Practical Verified Computation with Streaming Interactive Proofs

Justin Thaler, Harvard University

Outsourcing

- Many applications require outsourcing computation to untrusted service providers.
  - Main motivation: commercial cloud computing services.
  - Also, weak peripheral devices; fast but faulty co-processors.
  - Volunteer Computing (SETI@home, World Community Grid, etc.)

- User requires a guarantee that the cloud performed the computation correctly.

AT&T Cloud Services License Terms

AT&T… MAKES NO WARRANTY THAT THE APPLICATION… WILL BE UNINTERRUPTED, ACCURATE, RELIABLE, TIMELY, SECURE OR ERROR-FREE… [NOR THAT] ANY ERRORS IN THE APPLICATION WILL BE CORRECTED.

AWS Customer Agreement

WE… MAKE NO REPRESENTATIONS OF ANY KIND … THAT THE SERVICE OR THIRD PARTY CONTENT WILL BE UNINTERRUPTED, ERROR FREE OR FREE OF HARMFUL COMPONENTS, OR THAT ANY CONTENT … WILL BE SECURE OR NOT OTHERWISE LOST OR DAMAGED.

Goals of Verifiable Computation

- Provide user with correctness guarantee, without requiring her to perform full computation herself.
  - Ideally user will not even maintain a local copy of the data.
- Minimize extra effort required for cloud to provide correctness guarantee.
- Achieve protocols secure against malicious clouds, but lightweight for use in benign settings.

Possible Approaches

1. Approaches making strong assumptions.
   - Replication [ACKLW02, HKD07, …] assumes majority of responses are correct.
   - Trusted hardware [JSM01, CGJ+09, SSW10…]
2. Approaches with **minimal assumptions**.
   - Interactive proofs (this talk).
   - Argument systems (use crypto).
3. Approaches using two or more clouds.
   - Refereed Games: assumes 1 cloud is honest.
   - Multi-Prover Interactive Proofs: assumes clouds cannot communicate with each other.

Interactive Proofs

AT&T/Google/Microsoft/Amazon

Business/Agency/Scientist

Interactive Proofs

AT&T/Google/Microsoft/Amazon

Business/Agency/Scientist

Interactive Proofs

AT&T/Google/Microsoft/Amazon

Business/Agency/Scientist

Interactive Proofs

AT&T/Google/Microsoft/Amazon

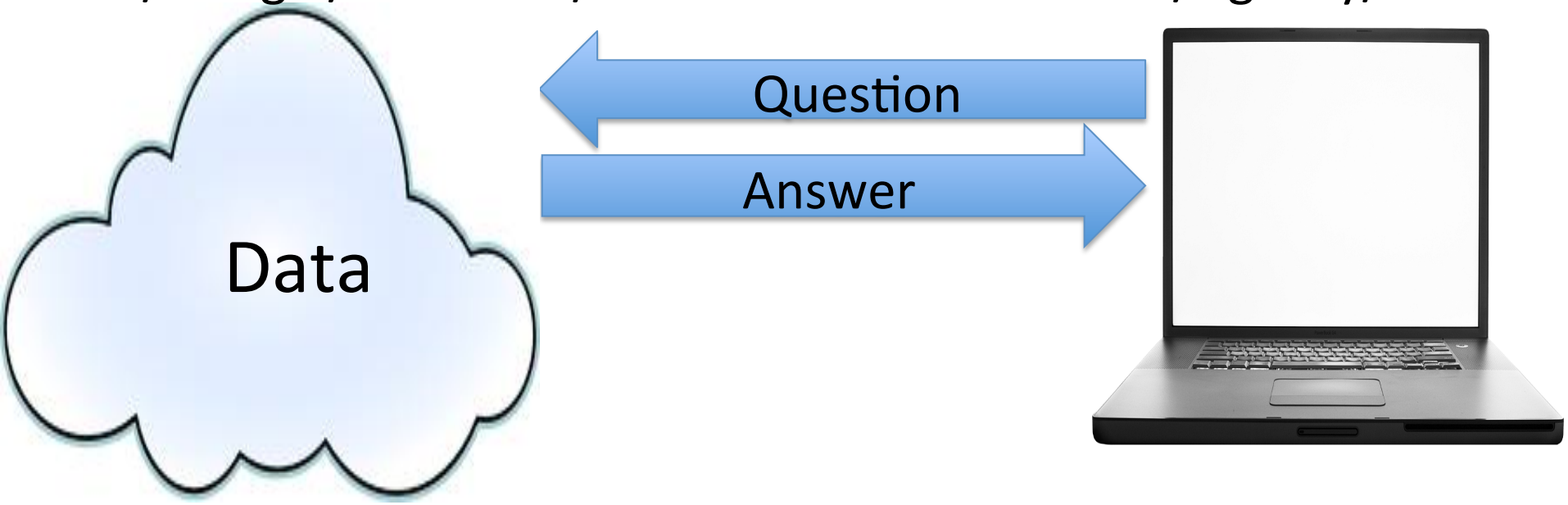
Business/Agency/Scientist

Question

Interactive Proofs

AT&T/Google/Microsoft/Amazon

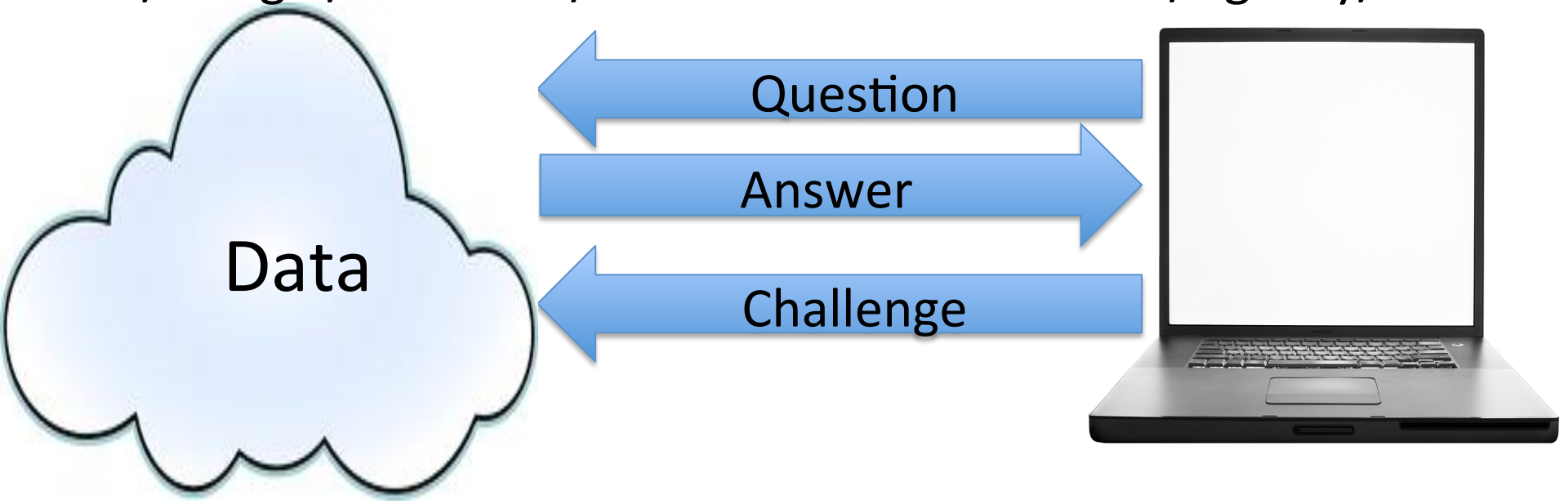
Business/Agency/Scientist



Interactive Proofs

AT&T/Google/Microsoft/Amazon

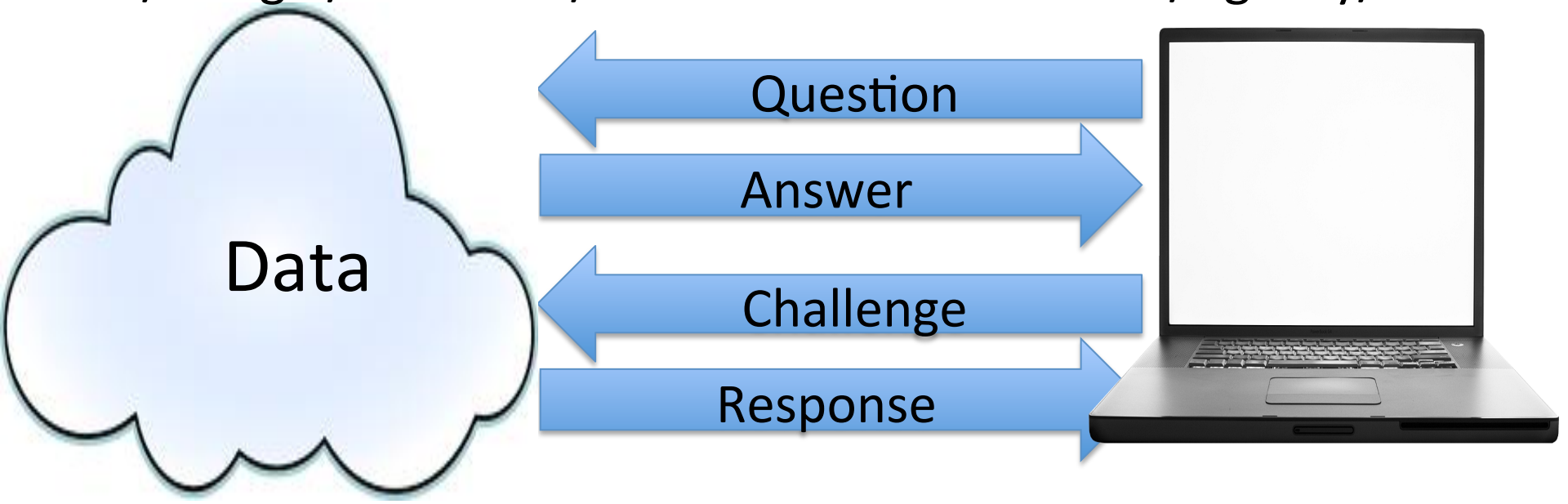
Business/Agency/Scientist



Interactive Proofs

AT&T/Google/Microsoft/Amazon

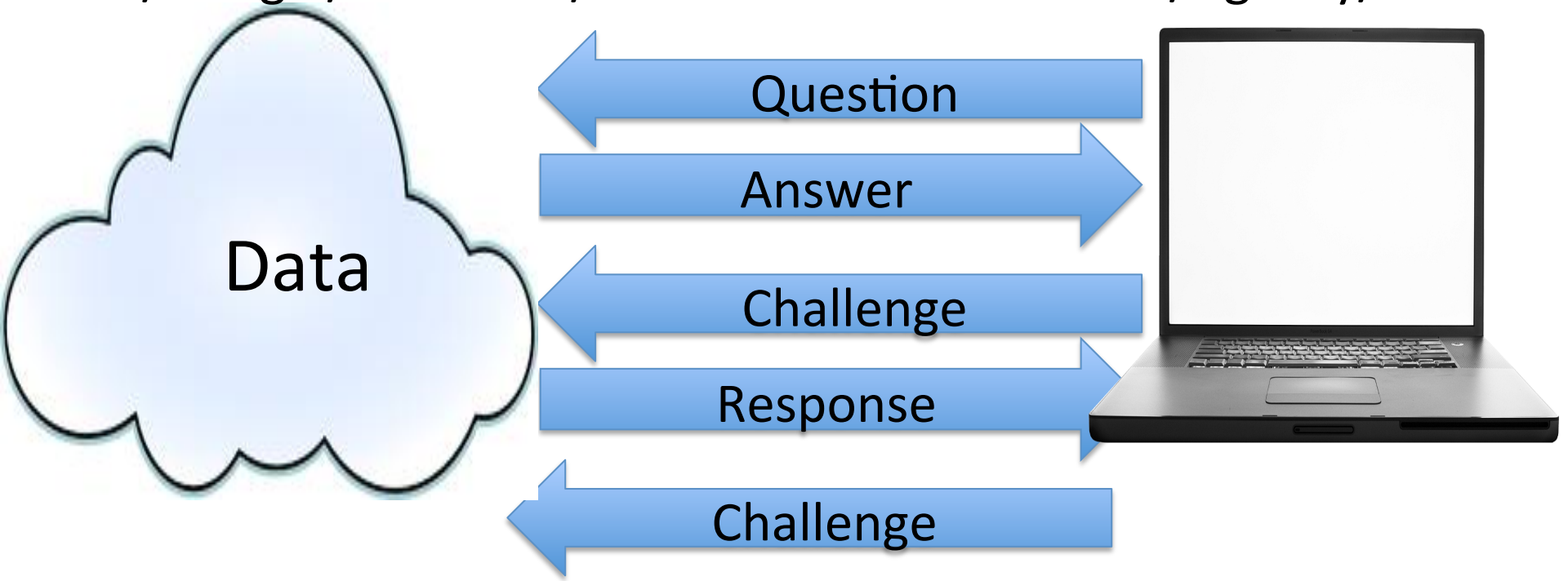
Business/Agency/Scientist



Interactive Proofs

AT&T/Google/Microsoft/Amazon

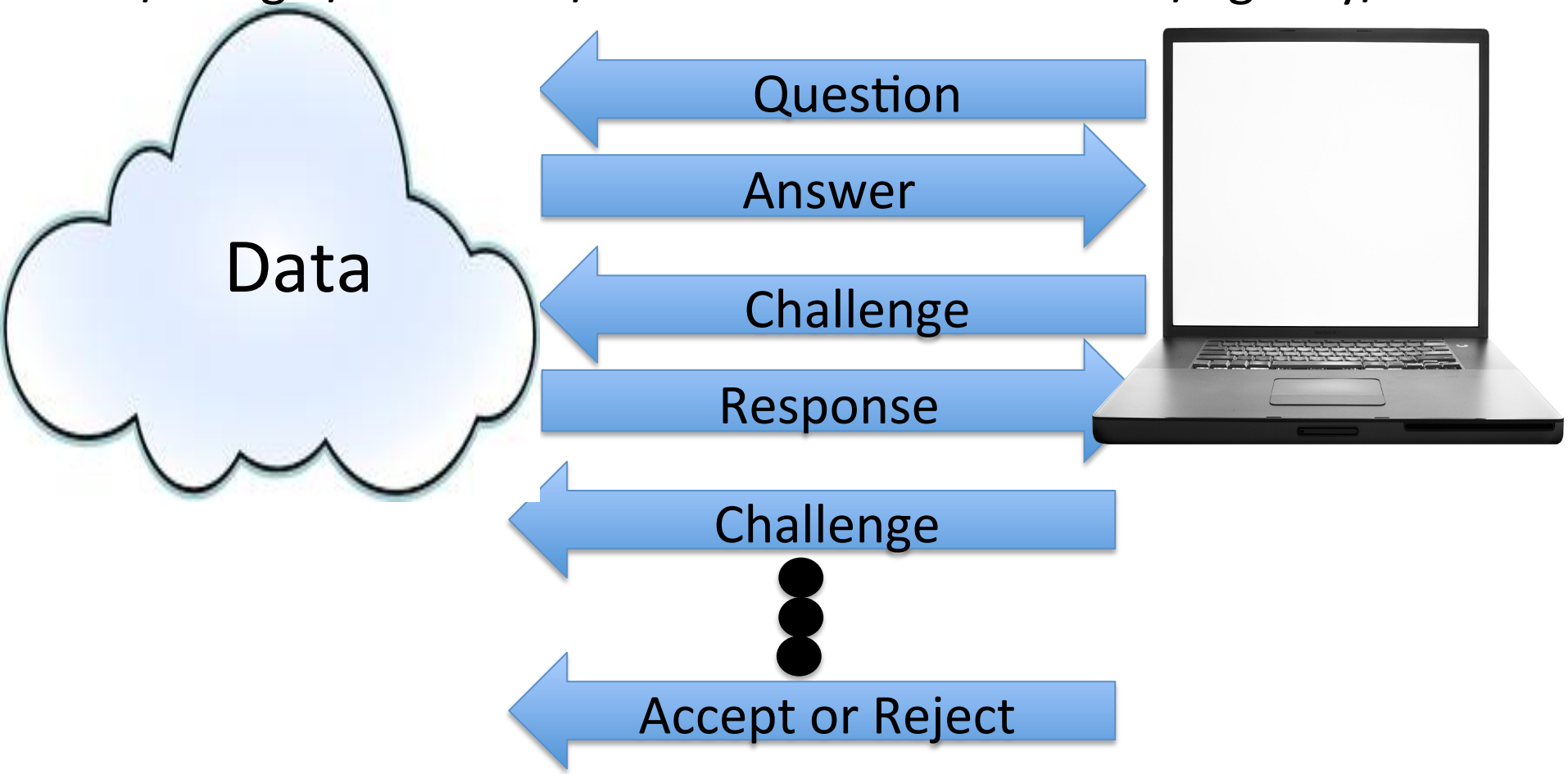
Business/Agency/Scientist



Interactive Proofs

AT&T/Google/Microsoft/Amazon

Business/Agency/Scientist



Interactive Proofs

- Prover P and Verifier V.

- P solves problem, tells V the answer.
  - Then P and V have a conversation.
  - P's goal: convince V the answer is correct.

- Requirements:
  - 1. Completeness: an honest P can convince V to accept.
  - 2. Soundness: V will catch a lying P with high probability (secure even if P is computationally unbounded).

Interactive Proofs

- IPs have revolutionized complexity theory in the last 25 years.
  - IP=PSPACE [LFKN90, Shamir90].
  - PCP Theorem [AS98, ALMSS98]. Hardness of approximation.
  - Zero Knowledge Proofs.

- But IPs have had very little impact in real delegation scenarios.
  - Why?
  - Not due to lack of applications!

Interactive Proofs

- Old Answer: Most results on IPs dealt with hard problems, needed P to be too powerful.
  - But recent constructions focus on "easy" problems (e.g. Interactive Proofs for Muggles [GKR 08]).
  - Allows V to run **very** quickly, so outsourcing is useful even though problems are "easy".
  - P does not need "much" more time to prove correctness than she does to just solve the problem!

Interactive Proofs

- Why does GKR **not** yield a practical protocol out of the box?
  - P has to do a lot of extra bookkeeping (**cubic** blowup in runtime).
  - Naively, V has to retain the full input.

This Talk: New Application of IPs

- To streaming problems: hard because V has to read input in one-pass streaming manner, but (might be) easy if V could store the whole input.

- Fits cloud computing well: streaming pass by V can occur while uploading data to cloud.

- V never needs to store entirety of data!

Data Streaming Model

- Stream: m elements from universe of size n.
  - e.g., $S=<x_1, x_2, \ldots, x_m> = 3,5,3,7,5,4,8,7,5,4,8,6,3,2, \ldots$

- Goal: Compute a function of stream, e.g., median, frequency moments, heavy hitters.

- Challenge:

  (i) Limited working memory, i.e., sublinear(n,m).

  (ii) Sequential access to adversarially ordered data.

  (iii) Process each update quickly.

Slide derived from [McGregor 10]

This Talk: Models

- Two models:
  1. One message (Non-interactive) [CCM 09/CCMT 12]: After both observe stream, P sends V an email with the answer, and a proof attached.
  2. Multiple rounds of interaction [CTY 10]: P and V have a *conversation* after both observe stream.

Costs in Our Models

- Two main costs: words of communication, and V's working memory.
- Other costs: running time, number of messages.

A Two-Pronged Approach

- First Prong: General purpose implementation to verify arbitrary computation [CMT12, TRMP12, T13].
  - Building on general-purpose GKR protocol.

- Second Prong: Develop highly optimized protocols for specific important problems [CCMT12, CMT10, CTY12, CCGT13].
  - Reporting queries (what value is stored in memory location x of my database?)
  - Matrix multiplication.
  - Graph problems like perfect matching.
  - Certain kinds of linear programs.
  - Etc.

Non-Interactive Protocols with Streaming Verifiers: A Sampling

A general technique

- Arithmetization: Given function *f* defined on small domain, replace *f* with its low-degree extension, LDE(*f*), as a polynomial defined over a large field.

- Can view LDE(*f*) as error-corrected encoding of *f*. Error-correcting properties give V considerable power over P.

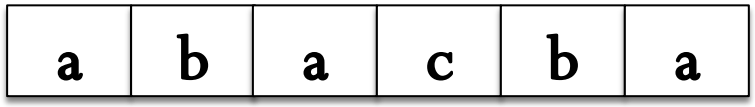
- If two (boolean) functions differ in one location, their LDE's will differ in almost all locations.

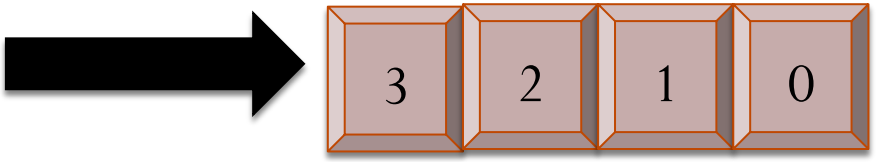
Second Frequency Moment (F_2)

- Second frequency moment is central streaming problem.
  - Captures sample variance, Euclidean norm, data similarity.

- Definition:
  - Let X be the frequency vector of the stream.
  - $F_2(X) = \sum_{i=1}^{n} X_i^2$

Raw data stream over universe {a, b, c, d}





Frequency Vector X

a b c d

$F_2(X) = 3^2 + 2^2 + 1^2 = 14$

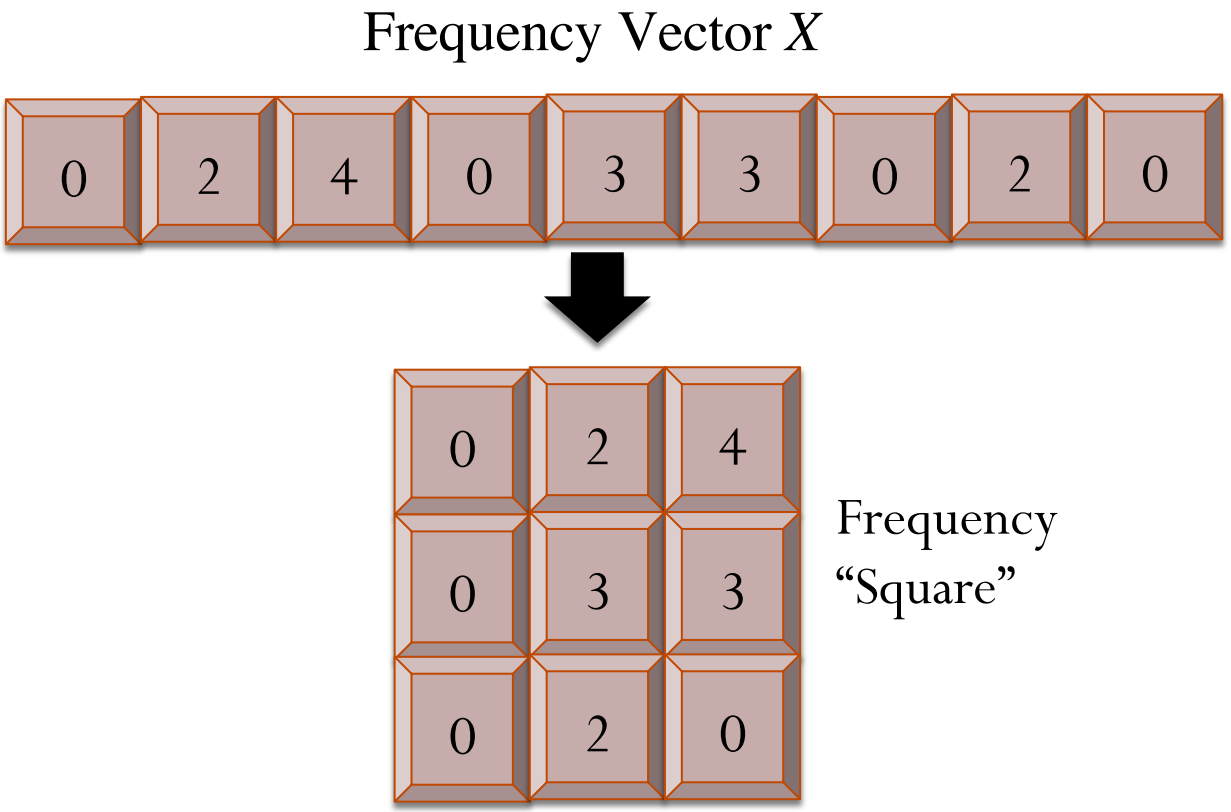
Second Frequency Moment

- [CCMT 12]: ($\sqrt{n}$ comm., $\sqrt{n}$ space)-protocol for F_2.
  - Terabytes of data translate to a few MBs of space and communication.

- Optimal. Lower bound of $\Omega(n)$ on comm. * space.

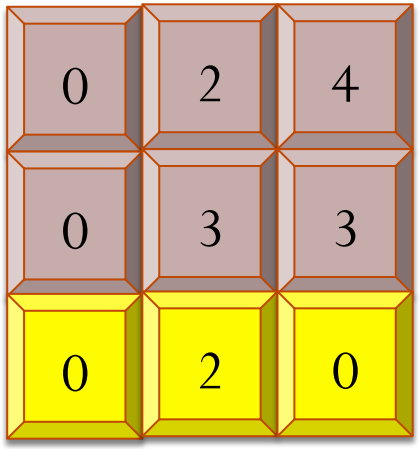
F_2 Protocol

- Recall: $F_2(X) = \sum_i X_i^2$
- View universe $[n]$ as $[\sqrt{n}]$ x $[\sqrt{n}]$.



- First idea: Have P send the answer "in pieces":
  - F_2(row 1). F_2(row 2). And so on. Requires $\sqrt{n}$ communication.
- V exactly tracks a row at random (denoted in yellow) so if P lies about any piece, V has a chance of catching her. Requires space $\sqrt{n}$.

Frequency Square



P sends

$20=2^2+4^2$

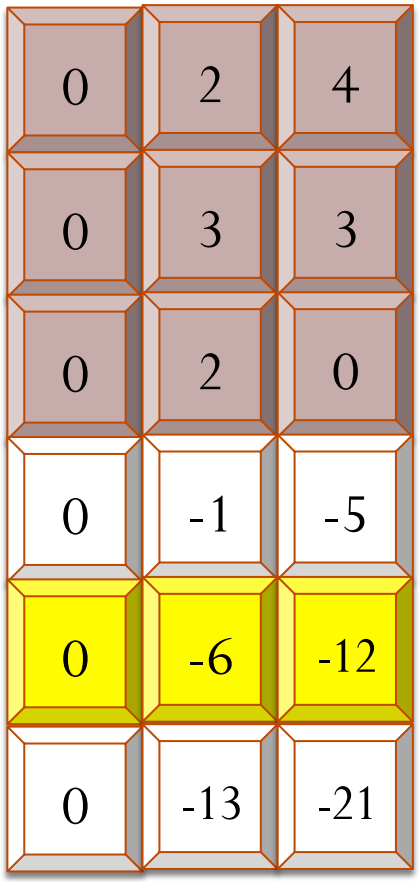
$18=3^2+3^2$

$4=2^2$

Slide derived from [McGregor 10]

- Problem: If P lies in only one place, V has small chance of catching her.
- What we'd like: if P lies about even one piece, she will have to lie about many.
- Solution: Have P commit (succinctly) to second frequency moment of rows of an **error-corrected encoding** of the input.
- Note: V can evaluate any row of the low-degree extension encoding in a streaming fashion.

Low-Degree Extension of Frequency Square



These values all lie on low-degree polynomial

P sends

$20=2^2+4^2$

$18=3^2+3^2$

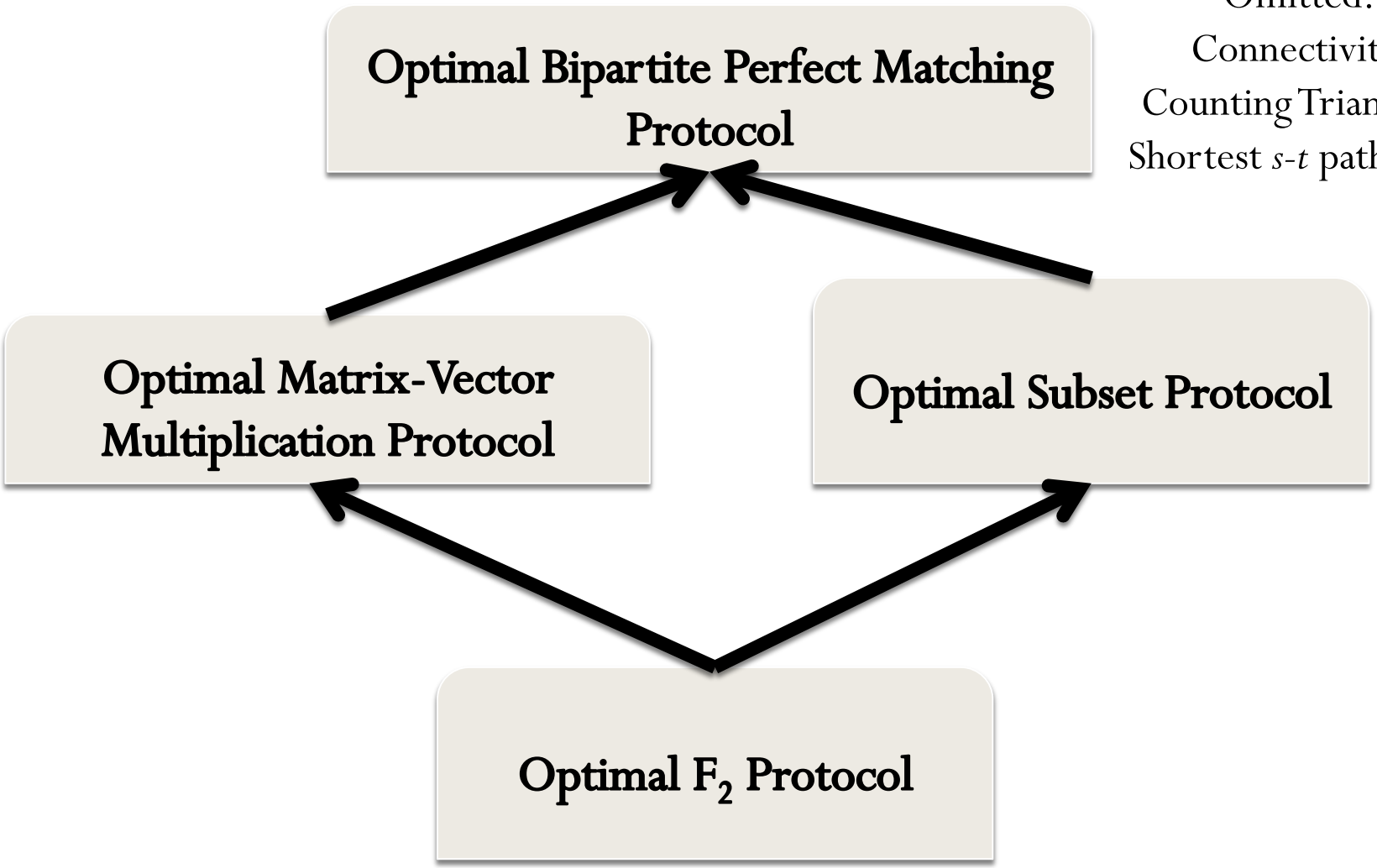
$4=2^2$

$26=(-1)^2+(-5)^2$

$180=(-6)^2+(-12)^2$

$610=(-13)^2+(-21)^2$

Why study F_2?



Omitted:
Connectivity,
Counting Triangles,
Shortest *s*-*t* path, etc.

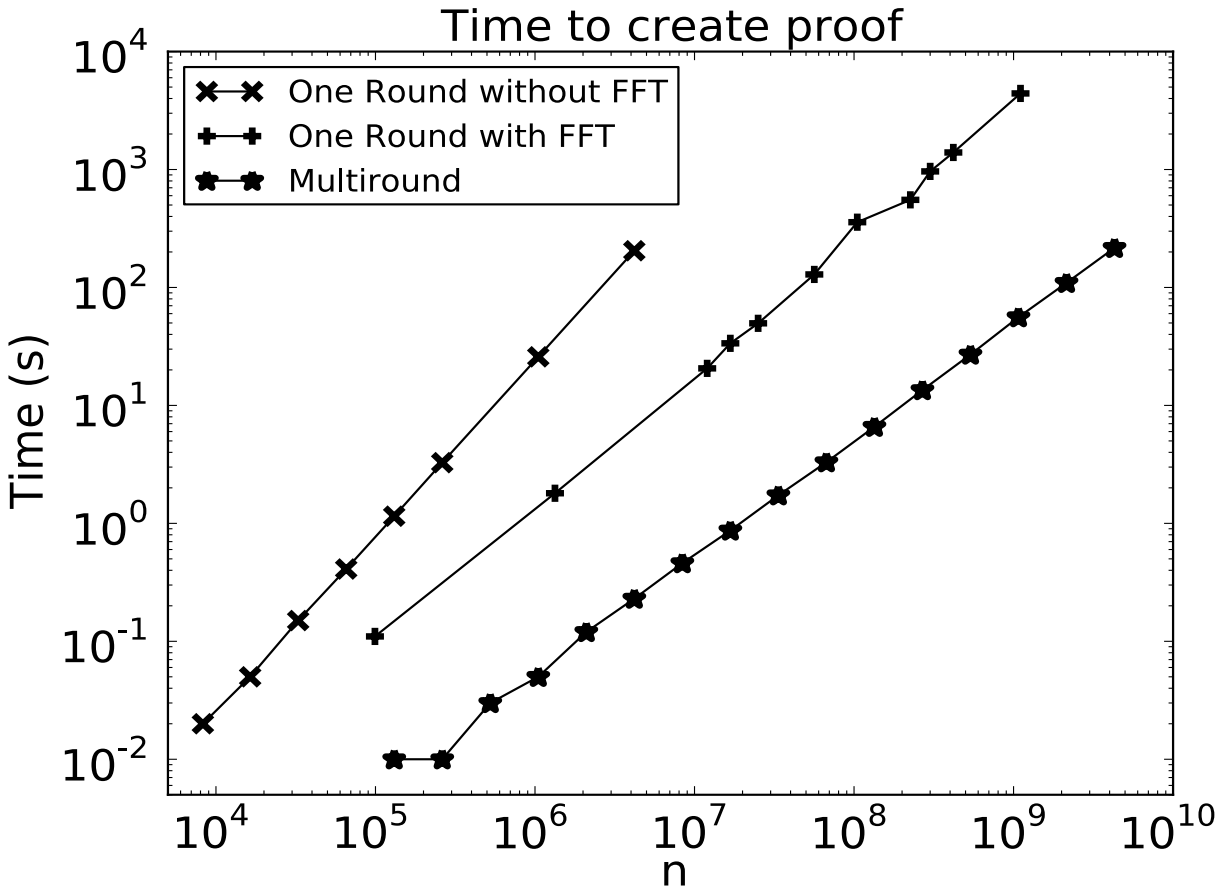
Protocol Engineering: Smart FFTs

- Naïve implementation of P in F_2 protocol requires $\Omega(n^{3/2})$ time, doesn't scale to large streams.

- Using FFT techniques, we reduce this to O(n log n) time.
  - "Standard" FFT algorithm is unsuitable.
  - Use Prime Factor FFT Algorithm instead. Works well over certain finite fields (which we can choose!), avoids precision issues entirely.

- Immediately yields fast protocols for many other problems.

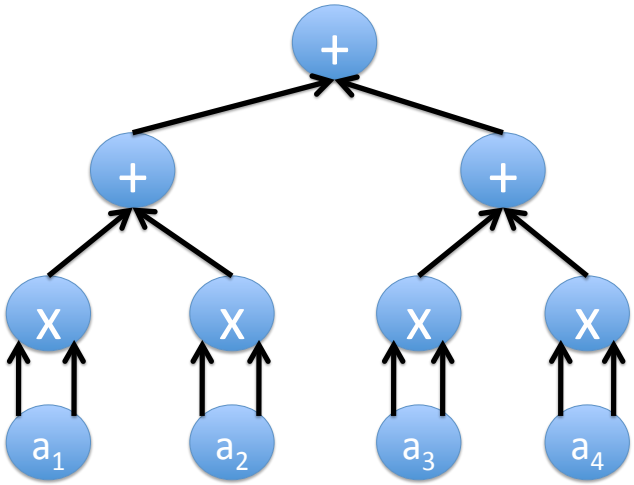
F_2 Experiments



Multi-round P from [CTY11] vs. Non-interactive P with and without FFT techniques

General Purpose IPs (Extending GKR)

Circuits, Fields, and All That



F_2 circuit

Saving V Space [CTY12]

- Can save V space "for free".
  - Reason: V only needs to store one LDE evaluation of the data.
  - Can be computed with one streaming pass over input.
- Fits cloud computing well: pass by V can occur while uploading data to cloud.
- The LDE evaluation is KBs in size, even if the input contains terabytes of data.

Saving V Time [CMT12]

- Can save V substantial amounts of time (at least for problems computable by small-depth circuits).
- E.g. when multiplying two 512x512 matrices, V requires .12s, while naive matrix multiplication takes .70s.
- Savings for V will be much larger at larger input sizes, and for more time-intensive computations.

Minimizing P's Overhead [CMT12]

- Brought P's runtime down from $\Omega(S^3)$, to $O(S \log S)$, where S is circuit size.
  - Key insight: use **multilinear** extension of circuit within the protocol.
  - Causes enormous cancellation in P's messages, allowing fast computation.

Minimizing P's Overhead [CMT12]

- Brought P's runtime down from $\Omega(S^3)$, to O(S log S), where S is circuit size.
  - Key insight: use **multilinear** extension of circuit within the protocol.
  - Causes enormous cancellation in P's messages, allowing fast computation.
- Lots of additional engineering.
  - Choosing the "right" finite field to work over.
  - Using the "right" circuits.
  - Etc.
- Practically speaking, still not good enough on its own.
  - 256 x 256 matrix multiplication takes P 27 minutes.
  - Naïve implementation of GKR would take trillions of times longer.

Reducing Overhead Further [T13]

- Downsides to [CMT12] implementation:
  - For "regular" circuits: log S factor runtime overhead for P.
  - For "irregular" circuits: log S factor runtime overhead for P, and expensive pre-processing phase for V.

Reducing Overhead Further [T13]

- Downsides to [CMT12] implementation:
  - For "regular" circuits: log S factor runtime overhead for P.
  - For "irregular" circuits: log S factor runtime overhead for P, and expensive pre-processing phase for V.

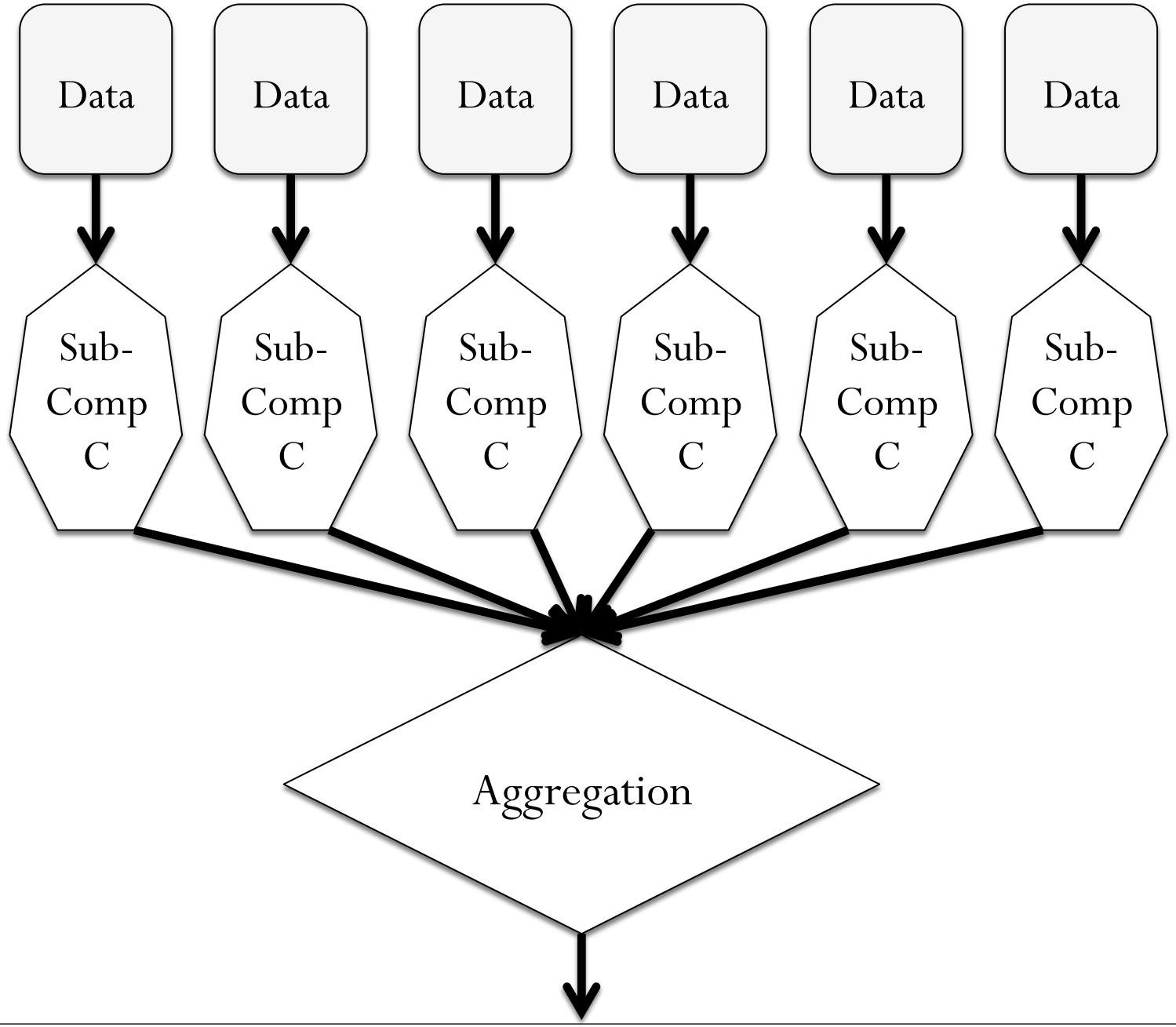
- Solution for "regular" circuits: Reduce P's runtime to O(S).
  - Key idea: use new arithmetization of the circuit, allowing P to reuse work across rounds.
  - Experimental results: 250x speedup over [CMT12].
  - P less than 10x slower than a C++ program that just evaluates the circuit.
  - Example applications: MatMult, DISTINCT, F_2, Pattern Matching, FFTs.

Results for Regular Circuits [T13]

Problem	P time [CMT12]	P time [T13]	V time [Both]	Rounds [T13]	Protocol Comm* [T13]	Circuit Eval Time
DISTINCT ($n=2^{20}$)	56.6 minutes	17.2 s	.2 s	236	40.7 KB	1.88 s
MatMult (512 x 512)	2.7 hours	37.8 s	.1 s	1361	5.4 KB	6.07 s

Dealing with Irregular Circuits [T13]

- No magic bullet for dealing with irregular wiring patterns.
  - Need *some* assumption about the computation being outsourced.
  - Is there structure in real-world computations?
- Yes: Data Parallel computation.
  - Any setting where a sub-computation C is applied to many pieces of data.
  - Make no assumptions about C itself.
  - These are the sort of problems getting outsourced!


Data
Data
Data
Data
Data
Data
Sub-
Comp
C
Sub-
Comp
C
Sub-
Comp
C
Sub-
Comp
C
Sub-
Comp
C
Sub-
Comp
C
Aggregation

Leveraging Parallelism [T13]

- Problem: Verify massive parallel computations.
  - Directly applying existing results has big overhead.
  - Costs depend on number of data pieces.
- Approach: take advantage of parallelism.
  - Reduce V's effort to proportional to size of C.
  - Reduce P's overhead to log size of C.
  - No dependence on number of data pieces.
- Key insight: C may be irregular internally, but the computation is maximally regular between copies of C.

Further Leveraging Parallelism [TRMP12, T13]

- In our protocols, P and V themselves can be parallelized (although V runs quickly even without parallelization).
- Using a GPU, achieved 40x-100x speedups for P, 100x speedups for V.

Independent Results

- Recent efforts to build practical *argument systems*.
  - Setty, McPherson, Blumberg, Vu, Braun, Parno, Walfish [NDSS12, Security12, EuroSys13]
- Vu et al. [Oakland13] build a system that:
  1. Starts with a high-level programming language.
  2. Automatically compiles any program in the language into an arithmetic circuit.
  3. Decides whether GKR implementation from [CMT12] (plus refinements) or state-of-the-art argument system is more efficient, and runs the better of the two.
- Experimental comparison in [Oakland13] shows [CMT12] significantly faster except for programs with complicated control flow or that are highly sequential.

Future Directions

1. Build a system that automatically verifies MapReduce programs.
   - Main obstacle: interactive proof technology cannot yet handle data-dependent control flow.
   - Address by **combining** crypto with interactive proofs?

2. Interactive proofs for deep circuits (i.e. non-parallel computation)?
   - Formalizes the question: "Is proving harder than computing?"

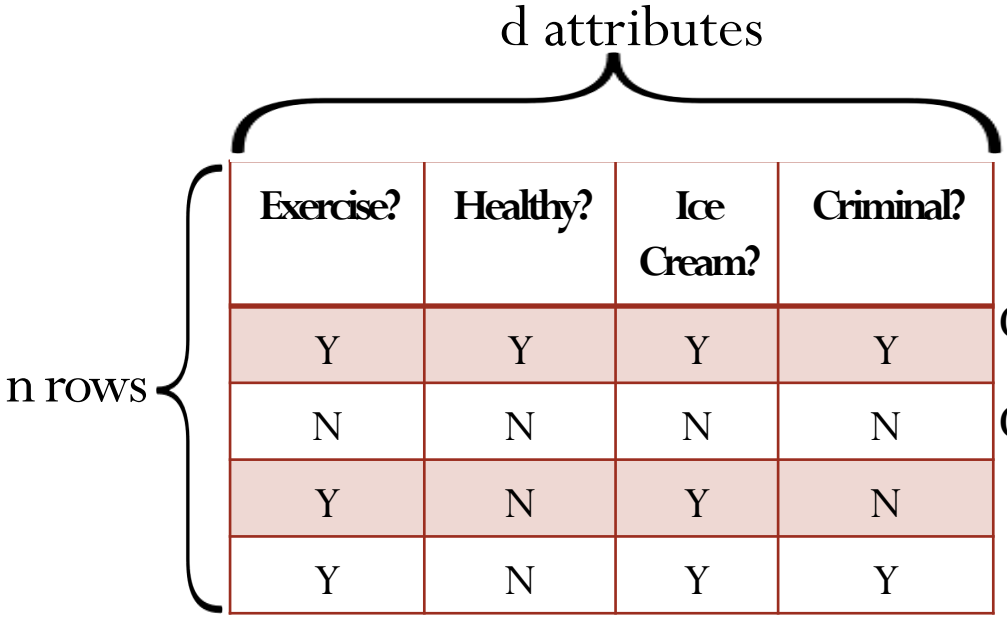
References

- Cormode, Mitzenmacher, T. (ESA 2010)
- Cormode, T., Yi (VLDB 2012)
- Cormode, Mitzenmacher, T. (ITCS 2012)
- T., Roberts, Mitzenmacher, Pfister (HotCloud 2012)
- Chakrabarti, Cormode, McGregor, T. (ICALP 2009, in submission 2012)
- T. (in submission, 2013)
- Chakrabarti, Cormode, Goyal, T. (ongoing, 2013)

Research Interests (So Far)

1. Verifiable computation (this talk).
2. Algorithms for massive data sets.
   - Streaming algorithms for pattern mining [MST12].
   - Hashing-based algorithms for set maintenance and sparse recovery [AGMT11, MT12, JMT13].
3. Computational learning theory, differential privacy, and their intersection [STT, COLT12], [TUV, ICALP12], [BT, in submission], [CTUW, ongoing].
   - Efficient learning in the presence of irrelevant information.
   - Faster algorithms for private data release.

k-way Marginal Queries



Query on a row: $q(x) = \text{Ice Cream}? \wedge \text{Criminal}?$

Query on database: $\frac{1}{n}\sum_i q(x_i)$.

- **k-way marginal queries**: q has at most k literals.
- Number of k-way marginal queries is $\sim d^k$.

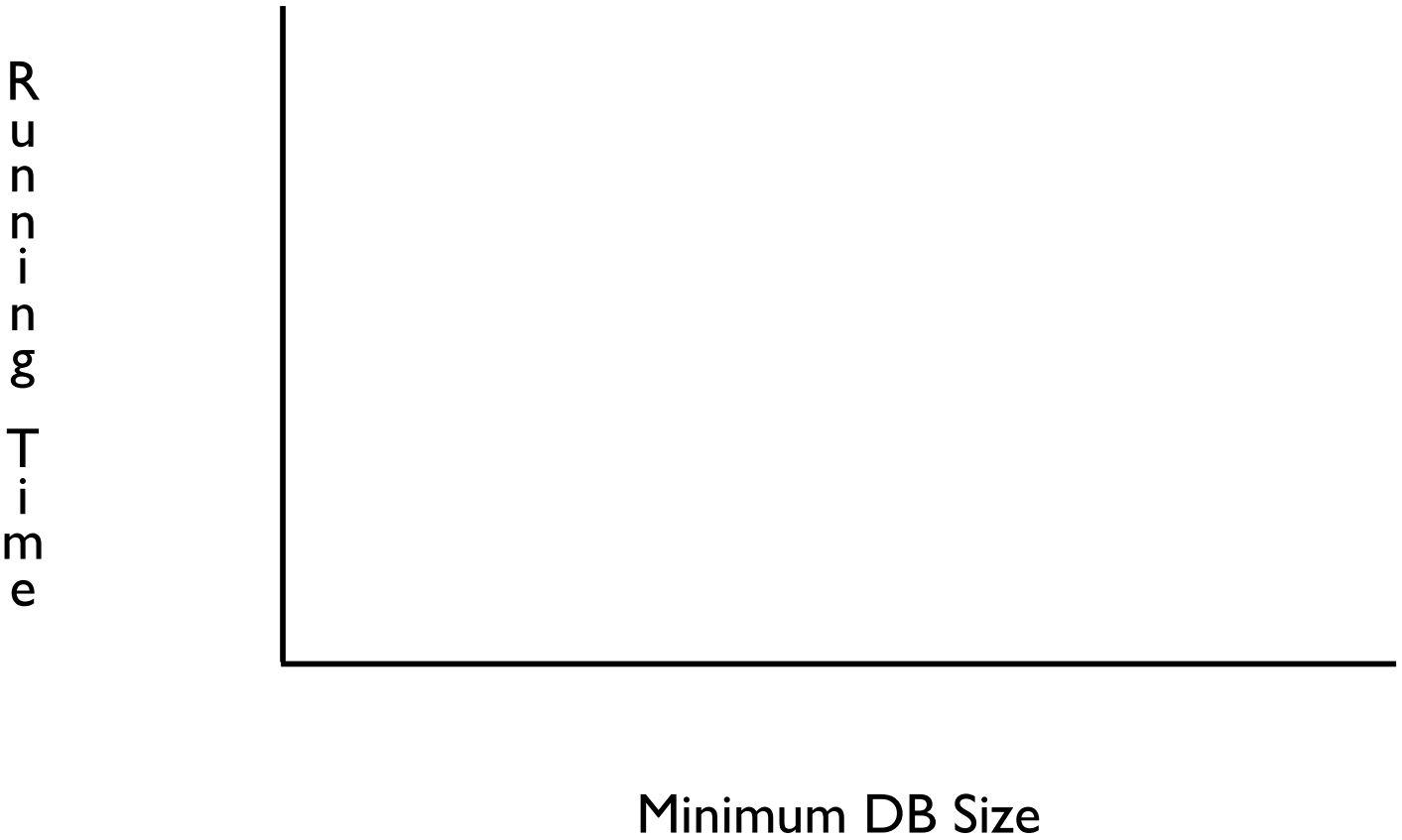
Goal: Private One-Shot Release Mechanism

- Want to release a private *summary* of database D such that for all k-way marginals q:

$$|q(\text{Summary}) - q(D)| \leq .01$$

- Two parameters to optimize: running time of sanitizer, and minimal database size required for non-trivial accuracy guarantees.

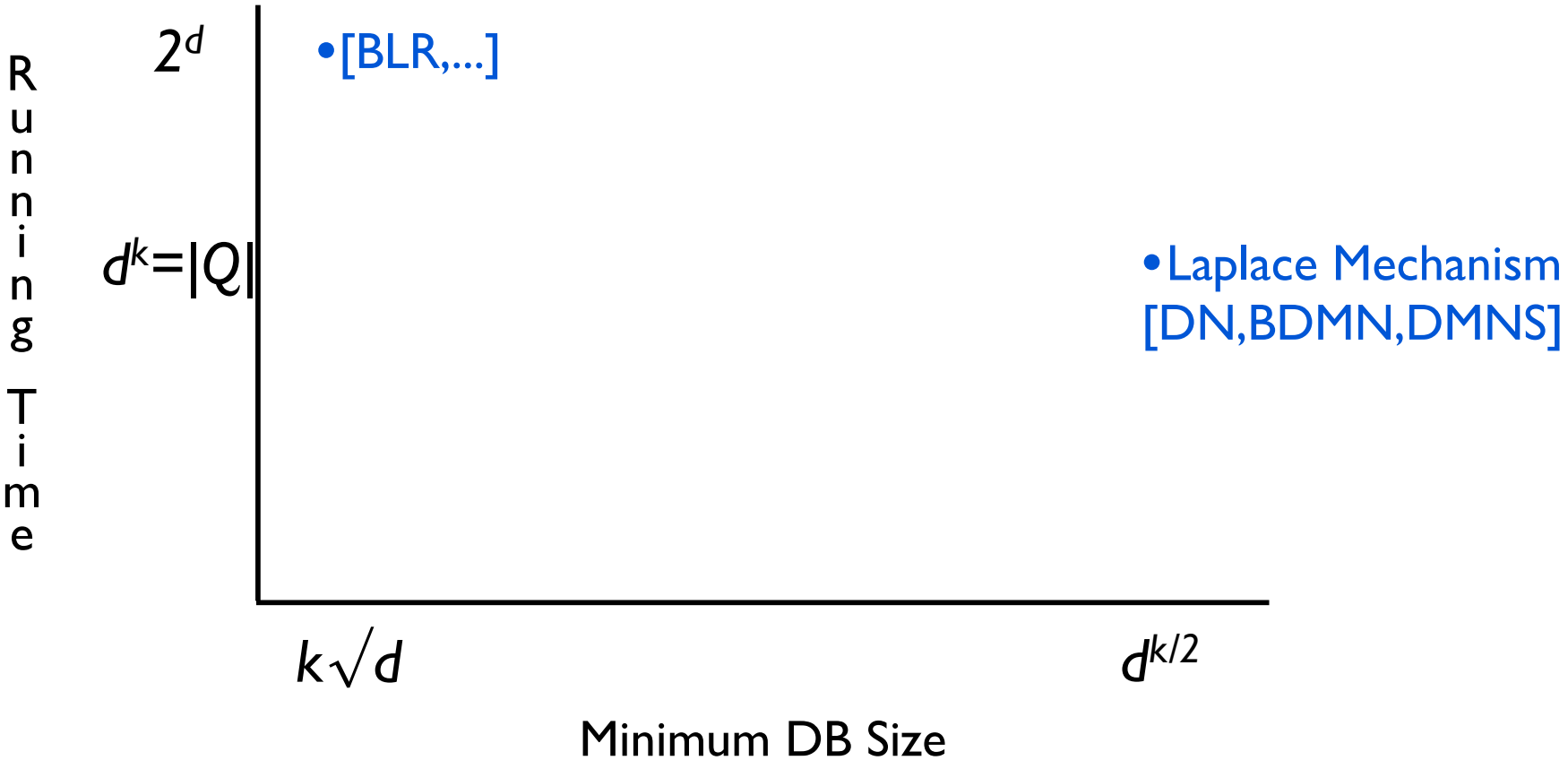
Prior Work on Marginals



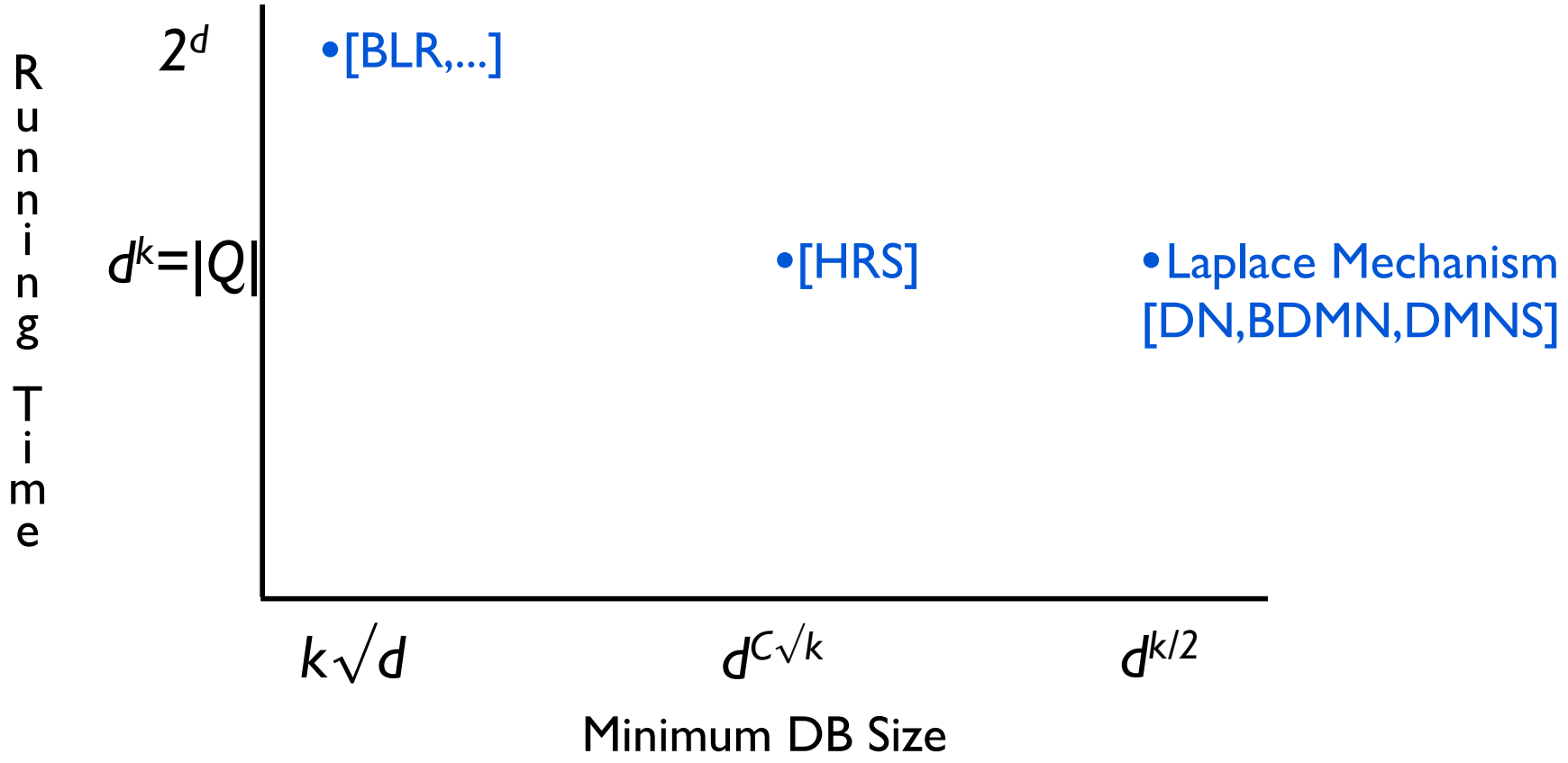
Prior Work on Marginals

- Laplace Mechanism [DN,BDMN,DMNS]

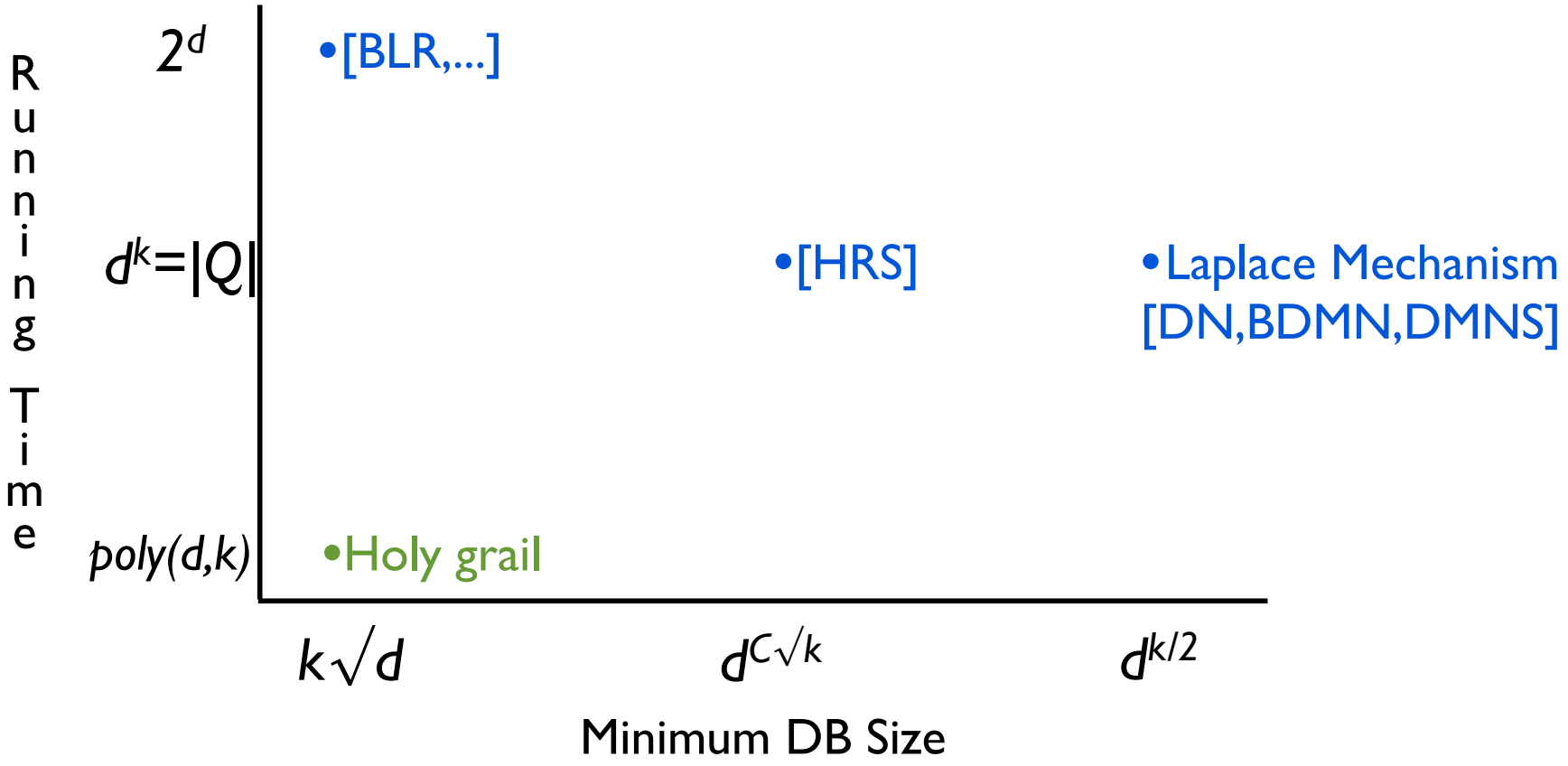
Prior Work on Marginals



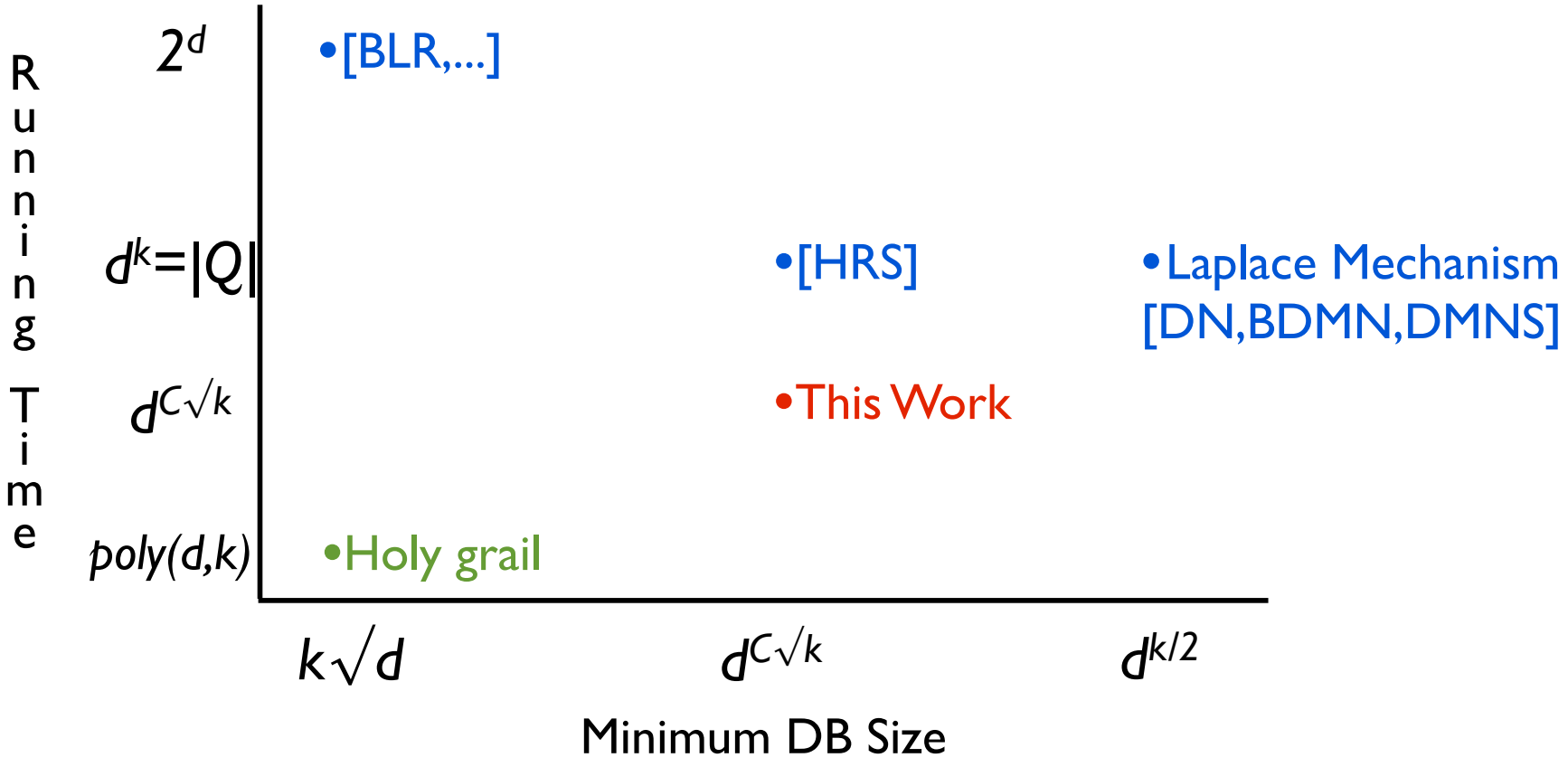
Prior Work on Marginals



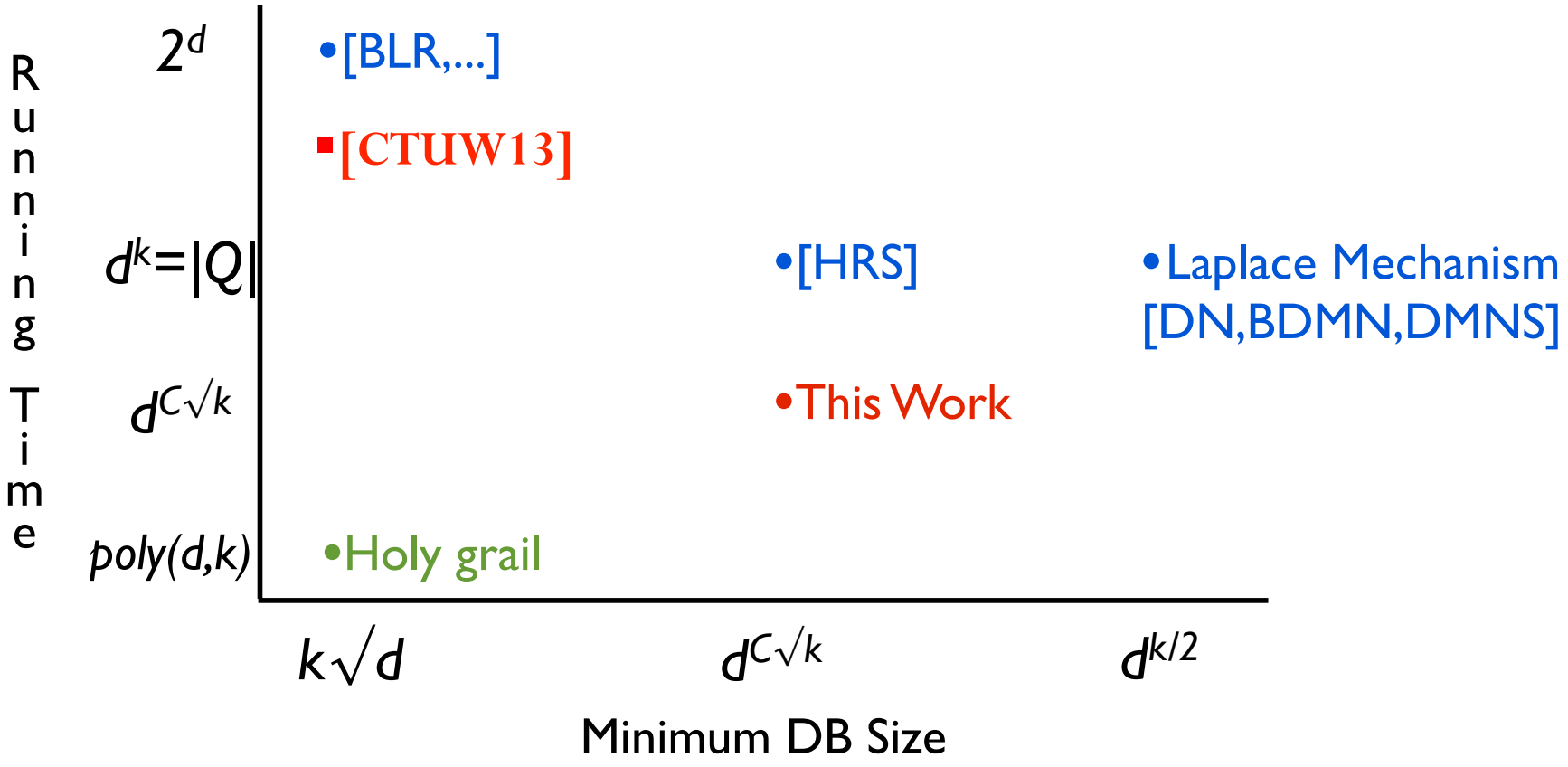
Prior Work on Marginals



Our Result [TUV, ICALP12]



Ongoing Work [CTUW13]



Thank you!

What About "Sparse" Streams? [CCGT13]

- Many streams are over enormous domain sizes (e.g. IPv6 flows)
  - Existing results depend domain size.
  - Want costs to depend on stream length instead.
- Idea: Domain reduction.
  - Ask P to provide 'perfect' hash function g mapping huge domain to small one.
  - Challenges: ensuring collisions in remapping do not cause errors (need a way for V to 'detect' collisions under g).
  - New protocols that allow P to 'correct' collisions online.
- Bottom line [CCGT13, to be submitted]: near-optimal tradeoffs in terms of stream length for frequency moments, graph problems, etc.

A Final Result: MatMult [T13]

- Let A be **any** time t, space s algorithm for n x n MatMult.
- New MatMult protocol:
  - P takes time $t + O(n^2)$ and space $s + o(n^2)$.
  - Optimal runtime up to leading constant assuming no $O(n^2)$ time algorithm for MatMult.

Problem Size	Naïve MatMult Time	Additional P time	V Time	Rounds	Protocol Comm
1024 x 1024	2.17 s	0.03 s	0.67 s	11	264 bytes
2048 x 2048	18.23 s	0.13 s	2.89 s	12	288 bytes