

Efficient Interactive Proofs for Linear Algebra

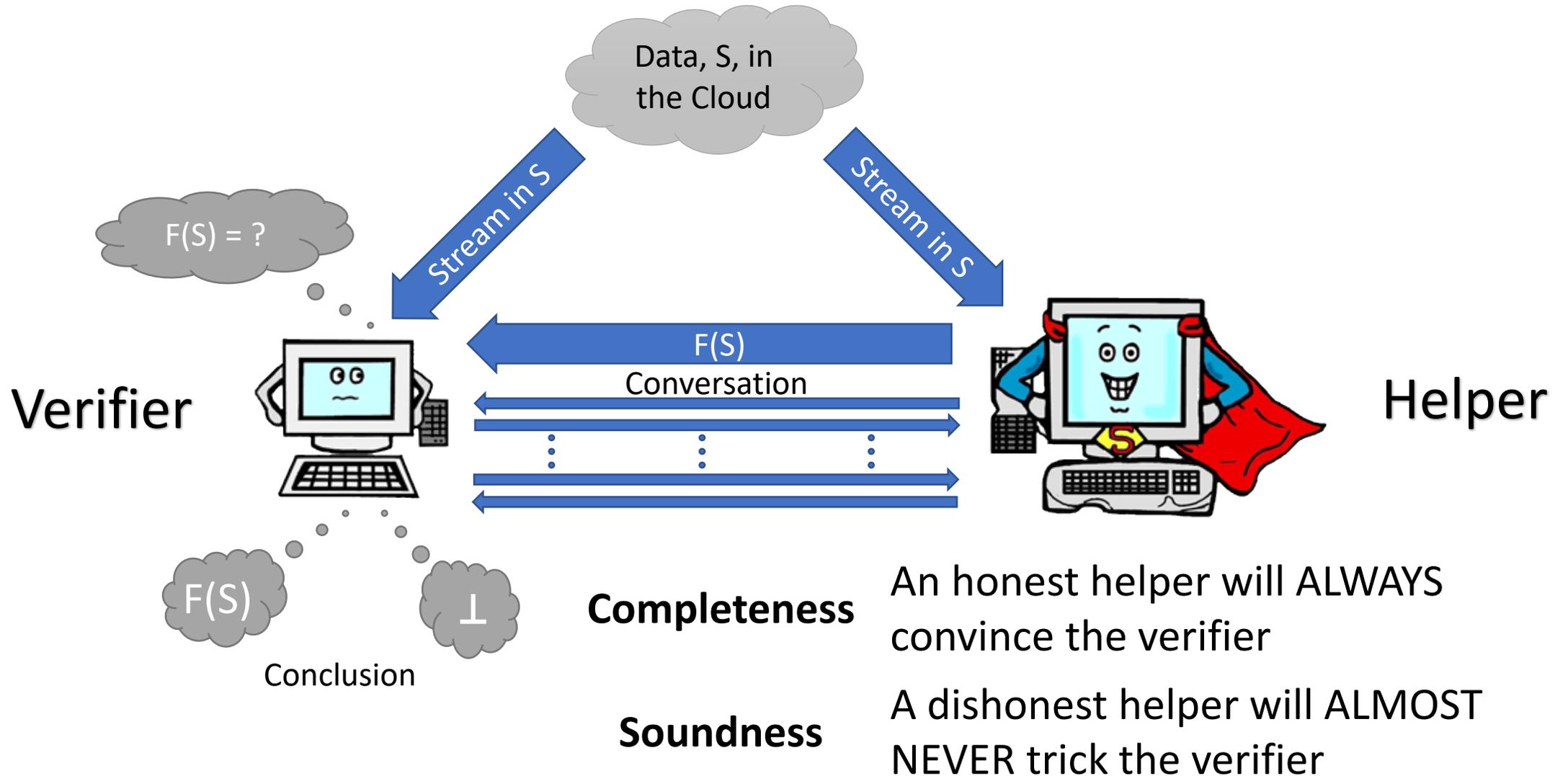
Christopher Hickey

Graham Cormode

University of Warwick

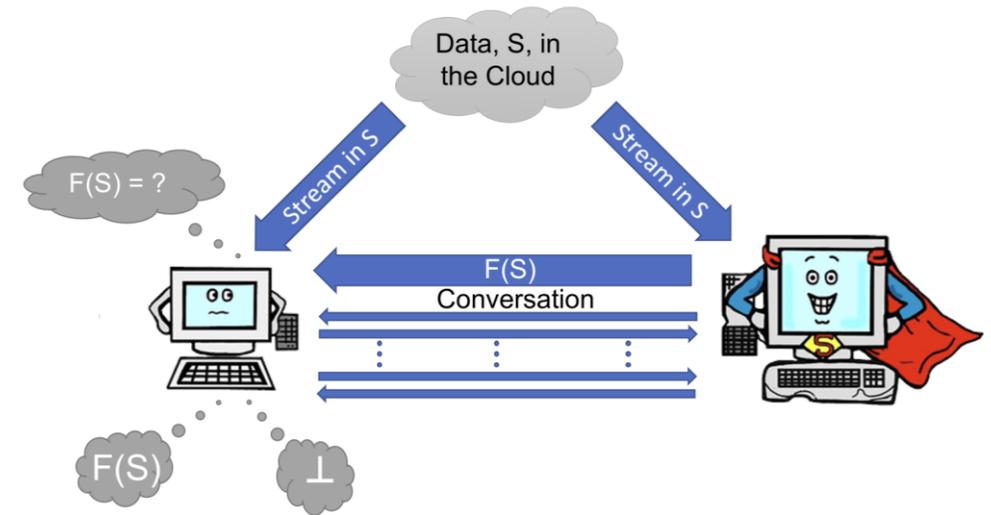
2202	2226	2256	2283	2310	2337	2364	2391	2418	2445	2472	2499	1	28	55	82	109	136	163	190	217	244	271	298	325	352	379	406	433	460	487	514	541	568	595	622	649	676	703	730	757	784	811	838	865	892	919	946	973	1000	1027	1054	1081	1108	1135	1162	1189	1216	1243	1270	1297	1324	1351	1378	1405	1432	1459	1486	1513	1540	1567	1594	1621	1648	1675	1702	1729	1756	1783	1810	1837	1864	1891	1918	1945	1972	1999	2026	2053	2080	2107	2134	2161	2188	2215	2242	2269	2296	2323	2350	2377	2404	2431	2458	2485	2512	2539	2566	2593	2620	2647	2674	2701	2728	2755	2782	2809	2836	2863	2890	2917	2944	2971	2998	3025	3052	3079	3106	3133	3160	3187	3214	3241	3268	3295	3322	3349	3376	3403	3430	3457	3484	3511	3538	3565	3592	3619	3646	3673	3700	3727	3754	3781	3808	3835	3862	3889	3916	3943	3970	3997	4024	4051	4078	4105	4132	4159	4186	4213	4240	4267	4294	4321	4348	4375	4402	4429	4456	4483	4510	4537	4564	4591	4618	4645	4672	4699	4726	4753	4780	4807	4834	4861	4888	4915	4942	4969	4996	5023	5050	5077	5104	5131	5158	5185	5212	5239	5266	5293	5320	5347	5374	5401	5428	5455	5482	5509	5536	5563	5590	5617	5644	5671	5698	5725	5752	5779	5806	5833	5860	5887	5914	5941	5968	5995	6022	6049	6076	6103	6130	6157	6184	6211	6238	6265	6292	6319	6346	6373	6400	6427	6454	6481	6508	6535	6562	6589	6616	6643	6670	6697	6724	6751	6778	6805	6832	6859	6886	6913	6940	6967	6994	7021	7048	7075	7102	7129	7156	7183	7210	7237	7264	7291	7318	7345	7372	7399	7426	7453	7480	7507	7534	7561	7588	7615	7642	7669	7696	7723	7750	7777	7804	7831	7858	7885	7912	7939	7966	7993	8020	8047	8074	8101	8128	8155	8182	8209	8236	8263	8290	8317	8344	8371	8398	8425	8452	8479	8506	8533	8560	8587	8614	8641	8668	8695	8722	8749	8776	8803	8830	8857	8884	8911	8938	8965	8992	9019	9046	9073	9100	9127	9154	9181	9208	9235	9262	9289	9316	9343	9370	9397	9424	9451	9478	9505	9532	9559	9586	9613	9640	9667	9694	9721	9748	9775	9802	9829	9856	9883	9910	9937	9964	9991	10018	10045	10072	10099	10126	10153	10180	10207	10234	10261	10288	10315	10342	10369	10396	10423	10450	10477	10504	10531	10558	10585	10612	10639	10666	10693	10720	10747	10774	10801	10828	10855	10882	10909	10936	10963	10990	11017	11044	11071	11098	11125	11152	11179	11206	11233	11260	11287	11314	11341	11368	11395	11422	11449	11476	11503	11530	11557	11584	11611	11638	11665	11692	11719	11746	11773	11800	11827	11854	11881	11908	11935	11962	11989	12016	12043	12070	12097	12124	12151	12178	12205	12232	12259	12286	12313	12340	12367	12394	12421	12448	12475	12502	12529	12556	12583	12610	12637	12664	12691	12718	12745	12772	12799	12826	12853	12880	12907	12934	12961	12988	13015	13042	13069	13096	13123	13150	13177	13204	13231	13258	13285	13312	13339	13366	13393	13420	13447	13474	13501	13528	13555	13582	13609	13636	13663	13690	13717	13744	13771	13798	13825	13852	13879	13906	13933	13960	13987	14014	14041	14068	14095	14122	14149	14176	14203	14230	14257	14284	14311	14338	14365	14392	14419	14446	14473	14500	14527	14554	14581	14608	14635	14662	14689	14716	14743	14770	14797	14824	14851	14878	14905	14932	14959	14986	15013	15040	15067	15094	15121	15148	15175	15202	15229	15256	15283	15310	15337	15364	15391	15418	15445	15472	15499	15526	15553	15580	15607	15634	15661	15688	15715	15742	15769	15796	15823	15850	15877	15904	15931	15958	15985	16012	16039	16066	16093	16120	16147	16174	16201	16228	16255	16282	16309	16336	16363	16390	16417	16444	16471	16498	16525	16552	16579	16606	16633	16660	16687	16714	16741	16768	16795	16822	16849	16876	16903	16930	16957	16984	17011	17038	17065	17092	17119	17146	17173	17200	17227	17254	17281	17308	17335	17362	17389	17416	17443	17470	17497	17524	17551	17578	17605	17632	17659	17686	17713	17740	17767	17794	17821	17848	17875	17902	17929	17956	17983	18010	18037	18064	18091	18118	18145	18172	18199	18226	18253	18280	18307	18334	18361	18388	18415	18442	18469	18496	18523	18550	18577	18604	18631	18658	18685	18712	18739	18766	18793	18820	18847	18874	18901	18928	18955	18982	19009	19036	19063	19090	19117	19144	19171	19198	19225	19252	19279	19306	19333	19360	19387	19414	19441	19468	19495	19522	19549	19576	19603	19630	19657	19684	19711	19738	19765	19792	19819	19846	19873	19900	19927	19954	19981	20008	20035	20062	20089	20116	20143	20170	20197	20224	20251	20278	20305	20332	20359	20386	20413	20440	20467	20494	20521	20548	20575	20602	20629	20656	20683	20710	20737	20764	20791	20818	20845	20872	20899	20926	20953	20980	21007	21034	21061	21088	21115	21142	21169	21196	21223	21250	21277	21304	21331	21358	21385	21412	21439	21466	21493	21520	21547	21574	21601	21628	21655	21682	21709	21736	21763	21790	21817	21844	21871	21898	21925	21952	21979	22006	22033	22060	22087	22114	22141	22168	22195	22222	22249	22276	22303	22330	22357	22384	22411	22438	22465	22492	22519	22546	22573	22600	22627	22654	22681	22708	22735	22762	22789	22816	22843	22870	22897	22924	22951	22978	23005	23032	23059	23086	23113	23140	23167	23194	23221	23248	23275	23302	23329	23356	23383	23410	23437	23464	23491	23518	23545	23572	23599	23626	23653	23680	23707	23734	23761	23788	23815	23842	23869	23896	23923	23950	23977	24004	24031	24058	24085	24112	24139	24166	24193	24220	24247	24274	24301	24328	24355	24382	24409	24436	24463	24490	24517	24544	24571	24598	24625	24652	24679	24706	24733	24760	24787	24814	24841	24868	24895	24922	24949	24976	25003	25030	25057	25084	25111	25138	25165	25192	25219	25246	25273	25300	25327	25354	25381	25408	25435	25462	25489	25516	25543	25570	25597	25624	25651	25678	25705	25732	25759	25786	25813	25840	25867	25894	25921	25948	25975	26002	26029	26056	26083	26110	26137	26164	26191	26218	26245	26272	26299	26326	26353	26380	26407	26434	26461	26488	26515	26542	26569	26596	26623	26650	26677	26704	26731	26758	26785	26812	26839	26866	26893	26920	26947	26974	27001	27028	27055	27082	27109	27136	27163	27190	27217	27244	27271	27298	27325	27352	27379	27406	27433	27460	27487	27514	27541	27568	27595	27622	27649	27676	27703	27730	27757	27784	27811	27838	27865	27892	27919	27946	27973	28000	28027	28054	28081	28108	28135	28162	28189	28216	28243	28270	28297	28324	28351	28378	28405	28432	28459	28486	28513	28540	28567	28594	28621	28648	28675	28702	28729	28756	28783	28810	28837	28864	28891	28918	28945	28972	29000	29027	29054	29081	29108	29135	29162	29189	29216	29243	29270	29297	29324	29351	29378	29405	29432	29459	29486	29513	29540	29567	29594	29621	29648	29675	29702	29729	29756	29783	29810	29837	29864	29891	29918	29945	29972	30000	30027	30054	30081	30108	30135	30162	30189	30216	30243	30270	30297	30324	30351	30378	30405	30432	30459	30486	30513	30540	30567	30594	30621	30648	30675	30702	30729	30756	30783	30810	30837	30864	30891	30918	30945	30972	31000	31027	31054	31081	31108	31135	31162	31189	31216	31243	31270	31297	31324	31351	31378	31405	31432	31459	31486	31513	31540	31567	31594	31621	31648	31675	31702	31729	31756	31783	31810	31837	31864	31891	31918	31945	31972	32000	32027	32054	32081	32108	32135	32162	32189	32216	32243	32270	32297	32324	32351	32378	32405	32432	32459	32486	32513	32540	32567	32594	32621	32648	32675	32702	32729	32756	32783	32810	32837	32864	32891	32918	32945	32972	33000	33027	33054	33081	33108	33135	33162	33189	33216	33243	33270	33297	33324	33351	33378	33405	33432	33459	33486	33513	33540	33567	33594	33621	33648	33675	33702	33729
------	------	------	------	------	------	------	------	------	------	------	------	---	----	----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

Our Model – Streaming Interactive Proofs



Costs in SIPs

Interactivity	Number of rounds
Verifier Memory	Working memory of the verifier
Communication	Total communication sent in both directions
Verifier's Streaming Cost	Computational complexity of streaming in S
Verifier's Checking Cost	Computational complexity of streaming the conversation
Helper Overhead	Additional work required by the helper beyond solving the problem



What Costs to Trade Off

“Non-interactive” costs
Verifier Memory
Verifier's Streaming Cost
Helper Overhead

“Interactive” costs
Communication
Interactivity
Verifier's Checking Cost

Rule of thumb: Decreasing a non-interactive cost usually increases some interactive cost, and vice-versa.

Our work attempts to see which cost is best to relax in order to minimize the **total time** of the protocol.

We focus on linear algebra, as this is a primitive for many problems, and yields interesting examples.

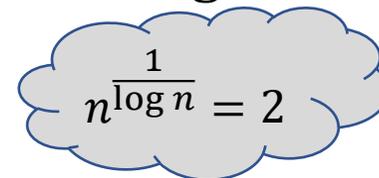
Warm-up: Inner Product

For two vectors of length n , ignoring constant factors.

	Method	This Work	[CTY11] Binary Sum-Check	[CMT12] FFT and LDEs
“Non-interactive” costs	Total Communication	$dn^{1/d}$	$\log n$	\sqrt{n}
	Verifier Checking Cost	$dn^{1/d}$	$\log n$	\sqrt{n}
	Rounds	$d - 1$	$\log n$	1
“interactive” costs	Helper Overhead	$n \log n$	$n \log n$	$n \log n$
	Verifier Streaming Cost	$dn^{1+1/d}$	$n \log n$	$n\sqrt{n}$
	Verifier Memory	$d + n^{1/d}$	$\log n$	\sqrt{n}

d is a variable parameter from 1 to $\log n$ determining the number of rounds.

Note that if we set $d = 2$, we get [CMT12], and if we set $d = \log n$ we get [CTY11].


$$n^{\frac{1}{\log n}} = 2$$

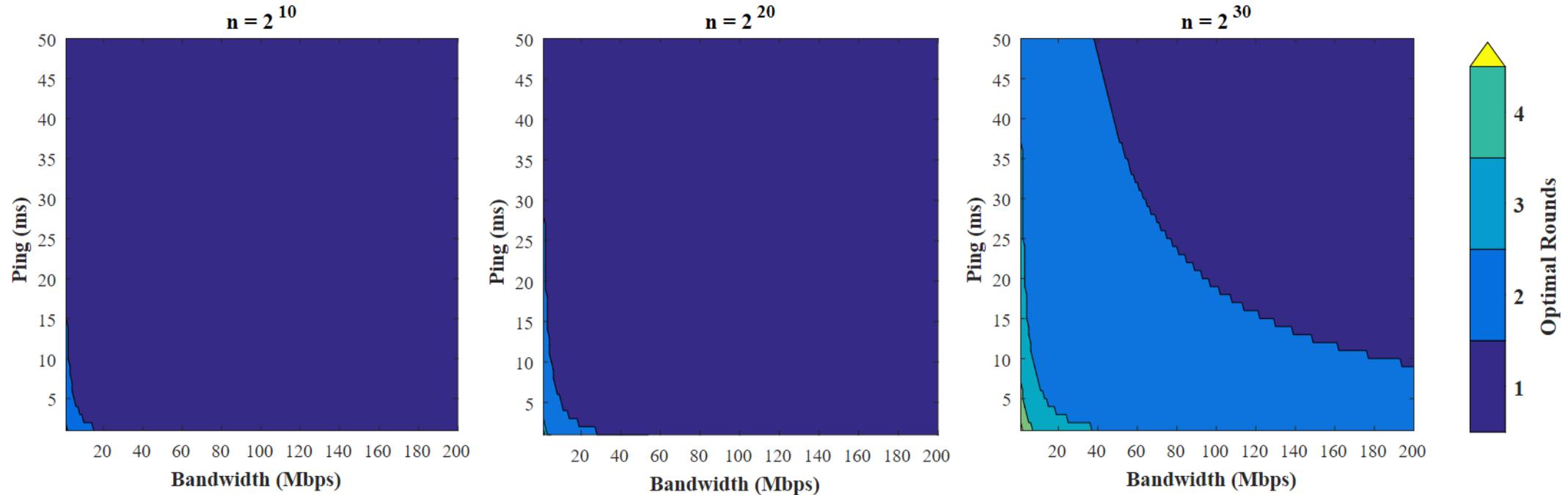
Matrix Multiplication

For two matrices of size $n \times n$, ignoring constant factors.

	Method	This Work	[Thaler13] Binary Sum Check	[CH18] Fingerprints
“Non-interactive” costs	Total Communication	$dn^{2/d}$	$\log n$	n^2
	Verifier Checking Cost	$n^2 + dn^{2/d}$	$n^2 + \log n$	n^2
	Rounds	d	$1 + \log n$	1
“interactive” costs	Helper Overhead	$n^2 \log n$	$n^2 \log n$	1
	Verifier Streaming Cost	$dn^{2+2/d}$	$n^2 \log n$	$n^2 \log n$
	Verifier Memory	$dn^{2/d}$	$\log n$	1

d is a variable parameter from 1 to $\log n$ determining the number of rounds.

Motivation: Minimizing Total Time Taken



Number of rounds considering *only* communication for Matrix Multiplication that decreases the total time to send all the data over all the rounds.

Less interactivity, even with more communication reduces overall time!

The question is now how much does this affect the other overheads?

Problem: Given streaming access to two data sets, how can we check they're the same (with high probability)?



Solution: Low Degree Extensions!

Consider a polynomial which passes through each data point (i, v_i) .

We index the data via a hypercube $[l]^d$ and create the **unique** polynomial of degree l in d variables that passes through each data point.

We can evaluate this LDE at a random point in \mathbb{F}^d as we stream the data!

LDEs share many useful properties,

- The probability of two different vectors having the same LDE evaluation at a random point is very small
- LDEs have linearity
- They can be constructed in $O(nld)$ time

LDEs can be used with the powerful sum-check protocol [LFKN92] to sum a function of the elements in a data set.

LDEs are very useful for making efficient protocols for inner product and matrix multiplication that use $d = \log n$ and $l = 2$.

Problem: Given $u, v \in \mathbb{F}^n$, how can we check the inner product $u^T v$? [CTY11]

[CTY11] uses LDEs with $n = l^d$, we represent the d -variate LDE of u by \tilde{u} and v by \tilde{v} . We want to find

$$u^T v = \sum_{i=1}^n u_i v_i = \sum_{k_1=0}^{l-1} \cdots \sum_{k_d=0}^{l-1} \tilde{u}(k_1, \dots, k_d) \tilde{v}(k_1, \dots, k_d)$$

They use a well-known protocol called ‘sum-check’ [LFKN92], a d -round protocol in which the prover allows the verifier to check the following sum against a ‘secret’ constructed in the streaming phase $\tilde{u}(r_1, \dots, r_d) \tilde{v}(r_1, \dots, r_d)$.

The messages the prover sends are degree $2l$ polynomials, which the prover can create in time $O(nld)$.

Problem: How were LDEs used to solve inner product? [CTY11]

The protocol uses sum-check, this is a d -round protocol involving d messages of $2l$ field elements.

Classification	Cost (ignoring constant factors)	Explanation
Interactivity	d	d rounds
Verifier Memory	$l + d$	Needs to store r , and l evaluations of g_j
Communication	ld	d messages of $2l$ field elements
Verifier's Streaming Cost	$dn^{1+1/d}$	Evaluating $\tilde{u}(r_1, \dots, r_d)\tilde{v}(r_1, \dots, r_d)$
Verifier's Checking Cost	ld	l evaluations of g_j , d times
Helper Overhead	nld	Forming g_j for j in $[1, d]$

[CTY11] note that using $l = 2$ and $d = \log n$ minimizes many costs, but with the cost of maximum interactivity.

Problem: How can we make [CTY11] variable-round without sacrificing Helper Overhead?

[CMT12] introduced a non-interactive protocol that massively reduced the helper overhead to $n \log n$ where the prover uses convolutions and fast fourier transforms.

We generalize this result to variable round protocols, as well as implementing a ‘stop-short’ reduction in sum-check to allow the protocol to run in $d - 1$ rounds.

Note that even with this adaptation, the memory efficient method is to use $d = \log n$. We aim to show experimentally that in practice, it’s often most time efficient to use as much memory as you have available.

However, the main motivation behind this protocol is how we can use it as a primitive for other protocols.

Problem: Vector-Matrix-Vector Multiplication

A first example of how to use this primitive is a nifty algebraic trick for multiplying two vectors $u, v \in \mathbb{F}^n$ and $A \in \mathbb{F}^{n \times n}$ we can verify $u^T A v$ by considering

$$u^T A v = \sum_{i=1}^n \sum_{j=1}^n u_i A_{ij} v_j = (uv^T)_{vec} \cdot A_{vec}$$

Where the subscript $_{vec}$ refers to a canonical transformation from a matrix to a vector.

Using the inner product protocol on the LDEs of A and uv^T gives us a protocol with communication and space costs $O(l^2 d)$ and d rounds.

Note we can use the inner product protocol as we can construct $\widetilde{uv^T}(r_1, r_2)$ using $\tilde{u}(r_1)\tilde{v}(r_2)$.

Problem: Matrix Multiplication

For matrices $A, B \in \mathbb{F}^{n \times n}$ we will have to verify that a sent **matrix** is correct, not just a scalar.

[Thaler13] uses LDEs for verification, and uses $\log n$ rounds and the inner product definition of matrix multiplication.

We use fingerprints in conjunction with our inner product protocol, however implement the outer-product definition of matrix multiplication.

For a vector $v \in \mathbb{F}^n$, the fingerprint of v with respect to $x \in_R \mathbb{F}$ is:

$$f_x(v) = \sum_{i=0}^{n-1} v_i x^i$$

Fingerprints have the property $f_x(u^T v) = f_x(u) f_x(v)$ [CH18].

We define fingerprints for matrices analogously.

Problem: Matrix Multiplication

For matrices $A, B \in \mathbb{F}^{n \times n}$ we will have to verify that a sent matrix is correct, not just a scalar.

Fingerprints are useful with the following identity

$$f_x(AB) = \sum_{i=1}^n f_{x^n}(A_i^\downarrow) f_x(B_i^\rightarrow) = \begin{pmatrix} f_{x^n}(A_1^\downarrow) \\ \vdots \\ f_{x^n}(A_n^\downarrow) \end{pmatrix} \cdot (f_x(B_1^\rightarrow) \quad \cdots \quad f_x(B_n^\rightarrow))$$

To use our inner product protocol, the verifier simply needs to be able to find the LDE of these two vectors at a random point, which it can using the linearity of fingerprints and LDEs.

Practical Analysis – Matrix Multiplication

Classification	Costs	How we'll time it	
Interactivity	d	The latency between each machine \times number of rounds \times 2	
Verifier Memory	$dn^{2/d}$	n/a	
Communication	$dn^{2/d}$	The bandwidth to send all the messages \times communication	
Verifier's Streaming Cost	$dn^{2+2/d}$	We will not time this, as it happens concurrently to seeing the data, which can happen at any point prior to the protocol starting.	
Verifier's Checking Cost	$n^2 + dn^{2/d}$	The time to fingerprint the matrix	The time taken to run the interactive protocol
Helper Overhead	$n^2 \log n$	The cost of producing the sum-check polynomials.	

Practical Analysis – Matrix Multiplication

Using bandwidth of 100Mbps And Latency of 20ms			Interactivity	Communication	Verifier's Checking cost	Verifier's Checking cost	Helper overhead
Matrix Size ($n = l^d$)	l	d	Latency (ms)	Bandwidth (ms)	Fingerprinting AB (ms)	Interactive Stage (ms)	Forming messages (ms)
2^{12}	2	12	440	0.014	150	0.009	0.23
	8	4	120	0.015	150	0.035	0.10
	64	2	40	0.041	150	0.043	0.11
2^{16}	2	16	600	0.019	40000	0.006	3.50
	16	4	120	0.031	40000	0.046	1.60
	256	2	40	0.163	40000	1.700	1.80
2^{18}	2	18	680	0.022	600000	0.006	14.10
	8	6	200	0.026	600000	0.030	6.30
	512	2	40	0.328	600000	6.400	7.80

Practical Analysis – Matrix Multiplication

Using bandwidth of 100Mbps And Latency of 20ms			Interactivity	Communication	Verifier's Checking cost	Verifier's Checking cost	Helper overhead
Matrix Size ($n = l^d$)	l	d	Latency (ms)	Bandwidth (ms)	Fingerprinting AB (ms)	Interactive Stage (ms)	Forming messages (ms)
2^{12}	2	12	440	0.014	150	0.009	0.23
	8	4	120	0.015	150	0.035	0.10
	64	2	40	0.041	150	0.043	0.11
2^{16}	2	16	600	0.019	40000	0.006	3.50
	16	4	120	0.031	40000	0.046	1.60
	256	2	40	0.163	40000	1.700	1.80
2^{18}	2	18	680	0.022	600000	0.006	14.10
	8	6	200	0.026	600000	0.030	6.30
	512	2	40	0.328	600000	6.400	7.80

This is independent of interactivity!

Practical Analysis – Matrix Multiplication

Using bandwidth of 100Mbps And Latency of 20ms			Interactivity	Communication	Verifier's Checking cost	Verifier's Checking cost	Helper overhead
Matrix Size ($n = l^d$)	l	d	Latency (ms)	Bandwidth (ms)	Fingerprinting AB (ms)	Interactive Stage (ms)	Forming messages (ms)
2^{12}	2	12	440	0.014	150	0.009	0.23
	8	4	120	0.015	150	0.035	0.10
	64	2	40	0.041	150	0.043	0.11
2^{16}	2	16	600	0.019	40000	0.006	3.50
	16	4	120	0.031	40000	0.046	1.60
	256	2	40	0.163	40000	1.700	1.80
2^{18}	2	18	680	0.022	600000	0.006	14.10
	8	6	200	0.026	600000	0.030	6.30
	512	2	40	0.328	600000	6.400	7.80

The latency dominates the other costs significantly, and this would still be the case even with a latency of 5ms.

This clearly demonstrates the location of the **time** bottle-neck in this protocol.

Interactivity and verifier memory

- The time bottleneck is the latency between the verifier and the prover, dominating the other costs that decrease with increased interactivity.
- This leads us to want to reduce the interactivity as much as the verifier's memory ($O(l^2 d)$) will let us.
- For example, for $n = 2^{18}$, optimality will likely be with a 6 round protocol.

For a matrix of size $n = 2^{18}$		
l	d	$l^2 d$
2	18	72
4	9	144
8	6	384
64	3	12288
512	2	524288

Closing Thoughts

- For our applications, where the problem is highly structured, the interactive protocols are very efficient.
- By adapting [CMT12]'s FFT protocol for binary sum-check to arbitrary sum-check, we achieve faster protocols than previously possible.
- We demonstrate how using certain applications are better with LDEs and some with fingerprints, and show some useful algebraic tricks to apply.
- Latency is the dominant time bottleneck.
- The most efficient protocol for the verifier will be to use as much memory as possible, even though the asymptotics say more interactivity is better.
- A large cost for the verifier is the initial streaming phase. Additional work could be done to uncover efficiency tricks to find the secret.

Any Questions? Email C.Hickey@warwick.ac.uk