Dark Matter Implementation

Configuration:
Platform: Ubuntu Server 18.04
Instance Type: t2.medium
Virtualization: hvm
vCPUs: 2
Memory: 4GB
Compiler: G++ 7.5
Code runs 1000 runs each time, time in document per run
Running environment: Amazon AWS

Code building notes:

Currently, the flags needed to run the program are in the mains.hpp file.

Options:

PACKED_PRF_CENTRAL = 1. - centralized packed PRF, both phases 2 and 3 are packed (no lookup table), key is Toeplitz:

UNPACKED_PRF_CENTRALIZED = 1. - Centralized naïve version unpacked

PACKED_PRF_CENTRAL_LOOKUP = 1 , centralized using lookup table

TEST_PRF= 1, Distributed dark matter version, packed, no lookup table

TEST_NP = 1, New protocol, packed, no lookup table

TEST_NP_LOOKUP = 1 , New protocol with lookup table

Building the code:

g++ -std=c++14 -O3 -o pDarkMatterPRF -I include/darkmatter/ src/*.cpp tests/*.cpp

Runtimes:
Runtime executed on Amazon AWS:

Centralized version:

Centralized PRF Implementation (Using Packing)

Phase	10^{-6} sec (μs)	Rounds/sec	Macbook Air (μs)
P1 (K * X)	3.16	$0.37 * 10^6$	4.8
P2	0.62	$2 * 10^6$	0.09
P3 (Mult by 81x256 Rand mat)	12.07	$0.082 * 10^6$	136.70
Full Protocol	18.5	65,400	142.70

Centralized PRF Implementation (Naïve/ Unpacked implementation)

Phase	10^{-6} sec (AWS)	Rounds/s (AWS)	Macbook Air
P1 (K * X)	2.52	~400K	3.889
Unpacking of 81 X 256 randomization matrix	0.23		0.356
P3 (Rmat* (K*X))	15.39	~65K	22.354
Full Protocol	20.19	~50K	28.448

Centralized PRF Implementation (Using Packing+ Lookup Table)

Phase	10^{-6} sec (AWS)	Rounds/s (AWS)	Local Macbook Air
Calling Lookup function	1.84	~544K	14.786
Full Protocol	6.08	~165K	21.188

Distributed version:

Notes: Preprocessing are excluded from timings.

Distributed Dark Matter PRF Implementation

Phases	AWS (μs)	Macbook Air(μs)
AX + B (Party 1)	10.72	14.65
AX + B (Party 2)	11.37	10.80
Phase 1(Total)	22.08	
Share Conversion (Party 1)	2.92	5.05
Share Conversion (Party 2)	3.87	2.83
Phase 2(Total)	6.80	
Phase 3 (Randomization)	23.73	285.22
PRF (entire PRF w/o preproc)	61.08	324.79

New Protocol (Z3 packing, no lookup table)

Phase	Phase Sub-Module	AWS Time(μs)	Number of Rounds/ Iterations	Macbook Air(μs)	Number of Rounds/ Iterations
Phase 1	**Party 1**	0.61	~811K	0.05	~100M
	Party 2	0.61		0.05	
	Mask	0.61		0.04	
	Total (phase 1)	1.23		0.10	
Phase 2	**Party 1**	6.09	~149K	3.38	~291K
	Party 2	6.09		3.38	
	Mask	0.59		0.04	
	Total (phase 2)	6.69		3.42	
Phase 3	**Party 1**	12.24	~81K	49.98	~19K
	Party 2	12.23		50.31	
	Total (phase 3)	12.24		50.31	
	Entire PRF	20.20	~49K	57.37	~17K

New Protocol (Z3 packing, LOOKUP TABLE)- Improvement expected

Phase	Phase Sub-Module	AWS Time(μs)	Number of Rounds/ Iterations	Macbook Air(μs)	Number of Rounds/ Iterations
Phase 1	**Party 1**	0.61	~821K	0.08	~6578K
	Party 2	0.60		0.06	
	Mask	0.60		0.07	
	Total (phase 1)	1.21		0.15	
Phase 2	**Party 1**	6.15	~148K	4.11	~239K
	Party 2	6.07		4.05	
	Mask	0.59		0.05	
	Total (phase 2)	6.75		4.16	
Phase 3	**Party 1**	6.20	~161K	15.05	~66K
	Party 2	6.05		14.68	
	Total (phase 3)	6.20		15.05	
	Entire PRF	14.17	~70K	19.37	~51K

Some rough preliminary insight (based on AWS):

1) Using lookup table improves the timing for phase 3 by ~50 %.
2) Using lookup table improves the overall time for executing new protocol by ~30%

Communication:
We can do back of the envelope estimates, but the actual communication time will depend on many factors. Below is a rough idea:

For the new protocol we have: ***each party sends 4N bits***.
For a 25 Mbps connection, we get 25*10^6/1024=~24K executions per sec

for 1 Gigabit, we get ~ 1M executions/sec

Synchronization is needed between the parties, though, so the assumption this is independent is an estimate.
Details: phase 1 - each party sends matrix K and vector X. K can be Toeplitz, so each party will send 3N bits. phase 2 - each party sends w' (N bits)