## Extracting Random Numbers and Seeding

For users of Microsoft Windows based systems, the PowerShell scripting language lends itself nicely to extracting random numbers from the number sequences generated by BubblesRNG. There’s however a limitation in that only 8-bit numbers can be extracted. The following PowerShell statement can be used out of the box to extract random numbers from a BubblesRNG generated sequence. The *Encoding* and *TotalCount* parameters represent the format in which data is read and the number of elements that are retrieved.

get-content .\test.bin -Encoding byte -TotalCount 10

To counteract this shortcoming, we developed a Java based utility called *getRnd()*which allows for the extraction of random numbers in 8, 16 and 32-bit formats. Once extracted, the random numbers are written to a text file using any of three delimiting options (comma, tab and space). Numbers can also be grouped in a user-defined number of columns. The Apache Common CLI library was used to parse and extract values from the command line arguments passed to *getRnd()*. The command line syntax and parameters list is as follows;

Usage: getRnd -f BubblesRNG.bin -t 32 -n 100 –seed

-c,--columns <arg> Group numbers by columns (max. 100). Default: c=1.

-d,--delimiter <arg> Delimiter type; c|s|t. Default: d='c'.

-f,--file <arg> Path to the binary sequence.

-h,--help Displays this help screen.

-i,--interval <arg> Re-seeding interval. Default: i=1.

-n,--total <arg> Total random numbers to extract. Default: n=1.

-s,--seed Seeds java.util.Random(seed). Default: No seeding.

-t,--type <arg> The format random numbers are extracted in;

8|16|32-bit. Default t=32.

The next example illustrates the syntax used to extract a total of fifty 32-bit random numbers from a BubblesRNG sequence file called *full.bin.* The output produced is a text file containing the extracted numbers grouped in five columns and tab delimited.

java -jar getRnd.jar -f full.bin -t 32 -n 50 -c 5 -d t

The utility also calculates the total number of random numbers that can be extracted from a sequence given all three possible number formats.

File size : 1048576

--------------------------

8-bit integers: 131072

16-bit integers: 65536

32-bit integers: 32768

--------------------------

The output generated by the example above is read from left to right, top to bottom.

146004700 1570884684 101148796 3334730218 301983954

3915235253 464959936 1898817400 4080329725 3314872584

3253479498 716281763 121755955 739364632 4271908411

2700330592 2608546019 1606669841 3765454255 1003511656

3339519586 2883254895 3112614014 3320150479 238062338

3731511387 1738171679 4098748843 4239415908 868101763

2196646017 356560547 2650995520 434776146 575495389

511404252 2723138791 2915476830 4154400171 594098724

883659793 2002961631 4022908911 969730223 2000156279

426333581 917470649 478188794 2331659507 3745954999

We include a PRNG seeding mechanism to demonstrate one possible use of the random numbers generated by BubblesRNG. The idea is to reseed the[*java.util.Random(*)](https://docs.oracle.com/javase/8/docs/api/java/util/Random.html) PRNG with a succession of random numbers extracted from sequences generated by BubblesRNG. The seeding interval is user-specified by the *–i* parameter. The PRNG is reseeded each time the number of random numbers generated by the Java PRNG equals *i*. Of course, reseeding does not alter in any way the periodicity of the Java generator which in this case is 248. What it does instead, is lower the probability that an attacker, armed with a history of events, is able to reconstruct the original random number sequence. Since the seed is changed frequently, preferably using short intervals, predicting any sequence, part of or replaying it becomes a hard problem. Below is a sample of the random numbers generated by the Java PRNG seeded with random numbers extracted from a BubblesRNG sequence. We exclude any negative numbers by logically shifting the values obtained (>>>1). If the seeding interval is set to any value greater than 1, *getRnd* will generate a total of (*i* *\* n)* random numbers where *n* is the total numbers extracted from the BubblesRNG sequence*.* The last example we give, extracts fifty 32-bit random numbers from a BubblesRNG sequence. The *java.util.Random()* method is called a hundred times (*i* *\* n)* during which it is reseeded every two iterations using the next random number extracted from the BubblesRNG sequence.

java -jar getRnd.jar -f full.bin -t 32 -n 50 -seed -i 2

The output in this case is not subject to any formatting and is written to a text file as follows;

6753507432878163773

7412323887963921583

2256056214628047983

4025353074638300297

1515008337270824104

3384374943480587425

7178013228310328253

7244661813984290529

2947778930595663759

5376721863615762028

8222212374078203909

1196067133645019165

The source code to getRnd.java is listed under [Appendix G](#_getRnd.java).