**Random Key Generation**

Currently the Redbacks Key Generation relies on pseudo-random number generation through the utilisation of the java randomnumber class. However, this can be considered insecure as the guess-ability of a pseudo-random number is much higher than that of a truly randomly generated number. This is because pseudo-random number generation requires a seed, and the same seed will produce the same number. Where are true random number generation does utilises a random point of physical data to generate a random number, such as Atmospheric Noise.

> "The difference between true random number generators(TRNGs) and pseudo-random number generators(PRNGs) is that TRNGs use an unpredictable physical means to generate numbers (like atmospheric noise), and PRNGs use mathematical algorithms (completely computer-generated)."[1]

Atmospheric Noise –

> Radio noise made by natural processes within the atmosphere, specifically lightning discharges in thunderstorms. Mostly caused by cloud-to-ground lightning strikes. The sum of all these lightning flashes results in atmospheric noise. It can be observed, with a radio receiver, in the form of a combination of white noise (coming from distant thunderstorms) and impulse noise (coming from a near thunderstorm) [2].
>
> Atmospheric noise and variation can be used to generate high quality random numbers

Atmospheric noise can be implemented similarly to as seen in [3].

## Approach

After reviewing various approaches I believe working with RandomNumber.org is the simplest way of implementing a truly random solution.

Approaches I assessed include utilising a TrueRNG (True Random Number Generation) device and the trueRNG library to generate a random number, introducing a random class in which the seed is generated from a .wav file as well as utilising RandomNumber.org.

For context, TrueRNG is a hardware device which utilises physical processes like thermal noise or radioactivity to generate random numbers and is used in conjunction with a software library.

Utilising Java's RandomClass and creating code which enables a seed to be generated from a .wav file relies on the idea that the noise from the audio will generate a genuinely random seed and consequently a truly random number, instead of relying on a pseudorandom seed like the time.

While using RandomNumber.org relies on atmospheric noise to generate random numbers and makes them available through APIs.

Some limitations of each method include:

**TrueRNG**

- Requires hardware / software
- Requires an input for randomness (which can be hard to acquire)
- Expensive compared to other options

**RandomClass with .wav input**

- The quality of the audio can affect the randomness of the generated numbers
- Would need to have different
- It is a computationally intensive

**Using RandomNumber.org**

- Randomness depends on the randomnumber.org algorithm
- Relies on http dependencies

After reflecting on the above as well as the resources and references below, I feel that using random.org is at current the most effective, efficient and manageable way of completing a truly random key generation class. I feel utilisation of reference 4 would be ideal for implementing a truly random key generation within the redbacks project.

References

*[1]https://www.researchgate.net/post/Difference_between_TRNG_or_PRNG#:~:text=The%20difference%20between%20true%20random,(completely%20computer%2Dgenerated)*.

*[2] Atmospheric noise - Wikipedia* (2023). Available at: https://en.wikipedia.org/wiki/Atmospheric_noise(Accessed: 25 April 2023).

[3] https://jeremytriplett06.medium.com/using-atmospheric-noise-to-generate-true-random-numbers-dc820ac9452d

[4] https://www.youtube.com/watch?v=lY0zX2Td3Cg&ab_channel=MVPJava