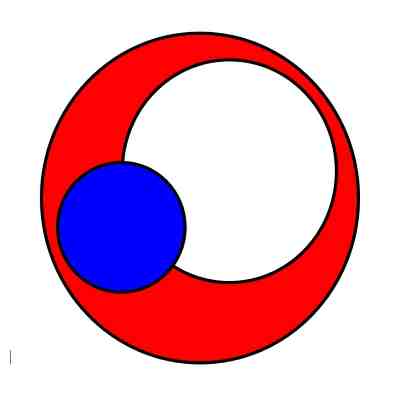
Synchronous random number generators PRNG

Translation into English (Google Translate)

Let's look at several ways to create software random number generators using external digital streams and without them. Let's write source code for Android Java - prototypes of applications showing the reality of using algorithms.



1. Generator of "random" numbers using expansion of source information, sampling and correction of source data. (**Czech Entropy** Application)

We will use TTS technology in this generator - converting the entered text into speech. This option is available in the Android operating system. TTS allows you to significantly increase the volume of data (30-100 thousand times), use different settings to obtain different digital results, ensure repeatability of the results (which makes it possible to build a PRNG), the transformation is “formulaless”. The disadvantages of the algorithm are obvious: the result is easily repeated (if you know the source text), large but not infinite expansion of the data, unstable level of entropy in the obtained data. Because the received data is a sound file, it may contain many repeating and dependent sequences of numbers.

Let's consider possible options for compensating for the shortcomings of such random number generators.

You can come up with (invent) quite a few algorithms for improving the results of such data expanders.

TTS conversion of the entered text several times with different parameters: sound quality, pronunciation speed, male/female voice, timbre, etc. With subsequent mixing of data from several sound files. The complexity of mixing algorithms directly affects the result. For example, you can "reverse" data in one or more audio files before mixing, introduce data shifts, block permutations, etc. At the same time, the repeatability of the results and the possibility of constructing synchronous generators of “random” numbers are preserved. To improve the quality of the result, it is necessary to evaluate entropy and use additional processing. One of the options readily available in Android is data compression - ZIP. A ZIP archive is created from the sound file, from which a section of compressed data is taken. According to a given algorithm, this data is mixed with the data of the previous stage. The specific mixing algorithm is selected after statistical evaluation of the data.

…

After all transformations, the required number of bytes is sampled.

TTS technology allows you to submit the received numbers (part of the numbers) to the input and receive a sound file from them... See the text above.

…

Now imagine that the described algorithm is a “cell” or “brick” from which a random number generator is built. Synchronous generator. 10 or 50 or 200 such “cells” with different parameters sequentially transform the entered text. If you enter the same text in another Android device and repeat the procedure, you will receive the same set of numbers.

And this is just the beginning. Here is another algorithm for a synchronous random number generator.

1. PRNG based on Internet radio or TV (**Czech Flow** Application)

We use an external digital stream to operate a “random” number generator. Synchronous generator. The ideal would be to create a high-quality digital noise generator on a satellite and dump the signal for your devices. But I don't have enough money for a companion yet. Therefore, we use the digital stream of Internet radio. Let's choose any Internet talk radio station. For example, news. The Android application will listen to Internet radio and search it for a given sequence of numbers. The application will work slightly differently (generate synchronous sets of "random" numbers) on the sending side and on the receiving side. For the application to work, you need to set several numbers that the application will wait for in the Internet radio stream. The shorter the array of numbers, the more often it will appear. It is necessary to make a tuning module that will select the optimal number of numbers. The app prototype used 12 numbers. Arrays of "random" numbers were generated approximately every 1-2 minutes. When the condition is met (the numbers in the stream are found), the application writes a block of Internet radio data. From which the required number of bytes is then selected using a given algorithm. However, these are not random numbers at all. Additional processing required. From the compiled array of “random” numbers, several numbers are selected and set for subsequent search in the stream. This is done several times. At each stage, several numbers are selected.

In parallel, similar modules listen to Internet radio streams No. 1, No. 2 ... and form the corresponding sets of numbers.

The resulting numbers are used for "cells" (generators) based on TTS at different stages (see point 1)

To ensure synchronization of generated arrays (and to decode encrypted data), time synchronization of receiving numeric arrays is required.

The disadvantages of such "random" number generators are that synchrony is ensured by reading data from streams at the same time. This can be successfully compensated for by compiling random number samples continuously and marking them with timestamps.

1. PRNG based on website data

PRNG "random" number generators based on the entire Internet. Synchronous PRNGs. To operate such generators, you need websites with changing news. News sites, stock trading sites, periodical publication sites. You can use RSS feeds.

The app reads updates from dozens of websites and uses texts as input to generate random numbers. Such generators as "cells" can be combined with dozens of TTS-based generators, used to form feedback in other types of generators, and used in value sampling operations.

Such generators are synchronized by time and a set of websites. And there is a subtlety here. It is necessary to ensure that websites display the same information for different locations.

Time synchronization is not convenient when working with mobile devices. It is much more convenient to use generators based on the accumulation of data and knowledge.

Disadvantages of such generators. The synchronicity of generating “random” numbers also depends on time, which is not convenient. It is necessary to use timestamps and generate more arrays of numbers.

A significant drawback of such generators is their dependence on regional versions of these websites. Information at one time may be different for different regions.

1. Generators based on "stretching" entropy.

Because Since electronic memory is now rapidly becoming cheaper (in terms of per unit of data), it is possible to pre-load a large volume of identical random numbers into both devices (sender and recipient). And then use them not only for direct encryption, but also as initial data for generating “random” numbers. As, for example, in TTS-based generators. By changing transformation and sampling algorithms, data can be “stretched” millions of times.

1. Generators based on data accumulation

Generators based on accumulated data are probably the most promising. To operate such generators, time synchronization is not required. The quality of random number generation can be improved as information from previous generation processes is accumulated. You can create very effective feedback loops and build algorithmic systems of enormous complexity. The process of generating random numbers moves from the realm of complex mathematical formulas to the realm of incredibly complex and constantly changing algorithms. Each time random numbers are generated, the application algorithm is rebuilt and the resulting algorithm will simply work. It will be impossible to describe it in simple and understandable words.

Moreover, if the device with such applications is “lost”, it will be impossible to interpret the algorithm. You can only use it (but there is protection against this too).

On the message receiving side, exactly the same application will repeat exactly the same actions, accumulate the same data and change the generation algorithm in the same way. There are no technical difficulties here other than the initial development (programming) process and system setup. Moreover, the system itself can evaluate and improve the quality of random numbers (again, synchronously).

The only obvious drawback of such systems is the higher computational load. However, it is acceptable for existing mobile devices.

In addition, in such generators (generation systems) it is possible to use separately effective programming work and the work of cryptographers. They simply won't know anything about each other.

Everything that was written above was written for creating synchronous PRNGs. With ordinary non-synchronous ones everything is much simpler. There are many more possibilities there.

Applications **Czech Entropy** and **Czech Flow** demonstrate the technical (software) possibility of creating such generators. There are many opportunities to improve the quality of the "random" numbers produced. Because Since these applications do not create real random numbers (sequences), it is impossible to use their public versions located in application stores or software archives for serious protection. You need to use custom versions with unpublished generation, sampling and improvement algorithms. However, now we will move on to the section of the document in which an algorithm (mechanism) will be proposed that can turn a public application into a custom one, improve the quality of random numbers and generate random numbers with an ideal distribution SYNCHRONOUSLY.

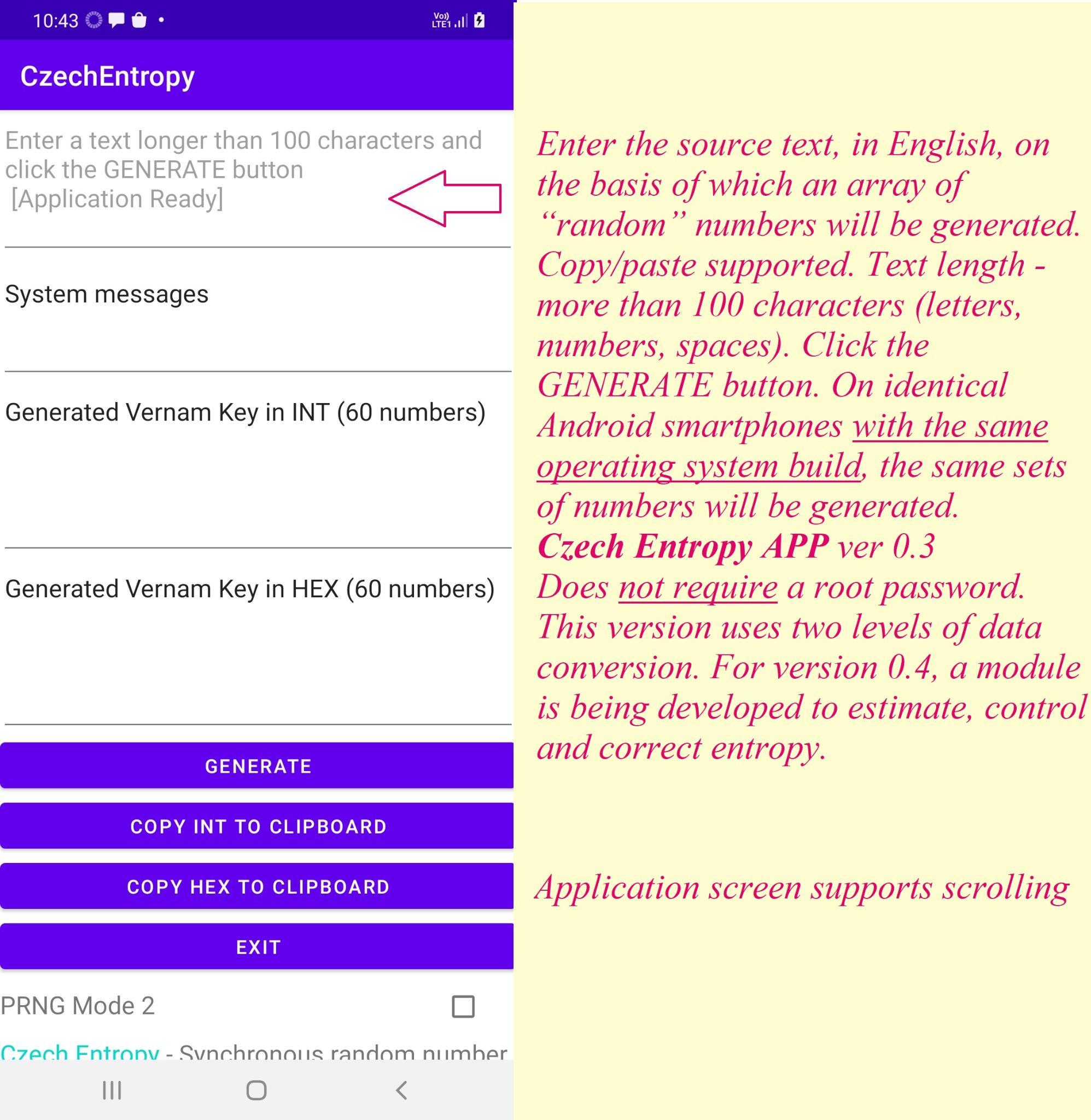
Czech Entropy APP ver 0.3

In version 0.3 of the Czech Entropy application, a second conversion level has been added that uses a different operating algorithm. You can enable it if you select CheckBox Mode 2

If the first level, Mode 1, uses extreme expansion of the source data, after which numbers are sampled (with constant coefficients for now), then the second level, Mode 2, performs entropy redistribution with data compression.

Feedbacks have not yet been used.

<https://drive.google.com/file/d/1in6sX45yDGty7NyDGCskZhJYJ4wo8V_T/view?usp=sharing> November 2023



Czech Entropy APP ver 0.35

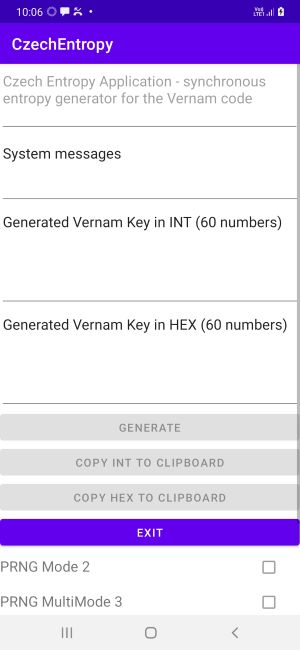
This version adds the MULTIMODE mode, which allows some of the generated “random numbers” to be used as input data for the next generation stages. The mode is selected by setting CheckBox **PRNG Multimode**

When selecting the MULTIMODE mode, the first array of numbers is generated in standard mode. And when you press the GENERATE button again, the mode with the supply of numbers to the application input begins to be used.

From this link you can download this version of the **Czech Entropy PRNG** application in APK format for installation on your Android smartphone. The application has been tested on Android 10. In addition to adding the MULTIMODE function, errors in the GUI have been fixed

<https://github.com/vallshmeleff/vernamkeygen> 18.Nov.2023

The entropy correction module is currently being tested. Once added to the **Czech Entropy PRNG** application, a field or entire screen will be provided to manually configure the parameters of ALL stages of generating "random numbers". It will also be possible to generate numbers with a large number of digits.



**PRNG with neural networks**

*A small digression. It's one thing to generate numbers synchronously. Another thing is to encrypt. But if they are also transmitted in such a way that no one pays attention…*

public Double Prediction (Integer DataInput1, Integer DataInput2){

return neurons.get(5).compute(

neurons.get(4).compute(

neurons.get(2).compute(DataInput1, DataInput2),

neurons.get(1).compute(DataInput1, DataInput2)

),

neurons.get(3).compute(

neurons.get(1).compute(DataInput1, DataInput2),

neurons.get(0).compute(DataInput1, DataInput2)

)

);

}

…

…

…

**PS. Warning. Please remember that this application Czech Entropy PRNG (Czech Entropy APP) is a FEATURE DEMONSTRATOR. For real use, individual (custom) application samples are required - only for the sender and recipient of messages.**