STREAMING INTERACTIVE PROOFS

CS5234 FINAL PROJECT

NGUYEN DANG PHUC NHAT, ANJU GOPALKRISHNAN, GRACE NGU SOOK ERN

OUTLINE

INTRODUCTION TO STREAMING INTERACTIVE PROOFS – PROJECT SCOPE

RECALL: FINITE FIELDS AND POLYNOMIALS

REVISIT: EQUALITY COMMUNICATION PROBLEM

IP TECHNIQUE: MULTILINEAR EXTENSION

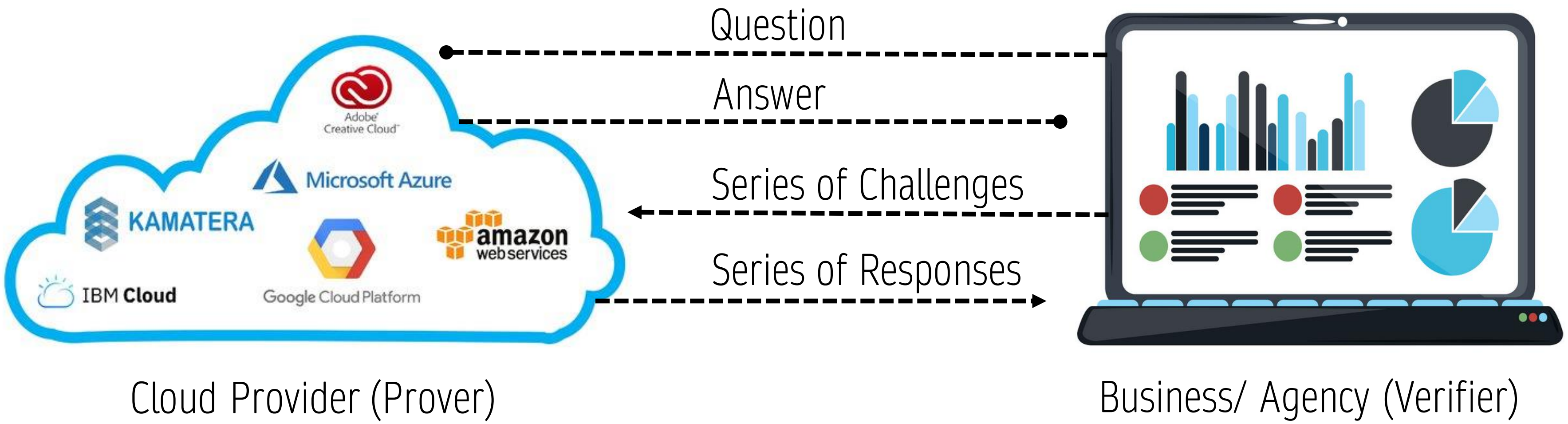
SUM-CHECK PROTOCOL

STREAMING APPLICATIONS: FREQUENCY MOMENT VERIFICATION

SUMMARY

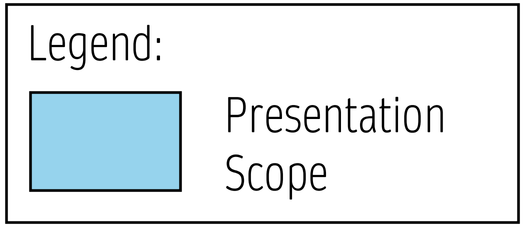
STREAMING INTERACTIVE PROOFS (SIP)

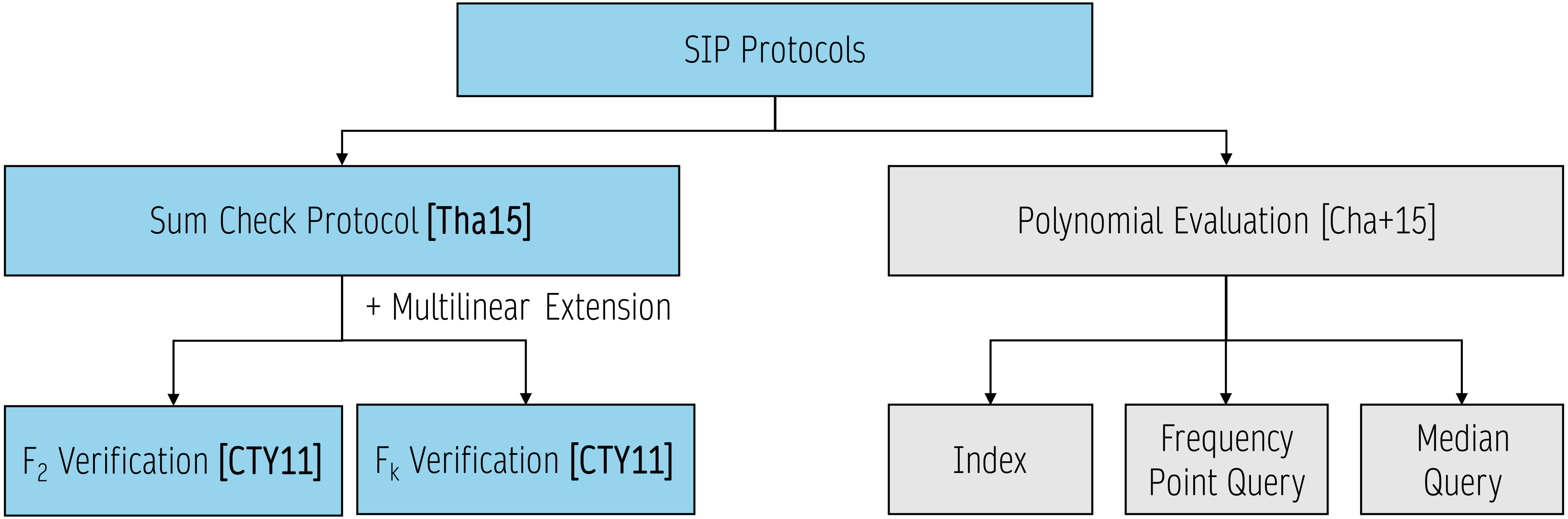
Problem Statement: A **computationally limited** client (Business/Agency) outsources a **powerful but untrusted** third-party service provider (Cloud). Example applications: Social Media Analytics, Financial Transaction Agency, etc.



Cloud service provider tries to convince the business of the truth of an 'answer' to the 'question' by interacting through a series of messages generated by randomized algorithms.

PROJECT SCOPE





Evaluation for both protocols: 1) Correctness, 2) Soundness, 3) Space Usage, and 4) Communication Cost

PRELIMINARIES: FINITE FIELDS AND POLYNOMIALS

Finite fields and polynomials are two algebraic concepts we utilize in the SIP setting:

Finite Fields, $\mathbb{F}$ in the SIP settings include both prime and binary fields. The common operations on $\mathbb{F}$ are 1) Addition 2) Multiplication

Polynomial is a mathematical expression involving a sum of powers in one or more variables multiplied by coefficients:

E.g.: Uni-variate polynomial - $a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$ where $a_n, a_{n-1}, \ldots, a_0$ are coefficients and $\mathbf{n}$ is degree of polynomials.

Multi-variate polynomial - $x^2 + 2xy + 2xz + y^2 + 2yz + z^2$

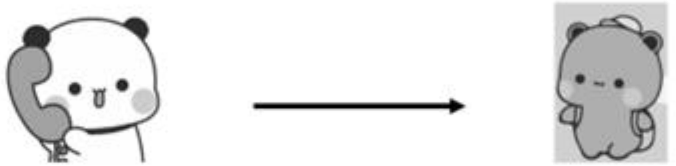
Why are they important for SIP?

1. Efficient to work with: Addition, Multiplication
2. Makes verification easy: A polynomial can be interpolated from a set of points and they can represent complex computations succinctly

POWER OF RANDOMNESS: EQUALITY PROBLEM

From our lectures… We learned about checking if two points are equal through Alice and Bob's one-way communication (non-interactive). Two claims arise from this protocol, and we will see here how randomness and polynomials plays an important role for SIPs [Tha20a].

Equality: Another Communication Problem



$x \in \{0,1\}^n$ $\qquad$ $y \in \{0,1\}^n$

- Alice and Bob gets n-length Binary strings x, y respectively
- Bob needs to determine whether $x = y$ or not deterministically
- **Lower Bound:** $\geq n$ bits of communication required

New Idea: $x \rightarrow p(r) = \sum_{i=1}^{n} x_i r^i$

$y \rightarrow q(r) = \sum_{i=1}^{n} y_i r^i$

Total cost: 2 field elements i.e. $(r, p(r))$, field size is n^2
$O(\log|F|) = O(\log n)$ bits

Protocol:

- Alice picks random $\mathrm{r} \in \mathbb{F}$ and sends (r, p(r)) to Bob.
- Bob checks p(r) = q(r)

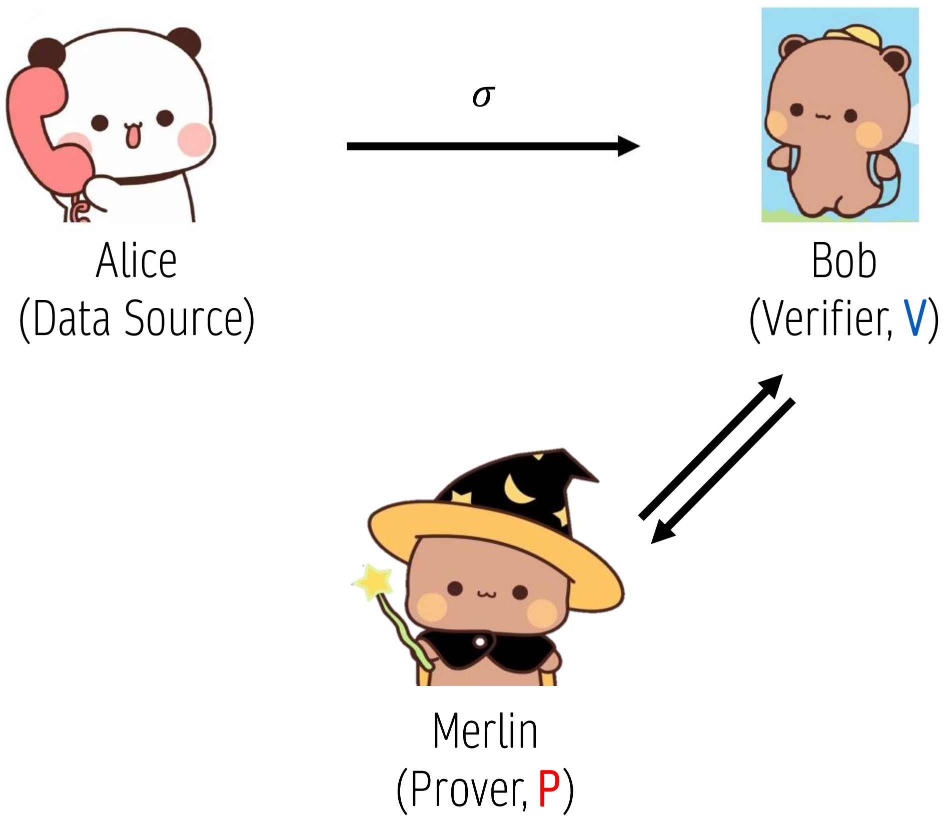
- Claim 1: if x= y, then Bob outputs **EQUAL** with high probability (1).

If 2 low-degree polynomial agree at randomly chosen input, then they are the same polynomial

- Claim 2: if x ≠ y, then Bob outputs **NOT-EQUAL** with probability at least $1 - \frac{1}{n}$ over the choice of $\mathrm{r} \in \mathbb{F}$

If 2 inputs differ at all, then they differ almost everywhere if interpreted as polynomials.

SIP GENERAL PROTOCOL & REQUIREMENTS



SIP Protocol:

- P computes and solves the problem and tells V the answer.
- P and V exchange a series of conversations.
- V's challenges are randomized in such a way that P cannot predict what V would ask.
- Goal: P wants to convince V that the answer is correct. V wants to detect if P is lying.

To prove that the answer is correct, two requirements must be met:

1. Completeness: An honest P can convince V to accept.

$$Pr[V\ accepts\ if\ P\ is\ honest]\ \geq\ 1-\delta_c$$

2. Soundness: V will catch a lying P with high probability. This must hold even if P is computationally unbounded and trying to trick V into accepting the incorrect answer.

$$Pr[V\ accepts\ if\ P\ is\ dishonest]\ \leq \delta_s$$

INTERACTIVE PROOF TECHNIQUES

[THA20A]

PRELIMINARIES

1. SHWARTZ-ZIPPEL LEMMA

Let $p \neq q$ be ℓ-variate polynomials of total degree at most d.

Then $\Pr_{r \in \boldsymbol{F}^{\ell}}[p(r) = q(r)] \leq \frac{d}{|\boldsymbol{F}|}$.

"Total degree" refers to the maximum sum of degrees of all variables in any term. E.g., $x_1^2 x_2 + x_1 x_2$ has total degree 3.

2. MULTILINEAR EXTENSIONS

Definition [**Extensions**]. Given a function $f\colon \{0,1\}^{\ell} \to \boldsymbol{F}$, a ℓ-variate polynomial g over **F** is said to **extend** f if $f(x) = g(x)$ for all $x \in \{0,1\}^{\ell}$.

Definition [**Multilinear Extensions**]. Any function $f\colon \{0,1\}^{\ell} \to \boldsymbol{F}$ has a **unique** multilinear extension (MLE), denoted $\tilde{f}$.

- Multilinear means the polynomial has degree at most 1 in each variable.
- $(1 - x_1)(1 - x_2)$ is multilinear, $x_1^2 x_2$ is not.

MULTILINEAR EXTENSION

$f: \{0,1\}^2 \to \mathbb{F}$

1	2
8	10

$\tilde{f}: \mathbf{F}^2 \to \mathbf{F}$

$\tilde{f}(x_1, x_2) = (1 - x_1)(1 - x_2) + 2(1 - x_1)x_2 + 8x_1(1 - x_2) + 10x_1x_2$

1	2	3	4	5	6
8	10	12	14	16	18
15	18	21	24	27	30
22	26	30	34	38	42
29	34	39	44	49	56

Can check:

$\tilde{f}(0,0) = 1$
$\tilde{f}(0,1) = 2$
$\tilde{f}(1,0) = 8$
$\tilde{f}(1,1) = 10$

SUM CHECK PROTOCOL

[THA20B], [THA15]

SUM-CHECK PROTOCOL

Input: V given oracle access to a ℓ-variate polynomial $g : F^{\ell} \to F$ over field $\boldsymbol{F}$.

Goal: compute the quantity:

$$\sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \dots \sum_{b_\ell \in \{0,1\}} g(b_1, \dots, b_\ell).$$

Start: P sends claimed answer C_1. The protocol must check that:

$$C_1 = \sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \dots \sum_{b_\ell \in \{0,1\}} g(b_1, \dots, b_\ell).$$

Round 1: P sends **univariate** polynomial $s_1(X_1)$ claimed to equal:

$$H_1(X_1) := \sum_{b_2 \in \{0,1\}} \dots \sum_{b_\ell \in \{0,1\}} g(X_1, b_2, \dots, b_\ell)$$

V checks that $C_1 = s_1(0) + s_1(1)$.

If this check passes, it is safe for V to believe that C_1 is the correct answer, so long as V believes that $s_1 = H_1$.

How to check this? Just check that s_1 and H_1 agree at a random point r_1!

V can compute $s_1(r_1)$ directly from P's first message, but not $H_1(r_1)$.

Start: P sends claimed answer C_1. The protocol must check that:

$$C_1 = \sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \dots \sum_{b_\ell \in \{0,1\}} g(b_1, \dots, b_\ell).$$

Round 1: P sends **univariate** polynomial $s_1(X_1)$ claimed to equal:

$$H_1(X_1) := \sum_{b_2 \in \{0,1\}} \dots \sum_{b_\ell \in \{0,1\}} g(X_1, b_2, \dots, b_\ell)$$

V checks that $C_1 = s_1(0) + s_1(1)$.

V picks r_1 at random from $\boldsymbol{F}$ and sends r_1 to P.

Round 2: They recursively check that $s_1(r_1) = H_1(r_1)$.

i.e., that $s_1(r_1) = \sum_{b_2 \in \{0,1\}} \dots \sum_{b_\ell \in \{0,1\}} g(r_1, b_2, \dots, b_\ell)$.

Start: P sends claimed answer C_1. The protocol must check that:

$$C_1 = \sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \dots \sum_{b_\ell \in \{0,1\}} g(b_1, \dots, b_\ell).$$

Round 1: P sends **univariate** polynomial $s_1(X_1)$ claimed to equal:

$$H_1(X_1) := \sum_{b_2 \in \{0,1\}} \dots \sum_{b_\ell \in \{0,1\}} g(X_1, b_2, \dots, b_\ell)$$

V checks that $C_1 = s_1(0) + s_1(1)$.

V picks r_1 at random from $\boldsymbol{F}$ and sends r_1 to P.

Round 2: They recursively check that $s_1(r_1) = H_1(r_1)$.

i.e., that $s_1(r_1) = \sum_{b_2 \in \{0,1\}} \dots \sum_{b_\ell \in \{0,1\}} g(r_1, b_2, \dots, b_\ell)$.

Round ℓ (Final round): P sends univariate polynomial $s_\ell(X_\ell)$ claimed to equal

$$H_\ell := g(r_1, \dots, r_{\ell-1}, X_\ell).$$

V checks that $s_{\ell-1}(r_{\ell-1}) = s_\ell(0) + s_\ell(1)$.

V picks r_ℓ at random, and needs to check that $s_\ell(r_\ell) = g(r_1, \dots, r_\ell)$.

- No need for more rounds. V can perform this check with one oracle query.

ANALYSIS: SUM-CHECK PROTOCOL

COMPLETENESS

Completeness holds by design: If P sends the prescribed messages, then all of V's checks will pass.

SOUNDNESS

Soundness: If P does not send the prescribed messages, then V rejects with probability at least $1 - \frac{\ell \cdot d}{|\boldsymbol{F}|}$, where d is the maximum degree of g in any variable.

Proof is by induction on the number of variables ℓ.

SOUNDNESS: SUMMARY

Summary: if $s_1 \neq H_1$, the probability V accepts is at most:

$$\Pr_{r_1 \in F}[s_1(r_1) = H_1(r_1)] + \Pr_{r_2,\ldots,r_\ell \in F}[\text{V accepts} | s_1(r_1) \neq H_1(r_1)]$$

$$\leq \frac{d}{|F|} + \frac{d(\ell-1)}{|F|} \leq \frac{d\ell}{|F|}.$$

COST ANALYSIS

Total communication is $O(d\ell)$ field elements.

- P sends ℓ messages, each a univariate polynomial of degree at most d. V sends $\ell - 1$ messages, each consisting of one field elements.
  - Space usage: $O(d \log |F|)$
- V's runtime is:

$O(d\ell + [\textit{time required to evaluate } g \textit{ at one point}])$.

- P's runtime is at most:

$O(d \cdot 2^{\ell} \cdot [\textit{time required to evaluate } g \textit{ at one point}])$.

STREAMING APPLICATIONS

STREAMING APPLICATIONS

- Modify **Sum-check** to design SIP protocols:
  - Compute the value $g(r)$ at a random point r
- Chosen problem: Frequency Moment Verification [CTY11]
  - F_2 Verification
  - Extension to F_k

STREAMING SETUP

- Universe: $[n]$ where $n = 2^b$
- Token: (i, c)
- Frequency vector: $(f_1, f_2, \dots, f_n)$
- Stream length m

- Frequency Moment: $F_k = \sum_{i=1}^{n} f_i^k$

APPLY MULTILINEAR EXTENSION

- Think of the Frequency Vector as a function P
  - $P : [n] \rightarrow [m]$
  - $P(i) = f_i$
- Transform the input of P into its binary representation
  - $P : \{0,1\}^b \rightarrow \mathbb{F}$
  - $P(i_1, i_2, \dots, i_b) = f_i$
- Choose a field $\mathbb{F}$ such that $[m] \subseteq \mathbb{F}$

APPLY MULTILINEAR EXTENSION

- Transform P into a multi-variate function
  - $P : \{0,1\}^b \to \mathbb{F}$
  - $P(i_1, i_2, \ldots, i_b) = f_i$
- Apply Multilinear Extension
  - $\tilde{P} : \mathbb{F}^b \to \mathbb{F}$
  - $\tilde{P}(z_1, z_2, \ldots, z_b) = \sum_{i \in [n]} P(i_1, i_2, \ldots, i_b)\, \chi_i(z_1, z_2, \ldots, z_b)$
- χ_i function acts as identity over $\{0,1\}^b$
  - $\chi_i : \mathbb{F}^b \to \mathbb{F}$
  - $\chi_i(z_1, z_2, \ldots, z_b) = 1$ if $(z_1, z_2, \ldots, z_b) = (i_1, i_2, \ldots, i_b)$
  - $\chi_i(z_1, z_2, \ldots, z_b) = 0$ for $(z_1, z_2, \ldots, z_b) \in \{0,1\}^b \backslash (i_1, i_2, \ldots, i_b)$

APPLY SUM-CHECK

- Goal: Apply Sum-check on $\tilde{P}$ (?) 🤔
  - $\tilde{P} : \mathbb{F}^b \to \mathbb{F}$
  - $\tilde{P}(z_1, z_2, \dots, z_b) = \sum_{i \in [n]} f_i \, \chi_i(z_1, z_2, \dots, z_b)$

- Randomly pick $(r_1, r_2, \dots, r_b) \in \mathbb{F}^b$
  → Verifier computes $\tilde{P}(r_1, r_2, \dots, r_b)$ from the stream

- Keep a variable Q
- Process each stream token (i, c):
  - $Q \leftarrow Q + c\chi_i(r_1, r_2, \dots, r_b)$
  - $O(\log |\mathbb{F}|)$ bits

APPLY SUM-CHECK

- Goal: Apply Sum-check on $\tilde{P}$ (?)
  - $\tilde{P} : \mathbb{F}^b \to \mathbb{F}$
  - $\tilde{P}(z_1, z_2, \dots, z_b) = \sum_{i \in [n]} f_i \, \chi_i(z_1, z_2, \dots, z_b)$

- $\sum_{j \in [n]} \left(\tilde{P}(j_1, j_2, \dots, j_b)\right)^2 = \sum_{j \in [n]} (P(j_1, j_2, \dots, j_b))^2 = \sum_{j \in [n]} f_j^2 = F_2$

- Final Protocol:
  - To verify F_2, apply Sum-check on $\tilde{P}^2$
  - Evaluate $\tilde{P}^2(r_1, r_2, \dots, r_b)$ by computing Q^2 after processing the stream

ANALYSIS

- Completeness: Perfect! If Prover is honest, Verifier always accepts
- Soundness: $\tilde{P}^2$ is b-variate with degree at most 2 at all variables. If Prover is dishonest, Verifier rejects with probability at least $1 - \frac{2b}{|\mathbb{F}|}$. With a large enough field $|\mathbb{F}| = \Theta\left(\frac{b}{\delta_s}\right) = \Theta(\frac{1}{\delta_s}\log n)$, we can bound soundness to any parameter δ_s
- Verifier's space usage: $O(\frac{1}{\delta_s}\log n)$ bits
  - $O(2 * \log|\mathbb{F}|) = O(\frac{1}{\delta_s}\log n)$ bits to keep Q^2
  - $O(2 * \log|\mathbb{F}|) = O(\frac{1}{\delta_s}\log n)$ for univariate polynomials degree at most 2 in Sum-check
- Number of messages: $O(b) = O(\log n)$ for Sum-check on b-variate polynomial

EXTENSION TO F_k

- $\sum_{j\in[n]}\left(\tilde{P}(j_1, j_2, \dots, j_b)\right)^k = \sum_{j\in[n]}(P(j_1, j_2, \dots, j_b))^k = \sum_{j\in[n]} f_j^k = F_k$

- Final Protocol:
  - To verify F_k, apply Sum-check on $\tilde{P}^k$
  - Evaluate $\tilde{P}^k(r_1, r_2, \dots, r_b)$ by computing Q^k after processing the stream

- Analysis is identical, but with a constant k blow-up in space: $O(\frac{k}{\delta_s}\log n)$ bits

- Space-Advantage over computation (AMS sketch): $O\left(k\epsilon^{-2}n^{1-\frac{1}{k}}\log\left(\frac{1}{\delta}\right)(\log m + \log n)\right)$ bits

TRIVIA

- Over a field $\mathbb{F}_p$ defined by the prime p, $\forall x \in \mathbb{F}_p, x^{p-1} = 1$ according to Fermat's Little Theorem.
  - $\sum_{j\in[n]}\left(\tilde{P}(j_1, j_2, \dots, j_b)\right)^{p-1} = \sum_{j\in[n]} f_j^0 = F_0$
- Cannot apply **Sum-check** directly. See arithmetization idea in **[CMT12]**.

SUMMARY

- We have seen:
  - Background knowledge for Streaming IP
  - A basic algebra concept: Multilinear Extension
  - A basic Verification protocol: Sum-check
  - Combination of both to Verify F_k

- IP in general and SIP in particular are deep subjects. Further study directions include, but not limited to:
  - Verifying Frequency Query and other Point Query problems [Cha+15] (in project write-up)
  - Communication Complexity classes in SIP [Cha+15]
  - Efficient SIP implementations [CMT12]

REFERENCES

- [Tha15] Justin Thaler. "Stream Verification". In: CoRR abs/1507.04188 (2015). arXiv: 1507.04188. URL: http://arxiv.org/abs/1507.04188.
- [Tha20a] Justin Thaler. COSC544 Lecture 2: The power of randomness: Fingerprinting and Freivalds' Algorithm for verifying matrix multiplication. Low-degree and multilinear extensions. 2020. URL: https://people.cs.georgetown.edu/jthaler/COSC544/Lecture2slides.pdf.
- [Tha20b] Justin Thaler. COSC544 Lecture 3: LFKN's sum-check protocol. 2020. URL: https://people.cs.georgetown.edu/jthaler/COSC544/Lecture3slides.pdf.
- [CTY11] Graham Cormode, Justin Thaler, and Ke Yi. "Verifying computations with streaming interactive proofs". In: arXiv preprint arXiv:1109.6882 (2011).
- [Cha+15] Amit Chakrabarti et al. "Verifiable stream computation and Arthur-Merlin communication". In: Leibniz international proceedings in informatics (LIPIcs) (2015), pp. 217–243.
- [CMT12] Graham Cormode, Michael Mitzenmacher, and Justin Thaler. "Practical verified computation with streaming interactive proofs". In: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference. 2012, pp. 90–112.

THANK YOU!