Mese a “zero-knowledge proof”-ok világáról

Kőműves Balázs

Faulhorn Labs & Codex
bkomuves@gmail.com

Budapest University of Technology
2024.11.11

Mi ez?

A “zero-knowledge proofs” néven futó témakör egy *absztrakt matematikán* alapuló technológia, ami röviden azt tudja, hogy:

> meg tudlak győzni egy állításról, nagyon hatékonyan, és anélkül hogy minden részletet elárulnék

Példák:

- van egy sakk feladványom, és meg akarlak győzni róla, hogy 2 lépésen belül lehet mattot adni, de nem akarom elárulni, hogy hogyan.
- van egy digitálisan aláírt dokumentumon, és ki akarok húzni belőle szenzitív részleteket
- egy online társasjátékban (pl. torpedó) meg akarlak győzni arról, hogy betartom a játékszabályokat

Mit jelentenek ezek a szavak?

A **“proof”** az itt egy füllentés: nem matematikai bizonyításról van szó, hanem *kriptográfiai bizonyításról*[1]. Ez praktikusan azt jelenti, hogy lehetetlen erőforrások, például egy univerzum méretű számítógép kellene ahhoz, hogy csalni tudjak.

A **“zero-knowledge”** azt jelenti, hogy semmilyen más információt nem árulok el[2], azon kívül hogy lehet 2 lépésen belül mattot adni.

Az **“argument of knowledge”** pedig azt jelenti, hogy én *tényleg tudok* 2 lépésen belül mattot adni, nem pedig csak tudom hogy lehetséges.

---

[1] hivtalosan ezt “argument”-nek hívják

[2] ez nem minden alkalmazásban fontos, de az egyszerűség kedvéért azokat is mindenki így hívja (ami rengeteg félreértésre ad okot)

Miért?

Miért jutna ez a kérdés egyáltalán eszünkbe?

Két fontos alkalmazási terület, ahol egy ilyen technológia jól jöhet:

- **privacy**: amikor tényleg nem akarunk elárulni valamilyen információt a másik félnek
- **hatékonyság**: egy meglepő tény, hogy ha eltitkolunk egy csomó mindent, akkor a "maradék" meggyőző információ lehet akár nagyon kicsi is, és nagyon gyorsan[3] (hatékonyan) megbizonyosodhat a másik fél arról hogy nem hazudunk

Mindenki tudja, hogy adatot nem lehet "a végtelenségig tömöríteni". Ami a meglepetés, hogy *állításokat* viszont lehet!

[3] a bizonyítékot létrehozni viszont kifejezetten lassú tud lenni

Mit lehet vele csinálni?

Sok mindent!

- a digitális aláírás például ennek egy speciális esete
- online identitás: én dönthetem el, hogy mennyi információt osztok meg magamról, miközben a másik fél is biztonságban érezheti magát
- biztonságtechnika általában (credentials, attestations, stb)
- blockchain / web3 világban a privacy is, és a hatékonyság is
- anonim online szavazás
- jog, egészségügy, bankolás: bármi ahol a privacy fontos
- megbízható(bb) machine learning alkalmazások
- stb

Határterületek

Ez egy rendkívűl izgalmas területe az alkalmazott matematikának, de vannak más hasolóan érdekes, közelálló területek is:

- **MPC** (Secure Multi-Party Computation): több résztvevő közösen szeretne kiszámolni valamit a saját adataikat összegezve, de senki sem akarja a titkos adatait átadni a másiknak
- **FHE** (Fully Homomorphic Encryption): egy bonyolult számolást szeretnénk egy harmadik félnek outsource-olni, de nem akarjuk hogy a privát adataink kikerüljenek - erre egy (főleg elméleti) megoldás, ha titkosított adaton is végre lehet hajtani a számításokat, anélkül hogy dekódolnánk őket

Ezek között a területek között viszonylag sok az átjárás, de most itt a **ZK**-ra koncentrálunk.

Hogyan?

Ez az egész nagyon mágikusan hangzik! Hogyan lehetséges ez?!

Nos igen, a kriptográfia lényegében mágia :)

Először két egyszerűbb példával próbálom ezt megvilágítani:

- az elsőben a színvak ismerősünket próbáljuk meggyőzni, hogy a piros és a zöld tényleg két különböző szín;
- a második már valódi matematika: a Schnorr protokol arról tud meggyőzni valakit, hogy ismerjük egy csoportelem diszkrét logaritmusát

A gyanakvó színvak

Egy intuitív példa a színek létezését nem elfogadó színvak esete:

Tegyük fel hogy van egy színvak ismerősünk, aki nem hiszi el hogy van különbség a piros és a zöld szín között. Hogyan győzzük meg?

A gyanakvó színvak

Egy intuitív példa a színek létezését nem elfogadó színvak esete:

Tegyük fel hogy van egy színvak ismerősünk, aki nem hiszi el hogy van különbség a piros és a zöld szín között. Hogyan győzzük meg?

Fogjunk egy A4-es lapot, fessük be az egyik felét pirosra, a másikat zöldre. A barátunknak ez egyszínű szürke, nem látja a különbséget ha elfordítjuk 180 fokkal, de mi látjuk. Ha most az ismerősünk a háta mögött véletlenszerűen elforgatja (ő tudja hogy milyen állásba, például a hátulján meg van jelölve az egyik sarok), és mi konzisztensen meg tudjuk mondani az "egyszínű" oldalról, hogy elforgatta vagy nem, akkor egy idő után *kénytelen lesz* elhinni hogy mi látunk valamit ami ő nem...

Schnorr protokol

Egy matematikai példa a **Schnorr protokol**. Ez egy *interaktív protokol*, amivel arról győzi meg az egyik fél a másikat, hogy ismeri egy N méretű $\mathbb{G} := \langle g \rangle$ ciklikus csoportban egy adott $x \in \mathbb{G}$ elem $w \in \mathbb{Z}_N$ *diszkrét logaritmusát*: $x = g^w$.

1. a **prover** választ egy egyenletesen véletlen $r \in \mathbb{Z}_N$ számot, és elküldi a $t := g^r \in \mathbb{G}$ csoportelemet
2. a **verifier** válaszol egy szintén véletlen $c \in \mathbb{Z}_N$ *challenge*-el
3. a **prover** az $s := r + cw \in \mathbb{Z}_N$ számmal válaszolja meg a challenge-t
4. a **verifier** ellenőrzi a $g^s = t \cdot x^c$ egyenletet

Ez a példa több fogalmat is illusztrál: a "hardness of discrete logarithm" feltevés; interaktív protokol; commitment ($t \in \mathbb{G}$ egy elköteleződés r mellett); zero knowledge (r mint "blinding factor").

Mi az hogy állítás?

Mit jelent pontosan az hogy "állítás", amiről meg akarlak győzni?

A gyakorlatban valamilyen objektíven *eldönthető* dolgot szeretnénk; ennek egy praktikus verziója egy *számítógép program*, ami vagy azt mondja hogy **OKÉ**, vagy azt hogy **NEM**.

Formálisabban, van egy

$$f : X \times W \to \{\text{elhiszem, vagy nem}\}$$

függvényünk (relációnak is hívhatnánk), ahol X jelöli a nyilvános információt, és W jelöli a titkos információt[4].

És arról akarlak meggyőzni, hogy egy nyilvánosan adott $x \in X$ mellé a birtokomban van egy olyan titkos $w \in W$, hogy

$$f(x, w) = \text{IGEN}$$

[4] ezt gyakran "witness"-nek hívják, emiatt a W betű

Sudoku

Egy jó példa erre a Sudoku feladvány[5]:

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Itt X (a feladvány) a baloldali ábra; W (a megoldás) a jobboldali ábra; és az f program ellenőrzi, hogy a megoldás szabályos, és kompatibilis a feladvánnyal.

[5] a képeket loptam az internetekről

Madártávlat

Egy tipikus[6] "blueprint" egy ZK bizonyítás elkészítéséhez:

1. az X bemenetet (ha van, akkor kimenetet), W witness-t és minden, a függvény/reláció kiértékelése közben keletkező részeredményt is *számokkal* (pontosabban, véges test elemekkel) reprezentáljuk
2. a relációnkant *polinom egyenletekre* fordítjuk le, úgy hogy az egyenletrendszernek pontosan akkor lesznek megoldásai a fenti számok, ha teljesül a reláció ("aritmetizáció")
3. majd valamilyen matematikai trükkel megmutatjuk, hogy az egyenleteknek van megoldása, olyan, ami konzisztens a bemenetekkel is ("interactive oracle proof")
4. végül ezt praktikusan megvalósíthatóvá tesszük kriptográfiai technológiákkal ("polynomial commitment schemes")

[6] de nem az egyetlen lehetséges

Polinomok

Sok, részben különböző konstrukció van ilyen ZK bizonyításokra, de majdnem mindegyik *véges testek feletti polinomokra* alapul.

Az egyváltozós[7] polinomoknak két fontos tulajdonságát fogjuk használni:

- **interpoláció:** tetszőleges n méretű (x_i, y_i) párhalmazhoz lehet talalálni egy $n-1$ fokú f polinomot amire $f(x_i) = y_i$;
- **gyökök:** egy n fokú, nem konstans zéró polinomnak pontosan n gyöke van.

Miért pont véges testek? Már csak azért is, mert a számítógép azzal tud számolni... (de más hasznos tulajdonságaik is vannak.)

[7] Többváltozós (multilineáris) polinomokat is szoktak használni, az egy párhuzamos világ (valamennyi átjárással).

Schwartz-Zippel lemma

Tegyük fel, hogy adott két, legfeljebb d-edfokú polinom egy $\mathbb{F}$ véges test felett:

$$p, q \in \mathbb{F}^{\leq d}[x]$$

És arra vagyunk kíváncsiak, hogy egyenlőek-e.

Ha nem egyenlőek, akkor a különbségük is legfeljebb d-edfokú, tehát maximum d gyöke lehet. Tehát ha véletlenszerűen választunk egy $\zeta \in \mathbb{F}$ testelemet, és ott kiértékeljük őket, akkor annak a valószínűsege, hogy $p(\zeta) = q(\zeta)$ legfeljebb $d/|\mathbb{F}|$.

A gyakorlatban általában d relatíve kicsi (például 10^6), míg $\mathbb{F}$ sokkal nagyobb (például 10^{40} vagy akár 10^{80}). Így már egyetlen[8] ponton kiértékelve is biztosak lehetünk abban hogy p és q egyenlőek-e!

Ez egy nagyon centrális ötlet a témakörben.

[8] ha $\mathbb{F}$ nem elég nagy, megismételhetjük 2-3 pontra is

Aritmetizáció

Hogyan fordítunk le egy függvényt / relációt polinom egyenletekre?

Két főbb elterjedt megközelítés van:

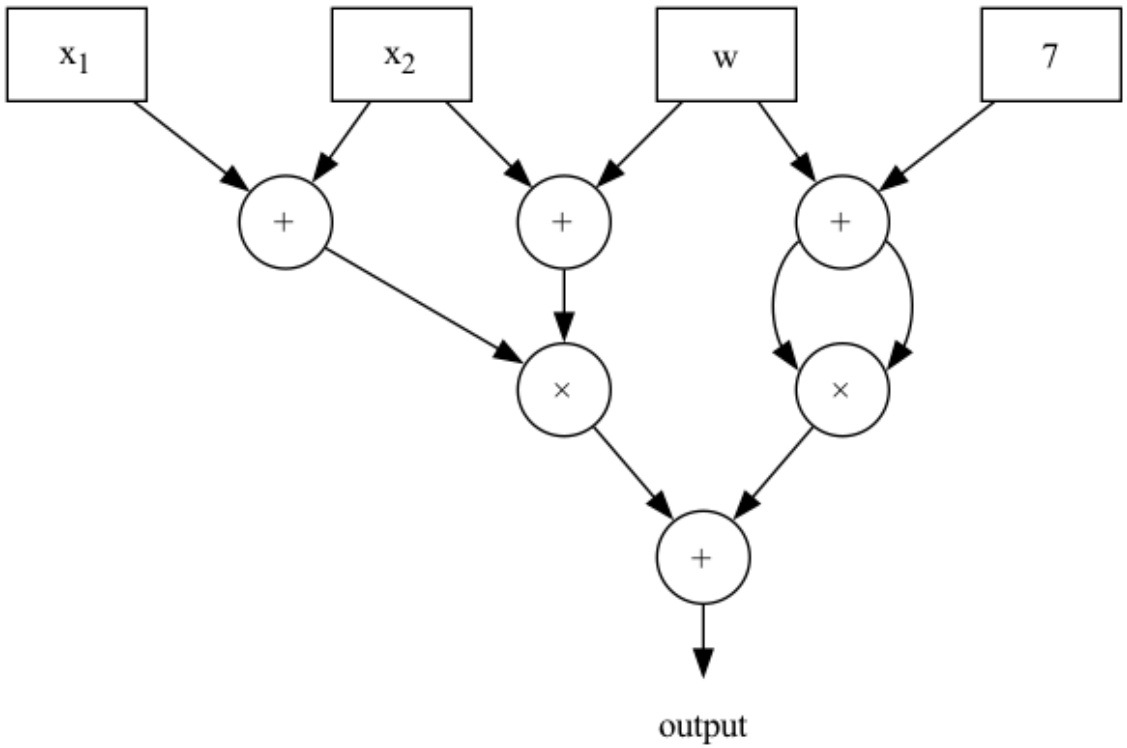
- aritmetikus áramkör
- és állapotgép

Ez a megkülönböztetés hasonló a fizikai chip-ek világához (célspecifikus vs. általános célú áramkör, avagy ASIC vs. CPU).

Utána egy következő lépésben ezeket konkrét egyenletekkel kódoljuk el; ezt is többféleképpen lehet csinálni (példák aritmetikus áramkör elkódolásokra: R1CS, Plonk, GKR, stb).

Aritmetikus áramkör

Egy “aritmetikus áramkör” arithmetikai kapukból (pl. összeadás, szorzás) és “drótokból” áll:



Ez az áramkör a következő függvényt valósítja meg:

$$f(x_1, x_2\,;\, w) := (x_1 + x_2)(x_2 + w) + (w + 7)^2$$

Állapotgép

Van egy állapotunk, amit egy $s \in \mathbb{F}^m$ vektorral kódolunk el, és egy $\texttt{step} : \mathbb{F}^m \to \mathbb{F}^m$ állapot-átmenet függvényünk. Ezt ismételgetjük egy megállási feltételig.

Például ez az állapotgép elkódolhat egy egyszerű virtuális processzort, és akkor az állításunkat megfogalmazhatjuk programként.

Ennek a megközelítésnek nagy előnye, hogy sokkal könnyebb használni: Az állítást csak egy programra kell fordítani, "alatta" minden más fix; így a lehetséges hibák halmaza is kisebb. Hátránya, hogy általában kicsit magasabbak a költségei.

Ezeket az (absztrakt) gépeket általában "zero-knowledge virtual machines"-nak hívják (zkVM).

Polinom interpoláció

Általában bármilyen felmerülő adatot (például X, W bemenet és witness) először véges test elemekkel reprezentálunk, majd a keletkező $Y \in \mathbb{F}^N$ vektort egy $N-1$ fokú P polinommal reprezentáljuk, úgy hogy az rögzített x_i helyeken az A_i értékeket vegye fel:

$$P(x_i) = Y_i$$

Gyakorlati megfontolásokból (például hogy ezt az interpolációs polinomot hatékonyan ki lehessen számolni), az $\{x_i\}$ halmazt egy $H \subset \mathbb{F}^\times$ multiplikatív részcsoportnak szokták választani: $x_i := \omega^i$,

$$H = \{1, \omega, \omega^2, \ldots, \omega^{N-1}\} \subset \mathbb{F}^\times \qquad (\text{megj.: } \omega^N = 1)$$

Tehát $P(\omega^i) = Y_i$. Ennek a konstrukciónak egy további előnye, hogy könnyű az indexeket eltolni: ha $P(x) = Y_k$ akkor $P(\omega x) = Y_{k+1}$, $P(\omega^2 x) = Y_{k+2}$, stb.

Fibonacci állapotgép, I.

Egy egyszerű példa egy állapotgépre a Fibonacci sorozatot kiszámoló gép: $F_{n+1} = F_n + F_{n-1}$. Itt az állapot a k-adik lépésben két szám: (a_i, b_i), és az állapot-átmenet függvény pedig:

$$\texttt{step}(a, b) := (a + b,\ a)$$

A kezdőállapot a hagyományos Fibonacci sorozatnál $(a_0, b_0) = (1, 0)$, de most a példa kedvéért lehet mondjuk a kezdőállapot titkos, és az állítás, hogy tudunk olyan kezdőállapotot, amire az N-ik Fibonacci szám[9] egy adott $x \in \mathbb{F}$.

Hogyan csináljuk ezt? Először az (a_i, b_i) állapotsoroztat két $A, B \in \mathbb{F}[x]$ interpolációs polinommal reprezentáljuk:

$$A(\omega^i) = a_i \qquad \text{és} \qquad B(\omega^i) = b_i$$

[9] természetesen itt most mindent az $\mathbb{F}$ testben értünk

Fibonacci állapotgép, II.

Emlékezzünk, hogy $A(\omega^i) = a_i$ és $B(\omega^i) = b_i$. Most tehát át kell fordítani az állapot-átmenet egyenleteket: $a_k = a_{k-1} + b_{k-1}$ és $b_k = a_{k-1}$ ezekre a polinomokra. Első lépés:

$$\forall x \in H,\, x \neq 1. \quad A(x) = A(\omega^{-1}x) + B(\omega^{-1}x) \quad \text{és} \quad B(x) = A(\omega^{-1})$$

Itt elég kellemetlen az $x \neq 1$ (azaz $k \neq 0$) megkötés: Univerzális, $\forall x \in H \ldots$ típusú egyenleteket szeretnénk. Ez könnyen megoldható:

$$0 = (x-1) \cdot \left[A(x) - A(\omega^{-1}x) - B(\omega^{-1}x)\right]$$
$$0 = (x-1) \cdot \left[B(x) - A(\omega^{-1}x)\right]$$

Szeretnénk még, hogy $A_N = t$, ahol $t \in \mathbb{F}$ egy nyilvános érték. Ezt többféleképpen el lehet kódolni; egy lehetőség: $\forall x \in H$. $\mathcal{L}_{N-1}(x) \cdot \left[A(x) - t\right] = 0$. Itt $\mathcal{L}_{N-1}$ egy Lagrange polinom:

$$\mathcal{L}_k(\omega^j) := \begin{cases} 1, & \text{ha} \quad k = j \\ 0, & \text{ha} \quad k \neq j \end{cases} \qquad \deg(\mathcal{L}_k) = N$$

Fibonacci állapotgép, III.

Utolsó lépésként a $\forall x \in H.\ P(x) = 0$ fajta egyenleteket szeretnénk tovább átírni olyanokká, amik már minden $x \in \mathbb{F}$-re teljesülnek.

Szerencsére ez is viszonylag egyszerűen megvalósítható: $P(\omega^i) = 0$ ekvivalens azzal, hogy $P(x)$ osztható $(x - \omega^i)$-vel. Ha minden i-re $P(\omega^i) = 0$, akkor pedig $P(x)$ osztható a $Z_H(x)$ "zéró polinommal"[10]:

$$Z_H(x) := \prod_{i=0}^{N-1} (x - \omega^i) = x^N - 1$$

Jelölje a hányadost $Q(x) := P(x)/Z_H(x)$. Fontos hogy ez is egy *polinom*. Ekkor a végső egyenleteink valahogy így fognak kinézni:

$$Q_1(x) \cdot Z_H(x) = (x - 1) \cdot \big[A(x) - A(\omega^{-1}x) - B(\omega^{-1}x)\big]$$
$$Q_2(x) \cdot Z_H(x) = (x - 1) \cdot \big[B(x) - A(\omega^{-1}x)\big]$$
$$Q_3(x) \cdot Z_H(x) = \mathcal{L}_{N-1}(x) \cdot \big[A(x) - t\big]$$

[10] Itt a hatékonyság szempontjából viszonylag fontos észrevétel, hogy $Z_H(x)$-nek nagyon egyszerű alakja van, és könnyű kiértékelni.

Fibonacci protokol, első próbálkozás

De hogyan lesz ezekből az egyenletekből *meggyőző bizonyíték*?

A Schnorr protokolhoz hasonlóan egy interaktív protokolt építünk. Az első probálkozás:

1. a **prover** kiszámolja az (a_n, b_n) Fibonacci sorozatot, a titkos $(a_0, b_0) \in \mathbb{F} \times \mathbb{F}$ kezdőállapotból kiindulva
2. ezekből elkészíti az $A(x)$, $B(x)$ interpolációs polinomokat
3. majd kiszámolja a Q_1, Q_2, Q_3 hányados polinomokat
4. elküldi az A, B, Q_1, Q_2, Q_3 polinomokat és a $t = a_{N-1}$ értéket
5. a **verifier** válasz egy véletlen $\zeta \in \mathbb{F}$ számot, és leellenőrzi a 3 egyenletet az $x \mapsto \zeta$ helyettesítéssel

Mi ezzel a baj?

Fibonacci protokol, első próbálkozás

De hogyan lesz ezekből az egyenletekből *meggyőző bizonyíték*?

A Schnorr protokolhoz hasonlóan egy interaktív protokolt építünk. Az első probálkozás:

1. a **prover** kiszámolja az (a_n, b_n) Fibonacci sorozatot, a titkos $(a_0, b_0) \in \mathbb{F} \times \mathbb{F}$ kezdőállapotból kiindulva
2. ezekből elkészíti az $A(x)$, $B(x)$ interpolációs polinomokat
3. majd kiszámolja a Q_1, Q_2, Q_3 hányados polinomokat
4. elküldi az A, B, Q_1, Q_2, Q_3 polinomokat és a $t = a_{N-1}$ értéket
5. a **verifier** válasz egy véletlen $\zeta \in \mathbb{F}$ számot, és leellenőrzi a 3 egyenletet az $x \mapsto \zeta$ helyettesítéssel

Mi ezzel a baj? Ezek a polinomok lehetnek nagyon nagyok (ha N nagy), ami sok kommunikációt, és a verifier részéről (viszonylag) sok számolást jelent; továbba könnyen kinyerhető belőlük a "titok", hiszen $a_0 = A(1)$ és $b_0 = B(1)$.

Polinom commitment sémák

A dilemma feloldása, hogy a prover polinomok helyett csak az "ujjlenyomataikat" (commitment) küldi el a verifiernek (ezek kicsik), és az $x \mapsto \zeta$ helyettesíteseket is a prover végzi el; ezekhez **bizonyításokat** is mellékel, amik garantálják hogy konzisztens az ujjlenyomattal.

Definíció: Egy *polynomial commitment scheme* (PCS) a következő algoritmusok gyűjteménye:

$$\begin{aligned} \texttt{commit} &: \mathbb{F}^{\leq D}[x] \to \mathbb{G} \\ \texttt{eval} &: \mathbb{F}^{\leq D}[x] \times \mathbb{F} \to \mathbb{F} \times \Pi \\ \texttt{check} &: \mathbb{G} \times (\mathbb{F} \times \Pi) \to \{\texttt{true}, \texttt{false}\} \end{aligned}$$

amik azt tudják, hogy:

- $\texttt{eval}(f, t) = \big(f(t)\,,\, \pi\big)$
- $\texttt{check}\big(\texttt{commit}(f), \texttt{eval}(f, t)\big) = \texttt{true}$
- reménytelenül nehéz másik $f' \in \mathbb{F}[x]$ polinomot találni, amire $\texttt{commit}(f) = \texttt{commit}(f')$
- reménytelenül nehéz másik $y' \in \mathbb{F}$-t és $\pi' \in \Pi$-t találni, amik átmennek az ellenőrzésen

Fibonacci protokol, második verzió

1. a **prover** kiszámolja az (a_n, b_n) Fibonacci sorozatot
2. ezekből elkészíti az $A(x)$, $B(x)$ interpolációs polinomokat, és elküldi ezek *commitment*-jét: $\mathtt{com}(A)$, $\mathtt{com}(B)$, és a $t = A_{N-1}$ értéket
3. majd kiszámolja a Q_1, Q_2, Q_3 hányados polinomokat, és elküldi ezek commitment-jeit is: $\mathtt{com}(Q_1)$, $\mathtt{com}(Q_2)$, $\mathtt{com}(Q_3)$
4. a **verifier** válasz egy véletlen $\zeta \in \mathbb{F}$ számot, és elküldi a prover-nek
5. a **prover** kiszámolja az $A(\zeta)$, $B(\zeta)$, $Q_{1,2,3}(\zeta)$ értékeket és a hozzájuk tartozó *bizonyításokat*, és ezeket elküldi a verifier-nek
6. a **verifier** leellenőrzi ezeket a bizonyításokat, és az egyenleteket is.

Ez már sokkal jobb: a commitmentek és az értekek nagyon "kicsik" (a polinomok méretéhez képest), as az ellenőrzések is csak keves időbe kerülnek.

Megjegyzés: Itt nagyon fontos, hogy a verifier csak az után választja ki a $\zeta \in \mathbb{F}$ challenge értéket, miután megkapta az összes commitment-et; ha a prover előre ismerné ζ-t, nagyon könnyen tudna csalni!

A véletlen lineáris kombináció trükk

Egy standard trükk polinomokat és egyenleteket véletlenszerűen választott együtthatókkal skálázva összeadni. Ezzel nagyon jelentős hatékonyságnövelést lehet elérni.

Tegyük fel hogy van m polinom-egyenletünk: $P_i(x) = 0$ (ide képzelhetjük például a korábbi 3 egyenletet). Válasszunk egy véletlen $\alpha \in \mathbb{F}$ együtthatót, és képezzük a

$$\mathcal{P}(x) := \sum_{i=0}^{m-1} \alpha^i P_i(x) = P_0(x) + \alpha \cdot P_1(x) + \alpha^2 \cdot P_2(x) + \dots$$

lineáris kombinációt. Annak esélye, hogy $\mathcal{P}(\zeta) = 0$, de legalább egy $P_i(\zeta) \neq 0$, nagyon kicsi: α polinomjaként tekintve láthatjuk, hogy legfeljebb $m/|\mathbb{F}|$.

Így tehát m egyenlet külön-külön leellenőrzése helyett elég egyet ellenőrizni.

Kombinált hányadospolinom

Ezt a trükköt alkalmazhatjuk a Q_i hányadospolinomokra és a 3 egyeletre is:

$$\mathcal{Q}(x) := Q_1(x) + \alpha \cdot Q_2(x) + \alpha^2 \cdot Q_3(x) =$$
$$= \sum_{i=1}^{3} \alpha^{i-1} \cdot \left[P_i(x)/Z_H(x)\right] = \frac{1}{Z_H(x)} \sum_{i=1}^{3} \alpha^{i-1} P_i(x)$$

Így csak egyetlen nagy egyenletünk marad: $\forall x \in \mathbb{F}$

$$\begin{aligned}\mathcal{Q}(x) \cdot Z_H(x) = 1 \ &\cdot (x-1) \cdot \left[A(x) - A(\omega^{-1}x) - B(\omega^{-1}x)\right] \\ &+ \alpha \ \cdot (x-1) \cdot \left[B(x) - A(\omega^{-1}x)\right] \\ &+ \alpha^2 \cdot \mathcal{L}_{N-1}(x) \cdot \left[A(x) - t\right]\end{aligned}$$

Fibonacci protokol, harmadik verzió

1. a **prover** kiszámolja az (a_n, b_n) Fibonacci sorozatot
2. elkészíti az $A(x)$, $B(x)$ interpolációs polinomokat, és elküldi a $\mathtt{com}(A)$, $\mathtt{com}(B)$ commitment-eket és a $t = A_{N-1}$ értéket
3. a **verifier** választ egy véletlen $\alpha \in \mathbb{F}$ értéket, és elküldi azt
4. a **prover** kiszámolja a $\mathcal{Q}(x)$ kombinált hányadospolinomokat ezzel az α együtthatóval, és elküldi a $\mathtt{com}(\mathcal{Q})$ commitmentjét
5. a **verifier** válasz egy véletlen $\zeta \in \mathbb{F}$ számot, és elküldi a prover-nek
6. a **prover** kiszámolja az $A(\zeta)$, $B(\zeta)$, $\mathcal{Q}(\zeta)$ értékeket és a hozzájuk tartozó bizonyításokat, és ezeket elküldi a verifier-nek
7. a **verifier** leellenőrzi ezeket a bizonyításokat, és a kombinált egyenletet.

A sorrend továbbra is nagyon fontos!!

Fiat-Shamir heurisztika

Eddig mindig *interaktív protokollokról* volt szó: a prover és a verifier "beszélget", és a beszélgetés végére a verifier (ideális esetben) meggyőződik az állítás igazságáról.

Azonban a gyakorlatban sokszor sokkal praktikusabb lenne egy *nem-interaktív* bizonyítás: A prover elkészíti az egész bizonyítást egy "csomagként", és utána azt bárki le tudja ellenőrizni (például az internetről letöltve).

A jó hír, hogy ezt (bizonyos feltételek mellett) mindig meg lehet csinálni. Vegyük észre, hogy a verifier üzenetei mindig egyenletesen véletlenül választott $\mathbb{F}$ testelemek voltak. Az ötlet, hogy ezeket a challenge-eket a prover *leszimulálja*, olyan módon hogy ne tudja őket csalás céljából érdemben befolyásolni.

Ehhez a prover az összes korábbi (szimulált) üzenetváltások egy *kriptografikus hash függvényét* (pl. SHA256) veszi alapul.

Polinom commitment sémák, második nekifutás

Már majdnem minden "mozgó alkatrészt" megnéztünk, kivéve a PCS-eket. Emlékeztetőül:

$$\begin{aligned} \texttt{commit} &: \mathbb{F}^{\leq D}[x] \to \mathbb{G} \\ \texttt{eval} &: \mathbb{F}^{\leq D}[x] \times \mathbb{F} \to \mathbb{F} \times \Pi \\ \texttt{check} &: \mathbb{G} \times (\mathbb{F} \times \Pi) \to \{\texttt{true}, \texttt{false}\} \end{aligned}$$

Több különböző ilyen séma van (és folyamatosan találnak ki új variációkat); ami közös bennük, hogy mindegyik valamilyen algebrai struktúrára épül: például

- dlog-nehéz csoportok (Bulletproofs)
- pairin-friendly elliptikus görbék (KZG, Dory)
- hibajavító kódok (FRI)
- groups of unknown order (DARK)
- rácsok, stb.

Pedersen vektor commitment

Bemelegítésképpen nézzük először a Pedersen (vektor) commitment-et. Ez NEM egy polinom commitment séma: csak annyit tud, hogy egy $x \in \mathbb{F}^n$ vektor "ujjlenyomatát" tudjuk képezni (hasonlóan egy kriptografikus hash függvényhez).

Szükségünk lesz egy rögzített $\mathbb{G}$ ciklikus csoportra, amire $|\mathbb{G}| = |\mathbb{F}|$, amiben nehéz a diszkrét logaritmus probléma, és rögített, véletlenszerűen választott, lineárisan független $g_1, \ldots g_n \in \mathbb{G}$ és $h \in \mathbb{G}$ generátor elemekre.

Ezek után, ha egy $x \in \mathbb{F}^n$ vektorhoz akarunk elköteleződni, akkor Pedersen commitment egyszerűen a:

$$\mathtt{com}(r; x) := h^r \cdot \prod_{i=1}^{n} g_i^{x_i} \in \mathbb{G}$$

csoportelem. Itt $r \in \mathbb{F}$ egy véletlenszerűen választott érték ("blinding factor"): Ez garantálja a zero-knowledge tulajdonságot. Amikor fel akarjuk fedni a vektort, akkor az r értéket is megmutatjuk.

KZG commitment, I.

A KZG commitment már egy igazi PCS. Itt egy kicsit bonyolultabb setup-ra lesz szükségünk: A csoportból két példány kell: $\mathbb{G}_1 = \mathbb{G}_2$, rögzített $g_i \in \mathbb{G}_i$ generátorokkal; valamint egy *bilineáris leképezés* (vagy "pairing"): $\langle -, - \rangle : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ egy harmadik csoportba. A gyakorlatban $\mathbb{G}_{1,2}$ elliptikus görbék részcsoportjai, és $\mathbb{G}_T$ egy véges test multiplikatív részcsoportja.

Ha $g \in \mathbb{G}_{1,2}$ és $|\mathbb{G}| = |\mathbb{F}| = p$, akkor egy $u \in \mathbb{F}_p$ prímtest-elemhez, amire gondolhatunk úgy is mint egy $0 \leq u < p$ egész szám, elkészíthetjük az $[u]_i := g_i^u \in \mathbb{G}_i$ csoportelemet.

Szükségünk lesz még e következő, ún. "trusted setup"-ra: Válasszunk egy véletlenszerű, *titkos* $\tau \in \mathbb{F}$ elemet, és készítsük el az

$$[1]_1 = g_1,\ [\tau]_1,\ [\tau^2]_1,\ \ldots, [\tau^D]_1 \in \mathbb{G}_1 \quad \text{és} \quad [1]_2 = g_2,\ [\tau]_2 \in \mathbb{G}_2$$

sorozatokat. Nagyon fontos, hogy a τ-t **egyetlen résztvevő se ismerje** (ezt a gyakorlatban meg lehet valósítani ún. "trusted setup ceremoniákkal").

KZG commitment, II.

Ekkor egy $P(x) := \sum_{k=0}^{n} a_k x^k$ polinom KZG commitmentje

$$\mathtt{com}(P) := \left[P(\tau)\right]_1 = g_1^{f(\tau)} = g_1^{\left(\sum a_k \tau^k\right)} = \prod_{k=0}^{n} g_1^{(a_k \tau^k)} = \prod_{k=0}^{n} [\tau^k]_1^{a_k}$$

Ez nagyon hasonló a Pedersen commitmenthez, csak véletlenszerű g_k generátorok helyett helyett *struktúrált* $[\tau^k]$ generátoraink vannak.

Ha most meg akarunk győzni valakit, hogy $P(x_0) = y_0$, akkor a bizonyíték a $Q(x) := (P(x) - y_0)/(x - x_0)$ hányados-polinom $\mathtt{com}(Q)$ commitmentje lesz.

Hogyan tudjuk ezt ellenőrizni? Azt szeretnénk látni, hogy $Q(x) \cdot (x - x_0) = P(x) - y_0$. Mindkét oldal commitmentjét képezve:

$$\mathtt{com}(Q)^{(\tau - x_0)} = \mathtt{com}(P) \cdot g_1^{-y_0}$$

Sajnos azonban τ-t nem ismerjük. Itt segít a bilineáris "pairing":

$$\left\langle \mathtt{com}(Q)\ ,\ [\tau]_2 \cdot g_2^{-x_0} \right\rangle = \left\langle \mathtt{com}(P) \cdot g_1^{-y_0}\ ,\ g_2 \right\rangle$$

Linkek

Néhány link kiindulópontnak, ha valaki többet szeretne tudni:

- Justin Thaler: Proofs, Arguments, and Zero-Knowledge (online könyv): `https://people.cs.georgetown.edu/jthaler/ProofsArgsAndZK.htm`
- Berkeley University ZKP MOOC (online kurzus): `https://zk-learning.org/`
- ZK Whiteboard Sessions (youtube előadások): `youtube.com/playlist?list=PLj80z0cJm8QErn3akRcqvxUsyXWC810Gq`
- zkp.science (linkgyűjtemény): `https://zkp.science/`
- Dankrad Feist: KZG polynomial commitments (blogposzt): `https://dankradfeist.de/ethereum/2020/06/16/kate-polynomial-commitments.html`
- Alan Szepieniec: Anatomy of a STARK (tutorial): `https://aszepieniec.github.io/stark-anatomy/`