**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.

Protocol 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**

**TEST\_OPRF = 1, oPRF protocol no lookup table**

**define TEST\_INTEGER\_PACKING = 1, integer packing**

Test flags:

**UNIT\_LOOKUP //tests the lookup table implementation**

**TEST\_PHASE1 //test AX+B class**

**TEST\_PackedMod2 //test class packedMod2**

**TEST\_PackedMod3 //test class packedMod3**

**TEST\_SC //Not used anymore**

**UNITTEST\_ROUND2 //for dark matter protocol - runs the unit test code that checks round 2 only**

**OPRF\_PRINT\_VAL // test the OPRF**

**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** | **sec )** | **Rounds/sec** | **Macbook Air )** |
| **P1** (K \* X) | 3.16 |  | 4.8 |
| **P2** | 0.62 |  | 0.09 |
| **P3** (Mult by 81x256 Rand mat) | 12.07 |  | 136.70 |
| **Full Protocol** | 18.5 | 65,400 | 142.70 |

**Centralized PRF Implementation (Naïve/ Unpacked implementation)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Phase** | **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** | **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** | **Phase**  **Sub-Module** | **AWS )** | **Number of Rounds/**  **Iterations** | **Macbook Air)** | **Number of Rounds/**  **Iterations** |
| **AX + B** | **Party 1** | 10.25 | ~46K | 6.00 | ~86K |
| **Party 2** | 11.47 | 5.62 |
| **Total** | 21.73 | 11.63 |
| **Share Conversion** | **(Party** | 2.85 | ~149K | 1.95 | ~341K |
| **Party 2** | 3.81 | 0.97 |
| **Total** | 6.67 | 2.93 |
| **Phase 3** | **Randomization** | 12.15 | ~82K | 61.50 | ~16K |
|  | **Entire PRF** | 40.56 | ~24K | 76.07 | ~13K |

**New Protocol (Z3 packing, no lookup table)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Phase** | **Phase**  **Sub-Module** | **AWS Time)** | **Number of Rounds/**  **Iterations** | **Macbook Air)** | **Number of Rounds/**  **Iterations** |
| **Phase 1(K’,X’ comp)** | **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 (w’ comp)** | **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**  **(mux+Z3 Rand)** | **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)** | **Number of Rounds/**  **Iterations** | **Macbook Air)** | **Number of Rounds/**  **Iterations** |
| **Phase 1** | **Party 1** | 0.61 | ~813K | 0.08 | ~6578K |
| **Party 2** | 0.61 | 0.06 |
| **Mask** | 0.61 | 0.07 |
| **Total (phase 1)** | 1.22 | 0.15 |
| **Phase 2** | **Party 1** | 6.15 | ~148K | 4.11 | ~239K |
| **Party 2** | 6.07 | 4.05 |
| **Mask** | 0.60 | 0.05 |
| **Total (phase 2)** | 6.75 | 4.16 |
| **Phase 3** | **Party 1** | 4.14 | ~241K | 15.05 | ~66K |
| **Party 2** | 4.04 | 14.68 |
| **Total (phase 3)** | 4.14 | 15.05 |
|  | **Entire PRF** | 12.12 | ~82K | 19.37 | ~51K |

**Some rough preliminary insight (based on AWS):**

1. Using lookup table improves the timing for phase 3(new protocol) by roughly 65-70%
2. Using lookup table improves the overall time for executing new protocol by ~40-60%

**\*\**Updated 23-OPRF timing (Z3 packed, no lookup)****\*\**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Phase** | **Phase**  **Sub-Module** | **AWS Time)** | **Number of Rounds/**  **Iterations** | **MacBook Air)** | **Number of Rounds/**  **Iterations** |
| **Step 1**  **(masking the inputs)** | **Client** | 0.67 | ~741K | 0.17 | ~235K |
| **Server** | 0.67 | 0.20 |
| **Total (step 1)** | 1.35 | 0.40 |
| **Step 2**  **(compute w’)** | **Client** | 3.30 | ~245K | 2.28 | ~381K |
| **Server** | 3.49 | 2.57 |
| **Combine w** | 0.60 | 0.05 |
| **Total (step 2)** | 4.09 | 2.62 |
| **Step 3**  **(Randomization)** | **Client** | 12.38 | ~75K | 62.53 | ~15K |
| **Server** | 13.21 | 59.14 |
| **Total (step 3)** | 13.21 | 62.53 |
|  | **Entire PRF** | 18.66 | ~53K | 65.58 | ~15K |

**\*\**Updated 23-OPRF timing (Z3 packed, lookup table)****\*\**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Phase** | **Phase**  **Sub-Module** | **AWS Time)** | **Number of Rounds/**  **Iterations** | **MacBook Air)** | **Number of Rounds/**  **Iterations** |
| **Step 1**  **(masking the inputs)** | **Client** | 0.63 | ~1550K | 0.15 | ~4950K |
| **Server** | 0.64 | 0.20 |
| **Total (step 1)** | 0.64 | 0.20 |
| **Step 2**  **(compute w’)** | **Client** | 3.32 | ~247K | 1.95 | ~459K |
| **Server** | 3.43 | 2.14 |
| **Combine w** | 0.60 | 0.03 |
| **Total (step 2)** | 4.03 | 2.17 |
| **Step 3**  **(Randomization)** | **Client** | 4.02 | ~206K | 10.48 | ~82K |
| **Server** | 4.84 | 12.17 |
| **Total (step 3)** | 4.84 | 12.17 |
|  | **Entire PRF** | 9.52 | ~104K | 14.54 | ~68K |

Libsodium (modular exponentiation ed 25519 curve timing)- Amazon AWS

|  |  |
| --- | --- |
| **Time to execute ed25519**  **(microseconds) - 1 run** | **Number of operations per second** |
| 28.69 | ~35K |

**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)

\* time in μs is for single instance