Using GPU as Hardware Random Number Generator

Taeill Yoo, Yongjin Yeom
Department of Financial Information Security, Kookmin University, South Korea

Abstract

Random number generator (RNG) is expected to provide unbiased, unpredictable bit sequences. RNG collects entropy from the environments and accumulates it into the pool. Then from the entropy pool, RNG generates seed with high entropy as input to the cryptographic algorithm called pseudo-random bit generator. Since the lack of entropy sources leads the output random bits predictable, it is important to harvest enough entropy from physical noise sources. Here, we show that we can harvest sufficient entropy from the race condition in parallel computations on GPU. According to the entropy estimations in NIST SP 800-90B, we measure the entropy obtained from NVIDIA GPU such as GTX90 and GTX780. Our result can be combined with high speed random number generating library like cuRAND. To sum up, GPU can be used as hardware random number generator with physical entropy source.

What is RNG?

- Random number generator (RNG) produces sequences of random numbers. (An ideal model is a fair coin toss)
- Pseudo-random number generator (PRNG) is a deterministic algorithm to generate random sequences (possibly a part of RNG).
- Applications: cryptography, computer simulation, game, etc.
- Requirements: unbiased (the same number of 0 and 1), unpredictable, etc.

CUDA Code for Collecting Entropy

CUDA Source Code: Generating random noise using race conditions in GPU

```
/* Kernel function generating random noise */
__global__ void RaceCondition(int *devArray, int nSize, int nIteration)
{
    int tid = threadIdx.x; //get thread ID
devArray[tid] = 0; //initialization of array in global memory
__shared__ int sharedArray[ARRAY_SIZE]; //default: ARRAY_SIZE = 512
sharedArray[tid] = devArray[tid]; //initialize shared memory
__syncthreads(); //confirm the initialization
/* Update shared memory that gives rise to Race condition */
for(int i=0; i<nIteration; i++) {
    for(int j=0; j<nSize; j++) {
        sharedArray[tid]++;
    }
__syncthreads();
devArray[tid] = sharedArray[tid]; //copy to global memory
__syncthreads();
}
```

Conclusion and Future Work

Collecting sufficient entropy for cryptographic module is challenging particularly for software module. We have shown that we can use GPU as entropy source and performed entropy estimation according to new methodology NIST 800-90B. We are planning to implement remaining parts of RNG on GPU including cryptographic algorithms so that GPU works as completely independent RNG containing entropy source.

Expériences

- GPU: GTX780 (We successfully run the same experiments on 610M & 690, too)
- Array Size: 512 elements
- The number of iterations: 1,000
- The number of experiments: 5 (colored lines in the graph below)

References