```
#include <iostream>
#include <windows.h>
#include <stdio.h>
#include <time.h>
#include <chrono>
#include <vector>
#include <string.h>

#include "NTL/ZZ.h"

using namespace NTL;

int mod(int x, int y)
{
  int z;
  z = x - x / y * y;
  return z;
}

ZZ mod(ZZ x, ZZ y)
{
  ZZ z;
  z = x - x / y * y;
  return z;
}

std::string to_binary(int x)
{
    std::string binary;
    for (int i = 8; i > 0; i--)
      {
        if (x >= pow(2, i - 1))
        {
            binary += '1';
            x -= pow(2, i - 1);
        }
        else
        {
            binary += '0';
        }
    }
    return binary;
}

int main()
{

  ZZ cap(255);

   //Iniciar el timer

   auto start = std::chrono::high_resolution_clock::now();

   SYSTEM_INFO siSysInfo;

   GetSystemInfo(&siSysInfo);

   //Obtener información del hardware de la PC

   printf("Hardware info: \n");

   std::cout << std::endl;

   printf("  OEM ID: %u\n",
      siSysInfo.dwOemId);
   printf("  Number of processors: %u\n",
      siSysInfo.dwNumberOfProcessors);
   printf("  Active processor mask: %u\n",
      siSysInfo.dwActiveProcessorMask);
   printf("  Processor level: %u\n",
      siSysInfo.wProcessorLevel);
   printf("  Processor tipe: %u\n",
      siSysInfo.dwProcessorType);

   //Convertir los datos a NTL

   ZZ a(siSysInfo.dwOemId);
   ZZ b(siSysInfo.dwNumberOfProcessors);
   ZZ c(siSysInfo.dwActiveProcessorMask);
   ZZ d(siSysInfo.wProcessorLevel);
   ZZ e(siSysInfo.dwProcessorType);

   //Detener el timer

      auto finish = std::chrono::high_resolution_clock::now();

   //Mostrar el tiempo obtenido en nanosegundos

      std::cout << "\nTime: " << std::chrono::duration_cast<std::chrono::milliseconds>(finish - start).count() << " milliseconds\n";

   //Convertir el tiempo obtenido a NTL

   ZZ t(std::chrono::duration_cast<std::chrono::milliseconds>(finish - start).count());

   //Multiplicar los datos de la PC por el tiempo y añadirlos a un array

      //Sacar modulo para que el número no pase de 255

   a = mod((a*t), cap);
   b = mod((b*t), cap);
   c = mod((c*t), cap);
   d = mod((d*t), cap);
   e = mod((e*t), cap);

   ZZ Array[5] = {a, b, c, d, e};

   std::cout << std::endl;

   //Mostrar los números aleatorios

   for(int i = 0; i < 5; i++)
   {
     std::cout << "Random number " << i+1 << ": " << Array[i] << std::endl;
   }

   int aux_a = conv<int>(a);
   int aux_b = conv<int>(b);
   int aux_c = conv<int>(c);
   int aux_d = conv<int>(d);
   int aux_e = conv<int>(e);

   std::string k1 = to_binary(aux_a);
   std::string k2 = to_binary(aux_b);
   std::string k3 = to_binary(aux_c);
   std::string k4 = to_binary(aux_d);
   std::string k5 = to_binary(aux_e);

   std::string K[64]

   //std::string K[5] = {k1, k2, k3, k4, k5};
```

```cpp

  std::cout << std::endl;

  for(int i = 0; i < 5; i++)
  {
    std::cout << "K[" << i << "]: ";
    for(int j = 0; j < 8; j++)
    {
      std::cout << K[i][j];
    }
    std::cout << std::endl;
  }

  std::cout <<

  return 0;
}
```