comunicación lineal. Considere la reconstrucción no de un sólo secreto, sino $d + 1$ secretos $[\![s_0]\!], \ldots, [\![s_d]\!]$ (el grado de los polinomios es d, el cual omitiremos). Sean $\alpha_1, \ldots, \alpha_n \in \mathbb{F}$ puntos diferentes de $\mathbb{F}$ (pueden ser diferentes a los puntos usados en Shamir secret-sharing). El protocolo es el siguiente:

1. *Para cada, las parties computan localmente $[\![u_i]\!] = \sum_{j=0}^{d} \alpha_i^j \cdot [\![s_j]\!]$, y envian sus shares de $[\![u_i]\!]$ a la party P_i.*
2. *Cada P_i receive las shares de $[\![u_i]\!]$, y usa detección de errores para reconstruir u_i (o abortar si la reconstrucción falla). En caso de reconstrucción exitosa, P_i envía u_i a las demás parties.*
3. *Cada party P_k recibe $(u_1, \ldots, u_n)$. Cada P_k busca un polinomio $f'(\mathtt{X})$ de grado $\leq d$ tal que $f'(\alpha_i) = u_i$ para cada $i = 1, \ldots, n$. Si existe, P_k toma como resultado de la reconstrucción los coeficientes $s'_0, \ldots, s'_d$ de $f'(\mathtt{X})$.*

Demuestre lo siguiente:

1. *Si $d = \Theta(n)$ (es decir, si $d = c \cdot n$ para alguna constante c), entonces la comunicación del protocolo por cada reconstrucción es lineal.*
2. *Si una party P_k no aborta y produce como resultado $s'_0, \ldots, s'_d$, entonces se cumple que $s'_i = s_i$ para cada $i \in \{0, \ldots, d\}$.*

5. Seguridad activa y perfecta para $t < n/3$ (13 de marzo)

Se presenta un método para generar double sharings $([\![r]\!]_t, [\![r]\!]_{2t})$ con seguridad activa y perfecta en el caso de $t < n/3$, que involucra una cantidad lineal $O(n)$ de comunicación por cada double sharing generado. Esto hace uso de matrices hiper-invertibles, en las cuales toda submatriz cuadrada es invertible. También se discute el concepto de corrección de errores, y cómo para $t < n/3$ sharings de grado t se pueden reconstruir correctamente sin abort. Esto se usa para obtener MPC con "guaranteed output delivery", asumiendo tuplas multiplicativas. Finalmente, se presenta brevemente un protocolo para $t = 1$ y $n = 3$ usando replicated secret-sharing, con el propósito de ilustrar cómo se pueden lograr protocolos más simples y eficientes y el número de parties es pequeño.

Video. https://youtu.be/GHcycX8WJng

Referencias principales. Capítulo 5 de [MPC-notes] para la generación de double-sharings, Sección 3.2.2 para corrección de errores.

Ejercicios

Ejercicio 5.1 (Corrección de errores con comunicación lineal)**.** *Demuestre cómo extender el ejercicio 4.3 para reconstruir $[\![s_0]\!]_t, \ldots, [\![s_d]\!]_t$ con corrección de errores (es decir, garantizando la reconstrucción correcta, sin abort), manteniendo comunicación lineal, en el caso de $t < n/3$.*

Para esto, use el siguiente hecho sobre corrección de errores: suponga que $2e < n - d$; dado un vector $\boldsymbol{x} = (x_1, \ldots, x_n)$ que es d-consistente, y dado $\boldsymbol{x}' = (x'_1, \ldots, x'_n)$ que difiere de $\boldsymbol{x}$ en a lo sumo e entradas, existe un algoritmo que toma $\boldsymbol{x}'$ como entrada, y regresa el vector $\boldsymbol{x}$.

Ejercicio 5.2 (Corrección de errores requiere $t < n/3$)**.** *El propósito de este ejercicio es mostrar que corrección de errores de Shamir sharings con polinomios de grado t, con t parties corruptas, requiere $t < n/3$. Para esto suponga que $t = n/3$, y considere un vector de sharings t-consistentes $[\![s]\!]_t = (s_1, \ldots, s_n)$. Muestre cómo, para cualquier ϵ, el adversario puede modificar las primeras t sharings (funciona para cualquier conjunto de t sharings) para obtener $(s'_1, \ldots, s'_t, s_{t+1}, \ldots, s_n)$ de tal manera que, si se remueven las t componentes $s_{t+1}, \ldots, s_{2t+1}$, el vector resultante $(s'_1, \ldots, s'_t, s_{2t+2}, \ldots, s_n) \in \mathbb{F}^{n-t}$ es t-consistente y más aún el secreto que se reconstruiría es $s + \epsilon$.*

Esto muestra a un nivel intuitivo que corrección de errores no es posible para $t = n/3$: existen dos conjuntos de tamaño t que se pueden remover y que llevan a vectores t-consistentes con secretos diferentes: $(s_{t+1}, \ldots, s_n)$ es t-consistente con el secreto s, y $(s'_1, \ldots, s'_t, s_{2t+2}, \ldots, s_n)$ es t-consistente con el secreto $s + \epsilon$.

6. Definiciones de seguridad pasiva y activa (18 de marzo)

Se presenta el paradigma de simulación para definir formalmente la noción de que un protocolo de MPC es seguro. También, se usa este paradigma para demostrar la seguridad de dos de los protocolos que vimos en sesiones anteriores

Video. Parte A: https://youtu.be/r_4RG6wtDxc, Parte B: https://youtu.be/hE8q7WBLP_0

Referencias principales. Secciones 2.2 y 2.3 de [Yehuda-Carmit] para las definiciones de seguridad pasiva y activa, respectivamente. Secciones 2.2, 2.3 y 2.4 de [CDN] para las definiciones de indistinguishability, y capítulo 4 de [CDN] para el UC-model (mucho más completo y complejo que como lo vimos en clase)

Ejercicios

Ejercicio 6.1 (Demostración de seguridad pasiva para DN07)**.** *Considere la función $w = f(x, y, z) = xy + z$ con tres parties que usamos como ejemplo en clase, y suponga que P_1 es corrupto. En clase, probamos que el protocolo BGW (ver ejercicio 3.2) en su versión pasiva es seguro, y también probamos que la versión activa que usa tuplas multiplicativas para verificar la multiplicación es segura (esta idea también se conoce como* sacrificing*, nombre cuyo origen está en el hecho de que una tupla se "sacrifica" para chequear si otra tupla es correcta). El objetivo de este ejercicio es demostrar que el protocolo DN07 que vimos en clase (este es el protocolo que usa double-sharings, ver ejercicio 3.3) es seguro. Comenzaremos con la vesión pasiva.*

Demuestre que el protocolo DN07 (para computar $f(x, y, z) = xy + z$ asumiendo P_1 corrupto), el cual asume la existencia de double-sharings $([\![r]\!]_t, [\![r]\!]_{2t})$, es pasivamente seguro. Note que en la demostración, cuando el simulador genera la view *del adversario, el simulador puede elegir las shares de $([\![r]\!]_t, [\![r]\!]_{2t})$ que P_1 posee. Dé la demostración en tanto detalle como la que se dio en clase para BGW*

Ejercicio 6.2 (Demostración de seguridad activa para DN07)**.** *Ahora extenderemos el ejercicio anterior para mostrar la seguridad activa del protocolo DN07. De nuevo, consideramos $n = 3$, $f(x, y, z) = xy + z$, y asumimos que P_1 es la party (activamente) corrupta. El protocolo es similar al que vimos en clase para la seguridad activa, con la diferencia que la primera multiplicación que obtiene $[\![x \cdot y]\!]_1$ a partir de $[\![x]\!]_1$ y $[\![y]\!]_1$ utiliza double-sharings como en DN07, y no las ideas del protocolo BGW como se hizo en clase.*

1. *Escriba el protocolo explícitamente como lo hicimos en clase. La única diferencia con respecto al que ya estudiamos es que se usan double-sharings para el producto. Para escribir esto formalmente, considere una funcionalidad $\mathcal{F}_{\mathsf{Rand}}$ que se encarga de distribuir double-sharings: la funcionalidad internamente muestrea $r \in \mathbb{F}$, genera shares $([\![r]\!]_t, [\![r]\!]_{2t})$, y distribuye estas shares a las parties.*
2. *Demuestre que este protocolo es seguro usando el paradigma de simulación para seguridad activa. Este protocolo usa tres funcionalidades: $\mathcal{F}_{\mathsf{Rand}}$, $\mathcal{F}_{\mathsf{Coin}}$ y $\mathcal{F}_{\mathsf{Triple}}$ (las últimas dos vistas en clase), recuerde que en la simulación, el simulador corre estas funcionalidades internamente.*

7. Dishonest majority con seguridad pasiva (27 de marzo)

Se presenta un protocolo de MPC para dos parties básico, usando multiplication triples. Luego, se crean estas triples usando Oblivious Transfer (OT) para el caso del campo F2, y usando Oblivious Linear Evaluation (OLE) para el campo Fp. Finalmente, se construye un protocolo de OT usando el criptosistema ElGamal, y se construye OLE usando el criptosistema de Paillier, que es aditivamente homomórfico.

Video. https://youtu.be/zF1Z5ahxojk

Referencias principales. Capítulo 7 de [MPC-notes].[1]

Ejercicios

OT Extension

Recuerde que OT se define como una funcionalidad de dos parties donde Alice tiene dos mensajes m_0, m_1, Bob tiene un bit $b \in \{0,1\}$, y Bob debe recibir m_b sin aprender nada sobre m_{1-b}, mientras que Alice no aprende nada sobre b. Supongamos que $m_0, m_1 \in \{0,1\}^\ell$, es decir, m_0 y m_1 son cadenas de ℓ bits. Denotemos por OT_ℓ una instancia de OT con mensajes de ℓ bits. En clase usamos OT_1 para producir multiplication triples en 2-party computation.

Esta primera colección de ejercicios tienen como propósito estudiar el tema de *OT extension*. Como se mencionaba en clase, OT *requiere* herramientas de criptografía de llave pública, que tienden a ser en general mucho más costosas que técnicas de criptografía simétrica. El objetivo de OT extension es construir muchas instancias de OT a partir de unas pocas instancias, usando solamente criptografía simétrica (por ejemplo, funciones hash, pseudorandom functions, encripción simétrica, etc.), generar muchos más OTs. Este resultado es crucial pues permite minimizar el uso de herramientas costosas. En el contexto de MPC donde necesitamos una instancia de OT por cada multiplication gate, OT extension nos permite sólo usar criptografía de llave pública para una cantidad limitada de OTs, independientemente de la cantidad de multiplicaciones que necesitemos.

Demostraremos este importante resultado en varios pasos. Primero, introduciremos unas variantes de OT. Denotamos por ROT_ℓ (proveniente de *Random OT*) la funcionalidad que es exactamente igual que OT_ℓ excepto que el input de Alice (r_0, r_1) no es elegido por ella, sino que son cadenas de ℓ bits *aleatorias*. Usemos OT_ℓ^m y ROT_ℓ^m para denotar m instancias diferentes de OT/ROT, es decir, Alice tiene m pares de mensajes de ℓ bits (elegidos por ella para el caso de OT, y aleatorios para ROT), y bob tiene m "choice" bits. Una última variante que consideraremos es COT_ℓ^m, que significa correlated OT. Esto es parecido a ROT_ℓ^m en el hecho de que Alice no elige sus mensajes $r_0, r_1 \in \{0,1\}^\ell$ en cada una de las m instancias, pero en COT estos mensajes no son aleatorios como en ROT. En vez de esto, los mensajes $r_0^{(i)}, r_1^{(i)}$ de la i-ésima instancia (para $i \in [m]$) son aleatorios pero restringidos a satisfacer la relación $r_0^{(i)} \oplus r_1^{(i)} = \Delta$, donde $\Delta \in \{0,1\}^\ell$ es una constante fija que Alice elige para todas las m instancias.

Nuestro objetivo final es obtener OT_ℓ^m para valores de m y ℓ "grandes", sin usar muchas operaciones costosas de public-key crypto. Esto lo haremos en varios pasos distribuidos en los siguientes ejercicios. Sea σ el parámetro de seguridad computacional.[2] Lo que demostraremos es lo siguiente (una flecha $A \implies B$ quiere decir que a partir de A podemos obtener B, sin mucho costo):

$$\mathsf{ROT}_\sigma^\sigma \overset{(1)}{\implies} \mathsf{OT}_m^\sigma \overset{(2)}{\implies} \mathsf{COT}_\sigma^m \overset{(3)}{\implies} \mathsf{ROT}_\ell^m \overset{(4)}{\implies} \mathsf{OT}_\ell^m.$$

Así, uniendo los dos extremos, vemos que podemos obtener m instancias de OT_ℓ a partir de σ instancias de ROT_σ. Recuerde que σ es *constante* (por ejemplo, $\sigma = 128$), así que esto dice que sólo tenemos que hacer cómputo con criptografía de llave pública para generar 128 instancias de

[1] Nota: el concepto de "products-to-sums conversion" que se vio en clase es un poco diferente al presentado en las notas.

[2] Es decir, queremos que para atacar nuestra construcción, el atacante tenga que hacer pasos que son polinomiales en σ, típicamente 2^σ. Para efectos prácticos σ será la longitud de las llaves para los PRFs, encripciones, etc. y suele ser por ejemplo $\sigma = 128$.

(random) OT, y a partir de estas podemos generar tantas instancias como queramos usando sólo symmetric key crypto.

Ejercicio 7.1 (Random OT)**.** *Este ejercicio muestra las implicaciones (1) y (4): mostraremos que una instancia de ROT implica una instancia de OT.*

1. *Primero, veamos que ROT puede ser útil también por sí solo. En clase usamos* OT_1 *para construir las multiplication triples requeridas para 2-party computation. Demuestre que Alice y Bob pueden usar* ROT_1 *en vez de chosen-input OT para generar estas triples (Hint: no hay que hacer cambios al protocolo).*

2. *El paso anterior muestra que para el propósito de generar triples, ROT es tan bueno como OT. Sin embargo, en otros casos chosen-input OT es realmente necesario (por ejemplo, si Alice y Bob quieren usar OT para muEjercicioslticar* $[\![x]\!]$ *y* $[\![y]\!]$ *directamente). Demuestre que* ROT_ℓ *se puede usar para implementar* OT_ℓ *(Hint: Alice puede usar* r_0, r_1 *como one-time-pads...)*

3. *Lo anterior muestra que* ROT_ℓ^m *implica* OT_ℓ^m*. Note que los parámetros* m *y* ℓ *son los mismos, lo cual está bien para la implicación (4). Sin embargo, la implicación (1) requiere empezar con longitud* σ *para ROT, y obtener longitud* m *para OT.*

   Considere una pseudorandom function PRF *que toma como input valores* $x \in \{0,1\}^\sigma$*, donde y produce output* $y \in \{0,1\}^\ell$*, con la garantía de que si* x *es aleatorio entonces* y *será esencialmente aleatorio. Demuestre que* ROT_ℓ *se puede implementar sin interacción a partir de* ROT_σ*. Esto muestra que la longitud* ℓ *de los mensajes de Alice en ROT no importa mucho, pues basta obtener ROT para mensajes aleatorios de longitud* σ*, para así obtener ROT para cualquier longitud* $\ell \geq \sigma$ *(en particular, obtenemos la implicación (1) tomando* $\ell = m$*).*

Ejercicio 7.2 (COT implica ROT)**.** *Ahora mostraremos la implicación (3), demostrando que una instancia de COT con longitud* σ *se puede usar para obtener una instancia de ROT con longitud* ℓ*. Considere una función hash* H *que toma inputs* $x \in \{0,1\}^\sigma$ *y da como output* $y \in \{0,1\}^\ell$*, garantizando que cada output* y *es esencialmente aleatorio.*[3] *Argumente que, dado* COT_σ*, se puede obtener* ROT_ℓ^m *para cualquier* $\ell \geq \sigma$*, sin interacción. (Hint: esto es simple, aplique la función* H *a los mensajes de Alice; no trate de demostrar formalmente pues para esto necesitaríamos definir las propiedades de* H *en más detalle.)*

Ejercicio 7.3 (OT Extension)**.** *Ahora demostraremos la implicación (2), que dice que* COT_σ^m *se puede construir a partir de* OT_m^σ*. Aquí es donde la "extensión" realmente ocurre, dado que* σ *instancias de (chosen-input) OT se convierten en* m *instancias de (correlated) OT.*

Queremos obtener m *instancias de* COT_σ*. Denotemos los bits de Bob en estas instancias como* $b^{(1)}, \ldots, b^{(m)} \in \{0,1\}$*. Lo que queremos es que para cada* $i \in [m]$ *Alice tenga pares de mensajes* $(r_0^{(i)}, r_1^{(i)})$ *donde* $r_0^{(i)} \oplus r_1^{(i)} = \Delta$ *y que Bob reciba* $r_{b^{(i)}}^{(i)}$*. Recuerde que tanto los* $r_j^{(i)}$ *como* Δ *son elementos de* $\{0,1\}^\sigma$*. Usaremos la notación* $r_j^{(i)}[k]$ *y* $\Delta[k]$ *para tomar el* k*-ésimo bit de estos vectores.*

Para crear estos vectores, Alice y Bob proceden así: primero, Alice elige $\Delta \in \{0,1\}^\sigma$ *localmente. Recuerde que podemos asumir que tenemos* OT_m^σ*. Alice y Bob usan esta funcionalidad, pero crucialmente, Alice y Bob intercambian roles: Bob será quien provee los dos mensajes en cada una de las* σ *instancias de OT, y Alice será quien toma uno de los dos mensajes. Concretamente, Bob va a tomar sus dos mensajes en cada instancia* $k \in [\sigma]$ *como* $m_0^{(k)} = (t_0^{(1)}[k], \ldots, t_0^{(m)}[k]) \in \{0,1\}^m$ *y* $m_1^{(k)} = m_0^{(k)} \oplus (b^{(1)}, \ldots, b^{(m)})$*, donde* $t_0^{(i)} \in \{0,1\}^\sigma$ *para* $i \in [m]$ *son vectores que Bob muestrea localmente, y Alice usa* $\Delta[1], \ldots, \Delta[\sigma]$ *como sus "choice-bits" en las* σ *instancias.*

[3] Esto se modela usando lo que se conoce como el "random oracle", pero omitiremos la formalización de esto.

1. *Demuestre que, después de llamar* OT_m^σ *con estos inputs, Alice obtiene* $t_0^{(i)}[k] \oplus \Delta[k] \cdot b^{(i)}$ *para* $k \in [\sigma]$ *y* $i \in [m]$.

2. *Sean* $r_0^{(i)} = t_0^{(i)} \oplus \Delta \cdot b^{(i)}$ *y* $r_1^{(i)} = r_0^{(i)} \oplus \Delta$, *vectores que pertenecen a* $\{0,1\}^\sigma$. *Demuestre que, para* $i \in [m]$*:*

   a) *Alice tiene* $(r_0^{(i)}, r_1^{(i)})$

   b) $r_0^{(i)} \oplus r_1^{(i)} = \Delta$

   c) *Bob tiene* $r_{b^{(i)}}^{(i)}$

   Esto demuestra que Alice y Bob obtienen m *instancias de COT, como se quería.*[4]

Otros ejercicios

Ejercicio 7.4 (Generar triples para n parties)**.** *Considere* n *parties* $P_1, \ldots, P_n$ *que quieren generar una multiplication triple* $([\![a]\!], [\![b]\!], [\![a \cdot b]\!])$. *Aquí,* $[\![\cdot]\!]$ *representa additive secret-sharing entre los* n *participantes, es decir,* $[\![x]\!] = (x_1, \ldots, x_n)$ *donde* $x = \sum_{i=1}^n x_i$. *Así, lo que queremos es que cada* P_i *tenga* (a_i, b_i, c_i) *tales que* $\sum_{i=1}^n c_i = (\sum_{i=1}^n a_i) \cdot (\sum_{i=1}^n b_i)$.

En clase vimos como generar triples para el caso $n = 2$. *Muestre cómo generalizar esto para* $n > 2$, *basado en OLE para* $\mathbb{F}_p$ *o OT para* $\mathbb{F}_2$. ***(Hint:*** *escriba* $(\sum_{i=1}^n a_i) \cdot (\sum_{i=1}^n b_i) = \sum_{i=1}^n \sum_{j=1}^n a_i b_j$ *y asuma que cada par de parties* (P_i, P_j) *puede usar OT/OLE para hacer conversiones de producto a suma)*

Ejercicio 7.5 (Comunicación lineal usando threshold AHE)**.** *La solución anterior tiene comunicación* n^2, *dado que hace uso de* n^2 *OTs/OLEs (uno para cada par de parties* (P_i, P_j)*). Mostraremos ahora cómo obtener una solución con comunicación lineal. Recuerde las propiedades de un additive homomorphic encryption (AHE) scheme: se pueden encriptar textos planos* x *usando la llave pública* pk, *obteniendo* $\mathsf{Enc}_{\mathsf{pk}}(x)$, *y este se puede desencriptar usando la llave privada* sk *para obtener* x*; más aún, dos textos cifrados* $\mathsf{Enc}_{\mathsf{pk}}(x)$ *y* $\mathsf{Enc}_{\mathsf{pk}}(y)$ *se pueden* sumar *para así obtener* $\mathsf{Enc}_{\mathsf{pk}}(x + y)$. *En clase, usamos un AHE para construir OLE.*

Una variante de AHE es threshold *AHE. En un esquema threshold AHE hay también una llave pública* pk*, pero ahora no hay una llave privada, sino que hay multiples llaves privadas* $\mathsf{sk}_1, \ldots, \mathsf{sk}_n$*, cada una conocida por cada party. Como antes, se pueden encriptar textos planos* x *usando la llave pública* pk*, obteniendo* $\mathsf{Enc}_{\mathsf{pk}}(x)$*, y se pueden sumar textos cifrados* $\mathsf{Enc}_{\mathsf{pk}}(x)$ *y* $\mathsf{Enc}_{\mathsf{pk}}(y)$ *para obtener* $\mathsf{Enc}_{\mathsf{pk}}(x+y)$. *La diferencia es en la desencripción: dada una encripción* $\mathsf{Enc}_{\mathsf{pk}}(x)$*, cada party* P_i *puede aplicar* sk_i *para obtener un valor* s_i*, los cuales satisfacen* $x = s_1 + \cdots + s_n$. *En otras palabras, cada party usa su llave* sk_i *para obtener una additive share del texto plano* x. *El esquema de Paillier que se vio en clase se puede convertir en threshold (no discutiremos cómo).*

Muestre cómo las parties pueden usar un esquema threshold AHE para obtener una triple con comunicación lineal. ***Hint:*** *Como en algunas de las construcciones que hemos visto antes, comunicación lineal se logra designando a una party que va a recibir mensajes, y luego envía mensajes de vuelta. El protocolo requerirá más de dos rondas. En la primera, las parties le envían encripciones a la party designada, la cual efectuará una suma, y devolverá la suma encriptada. ¿Puede ver cuales serán los siguientes pasos?*

[4] Técnicamente faltaría demostrar que los r's siguen la distribución correcta, es decir, son uniformemente aleatorios sujetos a que su XOR es Δ, y Bob sólo debe recibir $r_{b^{(i)}}$, sin aprender nada sobre $r_{1 \oplus b^{(i)}}$.

8. Seguridad activa para la fase online en dishonest majority (1 de abril)

Se discute cómo lograr seguridad activa para el caso de dishonest majority, en la fase online. Para esto se introduce la noción de "secret-sharing autenticado", y se muestra posteriormente cómo usar esta idea en el contexto de MPC con dos parties. Finalmente, se muestra cómo extender esto para el caso de múltiples parties, lo cual conlleva a lo que esencialmente es la fase online del protocolo que se conoce como "SPDZ".

Video. https://youtu.be/PTaDr4wHt7o

Referencias principales. Capítulo 8 de [MPC-notes]

Ejercicios

Ejercicio 8.1 (Probabilidad de error en SPDZ)**.** *Recordemos que en SPDZ queremos verificar que si una cantidad N the reconstrucciones se hicieron correctamente. Más precisamente, las parties tienen authenticated additive shares $\langle s_1 \rangle, \ldots, \langle s_N \rangle$, donde, recuerde, $\langle s_i \rangle$ significa $([\![s_i]\!], [\![\alpha \cdot s_i]\!], [\![\alpha]\!])$. Las parties reconstruyen cada $[\![s_i]\!]$ y el adversario introduce errores de tal manera que la reconstrucción da lugar a $s'_i = s_i + \epsilon_i$. Para verificar que estas reconstrucciones son correctas, con comunicación lineal, las parties ejecutan lo siguiente:*

1. *Computar localmente $[\![z_i]\!] = s'_i [\![\alpha]\!] - [\![s_i \alpha]\!]$*
2. *Muestrar valores públicos aleatorios $\sigma_1, \ldots, \sigma_N \in \mathbb{F}$. Esto se puede formular como una funcionalidad $\mathcal{F}_{\mathsf{coin}}$, como hemos hecho antes.*
3. *Computar localmente $[\![z]\!] = \sum_{i=1}^{N} \sigma_i [\![z_i]\!]$*
4. *Hacer "commit-and-open" para reconstruir $z' = z + \delta$ a partir de $[\![z]\!]$. Debido al uso de commit-and-open, podemos garantizar que el error aditivo δ es independiente de z.*
5. *Las parties verifican si $z' = 0$. Si este es el caso, entonces concluyen que todas las reconstrucciones fueron correctas. Si no, entonces abortan*

El objetivo de este ejercicio es demostrar que este protocolo es estadísticamente seguro, y más aún, caracterizar cuál es exactamente la probabilidad de que el adversario engañe a las honest parties reconstruyendo valores incorrectos que aún así pasan el check.

1. *Supongamos que existe al menos uno de los $\epsilon_1, \ldots, \epsilon_N$ no es cero (es decir, al menos una de las reconstrucciones no es correcta). Muestre que la probabilidad de que todos los z_i sean iguales a cero es a lo sumo $1/|\mathbb{F}|$.*
2. *Ahora asuma que existe al menos uno de los $z_1, \ldots, z_N$ que es no cero. Muestre que la probabilidad de que z' sea cero es a lo sumo $1/|\mathbb{F}|$.*
3. *Finalmente, muestre que si al menos uno de los $\epsilon_1, \ldots, \epsilon_N$ no es cero, entonces la probabilidad de que las parties acepten las reconstrucciones es a lo sumo $2/|\mathbb{F}|$. **Hint:** Sea A el evento en el cual el check es exitoso, y sea B el evento en el el cual $z_1 = \cdots = z_N = 0$. Queremos demostrar que $\Pr[A] \leq 2/|\mathbb{F}|$. Para esto, utilice la igualdad $\Pr[A] = \Pr[A \mid B] \Pr[B] + \Pr[A \mid \neg B] \Pr[\neg B]$ (llamada ley de probabilidad total, donde $\Pr[A \mid B]$ es la probabilidad condicional del evento A asumiendo el evento B), y utilice los dos ejercicios anteriores.*

Ejercicio 8.2 (SPDZ para campos pequeños)**.** *Como vimos en el ejercicio anterior, la probabilidad de fallo en SPDZ es proporcional a* $1/|\mathbb{F}|$*. Si el campo* $\mathbb{F}$ *es muy pequeño, esta probabilidad puede ser demasiado grande para ser útil en la práctica. Muestre como extender SPDZ para el campo* $\mathbb{F}_2$*, de tal manera que la probabilidad de error sea proporcional a* $1/2^\kappa$ *(donde* κ *es un valor dado).* ***Hint:*** *Use SPDZ pero con múltiples llaves* α*.*

9. MPC en la práctica (4 de abril)

Se hace una discusión sobre el estado de MPC en la práctica, incluyendo despliegues en el mundo real, implementaciones, esfuerzos por parte de la industria, entre otros temas. Se presenta un protocolo para generar firmas ECDSA a partir de una llave secreta que está "secret-shared", con el fin de ilustrar la aplicabilidad directa de las técnicas que se vieron en clase. También se discute cómo implementar otras aplicaciones más avanzadas usando bit decomposition. Finalmente, se da una presentación por parte de uno de los estudiantes sobre MP-SPDZ, la cual es una framework popular para implementar aplicaciones en MPC.

Video. Parte A https://youtu.be/U4NQVMN11TI, Parte B https://youtu.be/4n8NZInFELs

Referencias principales. Para ECDSA pueden revisar https://ia.cr/2019/768, pero la descripción allí es más compleja/completa que la que se vio en clase. Para MP-SPDZ, ver https://github.com/data61/MP-SPDZ/.

Ejercicios

Ejercicio 9.1 (Generación de secret-shared bits)**.** *En clase se vio que es útil tener shares de la forma* $(\llbracket r \rrbracket, \llbracket r_{\ell-1} \rrbracket, \ldots, \llbracket r_0 \rrbracket)$*, donde cada* r_i *son bits (es decir, son cero o uno) que satisfacen* $r = \sum_{i=0}^{\ell-1} 2^i r_i$*, es decir,* $(r_{\ell-1}, \ldots, r_1, r_0)$ *es la descomposición binaria de* r*. Aquí,* $\llbracket \cdot \rrbracket$ *es cualquier esquema de secret-sharing sobre* $\mathbb{F}_p$*. Representaremos a* $\mathbb{F}_p$ *como los enteros desde* $-(p-1)/2$ *hasta* $(p-1)/2$*.*

1. *Muestre que basta obtener shares de bits aleatorios* $\llbracket r_{\ell-1} \rrbracket, \ldots, \llbracket r_0 \rrbracket$ *para obtener shares de* $\llbracket r \rrbracket$ *sin interacción. Así, el problema de generar* $(\llbracket r \rrbracket, \llbracket r_{\ell-1} \rrbracket, \ldots, \llbracket r_0 \rrbracket)$ *se reduce a generar shares de bits aleatorios.*
2. *Ahora veamos cómo generar* $\llbracket b \rrbracket$ *donde* $b \in \{0, 1\}$ *es aleatorio. Para esto, el protocolo funciona de la siguiente manera:*
   a) *Las parties obtienen* $\llbracket s \rrbracket$ *donde* $s \in \mathbb{F}_p$ *es aleatorio (por ejemplo, usando una funcionalidad* $\mathcal{F}_{\mathsf{Rand}}$ *como se vio alguna vez en clase)*
   b) *Las parties ejecutan un protocolo de multiplicación para tomar el producto de* $\llbracket s \rrbracket$ *con sí mismo, para obtener* $\llbracket s^2 \rrbracket$
   c) *Las parties reconstruyen* $d = s^2$
   d) ***Demuestre que*** d *tendrá dos raíces cuadradas en* $\mathbb{F}_p$*, digamos* s_0 *y* s_1*, donde* s_0 *es "positiva" (pertenece a* $0, \ldots, (p-1)/2$*) y* s_1 *es "negativa" (pertenece a* $-(p-1)/2, \ldots, -1$*). Las parties computan localmente* $\llbracket b \rrbracket = 2^{-1}(s_0^{-1} \llbracket s \rrbracket + 1)$ *(donde* s_0^{-1} *y* 2^{-1} *son las inversas modulo* p*.*

e) *Las parties toman el resultado $[\![b]\!]$ como el output del protocolo.* ***Demuestre que*** *el valor b no sólo pertenece a $\mathbb{F}_p$, sino que pertenece a $\{0,1\}$, y es 0 con probabilidad $1/2$ y 1 con probabilidad $1/2$.*

Ejercicio 9.2 (One time pad sobre los enteros)**.** *En varios protocolos se ha usado el hecho de que si r es aleatorio en un* grupo*, entonces $d = x + r$ es aleatorio en el grupo e independiente de x. Esto se ha usado en el contexto en el cual el grupo es $\mathbb{F}_p$, y allí $x + r$ representa suma modulo p, por lo cual d es igual a $x + r$ reducido módulo p. Ahora, en otros contextos necesitamos que d sea igual a $x + r$ pero como enteros, sin reducción módulo p. Esto se puede lograr si x y r son lo suficientemente pequeños con respecto a p de tal manera que $x + r < p$, y así no ocurra la reducción modulo p. Sin embargo, ya no podemos demostrar que $x + r$ no revela nada sobre x, dado que r no se tomará uniformemente en $\mathbb{F}_p$, sino en un subconjunto más pequeño.*

1. *Para ilustrar la dificultad, sea $p = 31$, y suponga que $x \in \{0,1\}$ y que tomamos r aleatoriamente en $\{0,\ldots,10\}$. Claramente, la suma de x y r no excede p. Sin embargo, muestre que con cierta probabilidad no nula es posible adivinar cuál es el valor de x dado $x + r$. Esto muestra que $x + r$ no esconde completamente a x.* ***Hint:*** *si $x = 0$, ¿cómo puede $x + r$ ser igual a 0? y si $x = 1$, ¿cómo puede $x + r$ ser igual a 11?*
2. *Suponga ahora que $x \in \{0,\ldots,M\}$ y $r \in \{0,\ldots,M \cdot 2^s\}$, y suponga que $M(1+2^s) < p$ de tal manera que $x + r < p$. Demuestre que la probabilidad de que $x + r$ revele información sobre x es a lo sumo 2^{-s}. Esto muestra que eligiendo s lo suficientemente grande, podemos obtener seguridad estadística.* ***Hint:*** *hágase la pregunta "dado $x + r$, ¿cuáles valores de x son posibles?*

10. Fully Homomorphic Encryption (15 de abril)

Se presenta la definición de homomorphic encryption y las diferentes variantes, como additive, somewhat o fully HE. Se discuten algunas propiedades importantes como "strong homomorphism", "compactness", y "circuit privacy". Luego se muestra la técnica de bootstrapping para compilar un somewhat HE en un fully HE, y se presenta una construcción de somewhat HE usando el problema del máximo común divisor aproximado. Finalmente, se discuten algunas consideraciones sobre FHE en la práctica.

Video. Parte A https://youtu.be/61hcF08mA3s, Parte B https://youtu.be/nB-3QlhOAiU

Referencias principales. La survey de Shai Halevi (https://shaih.github.io/pubs/he-chapter.pdf) es muy útil para las definiciones básicas y las propiedades. Ese artículo presenta una construcción de SHE que es basada en látices, es más práctica que lo que se vio en clase, pero es mucho más compleja. La construcción que se discutió en la sesión es del paper "Fully Homomorphic Encryption over the Integers" (https://eprint.iacr.org/2009/616.pdf).

Ejercicios

Ejercicio 10.1 (Seguridad circular)**.** *Recuerde que un esquema de encripción de llave pública dado por tres algoritmos* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *se dice semánticamente seguro (intuitivamente) si un texto cifrado no revela nada del texto plano, y se dice* circularmente seguro *si su seguridad semántica no se rompe, incluso si el adversario tiene acceso a* $\mathsf{Enc}_{\mathtt{pk}}(\mathtt{sk})$*. Esto es fundamental para permitir*

bootstrapping. Intuitivamente esto no debería ser un problema pues la encripción no debería revelar nada sobre el texto plano, `sk` *en este caso. Sin embargo, este no es un texto plano cualquiera: es la llave secreta.*

Asuma la existencia de un esquema de encripción de llave pública que es semánticamente seguro (por ejemplo, RSA). Muestre cómo construir a partir de este un esquema de encripción que siga siendo semánticamente seguro, pero no es circularmente seguro. ***Hint:*** *modifique el algoritmo de encripción para que actúe de manera diferente si el input es la llave secreta; una tecnicalidad es ¿cómo puede el algoritmo de encripción decidir si el input es la llave secreta, si no tiene acceso a la llave secreta en sí?*

Ejercicio 10.2 (NAND es funcionalmente completo)**.** *Muestre que el operador* `NAND` *(negación del AND) es funcionalmente completo, es decir: cualquier función* $f : \mathbb{Z}_2^n \to \mathbb{Z}_2^m$ *puede describirse como un circuito donde la única compuerta es NAND.* ***Hint:*** *recuerde que ya conocemos otras compuertas que son funcionalmente completas.*

11. Introducción a zero knowledge proofs (17 de abril)

Se presenta la definición de interactive proofs, y se discute el potencial de esta clase computacional. Luego se define el concepto de zero knowledge, y se muestra una zero knowledge proof para el problema de quadratic residuosity. Finalmente, se muestra que todo problema en NP tiene una zero knowledge proof con simulación computacional.

Video. https://youtu.be/oWEtIOejVic

Referencias principales. En general, la Novena BIU Winter School (https://cyber.biu.ac.il/event/the-9th-biu-winter-school-on-cryptography/) y el curso masivo en ZK (https://zk-learning.org/) son referencias que usaremos. Para esta clase en concreto: Introducción a ZKPpor Shafi Goldwasser (https://youtu.be/uchjTIlPzFo). Charlas por Alon Rosen (https://youtu.be/6uGimDYZPMw, https://youtu.be/cQ-BI1WWzjU, https://youtu.be/cAI7Iw_bkZs).

12. Sigma Protocols y MPC-in-the-Head (24 de abril)

Se presenta la definición de sigma protocols, junto con algunas de sus propiedades básicas. También se muestra como ejemplo el protocolo de Schnorr, junto con su variante no interactiva usando la heurística de Fiat-Shamir, y también se discute cómo convertir esta construcción en un esquema de firmas digitales. Posteriormente se generaliza esta idea para construir firmas digitales a partir de otras funciones difíciles de invertir. Finalmente, se describe una familia de sigma protocols importante, que es aquellos basados en MPC-in-the-Head, la cual es una técnica para obtener zero knowledge a partir de MPC.

Video. https://youtu.be/cDN-FchIfGM

Referencias principales. Las notas de Ivan Damgård "On Sigma-protocols." (https://cs.au.dk/~ivan/Sigma.pdf). Aquí hay definiciones básicas de sigma protocols, junto construcciones y demostraciones de algunas de las propiedades que mencionamos en clase. También están las charlas de Benny Pinkas en la BIU winter school (https://youtu.be/XT1Pad0DM24 y https://youtu.be/m-NW75E8JIE).

Para MPC-in-the-Head:

- El paper original que introdujo la idea: https://web.cs.ucla.edu/~rafail/PUBLIC/77.pdf
- Paper conocido como "KKW", mejora la construcción original, y la presentación es muy limpia y "fácil" de seguir https://eprint.iacr.org/2018/475.pdf
- Un paper que presenta una construcción optimizada de MPC-in-the-head para circuitos arbitrarios, y lo aplica a firmas digitales https://eprint.iacr.org/2022/588.pdf
- Un paper reciente que explica de manera muy general MPC-in-the-Head, y explora el trade-off entre parámetros como el número de parties, threshold, etc. Es denso pero útil https://eprint.iacr.org/2022/1407.pdf
- Una serie de firmas digitales construidas a partir del paradigma del paper de arriba, basadas en problemas como MQ, MinRank y Syndrome Decoding. La presentación es densa pero limpia y entendible https://eprint.iacr.org/2022/1512.pdf

13. Panorama general de construcciones de SNARKs y KZG commitments (2 de mayo)

Se presenta la definición de una SNARK (succinct non-interactive argument of knowledge), y se discuten algunas de las variantes y métricas de interés sobre SNARKs. Se presentan los modelos de PCP, IOP, Linear PCP y Polynomial IOP, como alternativas al modelo de comunicación clásico que resulta ser más adecuado para construir SNARKs. Se discute que estos modelos abstractos se pueden instanciar en el mundo real usando functional commitments, y se muestra cómo hacer esto concretamente en el caso de PCP usando Merkle trees (vector commitments). Posteriormente se presenta una taxonomía a alto nivel que caracteriza algunas de las construcciones existentes dependiendo del modelo al que pertenecen, discutiendo ventajas y desventajas. Finalmente, se muestra la construcción de KZG commitments, los cuales son un ejemplo prominente de polynomial commitments, una herramienta fundamental para instanciar el modelo de polynomial IOP.

Video. https://youtu.be/3ON_e048DJQ

Referencias principales.

- https://youtu.be/bGEXYpt3sj0
- https://zkproof.org/2020/08/12/information-theoretic-proof-systems/
- https://zkproof.org/2020/10/15/information-theoretic-proof-systems-part-ii/

- Capítulo 19 de https://people.cs.georgetown.edu/jthaler/ProofsArgsAndZK.pdf
- Sección 2 de https://docs.zkproof.org/reference.pdf
- https://www.di.ens.fr/~nitulesc/files/Survey-SNARKs.pdf

14. Idea general de PLONK (8 de mayo)

Se presentan las ideas fundamentales que están detrás de la SNARK llamada PLONK, la cual es un polynomial IOP eficiente y popular que ha encontrado aplicaciones en la práctica. Concretamente, se muestra cómo el prover puede codificar la traza del cómputo en un polinomio, usar el Poly-IOP para permitirle al verifier tener acceso de oráculo a este polinomio, y cómo verificar que esta traza codifica un cómputo correcto del circuito deseado.

Video. https://youtu.be/nN6kKyFhwjE

Referencias principales.

- Dan Boneh: The Plonk Snark (https://youtu.be/A0oZVEXav24)
- Vitalik Buterin: how Plonk works (https://vitalik.ca/general/2019/09/22/plonk.html)
- Ethereum research: Plonk in Python (https://github.com/ethereum/research/tree/master/py_plonk)
- Paper original: PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge (https://eprint.iacr.org/2019/953.pdf)

15. Groth'16, SNARKs eficientes con trusted setup (23 de mayo)

Arantxa Zapico, investigadora de la Ethereum Foundation, presenta el la SNARK que se conoce como Groth'16, la cual es de las más eficientes en cuanto al tamaño de la prueba y costos de verifier. Para esto, Arantxa habla sobre non-falsifiable assumptions, quadratic arithmetic programs (QAPs), el protocolo GGPR, Groth'16, y también provee intuición sobre las diferentes pruebas de seguridad.

Video. https://youtu.be/Fp-QB9rCJLQ

Referencias principales.

- Paper original (Groth'16): https://eprint.iacr.org/2016/260.pdf
- Video donde se explican estas ideas bajo la notación de "linear PCPs": https://youtu.be/0LW-qeVe6QI

16. Sum-Check y GKR, pruebas doblemente eficientes (30 de mayo)

Eduardo Soria-Vazquez, investigador del Technology Innovation Institute, en Emiratos Árabes, presenta las técnicas detrás del protocolo GKR, el cual logra provers que corren en tiempo polinomial para generar pruebas compactas (no 'argumentos', los cuales necesitan hipótesis computacionales). Una componente fundamental para esta técnica es el protocolo Sum-Check, que permite verificar que la suma de ciertas evaluaciones de un polinomio multivariado da un valor determinado. Eduardo presenta en detalle el sum-check y provee intuición sobre su seguridad. También muestra cómo GKR hace uso del sum-check para poder demostrar la integridad del cómputo de manera compacta.

Video. https://youtu.be/nWTPtAZIX2E

Referencias principales.

- Paper original (GKR): https://www.microsoft.com/en-us/research/wp-content/uploads/2008/01/GoldwasserKR08a.pdf

17. FRI – Polynomial commitments transparentes (20 de junio)

En esta sesión se explican el test de grado bajo presente en FRI, el cual es esencial para la construcción de "STARKs", las cuales son unas pruebas compactas transparentes de integridad de cómputo. Transparente se refiere al hecho que no necesitan un "trusted setup". El test de grado bajo de FRI permite verificar interactiva y compactamente que un vector de valores para los cuales se conoce un Merkle hash corresponden a la evaluación de un polinomio de grado bajo. Esto resulta ser un componente fundamental para construir un polynomial commitment scheme, el cual se puede mezclar con otros Poly-IOPs como Plonk, visto anteriormente, para obtener una SNARK (de hecho, una STARK, por 'Transparente').

Video. https://youtu.be/vUu7nFLDSmo

Referencias principales.

- Paper original (FRI/STARK): https://eprint.iacr.org/2018/046.pdf
- Mejoras a FRI, más cercanas a lo usado en la práctica: https://eprint.iacr.org/2019/336.pdf (DEEP-FRI) y https://eprint.iacr.org/2020/654.pdf (analisis de seguridad mejorado)
- Serie de blogposts por Vitalik Buterin (alto nivel):
  1. https://vitalik.ca/general/2017/11/09/starks_part_1.html
  2. https://vitalik.ca/general/2017/11/22/starks_part_2.html
  3. https://www.vitalik.ca/general/2018/07/21/starks_part_3.html
- Serie de blogposts por Alan Szepieniec (más detallado, con código): https://aszepieniec.github.io/stark-anatomy/

18. ZKP y Blockchain, Polygon zk-EVM y Tornado Cash (30 de junio)

En esta sesión Ari Rodriguez de nos describe las aplicaciones de zk-SNARKs en el contexto de Blockchain, enfocándose en el escalamiento de Ethereum a través de Validity Rollups. Finalmente, estudiantes del curso comparten una presentación sobre el zk-EVM de Polygon, cuyo objetivo es escalar la máquina virtual de Ethereum usando SNARKs sin cambiar el lenguaje de programación, y también una presentación sobre el contrato inteligente Tornado Cash, cuyo diseño permite anonimizar fondos en Ethereum, usando zk-SNARKs.