ZK-SNARKs

Εισαγωγή και Εφαρμογές

Παναγιώτης Γροντάς

Με υλικό από διάλεξη Αλ. Ζαχαράκη (ακ. έτος 20-21), [Tha22] και ZKP MOOC *zk-learning.org*

Μέρος 1ο & Μέρος 2ο (23.03.23)

ΕΜΠ - Advanced Crypto (2022-2023)

Εισαγωγή

Interactive Proofs i

Κλασικές αποδείξεις:

- Ο prover P υπολογίζει μία απόδειξη π για μια πρόταση (offline)
- Την αποστέλλει στον verifier V
- Ο V την μελετά (offline)
- Την αποδέχεται ή όχι

Αλληλεπιδραστικές αποδείξεις (Interactive Proofs)

- Ο prover P υπολογίζει μία απόδειξη π για μια πρόταση (offline)
- Ο V τον ανακρίνει
- Ο P απαντά
- Ο V αποδέχεται ή όχι

- Χαρακτηριστικά: Αλληλεπίδραση και Τυχαιότητα
- Ασφάλεια:
  - Πληρότητα (Completeness)
  - Ορθότητα (Soundness) (ισχύει η απόδειξη)
  - Ορθότητα Γνώσης (Knowledge Soundness) - γνώση witness (ισχυρότερη έννοια)

Τεράστια υπολογιστική ισχύ $IP = PSPACE$

Interactive Proofs iii

Αποδείξεις μηδενικής γνώσης

- Αλληλεπιδραστική απόδειξη
- Ο V δεν μαθαίνει τίποτα παρά μόνο την αλήθεια της πρότασης
- Μοντελοποίηση: Ύπαρξη Sim που αλληλεπιδρά με V με μη διακρίσιμο τρόπο

Μειονεκτήματα:

- Ο V πρέπει να είναι online
- Μη δημόσια επαληθευσιμότητα λόγω ύπαρξης Sim
- Λύση - αφαίρεση αλληλεπίδρασης - NIZK

Σ Πρωτόκολλα

- Ο V είναι πάντα honest
    - επιλέγει πάντα ομοιόμορφα
- 3 μηνύματα
- Ειδική ορθότητα

Ευκολη αφαίρεση της αλληλεπιδραστικότητας με το Fiat - Shamir heuristic

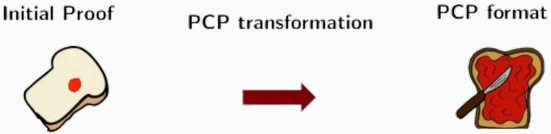
PCP - Probabilistically Checkable Proofs

Θεώρημα (Arora, Lund, Motwani, Sudan, Szegedy 1998)

NP = PCP($logn, 1$)

Επαλήθευση απόδειξης:

- Πιθανοτικός Verifier
- Σταθερό Πλήθος Από Ανεξάρτητα Queries σε οποιαδήποτε σημείο της απόδειξης
- Χρήση το πολύ $logn$ random coins



Από την θεωρία στην πράξη

(ZK)-SNARK

(**Z**ero-**K**nowledge) **S**uccinct **N**on-**I**nteractive **A**rgument Of **K**nowledge

Πολλές φορές το κομμάτι ZK είναι προαιρετικό

Με απλά λόγια

Μια απόδειξη π ότι μία πρόταση είναι αληθής όπου:

- Η απόδειξη είναι σύντομη
- Η επαλήθευση είναι γρήγορη
- Δεν αποκαλύπτει τίποτα άλλο για την πρόταση

Διαφορά με Σ πρωτόκολλα

- Γενικός υπολογισμός αντί για συγκεκριμένες τιμές
- Πραγματικό ZK αντί για HVZK

Βασικό Σενάριο

- Ένας client θέτει ερώτημα x (verifier V)
- Ένας server (prover P) έχει (προαιρετικά) μία ιδιωτική είσοδο w
- Ο V θέλει να μάθει το $y = C(x, w)$ όπου C δημόσια διαθέσιμη συνάρτηση
- V: ενδιαφέρεται ότι το y υπολογιστηκε σωστά. Δεν έχει τους πόρους ούτε το w ώστε να το υπολογίσει μόνος του
- P: ενδιαφέρεται να μη διαρρεύσει το w (αν υπάρχει)

Λύση: Ο P παράγει μια απόδειξη π ότι ο υπολογισμός εκτελέστηκε σωστά

Ο V επαληθεύει την απόδειξη

Τι προσφέρει ένα ZK-SNARK

- **Z**ero-**K**nowledge: Ο V μαθαίνει μόνο την εγκυρότητα του υπολογισμού
- **S**uccinct: Η απόδειξη είναι πολύ μικρή σε σχέση με τον υπολογισμό Στόχοι:
  - μέγεθος π: sublinear στο μέγεθος του w (ακόμα και σταθερό)
  - χρόνος επαλήθευσης π: υπογραμμικός στην περιγραφή της C, γραμμικό στο x - ανεξάρτητο από χρόνο εκτέλεσης της C
  - χρόνος δημιουργίας π: γραμμικός (ή quasilinear) στην περιγραφή της C
- **N**on **I**nteractive: Παράγονται μόνο από τον P και είναι δημόσια επαληθεύσιμες συμβολοσειρές.
- **A**rguments: Ένας πανίσχυρος υπολογιστικά P μπορεί να κλέψει - ορθότητα εγγυημένη μόνο για PPT prover.
- of **K**nowledge: Ο P πρέπει να γνωρίζει τον w

Εφαρμογές i

Προσωπικό σχόλιο (εκτός ύλης)

Η ανάπτυξη (θεωρητική - πρακτική) των zk-SNARKs είναι η μόνη γενικά θετική συνέπεια των blockchains!

Χωρίς ιδιωτικότητα (non ZK)

- Scalability: Επαλήθευση υπολογισμών που έγιναν off-chain
- Bridges: Μεταφορά αξίας μεταξύ blockchains (proof of consensus)

Με ιδιωτικότητα (ZK)

- Απόδειξη ότι μια συναλλαγή είναι έγκυρη χωρίς την αποκάλυψη του αποστολέα, παραλήπτη, ποσού (zCash, Tornado cash κλπ.)
- Απόδειξη ότι ένα exchange είναι solvent ή ότι τηρεί ένα κανονιστικό πλαίσιο

Εφαρμογές ii

Εκτός blockchain

- Επαλήθευση cloud computations
- Αυθεντικότητα φωτογραφιών [NT16]
  - Οι σύγχρονες φωτογραφικές μηχανές υπογράφουν τα pixels φωτογραφιών + timestamp + συντεταγμένες GPS
  - Τα αρχεία υφίστανται επεξεργασία πριν τη δημοσίευση οπότε η υπογραφή είναι άκυρη
  - zkSNARK που αποδεικνύει ότι:
    μια δημοσιευμένη φωτογραφία y όντως προύκυψε από μια αρχική φωτογραφία x μέσω ένος συγκεκριμένου μετασχηματισμού C

Γενική ιδέα

1. Κωδικοποίηση υπολογισμού C σε domain specific language (Arkworks, Zokrates, Cairo, ...)
2. Μεταγλώττιση σε πιο χρηστική ενδιάμεση αναπαράσταση (κύκλωμα, constraint system)
    - ορισμός transcript εκτέλεσης υπολογισμού
    - αντίστοίχιση εισόδων - εξόδων των πυλών
3. Μετατροπή transcript σε πολυώνυμο
    - Υπολογισμός πολυωνύμου από ζεύγη input output
    - Lagrange interpolation - extension σε σώμα αντί για $\{0, 1\}^\lambda$
4. Χρήση συστήματος απόδειξης γνώσης πολυωνύμου
    - Γνώση vectors με συγκεκριμένο εσωτερικό γινόμενο (συντελεστές, δυνάμεις)

Βασικές Έννοιες

Ορισμός

Μια τριάδα αλγορίθμων $(\mathsf{KGen}, \mathsf{Commit}, \mathsf{Open})$ τέτοια ώστε:

- $\mathsf{pk} \leftarrow \mathsf{KGen}(1^\lambda)$
- $(c, \mathsf{vk}) \leftarrow \mathsf{Commit}(\mathsf{pk}, m)$
- $\{0,1\} \leftarrow \mathsf{Vf}(\mathsf{pk}, \mathsf{vk}, c, m)$

Binding

$\forall$PPT $\mathcal{A}$:

$$\Pr\left[\begin{array}{c} m_1 \neq m_2 \wedge \\ \mathsf{V}(\mathsf{pk}, \mathsf{vk}_1, c, m_1) = 1 \wedge \\ \mathsf{V}(\mathsf{pk}, \mathsf{vk}_2, c, m_2) = 1 \end{array} \middle| \begin{array}{c} \mathsf{pk} \leftarrow \mathsf{Setup}(1^\lambda); \\ (c, m_1, m_2, \mathsf{vk}_1, \mathsf{vk}_2) \leftarrow \mathcal{A}(\mathsf{pk}) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Commitment schemes ii

Hiding

$\forall$PPT $\mathcal{A}$:

$$\Pr\left[\, b = b' \;\middle|\; \begin{array}{c} \mathsf{pk} \leftarrow \mathsf{Setup}(1^\lambda); \\ (m_0, m_1) \leftarrow \mathcal{A}(\mathsf{pk}); \\ b \leftarrow\!\$\; \{0,1\}; \\ (c_b, \mathsf{vk}_b) \leftarrow \mathsf{Commit}(\mathsf{pk}, m_b); \\ b' \leftarrow \mathcal{A}(\mathsf{pk}, c_b) \end{array} \right] \leq \mathsf{negl}(\lambda)$$

Homomorphic Commitments i

Pedersen commitments

- $(g, h, \mathbb{G}, q) \leftarrow \mathsf{KGen}(1^\lambda)$ με DLP δύσκολο στην $\mathbb{G}$
- $(g^{m} \cdot h^r, r) \leftarrow \mathsf{Commit}(\mathsf{pk}, m)$ με $r \leftarrow\$ \mathbb{Z}_q$
- $\mathsf{Vf}(\mathsf{pk}, \mathsf{vk}, C, m) \triangleq C := g^{m} \cdot h^r$

Θεώρημα

Το σχήμα δέσμευσης του Pedersen έχει perfect hiding και binding το οποίο εξαρτάται από τη δυσκολία του DLP στη $\mathbb{G}$.

Homomorphic Commitments ii

Σ-πρωτόκολλο για γνώση ανοίγματος

PoK$\{(m,r) \in \mathbb{Z}_q^2 : C = g^m \cdot h^r, \quad (C,g,h) \in \mathbb{G}^3\}$

- P : $(a,b) \leftarrow\$ \mathbb{Z}_q^2$. Υπολογισμός $T := g^a \cdot h^b$
- V : $e \leftarrow\$ \mathbb{Z}_q$
- P : $s_1 = a + em, s_2 = b + er$
- V : $g^{s_1} h^{s_2} g^{s_1'} h^{s_2'} = TC^e$

Απόδειξη Ασφάλειας

Completeness

$$
\begin{aligned}
g^{s_1} h^{s_2} &= g^{a+em} h^{b+er} = \\
&= g^a h^b \cdot (g^m h^r)^e = \\
&= T \cdot C^e
\end{aligned}
$$

Special-Soundness

Δύο accepting εκτελέσεις (T, e, s_1, s_2) και (T, e', s_1', s_2') οδηγούν σε witness (δηλ. valid opening) για το commitment C:

$$
\begin{aligned}
m^* &= \frac{s_1 - s_1'}{e - e'} \\
r^* &= \frac{s_2 - s_2'}{e - e'} \\
g^{m^*} h^{r^*} &= g^{(s_1 - s_1')(e-e')^{-1}} h^{(s_2 - s_2')(e-e')^{-1}} \\
&= (g^{s_1} h^{s_2} g^{-s_1'} h^{-s_2'})^{(e-e')^{-1}} \\
&= (TC^e \cdot (TC^e)^{-1})^{(e-e')^{-1}} \\
&= C
\end{aligned}
$$

HVZK Simulated transcript: $(g^m h^r C^{-e}, e, m, r)$ για $m, r, e \leftarrow\$ \mathbb{Z}_q$ ακολουθεί ακριβώς ίδια κατανομή με το (T, e, s_1, s_2).

Homomorphic property:

$$\mathsf{Commit}(\mathsf{pk}, m_1) \cdot \mathsf{Commit}(\mathsf{pk}, m_2) =$$
$$g^{m_1} h^{r_1} \cdot g^{m_2} h^{r_2} = g^{m_1+m_2} h^{r_1+r_2} =$$
$$\mathsf{Commit}(\mathsf{pk}, m_1 + m_2)$$

Generalized Pedersen Commitments - Vector commitments

Δίνεται vector $\mathbf{m} = (m_1, \cdots, m_n)$ και $h \leftarrow\$ \mathbb{G}$ και $\mathbf{g} = (g_1, \cdots, g_n) \leftarrow\$ \mathbb{G}^n$.

$\mathsf{Commit}(\mathbf{m}, r) = h^r \prod_{i=1}^{n} g_i^{m_i}$

Σε προσθετικό notation: $\mathsf{Commit}(\mathbf{m}, r) = rh + \sum_{i=1}^{n} g_i \cdot m_i$

$\mathsf{Commit}(\mathbf{m}, r) = rh + \langle \mathbf{g}, \mathbf{m} \rangle$ - εσωτερικό γινόμενο.

Συμβ. $\langle\langle \mathbf{g}, \mathbf{m} \rangle\rangle = \prod_{i=1}^{n} g_i^{m_i}$

Pairings i

$\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ πεπερασμένες κυκλικές ομάδες

Ζεύξη (pairing-bilinear map): Μία *αποδοτικά* υπολογίσιμη συνάρτηση

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$$

1) Διγραμμική (bilinear):

$$e(g_1 \cdot g_2, h_1) = e(g_1, h_1) \cdot e(g_2, h_1) \text{ και}$$

$$e(g_1, h_1 \cdot h_2) = e(g_1, h_1) \cdot e(g_1, h_2)$$

ή ισοδύναμα $e(g^a, h^b) = e(g, h)^{ab} \quad \forall g \in \mathbb{G}_1, h \in \mathbb{G}_2 \; a, b \in \mathbb{Z}$

2) Μη εκφυλισμένη (non-degenerate):
Αν $\mathbb{G} = \langle g \rangle$ τότε $\mathbb{G}_T = \langle e(g, g) \rangle$

Pairings ii

Μπορεί και $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$ (συμμετρικό pairing πχ. Weil)

Συνήθως: $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G} \subseteq \mathcal{E}(\mathbb{F}_p), \mathbb{G}_T \subseteq \mathbb{F}^*_{p^a}$

Συνέπεια ορισμού: Συμμετρία $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$

Pairings: γιατί είναι χρήσιμα;

Δίνονται 3 (computationally hiding) commitments $g^{m_1}, g^{m_2}, g^{m_3}$

Μπορώ να ελέγξω εύκολα αν $m_3 = m_1 + m_2$

ελέγχοντας αν $g^{m_3} = g^{m_1} \cdot g^{m_2}$

'Ομως:

Χωρίς pairing δεν μπορώ να ελέγξω αν $m_3 = m_1 \cdot m_2$ (δύσκολο DDHP)

Με pairing μπορώ, ελέγχω αν $e(g^{m_1}, g^{m_2}) = e(g^{m_3}, g) \Leftrightarrow e(g,g)^{m_1 m_2} = e(g,g)^{m_3}$ (εύκολο DDHP όχι όμως και DLP)

Pairings iv

Διγραμμικό Πρόβλημα Απόφασης Diffie-Hellman Διαχωρίζονται στοιχεία του $\mathbb{G}_T$

BDDHP

Δίνονται: δύο στοιχεία $h, g \in \mathbb{G}$ και τα στοιχεία $g^\alpha, g^\beta, e(h,g)^c$.

Ζητείται: Ισχύει $c = \alpha\beta$;

Παραδείγματα Elliptic Curves για Pairings:

- BLS12-381 (Barreto, Lynn, and Scott curves - Sean Bowe 2017)
- Εξίσωση: $y^2 = x^3 + 4 \bmod q$
- q πρώτος 381 bits
- $\mathbb{G}_1 \subset \mathcal{E}(\mathbb{F}_q)$
- $\mathbb{G}_2 \subset \mathcal{E}(\mathbb{F}_{q^2})$
- $\mathbb{G}_T \subset \mathbb{F}_{q^{12}}$ με τάξη 255 bits

Εφαρμογές PBC - Τριμερής ανταλλαγή κλειδιού

Έστω κυκλική ομάδα με $\mathbb{G} = \langle g \rangle$

Τρεις οντότητες A, B, C με ζευγάρια ιδιωτικών - δημοσίων κλειδιών $(x_A, Y_A = g^{x_A}), (x_B, Y_B = g^{x_B}), (x_C, Y_C = g^{x_C})$.

Μπορεί να συμφωνηθεί ένα κοινό κλειδί μεταξύ τους;

Pairings vi

Χωρίς pairings - σε 3 γύρους

1. Ο A στέλνει το Y_A στον B, ο B στέλνει το Y_B στον C, ο C στέλνει το Y_C στον A (κυκλικά).
2. Ο A υπολογίζει το $T_A = Y_C^{x_A} = g^{x_C x_A}$, ο B υπολογίζει το $T_B = Y_A^{x_B} = g^{x_B x_A}$ και ο C υπολογίζει το $T_C = Y_B^{x_C} = g^{x_B x_C}$
3. Ο A στέλνει το T_A στον B, ο B στέλνει το T_B στον C, ο C στέλνει το T_C στον A (πάλι κυκλικά).
4. Όλοι υπολογίζουν το κοινό κλειδί ως εξής:
    - Ο A με $T_C^{x_A} = g^{x_B x_C x_A}$
    - Ο B με $T_A^{x_B} = g^{x_C x_A x_B}$
    - Ο C με $T_B^{x_C} = g^{x_A x_B x_C}$

Pairings vii

Με pairings - σε 1 γύρο (Joux-2000)

Υποθέτουμε δύο ομάδες $\mathbb{G}$, $\mathbb{G}$ με τάξη ένα πρώτο q και μία συμμετρική διγραμμική ζεύξη $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$.

- Όλοι οι συμμετέχοντες εκπέμπουν τα δημόσια κλειδιά τους $Y_A = g^{x_A}, Y_B = g^{x_B}, Y_C = g^{x_C}$.
- Με την βοήθεια της ζεύξης το κοινό κλειδί μπορεί να υπολογιστεί ως εξής:
  - $e(Y_B, g^{Y_c})^{x_A} = e(g, g)^{x_B x_C x_A}$
  - $e(Y_A, Y_c)^{x_B} = e(g, g)^{x_A x_C x_B}$
  - $e(Y_A, Y_B)^{x_C} = e(g, g)^{x_A x_B x_C}$

Υπογραφές BLS [BLS01]

- Δημιουργία κλειδιών: $(\mathbb{G}, \mathbb{G}_T, e, x, Y) :=$ KGen(1^λ)
  - Ομάδες $(\mathbb{G} = \langle g \rangle, \mathbb{G}_T)$ τάξης q με δύσκολο CDH
  - Pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$
  - Συνάρτηση σύνοψης: H $: \{0,1\}^* \to \mathbb{G}$
  - Κλειδί υπογραφής sk: $x \leftarrow\$ \mathbb{Z}_q$
  - Κλειδί επαλήθευσης vk: $Y := g^x$
- Υπογραφή:
  - Υπολογισμός $h :=$ H(m)
  - Υπολογισμός $\sigma := h^x$
  - Επιστροφή $\sigma \in \mathbb{G}$
- Επαλήθευση:
  - Υπολογισμός $h :=$ H(m)
  - Έλεγχος $e(g, \sigma) = e(Y, h)$

Υπογραφές BLS - Ιδιότητες

Ορθότητα:

$e(g, \sigma) = e(g, h^x) = e(g, \mathsf{H}(m))^x$ και

$e(y, h) = e(g^x, \mathsf{H}(m)) = e(g, \mathsf{H}(m))^x$

Ασφάλεια:

Πλαστογράφηση οδηγεί στην παραβίαση του CDH στην $\mathbb{G}$ (άσκηση)

Aggregation:

Χρήστες: $\{(x_i, Y_i = g^{x_i})\}_{i=1}^n$, υπογραφές: $\{\sigma_i\}_{i=1}^n$

Δημιουργία κοινής υπογραφής: $\sigma := \prod_{i=1}^n \sigma_i$

Επαλήθευση: $\prod_{i=1}^n e(Y_i, \mathsf{H}(m_i)) = e(g, \sigma)$

Schwartz-Zippel Lemma

Θεώρημα

Δίνεται ένα σώμα $\mathbb{F}$ και $P : \mathbb{F}^m \to \mathbb{F}$ ένα (μη μηδενικό) πολυώνυμο βαθμού το πολύ d. Τότε για κάθε πεπερασμένο σύνολο $S \subseteq \mathbb{F}$:

$$\Pr_{x \leftarrow_{\$} S^m}[P(x) = 0] \leq \frac{d}{|S|}$$

Βαθμός: Μέγιστο άθροισμα εκθετών όρου

Θα χρησιμοποιούμε $\mathbb{F} = \{0, 1, \cdots p - 1\}$ με p πρώτο

Εφαρμογή:

Δύο διαφορετικά πολυώνυμα P, Q στο $\mathbb{F}^m \to \mathbb{F}$, βαθμού το πολύ d συμφωνούν το πολύ σε ποσοστό $\frac{d}{|\mathbb{F}|}$ σημείων. Δηλαδή $\Pr_{r \leftarrow_{\$} S^m}[P(r) = Q(r)] \leq \frac{d}{|S|}$

Ένα σημαντικό στοιχείο για το succinctness αν $d \ll |\mathbb{F}|$

Αναπαράσταση πολυωνύμων i

Έστω ένα πολυώνυμο βαθμού $d - 1$:

$$P(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{d-1}$$

Συνήθης Αναπαράσταση:
Χρησιμοποιώντας τους συντελεστές $(a_0, a_1, \cdots, a_{d-1})$

Ισοδύναμη Αναπαράσταση:
Χρησιμοποιώντας d ζεύγη σημείων
$((0, P(0)), (1, P(1)), \cdots, (d-1, P(d-1)))$

Αναπαράσταση πολυωνύμων ii

Παρεμβολή Lagrange

- Πολυώνυμα Lagrange $\lambda_i(x) = \prod_{k=0,k\neq i}^{d-1} \frac{x-k}{i-k}$
  Διαίσθηση: Selector polynomials
- $\lambda_i(j) = 0 \quad j \neq i$
  Πρέπει να είναι πολλαπλάσιο του $\prod_{k=0,k\neq i}^{d-1}(x-k)$
- $\lambda_i(i) = 1$
  Διαιρούμε με την αποτίμηση του για i $\prod_{k=0,k\neq i}^{d-1}(i-k)$
- Προκύπτει το

$$L(x) = \sum_{i=0}^{d-1} P(i)\lambda_i(x) = P(0)\lambda_0(x) + \cdots + P(d-1)\lambda_{d-1}(x)$$

αφού:

$$L(i) = P(i)\lambda_i(i) + \sum_{k=0,k\neq i}^{d-1} P(k)\lambda_k(x) = P(i)\cdot 1 + 0$$

Πολυπλοκότητα αποτίμησης Με βάση τον ορισμό: $\mathcal{O}(d^2)$ πράξεις (προσθέσεις, πολλαπλασιασμοί) στο $\mathbb{F}$

Όμως κάθε λ_i μπορεί να προκύψει από το λ_{i-1} με σταθερό αριθμό πράξεων:

$$\begin{aligned}
\lambda_i(j) &= \prod_{k=0, k\neq i}^{d-1} \frac{j-k}{i-k} \\
&= \frac{(j-0)(j-1)\cdots(j-(i-1))(j-(i+1)\cdots(j-(d-1))}{(i-0)(i-1)\cdots(i-(i-1))(i-(i+1)\cdots(i-(d-1))} \\
\lambda_{i-1}(j) &= \prod_{k=0, k\neq i-1}^{d-1} \frac{j-k}{(i-1)-k} \\
&= \frac{(j-0)(j-1)\cdots(j-(i-1)-1)(j-i)\cdots(j-(d-1))}{((i-1)-0)((i-1)-1)\cdots((i-1)-(i-1)-1)((i-1)-i))\cdots((i-1)-(d-1))}
\end{aligned}$$

$$\lambda_i(j) = \lambda_{i-1}(j) \cdot \frac{j-(i-1)}{(j-i)} \cdot \frac{(i-d)}{(i-0)}$$

Τελικά $\mathcal{O}(d)$ πράξεις (προσθέσεις, πολλαπλασιασμοί) στο $\mathbb{F}$

Επεκτάσεις ως PCP για πολυώνυμα

Επέκταση

Ένα πολυώνυμο $P : \mathbb{F}^m \to \mathbb{F}$ βαθμού d επεκτείνει μια συνάρτηση $f : \{0,1\}^m \to \mathbb{F}$ αν και μόνο αν:

$$\forall x \in \{0,1\}^m : \; f(x) = P(x)$$

Παρατήρηση

Αν δύο συναρτήσεις $f_1, f_2 : \{0,1\}^m \to \mathbb{F}$ διαφωνούν έστω και σε μία είσοδο, τότε οι επεκτάσεις τους με βαθμό το πολύ d, $P_1, P_2 : \mathbb{F}^m \to \mathbb{F}$ θα διαφωνούν σχεδόν παντού.

Αφού βαθμός P_1, P_2 το πολύ d από Schwartz-Zippel Lemma θα συμφωνούν σε $\frac{d}{|\mathbb{F}|}$ σημεία $d \ll |\mathbb{F}|$.

Arithmetization i

Μετατροπή boolean formula σε αριθμητικό κύκλωμα

Αριθμητικό Κύκλωμα C

Directed Acyclic Graph όπου:

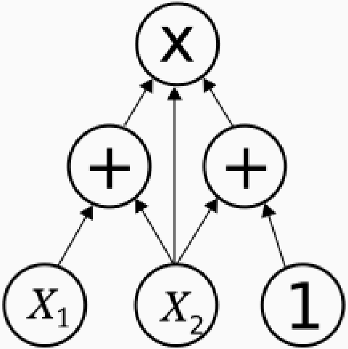
- Input gates: Κόμβοι με in-degree 0 - μεταβλητές $x_1, x_2, \cdots, x_n \in \mathbb{F}$ ή στοιχεία του $\mathbb{F}$
- Υπόλοιπες πύλες: Πράξεις $+, \cdot$

Ορίζει πολυώνυμο m-μεταβλητών και διαδικασία αποτίμησης

Μέγεθος C: $|C|$=πλήθος πυλών

Λογικό AND $x_1 + x_2$
Λογικό OR $x_1 + x_2 - x_1 x_2$
NOT: $1 - x$

Παράδειγμα: Το κύκλωμα:



αναπαρίσταται από το πολυώνυμο $P(x_1, x_2) = (x_1 + x_2)x_2(x_2 + 1)$

zk-SNARKS

Ορισμός zk-SNARK

(strong) zk-SNARK με preprocessing

Ένα (strong) zk-SNARK με preprocessing για ένα κύκλωμα C είναι μια τριάδα αλγορίθμων (Setup, P, V):

- $(\mathsf{pp}, \mathsf{vp}) \leftarrow \mathsf{Setup}(1^\lambda, C)$ PPT αλγόριθμος - Οι παράμετροι $(\mathsf{pp}, \mathsf{vp})$ είναι δημόσιες.
- $\pi \leftarrow \mathsf{P}(\mathsf{pp}, (x, y), w)$ με $|\pi| = \mathsf{o}(|w|)$ (sublinear) $(|\pi| = \mathcal{O}(\log |C|)$ για strong succinctness)
- $\{0, 1\} \leftarrow \mathsf{V}(\mathsf{vp}, x, \pi)$ με $T(\mathsf{V}) = \mathcal{O}(|x|) + \mathsf{o}(|C|)$ $(T(\mathsf{V}) = \mathcal{O}(|x| + \log |C|)$ για strong succinctness)

τέτοια ώστε το transcript $\langle \mathsf{P}, \mathsf{V} \rangle$ να διαθέτει completeness, knowledge soundness, zero-knowledge

Δεύτερο στοιχείο succinctness: Ο V δεν προλαβαίνει να διαβάσει το κύκλωμα. Η δημόσια παράμετρος vp λειτουργεί ως 'περίληψη'.

Παραλλαγές Preprocessing

- **Trusted** (per circuit) (Groth16 [Gro16])
  - Η τυχαιότητα πρέπει να παραμένει μυστική, αλλιώς ο P θα μπορεί να δημιουργεί unsound αποδείξεις
  - Υλοποίηση με Secure Multi Party Computation
- **Updatable** (Plonk [GWC19]): Δύο φάσεις
  - gp $\leftarrow$ $\text{Setup}_1(1^\lambda)$ Ο πιθανοτικός αλγόριθμος
  - $(\text{pp}, \text{vp}) \leftarrow \text{Setup}_2(\text{gp}, C)$ ντετερμινιστικός αλγόριθμος

  Η τυχαιότητα πρέπει να παραμένει μυστική μόνο στην πρώτη φάση, ώστε ο P να μην μπορεί να δημιουργεί unsound αποδείξεις
  Γενίκευση: Updatable MPC: Παίκτης i δέχεται ως input το gp_{i-1}, βάζει το δικό του randomness και εξάγει gp_i. Πιο εύκολο να βρεθεί ένας honest.
- **Transparent** (Bulletproofs [BCC+16, BBB+17] / STARK): Δεν χρειαζεται η τυχαιότητα να παραμένει μυστική

Ιδιότητες: Completeness

Αν P, V honest τότε ο V αποδέχεται.

Completeness

$\forall (x, y), w : C(x, w) = y$:

$$\Pr\left[\mathsf{V}(\mathsf{vp}, (x, y), \pi) = 1 \;\middle|\; \begin{array}{c} (\mathsf{pp}, \mathsf{vp}) \leftarrow \mathsf{Setup}(1^\lambda); \\ ((x, y), w) \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{vp}) \text{ note: } C(x, w) = y; \\ \pi \leftarrow \mathsf{P}(\mathsf{pp}, (x, y), w) \end{array} \right] = 1$$

Ιδιότητες: Soundness

Αν $\mathsf{V}(\mathsf{vp}, (x, y), \cdot) = 1$ τότε P γνωρίζει $w : C(x, w) = y$

Δηλαδή μπορεί να εξαχθεί w από τον P

Adaptive Knowledge Soundness

Το $\Pi = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ είναι adaptively knowledge sound αν για κάθε PPT αντίπαλο $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ με:

$$\Pr\left[\begin{array}{c|c} \mathsf{V}(\mathsf{vp}, (x, y), \pi) = 1 & \begin{array}{c} \mathsf{gp} \leftarrow \mathsf{Setup}_1(1^\lambda); \\ (C, (x, y), \mathbf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}_0(\mathsf{gp}); \\ (\mathsf{pp}, \mathsf{vp}) \leftarrow \mathsf{Setup}_2(C); \\ \pi \leftarrow \mathcal{A}_1(\mathsf{pp}, x); \end{array} \end{array}\right] \geq \mathsf{negl}(\lambda)$$

υπάρχει PPT αλγόριθμος $\mathcal{E}$ ώστε:

$$\Pr\left[\begin{array}{c|c} C(x, w) = y & \begin{array}{c} \mathsf{gp} \leftarrow \mathsf{Setup}_1(1^\lambda); \\ (C, (x, y)) \leftarrow \mathcal{A}_0(\mathsf{gp}); \\ w \leftarrow \mathcal{E}(\mathsf{gp}, C, (x, y), \mathbf{st}_{\mathcal{A}}); \end{array} \end{array}\right] \geq 1 - \mathsf{negl}(\lambda)$$

Ιδιότητες: Zero-Knowledge

Zero-Knowledge

$\forall (x, y), w : C(x, w) = y$ υπάρχει PPT αλγόριθμος Sim ώστε:

$$\Pr\left[b = b' \;\middle|\; \begin{array}{c} (\mathsf{pp}, \mathsf{vp}) \leftarrow \mathsf{Setup}(1^\lambda); \\ b \leftarrow\$ \{0,1\}; \\ \textbf{if } b = 1 : \quad \pi \leftarrow \mathsf{P}(\mathsf{pp}, (x, y), w); \\ \textbf{if } b = 0 : \quad \pi \leftarrow \mathsf{Sim}(\mathsf{pp}, (x, y)); \\ b' \leftarrow \mathcal{A}(\mathsf{vp}) \end{array} \right] \leq \mathsf{negl}(\lambda)$$

Συστατικά zk-SNARKS i

- Functional Commitment Scheme
  - Δέσμευση σε συναρτήσεις
    - Polynomial: Δέσμευση σε πολυώνυμο μίας μεταβλητής βαθμού το πολύ d
    - Multilinear: Δέσμευση πολυώνυμο πολλών μεταβλητών βαθμού 1 πχ. $P(x_1, x_2, x_3) = x_1x_2 + x_3 + x_1x_2$
    - Vector commitment: Δέσμευση σε $\mathbf{u} = (u_1, u_2, \cdots u_d)$ Μπορεί να ζητηθεί ανοιγμα οποιουδήποτε στοιχείου
    - Υπολογισμός εσωτερικού γινομένου $f_u(v) = \langle v, u \rangle = \sum_{i=1}^{n} v_i u_i$
  - Ασφάλεια με κρυπτογραφικές υποθέσεις

Συστατικά zk-SNARKS ii

- Interactive Oracle Proof (IOP)
  - Σε κάθε γύρο του πρωτοκόλλου ο verifier **δεν** χρειάζεται να διαβάσει όλα τα μηνύματα του prover
  - Αποκτά oracle access σε αυτά, δηλ. query σε σταθερό πλήθος συμβόλων
  - Διαίσθηση:
    - Για succinctness, ο verifier δεν πρέπει (δεν μπορεί) να διαβάσει ολόκληρο το witness
    - Oracle access: Επιτρέπει την αποκάλυψη 'τμημάτων' του witness
  - Ασφάλεια *χωρίς* κρυπτογραφικές υποθέσεις

zk-SNARKs με Polynomial Commitments

- Δημιουργία polynomial IOP
  - Τα μηνύματα του prover ορίζουν ένα πολυώνυμο P συγκεκριμένου βαθμού d σε κάποιο σώμα $\mathbb{F}$ με πάρα πολλούς συντελεστές (μεγάλο κύκλωμα)
  - Ζητάει αποτιμήσεις σε συγκεκριμένα σημεία επιλογής του τα οποία απαντάει ο prover (query access)
  - Πρέπει να μπορεί να επαληθεύσει τις απαντήσεις
- Αντικατάσταση μηνυμάτων με polynomial commitment schemes για να μπορεί να δεσμεύεται ο prover.
- Non-interactive με Fiat-Shamir Heuristic

Polynomial Commitments

Ένας P θέλει να δεσμευτεί σε ένα πολυώνυμο P βαθμού το πολύ d (Στην πραγματικότητα σε όλες τις αποτιμήσεις του!)

O V επιλέγει ένα σημείο u στο οποίο ζητάει την αποτίμηση του P

O P υπολογίζει $P(u) = v$ χωρίς να μπορεί να κλέψει (πχ. άλλο πολυώνυμο)

Trivial Λύση

- P στέλνει τους συντελεστές του P
- V στέλνει το u
- P υπολογίζει $P(u) = v$
- V υπολογίζει και αυτός το $P(u) = v$

Παρατήρηση:
$P(u) = \sum_{i=0}^{d} a_i u^i = \langle \mathsf{a}, \mathsf{u} \rangle$ με $\mathsf{a} = (a_0, a_1, \cdots, a_d)$ και $\mathsf{u} = (1, u, \cdots, u^d)$

Polynomial Commitment Schemes (PCS) ii

Τέλειο knowledge soundness ομως:

Μειονεκτήματα:

- Μεγάλη (γραμμική) περιγραφή του P
- Μεγάλος (γραμμικός) υπολογισμός για τον V
- Όχι μηδενική γνώση

Λύση:

- Αποστολή δέσμευσης για το P
- Για να μην μπορεί να κλέψει και απόδειξη π ότι $P(u) = v$ (ProveEval)
- Ο verifier επαληθεύει το π (VerifyEval)

Σύνταξη PCS

- gp $\leftarrow$ Setup$(1^\lambda, \mathcal{P})$
  Επιλογή πολυωνύμου $P \in \mathcal{P}$
- $C_P \leftarrow$ Commit(gp$, P)$
  Δέσμευση σε πολυώνυμο P. Στην πραγματικότητα δέσμευση σε αποτίμηση για κάποιο σημείο του.
- $\pi \leftarrow$ ProveEval(gp$, P, u, v)$
  Απόδειξη ότι $P(u) = v$. Το σημείο u επιλέγεται από τον V.
- VerifyEval(gp$, C_P, \pi, u, v)$
  Επαλήθευση αποτίμησης

Ασφάλεια PCS

Binding

Υπάρχει μοναδικό πολυώνυμο P ώστε ο P να μπορεί να εξηγήσει τις αποτιμήσεις.

Extractability

Ο P γνωρίζει το πολυώνυμο P

Για κάθε PPT κακό prover P* που πείθει τον V υπάρχει αποδοτικός αλγόριθμος $\mathcal{E}$ ο οποίος μπορεί να εξάγει το P από το transcript με τον P*

Extractability = Ισχυρότερο Binding

Εφαρμογές: Vector Commitments i

Ο P θέλει να δεσμευτεί σε ένα vector $\mathbf{u} = (u_1, u_2, \cdots u_d)$.

1ος τρόπος: Με Merkle Tree

Τα στοιχεία του vector είναι τα φύλλα στο Merkle Tree.

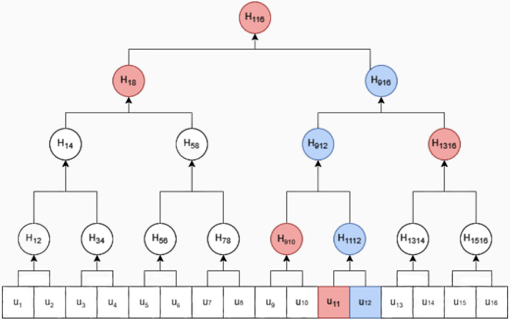
Η δέσμευση είναι η ρίζα

Επιβεβαίωση τιμών για στοιχεία του vector: $u_i = x$? Αποκάλυψη του μονοπατιού για ανακατασκευή της απόδειξης.

Κόστος για 1 ερώτηση: $\mathcal{O}(\log d)$

Κόστος για n ερωτήσεις $\mathcal{O}(n \cdot \log d)$

Παράδειγμα: $\mathbf{u} = (u_1, \cdots u_{16})$



Εφαρμογές: Vector Commitments iii

2ος τρόπος: Με Polynomial Commitments

Υπολογισμός πολυωνύμου P ώστε $P(i) = u_i \; i \in [d]$

Χρήση $C_P = \text{PCS.Commit}(\text{gp}, P)$

Επιβεβαίωση τιμών για στοιχεία του vector: $u_i = x?$
$\text{PCS.ProveEval}(\text{gp}, P, i, x)$

Κόστος για 1 ερώτηση: Ακόμα και $\mathcal{O}(1)$

Κόστος για n ερωτήσεις: Ακόμα και $\mathcal{O}(n)$

Άλλη λύση για vector commitments: **Verkle Trees** [Kus18]

Ερώτηση: Θα μπορούσε να χρησιμοποιηθεί Merkle tree για polynomial commitment?

Γραμμικό PCS με ZK

- Setup. Επιλογή $g, h \leftarrow\$ \mathbb{G}$
- Commit. P Αποστολή Pedersen Commitments για όλους τους συντελεστές $\mathbf{c} = (\text{Commit}(\text{pk}_i, a_i))_{i=0}^{d}$
- V: Επιλογή u
- ProveEval. P: Αποστολή $P(u)$
- V: Για $\mathbf{u} = (1, u, ..., u^d)$ υπολογισμός $\mathbf{c}^{\mathbf{u}} = \prod_{i=0}^{d} c_i^{u^i}$
- VerifyEval. V: Επαλήθευση ότι το $\mathbf{c}^{\mathbf{u}}$ ανοίγει σε $P(u)$

Πώς μπορεί ο P αν γνωρίζει το u αλλά όχι το P, να αποδείξει ότι $P(u) = v$? Δημιουργία $\mathbf{c}$ ως εξής:

- $c_0 \leftarrow\$ \mathbb{G}$
- $c_i = (c' \cdot c_{i-1}^{-1})^{u^{-i}}$ με $c' \leftarrow\$ \mathbb{G}$
- $c_d = (g^v c_{d-1}^{-1})^{u^{-d}}$

Για soundness:

- παραγωγή απόδειξης γνώσης για τα openings a_i
- ο prover δεν πρέπει να ξέρει το u

PCS με σταθερό μέγεθος απόδειξης - γραμμικό χρόνο επαλήθευσης i

- Setup. Επιλογή $h, g, g_0, \cdots g_d \leftarrow\$ \mathbb{G}$
- Commit. P Χρήση generalised vector commitment για τους συντελεστές $\mathsf{a} = (a_0, a_1, \cdots, a_d)$: $c_{\mathsf{a}} = h^{r_{\mathsf{a}}} \prod_{i=0}^{d} {g_i}^{a_i}$ με $r_{\mathsf{a}} \leftarrow\$ \mathbb{Z}_q$
- V. Αποστολή u.
- ProveEval. Ο P
    - Υπολογισμός $P(u) = v$
    - Υπολογισμός $c_v = g^v h^{r_v}$
    - Αποστολή c_{a}, c_V
- VerifyEval (με ZK).
    - Θα ανοίξουν blinded commitments (όπως σε Schnorr)
    - Ο P επιλέγει $\mathsf{x}, r_{\mathsf{x}}, r'_v \leftarrow\$ \mathbb{Z}_q^{d+3}$ και υπολογίζει το $c_{\mathsf{x}} = \text{Commit}(\mathsf{x}, r_{\mathsf{x}})$ και $c_{v'} = \text{Commit}(\langle \mathsf{a}, \mathsf{x} \rangle, r_{v'})$
    - Αποστέλλει $c_{\mathsf{x}}, c_{v'}$ στον V
    - V Αποστολή $e \leftarrow\$ \mathbb{Z}_q$

PCS με σταθερό μέγεθος απόδειξης - γραμμικό χρόνο επαλήθευσης ii

- Ο P υπολογίζει $\mathsf{s} := \mathsf{x} + e\mathsf{a}$ και $s_{v'} = \langle \mathsf{a}, \mathsf{x} \rangle + ev$
- Ο P δεσμεύεται στο $s, s_{v'}$ ως $c_{\mathsf{s}} = \text{Commit}(\mathsf{s}, r_{\mathsf{s}})$ και $c_{s_{v'}} = \text{Commit}(s_{v'}, r_{s_{v'}})$ με $r_{\mathsf{s}}, r_{s_{v'}}$
- Ο P στέλνει στον V $(\mathsf{s}, r_{\mathsf{s}}, s_{v'}, r_{s_{v'}})$
- Ο V υπολογίζει:

$$c_{\mathsf{s}} := c_{\mathsf{a}}^{e} \cdot c_{\mathsf{x}}$$
$$c_{s_{v'}} := c_{v}^{e} \cdot c_{v'}$$

- Ο V ελέγχει αν:

$$c_{\mathsf{s}} = h^{r_{\mathsf{s}}} \prod_{i} g_i^{s_i}$$
$$c_{s_{v'}} = h^{r_{s_{v'}}} g^{s_{v'}}$$

KZG Commitments

The KZG [KZG10] PCS i

Βασική ιδέα

Για κάθε πολυώνυμο βαθμού d ισχύει $P(u) = v$ αν και μόνο αν υπάρχει πολυώνυμο W (witness) βαθμού $d-1$ τέτοιο ώστε:

$$P(x) - v = W(x) \cdot (x - u)$$

Πιθανοτική επαλήθευση:

- P δεσμεύεται στο $P(s)$, $s \leftarrow\$ \mathbb{Z}_q$
- V διαλέγει u και υπολογίζει $s - u$
- P δεσμεύεται στο $W(s)$
- V ελέγχει μέσω των δεσμεύσεων αν $P(s) - v = W(s)(s - u)$ - χωρίς να τις ανοίξει.

Λεπτομέρεια: Χωρίς κανείς να γνωρίζει το s!

The KZG [KZG10] PCS ii

Χαρακτηριστικά

- Ανάγκη pairing - θέλουμε πολλαπλασιαστικό ομομορφισμό μόνο για μία πράξη.
- Trusted setup
- Μέγεθος απόδειξης $\mathcal{O}(1)$: 1-2 group elements
- Χρόνος επαλήθευσης $\mathcal{O}(1)$: 1-2 έλεγχοι pairings
- Χρόνος δημιουργίας απόδειξης $\mathcal{O}(d)$: ανάλογος με το μέγεθος του πολυωνύμου

The KZG [KZG10] PCS iii

Υπολογιστικές υποθέσεις - d-strong Diffie Hellman Assumption

$\forall$ PPT $\mathcal{A}$:

$$\Pr\left[\; g^{\frac{1}{s-u}} = x \;\middle|\; \begin{array}{c} (g, \mathbb{G}) \leftarrow \mathsf{Setup}(1^{\lambda}); \\ (s, u) \leftarrow\!\!\$\; \mathbb{Z}_q; \\ x \leftarrow \mathcal{A}(u, g, g^{s}, g^{s^2}, \cdots, g^{s^d}) \end{array} \right] \leq \mathsf{negl}(\lambda)$$

Boneh and Boyen (2004)

d-strong Diffie Hellman Assumption $\leq DLP$

The KZG [KZG10] PCS iv

Setup(1^λ,$\mathbb{F}_p^d$)

- $\mathcal{P} = \mathbb{F}_p^d$
- Επιστρέφει $\mathbb{G}, \mathbb{G}_T$ με τάξη πρώτο q ώστε να υπάρχει συμμετρικό bilinear pairing μεταξύ τους $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, όπου ισχύει το DLP για το $\mathbb{G}$ και το $d - SDH$.
- $g \leftarrow\$ \mathbb{G}$
- sk $= s \leftarrow\$ \mathbb{Z}_q$ (trusted setup)
- Υπολογισμός $g^s, \cdots g^{s^d} \in \mathbb{G}$ (powers of tau)
- Διαγραφή s!
- Επιστροφή gp $= (e, \mathbb{G}, \mathbb{G}_T, g, g^s, g^{s^2}, \cdots g^{s^d})$ (structured reference string)

The KZG [KZG10] PCS v

Commit(pk, P)

- Υπολογίζει τη δέσμευση $C_P = g^{P(s)}$ χωρίς το s μέσω των ομομορφικών ιδιοτήτων
- Αν $P(x) = \sum_{i=0}^{d} a_i x^i$ τότε:

$$C_P = g^{P(s)} = \prod_{i=0}^{d} g^{a_i s^i} = \prod_{i=0}^{d} (g^{s^i})^{a_i}$$

Παρατήρηση: Το commitment είναι μόνο ένα στοιχείο της $\mathbb{G}$ ανεξάρτητα από τον βαθμό του P. Σταθερό μέγεθος.

ProveEval(pk, P, u, v)

Ο V επιλέγει το u

Ο P αποδεικνύει $P(u) = v$ χωρίς να αποκαλύψει το πολυώνυμο:

$$P(u) = v \Leftrightarrow (x - u) | (P(x) - v) \Leftrightarrow$$
$$\exists W \in \mathbb{F}_p^{d-1} : W(x)(x - u) = P(x) - v$$

Ο P υπολογίζει το W και δεσμεύεται σε αυτό

Στέλνει το $y = \text{Commit}(\text{pk}, W)$ δηλ. δεσμεύεται στο $g^{W(s)}$

VerifyEval(pk, c, y, u, v)

Ο verifier πρέπει να ελέγξει αν:

$$P(u) = v \Leftrightarrow W(s) \cdot (s - u) = P(s) - v$$

Δεν γνωρίζει όμως το s. Θα χρησιμοποιήσει το pairing, ελέγχοντας αν:

$$e(C_P \cdot g^{-v}, g) = e(g^{s-u}, \pi)$$

Πράγματι:

$$\begin{aligned} e(C_P \cdot g^{-v}, g) = e(g^{s-u}, \pi) \Leftrightarrow \\ e(g^{P(s)-v}, g) = e(g^{s-u}, g^{W(s)}) \Leftrightarrow \\ e(g, g)^{P(s)-v} = e(g, g)^{W(s)\cdot(s-u)} \end{aligned}$$

Απόδειξη ασφάλειας KZG

Binding

Το άνοιγμα του $C_P = P(s)$ σε $v, v'(v \neq v')$ σημαίνει ότι υπάρχουν $y = g^{W(s)}, y' = g^{W'(s)}$:

$$e(g,g)^{W(s)\cdot(s-u)} = e(g,g)^{P(s)-v} \quad \text{και} \quad e(g,g)^{W'(s)\cdot(s-u)} = e(g,g)^{P(s)-v'} \Rightarrow$$

$$e(g,g)^{W(s)\cdot(s-u)\cdot v} = e(g,g)^{P(s)} \quad \text{και} \quad e(g,g)^{W'(s)\cdot(s-u)\cdot v'} = e(g,g)^{P(s)} \Rightarrow$$

$$g^{W(s)\cdot(s-u)\cdot v} = g^{W'(s)\cdot(s-u)\cdot v'} \Rightarrow$$

$$g^{v-v'} = g^{(W'(s)-W(s))\cdot(s-u)} \Rightarrow$$

$$g^{\frac{1}{s-u}} = (y' \cdot y^{-1})^{\frac{1}{v-v'}}$$

Δηλαδή ο prover έλυσε το d-Strong Diffie Hellman Assumption στο $\mathbb{G}$

Τροποποίηση KZG για extractability i

Setup$(1^\lambda, \mathbb{F}_p^d)$

Αντί για

$$(e, \mathbb{G}, \mathbb{G}_T, g, g^s, g^{s^2}, \cdots g^{s^d})$$

επιστρέφεται

$$(e, \mathbb{G}, \mathbb{G}_T, (g, g^b), (g^s, g^{bs}), (g^{s^2}, g^{bs^2}), \cdots (g^{s^d}, g^{bs^d}))$$

με

$b \leftarrow\$ \mathbb{Z}_q$ το οποίο όπως και το s πρέπει να διαγραφεί με ασφάλεια.

Διπλάσιο μέγεθος SRS

Τροποποίηση KZG για extractability ii

Commit(pk, P)

Επιστρέφεται το $C_P = (C_{P_1}, C_{P_2}) = (g^{P(s)}, g^{bP(s)})$ με:

$$C_{P_1} = g^{P(s)} = \prod_{i=0}^{d} g^{a_i s^i} = \prod_{i=0}^{d} (g^{s^i})^{a_i}$$

και

$$C_{P_2} = g^{bP(s)} = \prod_{i=0}^{d} g^{b a_i s^i} = \prod_{i=0}^{d} (g^{b s^i})^{a_i}$$

Διπλάσιο μέγεθος commitment

Μπορεί να υπολογιστεί εύκολα από το SRS

Τροποποίηση KZG για extractability iii

ProveEval(pk, P, u, v)

Ίδια ιδέα με το απλό σχήμα $C_W = \text{Commit}(\text{pk}, W)$

VerifyEval(pk, C_P, C_W, u, v)

O V ελέγχει

$$e(C_{P_1}, g^{-v}) = e(Y, g^s g^{-u})$$

και

$$e(C_{P_1}, g^b) = e(C_{P_2}, g)$$

Πληρότητα: Όμοια + bilinearity

Δυσκολία Extractability i

d-Power Knowledge of Exponent

$\forall$ PPT $\mathcal{A}$ $\exists$ PPT αλγόριθμος $\mathcal{E}$ ώστε:

$$\Pr\left[\begin{array}{c|c} & (e, \mathbb{G}, \mathbb{G}_T, S = ((g, g^b), \\ c = c'^b \wedge & (g^s, g^{bs}), (g^{s^2}, g^{bs^2}), \cdots (g^{s^d}, g^{bs^d}))) \\ c \neq \prod_{i=0}^{d} g^{a_i s^i} & \leftarrow \mathsf{Setup}(1^\lambda); \\ & (c, c') \leftarrow \mathcal{A}(S); \\ & (a_0, \cdots a_d) \leftarrow \mathcal{E}(S) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Οποιαδήποτε στοιχεία της ομάδας c, c' για τα οποία $c = c'^b$ μπορούν να υπολογιστούν μόνο από το SRS, δηλαδή: $c = \prod_{i=0}^{d} (g^{s^i})^{a_i}$ και $c' = \prod_{i=0}^{d} (g^{bs^i})^{a_i}$

Αν ισχύει τότε έχουμε απευθείας Extractability.

Γενίκευση της παραπάνω παλαιότερης υπόθεσης:

Knowledge of Exponent (KEA)

$\forall$ PPT $\mathcal{A}$, $\exists$ PPT αλγόριθμος $\mathcal{E}$ ώστε:

$$\Pr\left[\; Y = X^b \wedge X \neq g^a \;\middle|\; \begin{array}{c} (\mathbb{G}, q, g) \leftarrow \mathsf{Setup}(1^\lambda); \\ b \leftarrow\$\; \mathbb{Z}_q; \\ (X, Y) \leftarrow \mathcal{A}(\mathbb{G}, q, g, g^b) \\ a \leftarrow \mathcal{E}(\mathbb{G}, q, g, g^b) \end{array} \right] \leq \mathsf{negl}(\lambda)$$

Ο υπολογισμός του Y μπορεί να γίνει **μόνο** ως: $g^b \to (g^b)^a$

Knowledge of Exponent ii

Γιατί:

Ο $\mathcal{E}$ έχει πρόσβαση στο ίδιο input με τον $\mathcal{A}$ δηλ. στο $(\mathbb{G}, q, g, g^b)$

Ο $\mathcal{E}$ εξαρτάται από τον $\mathcal{A}$ $(\forall\exists)$. πχ. έχει πρόσβαση στο εσωτερικό του (non black-box access)

Άρα το a προέρχεται από τον $\mathcal{A}$ και ο $\mathcal{E}$ το ανακτά. Ουσιαστικά $\mathcal{A} \equiv \mathcal{E}$ μόνο που έχουν διαφορετικό αποτέλεσμα.

Και είναι ο μονος τρόπος να συμβεί. Γιατί αν υπήρχε και άλλος τρόπος (δηλ. χωρίς ενδιάμεσο υπολογισμό a) Θα υπήρχε $\mathcal{A}^*$ που τον χρησιμοποιούσε και άρα γι' αυτόν δεν θα μπορούσε να υπάρξει extractor.

Σύγκριση με παίγνιο DLP

DLP

$\forall$ PPT $\mathcal{A}$

$$\Pr\left[\begin{array}{c|c} x' = x & \begin{array}{c} (\mathbb{G}, q, g) \leftarrow \mathsf{Setup}(1^\lambda); \\ x \leftarrow\$ \mathbb{Z}_q; \\ y := g^x; \\ x' \leftarrow \mathcal{A}(\mathbb{G}, q, g, y) \end{array} \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Non-Falsifiable Assumptions i

DLP

Πώς θα μπορούσε να καταρριφθεί falsifiable assumption?

Εύρεση αντιπάλου που κερδίζει το παίγνιο πχ. υπολογίζει DLP με μη-αμελητέα πιθανότητα (θετικό αποτέλεσμα)

Χρησιμότητα falsifiable assumption

Εύρεση καλύτερης επίθεσης - Ρύθμιση 1^λ

Non-Falsifiable Assumptions ii

KEA

Πώς θα μπορούσε να καταρριφθεί η υπόθεση (KEA)?

Εύρεση $\mathcal{A}$ ο οποίος παράγει X, Y με $Y = X^b$ χωρίς χρήση a

Εύρεση $\mathcal{A}$ χωρίς αντίστοιχο $\mathcal{E}$ - Ο $\mathcal{A}$ δεν ξέρει το a

Πώς μπορεί να αποδειχθεί ότι ο αντίπαλος δεν ξέρει κάτι (χωρίς πρόσβαση στον κώδικά του);

Αρνητικό αποτέλεσμα - περιορισμένη χρησιμότητα

Controversial

Τόσο falsifiable όσο και non-falsifiable assumptions έχουν βρεθεί ψευδείς!

Αλλά λόγω impossibility result [GW10] - δεν μπορούν να υπάρχουν SNARGs χωρίς non-falsifiable assumptions

Κατανεμημένος τρόπος υπολογισμού του gp. Κανένας δε γνωρίζει το s

Αρκεί ένας τίμιος παίκτης

Βασική ιδέα: Κάθε συμμετέχων:

- Λαμβάνει input το παλιό SRS: $\mathsf{gp} = (g, g^s, \cdots, g^{s^d})$
- Διαλέγει $\tau \leftarrow\$ \mathbb{Z}_q$
- Υπολογίζει νέο SRS: $\mathsf{gp}' = (g^\tau, (g^s)^\tau, \cdots, (g^{s^d})^\tau)$
- Δημοσιοποιεί το νέο SRS
  - Απόδειξη γνώσης τ (Schnorr protocol)
  - Απόδειξη ότι στο gp' οι δυνάμεις είναι συνεχόμενες. Ευκολα με d χρήσεις του pairing

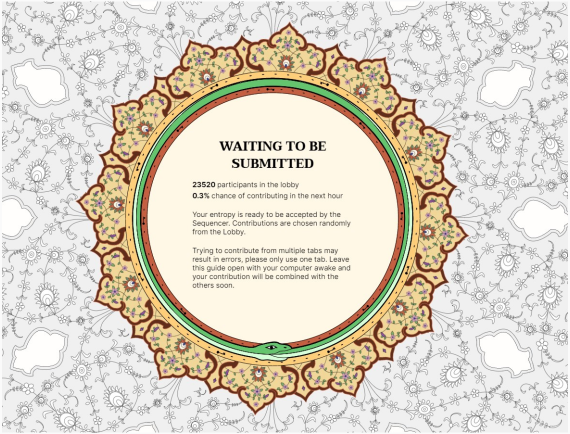
$$\forall i : e(g_1, g_i) = e(g_0, g_{i+1}) \Leftrightarrow$$
$$e(g^{\tau s}, g^{\tau s^i}) = e(g^\tau, g^{\tau s^{i+1}})$$

Powers of tau ceremony ii

Κεντρική οντότητα για υπολογισμό επόμενου (sequencer) αλλά γίνεται και χωρίς αυτήν [NRBB22]

Αυτή τη χρονική στιγμή είναι σε εξέλιξη μία παρόμοια διαδικασία στο Ethereum. *https://ceremony.ethereum.org/*

Powers of tau ceremony iii



KZG - Multiproofs i

Θέλουμε να αποδείξουμε πολλές αποτιμήσεις ($m < d$) του ίδιου πολυωνύμου δηλ.

$$\{P(u_i) = v_i\}_{i=1}^{m}$$

Μπορούμε να το κάνουμε χρησιμοποιώντας μόνο ένα group element.

ProveEval(pk, $P, \{u_i\}, \{v_i\}$)

Χρήση πολυωνυμικής παρεμβολής για $(u_i, P(u_i))$

- Κατασκευάζουμε πολυώνυμο βαθμού $m-1$ ως $H(x) = \sum_{i=1}^{m} P(u_i) \prod_{k=1}^{m} \frac{x-u_k}{u_i-u_k}$
- Παρατηρούμε ότι $P(x) - H(x) = \prod_{k=1}^{m}(x-u_k) \cdot W(x)$ αφού από την κατασκευή $P(u_i) - H(u_i) = 0$
- Άρα πάλι: ProveEval(pk, $P, \{u_i\}, \{v_i\}) = \pi = g^{W(s)}$

KZG - Multiproofs ii

VerifyEval(pk, $C_P, \pi, \{u_i\}, \{v_i\}$)

- Υπολογισμός $H(x)$ και $\prod_{k=1}^{m}(x-u_k)$
- Χρήση pairing:

$$
\begin{aligned}
e(C_P, g^{-H(s)}) &= e(g^{\prod_{k=1}^{m}(s-u_k)}, \pi) \Leftrightarrow \\
e(g^{P(s)}, g^{-H(s)}) &= e(g^{\prod_{k=1}^{m}(s-u_k)}, g^{W(s)}) \Leftrightarrow \\
e(g,g)^{P(s)-H(s)} &= e(g,g)^{W(s)\prod_{k=1}^{m}(s-u_k)}
\end{aligned}
$$

Επέκταση και για πολλαπλά διαφορετικά πολυώνυμα

Βασική Ιδέα

Το commitments $g^{P(s)}$ δεν είναι perfectly hiding (deterministic)
Γενική λύση με χρήση randomisers - ο P δίνει τιμές που μοιάζουν με ομοιόμορφες.
Διαίσθηση: Μετατροπή $g^{P(s)}$ σε Pedersen commitment δηλ. $g^{P(s)}h^r$

Κατασκευή:

- Setup$(1^\lambda, \mathcal{P})$
  - Επιλογή $\eta \leftarrow\$ \mathbb{Z}_q$ και υπολογισμός $h := g^\eta$
  - Προσθήκη h, h^s SRS
  - gp $= (g, g^s, g^{s^2}, \cdots, g^{s^d}, h, h^s)$
  - Διαγραφή s, η
- Commit(gp, P)
  - Επιλογή $r \leftarrow\$ \mathbb{Z}_q$
  - Δέσμευση στο $g^{P(s)+\eta \cdot r} = g^{P(s)}h^r$

KZG με ZK ii

- ProveEval(gp, P, u, v)
  - Αν σταλεί μόνο το $g^{W(s)}$ θα υπάρχει διαρροή. Οπότε πρέπει να κρυφτεί και αυτό.
  - Επίσης πρέπει να προσαρμοστεί η επαλήθευση στη νέα δέσμευση $g^{P(s)+\eta \cdot r}$
  - Επιλογή $r' \leftarrow\$ \mathbb{Z}_q$
  - Μετατρέπουμε την εξισωση από $P(x) - P(u) = (x-u) \cdot W(x)$ σε:

$$(P(x) + ry) - P(u) = (x-u) \cdot (W(x) + r'y) + ry - (x-u)r'y$$

  - Ισοδύναμα:

$$(P(x) + ry) - P(u) = (x-u) \cdot (W(x) + r'y) + y(r - r'(x-u))$$

  - $\pi = (g^{W(s)+r'\eta}, h^{(r-r'(s-u))})$

- VerifyEval(gp, C_p, π, u, v)
  Έλεγχος αν $e(C_p/(g^v \cdot h^{(r-r'(s-u))}), g) = e(g^{s-u}, g^{W(s)+r'\eta})$

$$e(C_p/(g^v \cdot g^{-\eta(r-r'(s-u))}), g) = e(g^{s-u}, g^{W(s)+r'\eta}) \Leftrightarrow$$
$$e(g^{P(s)+\eta\cdot r-v-\eta(r-r'(s-u)}, g) = e(g^{s-u}, g^{W(s)+r'\eta}) \Leftrightarrow$$
$$e(g,g)^{P(s)+\eta\cdot r-v-\eta(r-r'(s-u))} = e(g,g)^{(s-u)(W(s)+r'\eta)}$$

Bulletproofs

Bulletproofs [BCC+16, BBB+17] i

Χαρακτηριστικά

- Δεν απαιτούν pairings, ούτε non-falsifiable assumptions
- Transparent Setup: $d + 1$ group elements
- Proof size: $\mathcal{O}(\log d)$
- Prover complexity: $\mathcal{O}(d)$
- Verifier complexity: $\mathcal{O}(d)$

Bulletproofs [BCC+16, BBB+17] ii

Διαίσθηση

Εφαρμογή Διαίρει και Βασίλευε

P: Αρχικό πολυώνυμο: $P(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_d x^d$

P: Δέσμευση: $C_P = g_0^{a_0} g_1^{a_1} g_2^{a_2} \cdots g_d^{a_d}$

V: Αποστολή u

P: Υπολογισμός $v = a_0 + a_1 u + a_2 u^2 + \cdots + a_d u^d$

P: Μείωση πολυωνύμου σε βαθμό $\frac{d}{2}$ σε συνεργασία με τον V

P: Ενημέρωση gp, C_P, v με τρόπο που μπορούν να επαληθευτούν

Συνολική Δουλειά από P, V $\mathcal{O}\left(\frac{d}{2} + \frac{d}{4} + \cdots + 1\right) = \mathcal{O}(d)$

Αρχική

P: Γνωρίζει πολυώνυμο: $P(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$
CRS: $\mathsf{gp} = (g_0, g_1, g_2, g_3)$ με $g_0, g_1, g_2, g_3 \leftarrow\$ \mathbb{G}$
$\mathsf{Commit}(\mathsf{gp}, P) = C_P = g_0^{a_0} g_1^{a_1} g_2^{a_2} g_3^{a_3}$

Ο V αποστέλει u
Ο P υπολογίζει το $v = P(u) = a_0 + a_1 u + a_2 u^2 + a_3 u^3$.

Η επαλήθευση θα γίνει μέσω ενός πολυωνύμου με 2 όρους $P'(x) = a'_0 + a'_1 x$:

1. Ο P σπάει v σε $v_L = a_0 + a_1 u, v_R = a_2 + a_3 u$.
2. Ο V ελέγχει αν: $v = v_L + v_R u^2$.
3. Ο V αποστέλλει $r \leftarrow\$ \mathbb{Z}_q$ (ώστε να συμμετάσχει στον υπολογισμό του P' και να μην μπορεί να κλέψει ο P)

4. Ο P υπολογίζει

$$a'_0 = a_0 r + a_2$$
$$a'_1 = a_1 r + a_3$$
$$P'(x) = (a_0 r + a_2) + (a_1 r + a_3)x$$

5. $P'(u) = v' = (a_0 r + a_2) + (a_1 r + a_3)u$ η νέα τιμή που πρέπει να ελέγξει ο V. Μπορεί να προκύψει από την παλιά ως: $v' = r v_L + v_R$ (με v_L, v_R να έχουν αποσταλεί νωρίτερα).
6. Η νέα δέσμευση πρέπει θα έχει μορφή $(g'_0)^{a'_0} \cdot (g'_1)^{a'_1}$ με $(g'_0)^{a'_0} = (g'_0)^{a_0 r + a_2}$ και $(g'_1)^{a'_1} = (g'_1)^{a_1 r + a_3}$ Όμως ο V δεν μπορεί να την υπολογίσει χωρίς τα a_0, a_1, a_2, a_3.
7. Πρέπει να βοηθήσει ο P.
Πριν τη λήψη του r στέλνει $L = g_2^{a_0} g_3^{a_1}$ και $R = g_0^{a_2} g_1^{a_3}$ (μαζί με $v_L.v_R$)

Bulletproofs reduction μέσω απλουστευμένου παραδείγματος iii

8. Ο V υπολογίζει $C'_P = L^r C_P R^{r^{-1}}$ και $\mathsf{gp}' = (g_0^{r^{-1}} g_2, g_1^{r^{-1}} g_3)$

επειδή:

$$\begin{aligned} L^r C_P R^{r^{-1}} &= g_2^{ra_0} g_3^{ra_1} \cdot g_0^{a_0} g_1^{a_1} g_2^{a_2} g_3^{a_3} \cdot g_0^{a_2 r^{-1}} g_1^{a_3 r^{-1}} \\ &= g_0^{a_0 + a_2 r^{-1}} g_1^{a_1 + a_3 r^{-1}} g_2^{ra_0 + a2} g_3^{ra_1 + a_3} \\ &= (g_0^{r^{-1}} g_2)^{ra_0 + a2} \cdot (g_1^{r^{-1}} g_3)^{ra_1 + a_3} \end{aligned}$$

Αναγωγή από πολυώνυμο 4 όρων σε πολυώνυμο 2 όρων.

Πολυώνυμο 2 όρων

P: Γνωρίζει πολυώνυμο: $P'(x) = a'_0 + a'_1 x$

CRS: gp $= (g'_0, g'_1)$

Commit(gp, P') $= C'_P = {g'_0}^{a'_0} {g'_1}^{a'_1}$

Ο P υπολογίζει και στέλνει

$L' = (g'_1)^{a'_0} = (g_1^{r^{-1}} g_3)^{a_0 r + a_2}$,

$R' = (g'_0)^{a'_1} = (g_0^{r^{-1}} g_2)^{a_1 r + a_3}$,

$v'_L = a'_0$ και $v'_R = a'_1$

Ο V ελέγχει αν $v' = v'_L + v'_R u$ και στέλνει νέο challenge $e \leftarrow\$ \mathbb{Z}_q$

Ο P υπολογίζει σταθερό πολυώνυμο

$P''(x) = a''_0 = a'_0 e + a'_1 = a_0 r e + a_2 e + a_1 r + a_3$

Ο P στέλνει το a''_0 (όχι ZK) στον V ο οποίος υπολογίζει:

$$\mathsf{gp}'' = g_0'' = (g_0^{r^{-1}} g_2)^{e^{-1}} g_1^{r^{-1}} g_3$$
$$= g_0^{r^{-1}e^{-1}} g_2^{e^{-1}} g_1^{r^{-1}} g_3$$

και ελέγχει αν:

$$P''(u) = v'' = a_0'' \text{ και } v'' = ev_L' + v_R = ea_0' + a_1'$$

που ισχύει και

$$L'^e C_P' R'^{e^{-1}} = (g_0'')^{a_0''}$$

Πράγματι μετά από πολλές ομαδοποιήσεις:

Bulletproofs reduction μέσω απλουστευμένου παραδείγματος vi

$$L'^{e} C'_P R'^{e^{-1}} =$$

$$(g_1^{r^{-1}} g_3)^{ea_0 r + ea_2} \cdot (g_0^{r^{-1}} g_2)^{ra_0 + a2} \cdot (g_1^{r^{-1}} g_3)^{ra_1 + a_3} (g_0^{r^{-1}} g_2)^{e^{-1} a_1 r + e^{-1} a_3} =$$

$$g_0^{a_0 + r^{-1} a_2 + e^{-1} a_1 + e^{-1} r^{-1} a_3} g_1^{ea_0 + r^{-1} ea_2 + a_1 + r^{-1} a_3} \cdot$$

$$g_2^{ra_0 + a_2 + e^{-1} ra_1 + e^{-1} a_3} g_3^{era_0 + ea_2 + ra_1 + a_3} =$$

$$((g_0^{r^{-1}} g_2)^{e^{-1}} g_1^{r^{-1}} g_3)^{a_0 re + a_2 e + a_1 r + a_3} =$$

$$(g_0'')^{a_0''}$$

Ο P θέλει να σπάσει τα a και g σε $\mathsf{a}_L, \mathsf{a}_R$ και $\mathsf{g}_L, \mathsf{g}_R$ έτσι ώστε:

$$C_P = \langle\langle \mathsf{a}, \mathsf{g} \rangle\rangle = \langle\langle \mathsf{a}_L, \mathsf{g}_L \rangle\rangle \cdot \langle\langle \mathsf{a}_R, \mathsf{g}_R \rangle\rangle$$

Το σπάσιμο πρέπει να είναι επαληθεύσιμο χωρίς να είναι γνωστό το a. Έτσι ο V διαλέγει r και ο P συνδυάζει τα δύο μισά σε νέο μισό:

$$\mathsf{a}' = r\mathsf{a}_L + r^{-1}\mathsf{a}_R$$
$$\mathsf{g}' = \mathsf{g}_L^{r^{-1}} \cdot \mathsf{g}_R^{r}$$

Νέα τιμή του C_P:

$$\begin{aligned} C'_P = \langle\langle \mathsf{a}', \mathsf{g}' \rangle\rangle &= \langle\langle r\mathsf{a}_L + r^{-1}\mathsf{a}_R, \mathsf{g}_L^{r-1} \cdot \mathsf{g}_R^r \rangle\rangle \\ &= \langle\langle \mathsf{a}_L, \mathsf{g}_L \rangle\rangle \cdot \langle\langle \mathsf{a}_R, \mathsf{g}_R \rangle\rangle \cdot \langle\langle r\mathsf{a}_L, \mathsf{g}_R^r \rangle\rangle \cdot \langle\langle r^{-1}\mathsf{a}_L, \mathsf{g}_R^{r^{-1}} \rangle\rangle \\ &= \langle\langle \mathsf{a}_L, \mathsf{g}_L \rangle\rangle \cdot \langle\langle \mathsf{a}_R, \mathsf{g}_R \rangle\rangle \cdot \langle\langle \mathsf{a}_L, \mathsf{g}_R \rangle\rangle^{r^2} \cdot \langle\langle \mathsf{a}_L, \mathsf{g}_R \rangle\rangle^{r^{-2}} \end{aligned}$$

Από τα παραπάνω:

- $\langle\langle \mathsf{a}_L, \mathsf{g}_L \rangle\rangle \cdot \langle\langle \mathsf{a}_R, \mathsf{g}_R \rangle\rangle$ είναι γνωστό
- $\langle\langle \mathsf{a}_L, \mathsf{g}_R \rangle\rangle$ και $\langle\langle \mathsf{a}_L, \mathsf{g}_R \rangle\rangle$ πρέπει να δοθούν πριν σταλεί το r

Έτσι υπολογίζεται το νέο commitment και η νέα τιμή που αντιστοιχούν σε vectors μισού μεγέθους

Στόχος

P γνωρίζει vector $\mathbf{a} = (a_0, \cdots, a_d)$ $d+1$ στοιχείων τέτοιο ώστε $C = \text{Commit}(\mathbf{a}) = \prod_{i=0}^{d} g_i^{a_i}$ και $v = \prod_{i=0}^{d} (u^i)^{a_i}$ για $u \in \mathbb{F}_p$.

- Δημόσια Είσοδος: $\text{gp} = (g_0, g_1, \cdots g_d) = \mathbf{g}$ και $i \in \{1, \cdots, \log d\}, u, v, \mathbf{u} = (1, u, u^2, \cdots, u^d)$
- Ιδιωτική είσοδος P: $P(x) = a_0 + a_1 x + \cdots + a_d x^d = \mathbf{a}$
- Είσοδος γύρου i : $c_i = \text{Commit}_i(\mathbf{a}_i) = \prod_{k=1}^{d_i} g_k^{a_k}$
- Αν $i = 1$ ο P στέλνει: $\mathbf{a}_1 = a_0$ και ο verifier ελέγχει αν $g_0^{a_0} = c_1$ και $u^{a_0} = v_1$
- Αλλιώς:
    - ο P σπάει το $\mathbf{a}_i$ σε $(\mathbf{a}_{iL}, \mathbf{a}_{iR})$ το $\mathbf{g}_i$ σε $(\mathbf{g}_{iL}, \mathbf{g}_{iR})$ και το $\mathbf{u}_i$ σε $(\mathbf{u}_{iL}), \mathbf{u}_{iR})$.
    - ο P υπολογίζει:
        - $v_L = \langle\langle \mathbf{a}_{iL}, \mathbf{g}_{iR} \rangle\rangle$

- $v_R = \langle\langle \mathsf{a}_{iR}, \mathsf{g}_{iL} \rangle\rangle$
- $v'_L = \langle \mathsf{a}_{iL}, \mathsf{u}_{iR}, \rangle$
- $v'_R = \langle \mathsf{a}_{iR}, \mathsf{u}_{iL} \rangle$

- ο P στέλνει v_L, v_R, v'_L, v'_R
- ο V στέλνει $r_i \leftarrow\!\$ \, \mathbb{Z}_q$
- Ο P υπολογίζει νέο vector συντελεστών μισού μήκους

$$\mathsf{a}_{i-1} := {\mathsf{a}_{iL}}^{r} \cdot {\mathsf{a}_{iR}}^{r^{-1}}$$

- Και P και V υπολογίζουν
  - $\mathsf{g}_{i-1} := {\mathsf{g}_{iR}}^{r} \cdot {\mathsf{g}_{iL}}^{r^{-1}}$ νέο vector gp
  - $\mathsf{u}_{i-1} := {\mathsf{u}_{iR}}^{r} \cdot {\mathsf{u}_{iL}}^{r^{-1}}$ νέα τιμή vector αποτίμησης
  - $c_{i-1} = c_i \cdot v_L^{r^2} v_R^{r^{-2}}$ ενημέρωση commitment
  - $v_{i-1} = v_i \cdot v_L^{r^2} v_R^{r^{-2}}$ ενημέρωση τιμής

Knowledge Soundness

Αν $v_L \neq \langle\langle \mathsf{a}_L, \mathsf{g}_R \rangle\rangle \vee v_R \neq \langle\langle \mathsf{a}_R \mathsf{g}_L \rangle\rangle$ τότε $v' \neq \langle\langle \mathsf{a}', \mathsf{g}' \rangle\rangle$ και κατά συνέπεια $C'_P \neq \langle\langle \mathsf{a}', \mathsf{g}' \rangle\rangle$.

Θέλουμε:

$$C'^{r^2}_P = (C_P \cdot v_L^{r^2} \cdot v_R^{r^{-2}})^{r^2} = C_P^{r^2} \cdot v_L^{r^4} \cdot v_R$$

και

$$C'^{r^2}_P = \langle\langle \mathsf{a}', \mathsf{g}' \rangle\rangle^{r^2} = C_P^{r^2} \cdot \langle\langle \mathsf{a}_L, \mathsf{g}_R \rangle\rangle^{r^4} \cdot \langle\langle \mathsf{a}_L, \mathsf{g}_R \rangle\rangle$$

Οι τιμές αυτές θα είναι ίσες με πιθανότητα $1 - \frac{4}{p}$ στην επιλογή του r

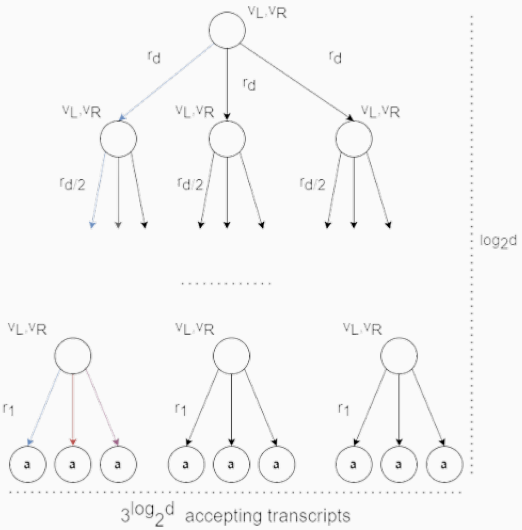
Ανάλυση ασφάλειας ii

Κατασκευή extractor $\mathcal{E}$ που για accepting transcripts μπορεί να εξάγει τον witness (συντελεστές **a**)

Γενίκευση forking lemma για πολλαπλούς γύρους

Κατασκευή transcript tree depth-first:

- Τριαδικό δέντρο με βάθος $log_2 d$
- Ακμές: verifier challenges
- Εσωτερικοί κόμβοι: απαντήσεις prover
- Φύλλα: accepting transcripts
- Πολυπλοκότητα κατασκευής: $\mathcal{O}\left(d^{log_2 3}|\mathsf{V}|\right)$
- Ανακατασκευή σταδιακά **a** ξεκινώντας από τα φύλλα



vL, vR
rd
rd
rd
vL, vR
vL, vR
vL, vR
rd/2
rd/2
rd/2
log2d
vL, vR
vL, vR
vL, vR
r1
r1
r1
a
a
a
a
a
a
a
a
a
3log2d accepting transcripts

Ανάλυση ασφάλειας iv

Για μηδενική γνώση

Όλες οι τιμές που στέλνει ο prover είναι Pedersen commitments

Εκμεταλλευόμαστε τον προσθετικό ομορφισμό

Non interactive version

Fiat Shamir heuristic σε όλες τις προηγούμενες τιμές

Βελτιώσεις:

- Hydra [WTS+18] $\mathcal{O}\left(\sqrt{d}\right)$ proof and verifier size
- Dory [Lee21] $\mathcal{O}(\log d)$ proof and verifier size

Sum-Check IOP

The Sum-Check protocol [LFKN92] i

Δίνεται πολυώνυμο $g : \mathbb{F}^n \rightarrow \mathbb{F}$ με βαθμό d.
Θέλουμε ο V να μάθει την τιμή:

$$G := \sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} g(b_1, b_2, \cdots b_n)$$

- Μπορεί να γίνει με 2^n αποτιμήσεις του g: μεγάλο κόστος
- Εναλλακτική: Θα το υπολογίσει ο P και θα αποδείξει στον V ότι είναι σωστό.
- Ο V μπορεί να κάνει χρήση ενός oracle query
- Πολυπλοκότητα:
  - Prover $\mathcal{O}(d2^n)$
  - Verifier $\mathcal{O}(dn)$
  - Επικοινωνία $\mathcal{O}(n)$

Το πρωτόκολλο

1. P: Στέλνει την τιμή
   $G = \sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} g(b_1, b_2, \cdots b_n)$
2. Γύρος 1:
   - P: Δημιουργεί $G_1(X_1) = \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} g(X_1, b_2, \cdots b_n)$
     Δηλαδή αποτιμά για $x_2 \in \{0,1\}, \cdots, x_n \in \{0,1\}$ και αφήνει X_1 ελεύθερη μεταβλητή. Στέλνει πολυώνυμο $W_1(X_1), X_1 \in \mathbb{F}$ για το οποίο ισχυρίζεται ότι $W_1 = G_1$
   - V:
     - Ελέγχει ότι $W_1(0) + W_1(1) = G$
     - Αρα ο prover έχει δίκιο, αρκεί $W_1 = G_1$
     - Ο V επιλέγει $r_1 \in \mathbb{F}$ για να ελέγξει ότι $W_1(r) = G_1(r)$
     - Αλλά δεν γνωρίζει G_1
     - Στέλνει το r_1 στον P
     - Επαλήθευση μέσω πολυωνύμου με μία μεταβλητή λιγότερη $g(r_1, x_2, \cdots, x_n)$

3. Γύρος i:
    - P: Στέλνει πολυώνυμο $W_i(X_i), X_i \in \mathbb{F}$ για το οποίο ισχυρίζεται ότι $W_i = G_i$ δηλ.
      $G_i(X_i) = \sum_{b_{i+1} \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} g(r_1, \cdots, X_i, b_{i+1}, \cdots b_{i+1})$
    - V:
        - Ελέγχει ότι $W_i(0) + W_i(1) = W_{i-1}(r_{i-1})$
        - Ο V επιλέγει $r_i \in \mathbb{F}$ και το στέλνει στον P
4. Γύρος n:
    - $G_n(X_n) = g(r_1, r_2, \cdots, X_n)$
    - Ο P στέλνει $W_n(X_n)$
    - Ο V ελέγχει ότι $W_n(0) + W_n(1) = W_{n-1}(r_{n-1})$
    - Πώς θα ελέγξει ότι W_n είναι σωστό;
    - Επιλέγει $r_n \xleftarrow{\$} \mathbb{F}$ και χρησιμοποιεί το oracle query για να ελέγξει αν $g(r_1, r_2, \cdots r_n) = W_n(r_n)$

Παράδειγμα: $g(X_1, X_2, X_3) = 2X_1^3 + X_1X_3 + X_2X_3$
Ο P ισχυρίζεται ότι: $\sum_{\mathsf{d}\in\{0,1\}^3} g(\mathsf{d}) = 12$

- Ο P υπολογίζει $W_1(X_1) = g(X_1, 0, 0) + g(X_1, 0, 1) + g(X_1, 1, 0) + g(X_1, 1, 1) = 2X_1^3 + (2X_1^3 + X_1) + 2X_1^3 + (2X_1^3 + X_1 + 1) = 8X_1^3 + 2X_1 + 1$
- Ο V επαληθεύει: $W_1(0) + W_1(1) = G = 12$
- Ο V επιλέγει $r_1 = 2$
- Ο P υπολογίζει $W_2(X_2) = g(2, X_2, 0) + g(2, X_2, 1) = X_2 + 34$
- Ο V επαληθεύει: $W_1(2) = W_2(0) + W_2(1) = 69$
- Ο V επιλέγει $r_2 = 3$
- Ο P υπολογίζει $W_3(X_3) = g(2, 3, X_3) = 16 + 5X_3$
- Ο V επαληθεύει: $W_2(3) = W_3(0) + W_3(1) = 37$
- Ο V επιλέγει $r_3 = 6$
- Ο V επαληθεύει $W_3(6) = g(2, 3, 6) = 46$ μέσω oracle query

Έλεγχος βαθμού:

Σε κάθε γύρο i ο V ελέγχει και αν το πολυώνυμο W_i έχει το σωστό βαθμό d_i για την μεταβλητή X_i. Αυτό εξαρτάται από την αναπαράστη του W_i (λήψη $d_i + 1$ συντελεστών ή $d_i + 1$ σημείων).

Sum-Check soundness

Αρκεί σε έναν γύρο ο κακός P* να πείσει τον V ότι $W_i(r_i) = G_i(r_i)$ (μετά το πρωτόκολλο θα σταματήσει)

Πιθανότητα $W_i(r_i) = G_i(r_i)$ όταν $W_i \neq G_i$: $\leq \frac{d}{|\mathbb{F}|}$

Πιθανότητα αυτό να συμβεί σε έναν από τους n γύρους $\leq n\frac{d}{|\mathbb{F}|}$

Άρα πιθανότητα ο V να κάνει reject P: $\geq 1 - n\frac{d}{|\mathbb{F}|}$

Το πρωτόκολλο GKR [GKR15]

Αποτίμηση Αριθμητικού Κυκλώματος

Οι P και V συμφωνούν σε ένα αριθμητικό κύκλωμα C με τιμές σε $\mathbb{F}$, fan-in 2 και λογαριθμικό χώρο (log-space).

- d: μέγιστο βάθος του κυκλώματος σε επίπεδα: 0 (output) μέχρι d (input)
- S: πλήθος πυλών
- S_i: πλήθος πυλών στο επίπεδο i. Υποθέτουμε $S_i = 2^{k_i}$
- n: πλήθος μεταβλητών

Στόχος: Να υπολογιστεί η έξοδος του C με succinct τρόπο.

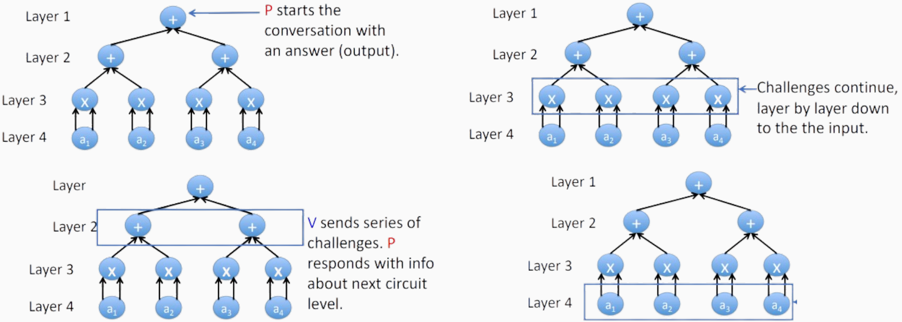
- Πολυπλοκότητα P: $\mathcal{O}(S)$
- Πολυπλοκότητα V: $\mathcal{O}(dlogS)$

Βασική ιδέα GKR i

- Αρχικά ο P ανακοινώνει το $C(\mathsf{x})$ (επίπεδο 0)
- Αναδρομικά ο ισχυρισμός για την έξοδο στο επίπεδο i μετατρέπεται σε ισχυρισμό για την έξοδο στο επίπεδο $i+1$
- **Πώς**: Χρησιμοποιώντας το πρωτόκολλο Sum-Check
- Κατάληξη: Ισχυρισμός για το επίπεδο 0 (input x) το οποίο είναι γνωστό στον V

Βασική ιδέα GKR ii

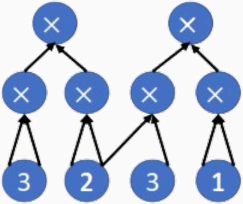
Σχηματικά [Tha22]



Αναπαράσταση κυκλώματος με συναρτήσεις i

Transcript (Execution Trace) Κυκλώματος

- Αριθμούμε τις πύλες στο επίπεδο i με έναν αριθμό (στο δυαδικό) $\{0, \cdots S_i = 2^{k_i} - 1\}]$
- Ορίζουμε $W_i : \{0,1\}^{k_i} \to \mathbb{F}$ η συνάρτηση που παίρνει ως είσοδο τον αριθμό πύλης (στο δυαδικό) και έχει έξοδο την τιμή της.



$W_0(0) = 36, \quad W_0(1) = 6$

$W_1(0{,}0) = 9, \ W_1(0{,}1)=4, W_1(1{,}0) = 6, W_1(1{,}1) = 1$

$W_2(0{,}0) = 3, \ W_2(0{,}1)=2, W_2(1{,}0) = 3, W_2(1{,}1) = 1$

Wiring predicate

- Ορίζουμε $\text{in}_{1i}, \text{in}_{2i} : \{0,1\}^{k_i} \to \{0,1\}^{k_{i+1}}$ ποιες γραμμές συνδέονται σε μια πύλη από το προηγούμενο επίπεδο πχ. $\text{in}_{11}(10) = 01$ και $\text{in}_{21}(10) = 10$
- $\text{add}_i(a,b,c) = 1 \Leftrightarrow \text{in}_{1i} = b \wedge \text{in}_{2i} = c \wedge$ a είναι addition gate
- $\text{mult}_i(a,b,c) = 1 \Leftrightarrow \text{in}_{1i} = b \wedge \text{in}_{2i} = c \wedge$ a is είναι multiplication gate
- πχ. $\text{mult}_0(0,00,01) = 1$ και $\text{add}_0(0,10,11) = 0$
- πχ. $\text{mult}_1(01,00,01) = 0$ και $\text{mult}_i(01,01,01) = 1$
- $D : \{0,1\}^{k_0} \to \mathbb{F}$ η έξοδος του κυκλώματος στην πύλη που δίνεται στο input πχ. $D(0) = 36$

Στην πραγματικότητα αναφερόμαστε στα multilinear extensions των add, mult, D (με ορίσματα από $\mathbb{F}^{k_i}$ αντί για $\{0,1\}^{k_i}$

Το πρωτόκολλο i

1. Ο P δημοσιοποιεί τη συνάρτηση $D : \{0,1\}^{k_0} \to \mathbb{F}$ μέσω των τιμών της.
2. Ο V επιλέγει $r_0 \in \mathbb{F}^{k_0}$ και υπολογίζει $D(r_0)$
3. Αν $D_(r_0) = W_0(r_0)$ τότε το κύκλωμα είναι σωστό με μεγάλη πιθανότητα)
4. Ο V δεν μπορεί να υπολογίσει το W_0 μόνος του για να επαληθεύσει
5. Ο P μετατρέπει τον ισχυρισμό για το W_0 σε ισχυρισμό για το W_1 και ...

6. Επαναληπτικά για $i = 0, 1, \cdots d-1$, ο P μετατρέπει τον ισχυρισμό για το W_i σε ισχυρισμό για το W_{i+1} ως εξής:

$$W_i(r_i) = f_{r_i}^{(i)}(b, c) =$$
$$\sum_{b,c \in \{0,1\}^{k_i+1}} \text{add}_i(r_i, b, c) \cdot (W_{i+1}(b) + W_{i+1}(c)) +$$
$$\text{mult}_i(r_i, b, c) \cdot (W_{i+1}(b) \cdot W_{i+1}(c))$$

   το οποίο μπορεί να υπολογιστεί χρησιμοποιώντας το Sum-Check.

7. Στο τέλος του πρωτοκόλλου ο V πρέπει να αποτιμήσει το $f_{r_i}^{(i)}$ σε ένα σημείο (b^*, c^*). Μπορεί να γίνει χρησιμοποιώντας ένα oracle query σε polynomial commitment πχ.
8. Ο V ελέγχει απευθείας το τελευταίο επίπεδο $W_d(r_d)$ (κατασκευάζοντας το W_d από την είσοδο και αποτιμώντας το).

PLONK IOP

Απόδειξη Σύνθετων Ιδιοτήτων Πολυωνύμων

- Ο P κατέχει κάποια πολυώνυμα P, Q
- Θέλει να αποδείξει ότι έχουν κάποιες ιδιότητες
- Στέλνει commitments C_P, C_Q στον V
- Ο V επιλέγει $r \leftarrow\$ \mathbb{Z}_q$
- Ο P απαντά με βάση τα P, Q και παρέχει απόδειξη ως προς τα C_P, C_Q (ProveEval)
- Ο V αποδέχεται ή όχι (VerifyEval)

Πρέπει να γίνει διατηρώντας γραμμικό P και υπο-γραμμικό V!

Zero - Test i

Zero - Test

Δίνεται $\Omega \subseteq \mathbb{F}_q$. Ο P γνωρίζει P και θέλει να αποδείξει ότι
$P(\omega) = 0 \quad \forall \omega \in \Omega$

Λήμμα

Το P είναι μηδενικό στο Ω ανν διαιρείται με το πολυώνυμο
$Z_\Omega(x) = \prod_{\omega \in \Omega}(x - \omega)$ (vanishing πολυώνυμο)

1. Ο P υπολογίζει $C_P = \text{Commit}(\text{gp}, P)$
2. Ο P υπολογίζει $Q = P/Z_\Omega$
3. Ο P δεσμεύεται στο Q: (υπολογίζει $C_Q = \text{Commit}(\text{pk}, Q)$)
4. Ο V επιλέγει $u \leftarrow\$ \mathbb{Z}_q$
5. Ο P υπολογίζει
   $\pi_P = \text{ProveEval}(\text{gp}, P, u, P(u)), \quad \pi_Q = \text{ProveEval}(\text{gp}, Q, u, Q(u))$

Zero - Test ii

6. Ο V επαληθεύει:
   - VerifyEval(gp, $C_P, \pi_P, u, P(u)$)
   - VerifyEval(gp, $C_Q, \pi_Q, u, Q(u)$)
   - και $P(u) = Q(u)Z_\Omega(u)$ (ισοδύναμη σχέση με τα π_Q, π_P)

Παρατηρήσεις:

- Ο V υπολογίζει μόνος του το Z_Ω στο $u \in \mathbb{F}_q$
- Για να γίνεται αποδοτικά $Z_\Omega(x) = x^k - 1$ και $\Omega = \{1, \omega, \cdots, \omega^{k-1}\}$ με ω: k root of unity (δηλ. $\omega^k = 1$)
- V complexity για αποτιμήση r^k: $\mathcal{O}(\log k)$
- P complexity $\mathcal{O}(d)$

Product - Test

Δίνεται $\Omega = \{1, \omega, \cdots, \omega^{k-1}\} \subseteq \mathbb{F}_q$. O P γνωρίζει P και θέλει να αποδείξει ότι $\prod_{\omega \in \Omega} P(\omega) = 1$

- O P υπολογίζει (με παρεμβολή) πολυώνυμο T τέτοιο ώστε:
  - $T(1) = P(1)$
  - $T(\omega^s) = \prod_{i=0}^{s} P(\omega^i) \qquad = (\prod_{i=0}^{s-1} P(\omega^i)) \cdot P(\omega^s) = T(\omega^{s-1}) P(\omega^s)$
- $T(\omega^{k-1}) = \prod_{i=0}^{k-1} P(\omega^i) = \prod_{\omega \in \Omega} P(\omega) = 1$ (αν το P είναι σωστό)
- $\forall x \in \Omega : \quad T(\omega \cdot x) = T(x) \cdot P(\omega \cdot x)$

Λήμμα

Αν $T(\omega^{k-1}) = 1$ και $T(\omega \cdot x) = T(x) \cdot P(\omega \cdot x)$ τότε: $\prod_{\omega \in \Omega} P(\omega) = 1$

Product - Test ii

1. Ο P υπολογίζει $C_P = \text{Commit}(\text{gp}, P)$
2. Ο P κατασκευάζει το T (βαθμού $k-1$)
3. Ο P υπολογίζει πολυώνυμο $R(x) = T(\omega \cdot x) - T(x) \cdot P(\omega \cdot x)$ και δεσμεύεται σε αυτό $C_R = \text{Commit}(\text{gp}, R)$
4. Ιδέα: Χρήση zero test στο πολυώνυμο R
5. Ο V πρέπει να επιπλέον ότι: $T(\omega^{k-1}) = 1$ και $T(\omega \cdot u) = T(u) \cdot P(\omega \cdot u)$ για $u \xleftarrow{\$} \mathbb{Z}_q$
6. Αυτό θα γίνει μέσω των $\text{ProveEval}(\text{gp}, T, \omega^{k-1}, T(\omega^{k-1}))$ $\text{ProveEval}(\text{gp}, T, u, T(u))$, $\text{ProveEval}(\text{gp}, T, \omega u, T(\omega u))$ και $\text{ProveEval}(\text{gp}, P, \omega u, P(\omega u))$

Παρατήρηση: Με το ίδιο πρωτόκολλο μπορεί να αποδειχθεί ότι $\prod_{\omega \in \Omega} \frac{P}{Q}(\omega) = 1$

Permutation - Test

Permutation - Test

Δίνεται $\Omega = \{1, \omega, \cdots, \omega^{k-1}\} \subseteq \mathbb{F}_q$ και πολυώνυμα P, Q γνωστά στον P.

Ο P θέλει να αποδείξει ότι υπάρχει μετάθεση W ώστε το σύνολο $P(\Omega) = \{P(x)\}_{x \in \Omega} = W(Q(\Omega))$

Βασική παρατήρηση

- Ορίζουμε $P'(x) = \prod_{x \in \Omega}(x - P(x))$
- και $Q'(x) = \prod_{x \in \Omega}(x - Q(x))$

Τότε: $P(\Omega) = W(Q(\Omega)) \Leftrightarrow P'(x) = Q'(x)$

Χρήση παραλλαγής Product Check για $\frac{P'}{Q'}$ χωρίς να χρειαστεί να κατασκευαστούν τα P', Q'.

Prescribed Permutation Check

Permutation - Test

Δίνεται $\Omega = \{1, \omega, \cdots, \omega^{k-1}\} \subseteq \mathbb{F}_q$ και πολυώνυμα P, Q γνωστά στον P.

Ο P θέλει να αποδείξει υπάρχει **γνωστή** μετάθεση W ώστε το σύνολο $P(\omega) = \{P(x)\}_{x \in \Omega} = W(Q(\Omega))$

Ο P γνωρίζει P, Q, W και ο V C_P, C_Q, C_W.

Δεν αρκεί να γίνει zero - test στο $R(x) = P(x) - Q(W(x))$ γιατί το πολυώνυμο $Q.W$ χρειάζεται τετραγωνικό χρόνο για να υπολογιστεί.

Παρατήρηση

Αν $(W(x), P(x))_{x \in \Omega}$ είναι μετάθεση του $(x, Q(x))_{x \in \Omega}$ τότε $\forall x \in \Omega : P(x) = Q(W(x))$.

Αρκεί να κάνουμε Product - Check στα πολυώνυμα *δύο* μεταβλητών:

$$P(x, y) = \prod_{a \in \Omega} (x - yW(a) - P(a))$$

$$Q(x, y) = \prod_{a \in \Omega} (x - ya - Q(a))$$

PLONK IOP [GWC19]

- Polynomial IOP γενικής χρήσης.
- Συνδυάζεται με διαφορετικό polynomial commitments για να δώσει διαφορετικά SNARKs
- Βήματα
  - Arithmetization - Μετατροπή κυκλώματος fan-in 2 σε computation transcript / trace και μετά σε πολυώνυμο
  - Εκτέλεση IOP για:
    - Σωστή κωδικοποίηση εισόδου
    - Σωστή αποτίμηση πυλών
    - Σωστή σύνδεση πυλών (σύνδεση εξόδων - εισόδων)
    - Σωστό αποτέλεσμα

Soundness: Αν $\frac{7|C|}{q} \leq \mathrm{negl}(\lambda)$

PLONK Arithmetization

Θέτουμε $d = 3|C| + |I|$ και $\Omega = \left\{1, \omega, \omega^2, \cdots, \omega^{d-1}\right\}$

Το πολυώνυμο T για το κύκλωμα θα σχηματιστεί με Lagrange Interpolation από τις παρακάτω τιμές

- $T(\omega^{-j}) = I_j \quad \forall j \in [|I|]$ οι τιμές που αντιστοιχούν στην είσοδο
- Για κάθε πύλη l:
  - $T(\omega^{3l})$ η τιμή για την αριστερή είσοδο
  - $T(\omega^{3l+1})$ η τιμή για την δεξιά είσοδο
  - $T(\omega^{3l+2})$ η τιμή για την έξοδο

P-IOP για σωστή κωδικοποίηση εισόδου

- Ο V γνωρίζει το σύνολο I.
- Ο V θα κατασκευάσει πολυώνυμο V: $V(\omega^{-j}) = I_j \quad \forall j \in [|I|]$
- Ο P θα κατασκευάσει αποδείξη ότι $T(x) = V(x) \quad \forall x \in \Omega_I$

Μπορεί να γίνει εύκολα με το Zero - Test.

Πολυπλοκότητα: Γραμμική στην είσοδο I

P-IOP για σωστή αποτίμηση πύλης

- Ο P θα ορίσει selector-πολυώνυμο S ώστε:
  - $S(\omega^{3l}) = 0$ αν η πύλη l είναι πύλη πρόσθεσης
  - $S(\omega^{3l}) = 1$ αν η πύλη l είναι πύλη πολλαπλασιασμού
- Το S δεν εξαρτάται από το input οπότε μπορεί να είναι τμήμα του preprocessing
- Για κάθε $x \in \left\{1, \omega^3, \omega^6, \cdots, \omega^{(3|C|-1)}\right\}$

$$S(x)(T(x) + T(\omega x)) + (1-S(x))(T(x) \cdot T(\omega x)) - T(\omega^2 x) = 0$$

Μπορεί να πάλι να γίνει εύκολα με το Zero - Test

P-IOP για σύνδεση πυλών

Κάθε πύλη κάνει σωστά τη δουλειά της

Για να είναι σωστό το κύκλωμα πρέπει να συνδεθούν και σωστά.

Κάποιες τιμές του T πρέπει να είναι ίσες μεταξύ τους (copy constraints)

Μπορεί να εκφραστεί με rotation του Ω

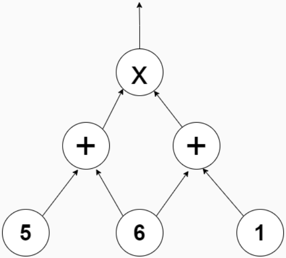
Rotation Lemma

Δίνεται πολυώνυμο $W : \Omega \to \Omega$ το οποίο υλοποιεί ένα rotation του Ω
Αν $\forall x \in \Omega : T(x) = T(W(x))$ τότε οι πύλες έχουν συνδεθεί σωστά.

Μπορεί να γίνει με το Prescribed Permutation Check

Παράδειγμα

$$C(x,w) = (x_1+x_2)(x_2+w)$$
$$\Omega = \{1, \omega, \cdots, \omega^{11}\}$$



Είσοδος:	5	6	1	$T(\omega^{-1})$	$T(\omega^{-2})$	$T(\omega^{-3})$
Πύλη 0: $S(\omega^0) = 1$	5	6	11	$T(\omega^0)$	$T(\omega^1)$	$T(\omega^2)$
Πύλη 1: $S(\omega^3) = 1$	6	1	7	$T(\omega^3)$	$T(\omega^4)$	$T(\omega^5)$
Πύλη 2: $S(\omega^6) = 0$	11	7	77	$T(\omega^6)$	$T(\omega^7)$	$T(\omega^8)$

Σύνδεση Πυλών και Permutation

$$T(\omega^{-2}) = T(\omega^1) = T(\omega^3)$$
$$W(\omega^{-2}, \omega^1, \omega^3) = (\omega^3, \omega^{-2}, \omega^1)$$
$$T(\omega^{-3}) = T(\omega^4)$$
$$W(\omega^{-3}, \omega^4) = (\omega^4, \omega^{-3})$$
$$T(\omega^{-1}) = T(\omega^0)$$
$$W(\omega^{-1}, \omega^0) = (\omega^0, \omega^1)$$
$$T(\omega^2) = T(\omega^6)$$
$$W(\omega^2, \omega^6) = (\omega^6, \omega^2)$$
$$T(\omega^5) = T(\omega^7)$$
$$W(\omega^5, \omega^7) = (\omega^5, \omega^7)$$

Βιβλιογραφία

Βιβλιογραφία i

- Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell, *Bulletproofs: Short proofs for confidential transactions and more*, Cryptology ePrint Archive, Paper 2017/1066, 2017, https://eprint.iacr.org/2017/1066.
- Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit, *Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting*, EUROCRYPT 2016, 2016.
- Dan Boneh, Ben Lynn, and Hovav Shacham, *Short signatures from the weil pairing*, ASIACRYPT 2001 (Colin Boyd, ed.), 2001.
- Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum, *Delegating computation: Interactive proofs for muggles*.

Βιβλιογραφία ii

- Jens Groth, *On the size of pairing-based non-interactive arguments*, Cryptology ePrint Archive, Paper 2016/260, 2016, https://eprint.iacr.org/2016/260.
- Craig Gentry and Daniel Wichs, *Separating succinct non-interactive arguments from all falsifiable assumptions*, Cryptology ePrint Archive, Paper 2010/610, 2010, https://eprint.iacr.org/2010/610.
- Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru, *Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge*, Cryptology ePrint Archive, Paper 2019/953, 2019, https://eprint.iacr.org/2019/953.
- John Kuszmaul, *Verkle trees*.

- Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg, *Constant-size commitments to polynomials and their applications*, ASIACRYPT 2010, 2010.
- Jonathan Lee, *Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments*, Theory of Cryptography (Cham), 2021.
- Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan, *Algebraic methods for interactive proof systems*.
- Valeria Nikolaenko, Sam Ragsdale, Joseph Bonneau, and Dan Boneh, *Powers-of-tau to the people: Decentralizing setup ceremonies*, Cryptology ePrint Archive, Paper 2022/1592, 2022, https://eprint.iacr.org/2022/1592.

Βιβλιογραφία iv

- Assa Naveh and Eran Tromer, *Photoproof: Cryptographic image authentication for any set of permissible transformations*, 2016 IEEE Symposium on Security and Privacy (SP), 2016.
- Justin Thaler, *Proofs, arguments, and zero-knowledge*.
- Riad S. Wahby, Ioanna Tzialla, Abhi Shelat, Justin Thaler, and Michael Walfish, *Doubly-efficient zksnarks without trusted setup*, 2018 IEEE Symposium on Security and Privacy (SP), 2018, pp. 926–943.