Leveraging Quantum Random Number Generation in Differential Privacy

Kaya Tacer, Alex Chrzanowski, Ryan Zhang, Nikita Gorshkov

May 2025

Abstract

Differential privacy is a method that ensures the privacy of individuals in a dataset by mixing in a small amount of noise to the results. The unpredictability of random number generation is crucial in this process. Deviations from randomness can risk leakage of individuals' sensitive data. Classical methods of random number generation are fundamentally limited in their randomness, thus, their risk of data leakage always remains non-zero. Therefore, we leverage QRNG - quantum random number generation - to create a noising process with no risk of data leakage due to compromise of the random number generator.

1 Differential Privacy

Differential privacy is a method of obfuscating sensitive results arising from data analysis. It protects the data subjects' privacy while still enabling useful results to be published, ensuring that the inclusion or exclusion of a single data point will not change the results. This way, data subjects cannot be re-identified with additional information. The importance of this cannot be understated. An internal investigation using the 2010 US census found that without differential privacy, 52 million people were able to be re-identified by using commercial databases. Moreover, the census predicted that with access to more databases, up to 179 million people could be re-identified.

To demonstrate its use and the danger of its absence, consider a school report, which reports that 10 children are on financial aid in March, and 9 children are on financial aid in April. If, as a student at the school, you know that between March and April, Tim dropped out of school, you can then infer that Tim was on financial aid. This breaches privacy and allows Tim to be re-identified through the results of the data analysis. Differential Privacy eliminates this risk by adding noise into the result, preserving accuracy and safeguarding Tim's privacy.

1.1 ε-Differential Privacy

The specific type of differential privacy we utilized to leverage the quantum advantage was ϵ-differential privacy. ϵ-differential privacy for our purpose requires that for any two data sets D_1 and D_2, where D_1 and D_2 differ by a single data instance, if run through an arbitrary algorithm A, the probability of getting any specific result is:

$$P(A(D_1)) \leq e^{\varepsilon} \cdot P(A(D_2))$$

Essentially, this inequality ensures that the result from running any algorithm over the datasets will not produce a difference larger than the result multiplied by e^{ε}. ε is a chosen number that defines how much accuracy should be sacrificed to safeguard privacy, with lower ε values corresponding to more privacy. Typical ε values range from 0.01 - 1 for high security data, and ε values as high as 10 for low security data.

1.2 The Role of Randomness

Random numbers are vital for the confidentiality that differential privacy provides. In order to generate the noise that is added to the data, we sample from a Laplace distribution with $\mu = 0$ and $b = \frac{1}{\varepsilon}$. We want $\mu = 0$ to prevent any bias within the noise. Using a Laplace distribution to generate noise for the dataset works under the assumption that each sample is random and completely independent from each other sample. This is where any bias from random samples can compromise the integrity of the noise generation algorithm. If the noise generation algorithm is compromised, the generated noise for each data point can be found, and the exact data can be extracted. This undermines the utility of differential privacy and reveals the sensitive data without noise.

2 Limitations of Classical Random Number Generators

2.1 Pseudo-Random Number Generators (PRNGs)

Pseudo-random number generators are deterministic processes that give the appearance of randomness based on a seed. Some PRNGs sample environmental noise for their seed, but fundamentally, all PRNGs are predictable. Additionally, once a seed has been chosen, no new information is being produced, so even sampling environmental noise does not lead to true randomness. As shown in Task 1 Part C, machine intelligence can predict LCG PRNGs because they are reversible. This compromises the integrity of any noise that they could generate for differential privacy. With a more advanced PRNG such as AES, this machine learning challenge is overcome, but they are still not foolproof. With AES, if the seed is known or leaked in any way, the random numbers can be predicted, and therefore, the noise can be ignored, compromising the differential privacy. This is especially an issue for high confidentiality data sets, such as those that concern national security. If the seed were to leak in any way, the data would be compromised. Additionally, AES is not theoretically guaranteed to give random numbers like quantum random number generators are, so it is not impossible that AES could be cracked sometime in the future. Therefore, a more foolproof method of noise generation is needed.

2.2 True Random Number Generators (TRNGs)

True random number generators rely on seemingly random quantities and processes to generate random numbers. This includes readings such as temperature, noise, and aspects of hardware. Even though true random number generators are generally reliable, they still have shortcomings. Firstly, TRNGs cannot be proven to be random and can still fall victim to biases in the environment because their randomness is opaque. This means that it is difficult to recognize that a TRNG is flawed or influenced. Additionally, TRNGs can be influenced by outside means to bias the random number generation. An example of this occurred in the early 2000s. Intel CPUs had the capacity to generate true random numbers based on thermal noise. However, researchers noticed that if put under certain voltages, the noise decreased. It also started producing repeating patterns of bits, completely undermining the utility of a TRNG. It was also possible to change the output by changing the power supply voltage, allowing the random numbers to be influenced by outside actions. Even though TRNGs might be tricky to take advantage of, in the case of differential privacy for important data, the possibility of a vulnerability is a driving factor towards using quantum random number generators instead.

3 The Quantum Advantage

With the use of quantum random number generators (QRNGs), we can get truly random numbers with which to create noise. With this, the noise added to the data is guaranteed to be bias-free and random. This ensures that even with unlimited data, the added noise cannot be discovered and the original exact data extracted. Additionally, since there is no seeding and the random numbers come directly from quantum uncertainty, there is no way to predict the noise even with full information. This eliminates the limitation of AES random number generators, which can still be compromised with a known seed. QRNGs make it possible for an attacker to read memory or reverse engineer the code, and still have no clue as to the noise

that was added to the dataset. It also doesn't share the same limitations of TRNGs, as it is guaranteed by theory that the result of a QRNG is uncertain. This guarantee also eliminates the possibility of interference from outside actions, as was the case with Intel's CPU TRNGs. Overall, for confidential datasets that need to ensure the privacy of their data subjects, quantum randomness is vital to ensure their enduring security.

4 Implementation

By using QRNGs, we were able to generate a random and independent number between 0 and 1, feed it into an inverse Laplace CDF, and get an equally random and independent sample of a Laplace distribution. This random sample of a Laplace distribution is used to dictate the noise that is added to each attribute that a data scientist may desire to obfuscate, completing the process of differential privacy. Our implementation included separate functions to generate these random numbers named get_qrng_floats(), as well as a function to transform those random numbers into a random Laplace distribution sample called get_laplace_transform(). To make it easy to use, we also created a wrapper function called add_noise() that takes in a one-dimensional array, an ε value, and an optional boolean to cast the resulting array into ints. This function takes in an array with the data that needs to be private, for example, the number of students on financial aid over the past 24 months. It then adds noise to the function according to the chosen ε value and random quantum numbers, and if desired, casts the new noisy data to integers, before returning the modified array. By doing this, we have made it simple to put a data set through the process of adding differential privacy while also eliminating any possibility of a seed being leaked or noise being predictable because of our use of a QRNG.

5 Statistical Analysis

5.1 Randomness of TRNG

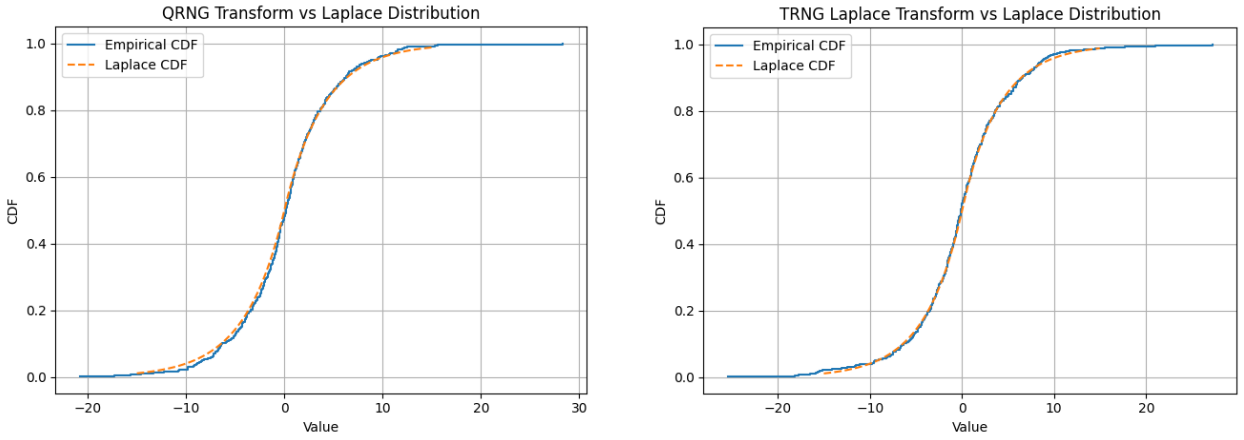
During our testing, we sought to understand whether the additional cost and complexity of quantum random number generators (QRNGs) are justified compared to simpler, hardware-based true random number generators (TRNGs), such as Jitter entropy sources. While TRNGs are generally more secure than pseudorandom number generators (PRNGs), they may still exhibit structural patterns or slight deviations from ideal statistical behavior. To assess one of QRNG's purported advantages—its ability to produce output that more closely approximates theoretical randomness—we generated 500 sets of 500 floating-point values in the range $[0, 1)$ using both the IDQuantique QRNG API and the Jitter-based TRNG.

Each sample was transformed into Laplace-distributed noise using an inverse Laplace mechanism with $\varepsilon = 0.25$ (depicted in figures 1a and 1b), then tested for distributional conformity using the one-sample Kolmogorov-Smirnov (KS) test. This test measures how closely each sample matches the ideal Laplace $(0, \frac{1}{\varepsilon})$ distribution. Across the 500 trials, the average KS statistic for the QRNG data was 0.038649 ($\sigma = 0.011399$), compared to 0.037526 ($\sigma = 0.010827$) for the TRNG data. We performed a two-sample t-test comparing the KS statistics from both sources under the null hypothesis that they are equally close to Laplace-distributed noise. With a significance level of $\alpha = 0.08$, the resulting p-value was approximately 0.055, allowing us to reject the null hypothesis at that level.

While this analysis does not directly evaluate entropy or unpredictability in the cryptographic sense, it provides modest statistical evidence that the QRNG output conforms more consistently to the expected Laplace distribution. This result suggests that, under this transformation-based test, the QRNG exhibits behavior more consistent with idealized random sampling, supporting its potential advantage in producing higher-quality randomness.

5.2 Applicability to Real Data

Further, we wanted to evaluate the practical applicability of our QRNG in real-world privacy-preserving computations. Using data from the U.S. Census, we applied the Laplace mechanism to privatize the original dataset by adding noise sampled using the IDQuantique QRNG API, with $\varepsilon = 0.25$. To assess the utility of the privatized data, we computed the Pearson correlation coefficient between the original and privatized values. The resulting correlation was approximately $r = 0.995$ with a p-value near zero, indicating a very



(a) QRNG Distribution

(b) TRNG Distribution

Figure 1: Comparison of QRNG and TRNG Distributions

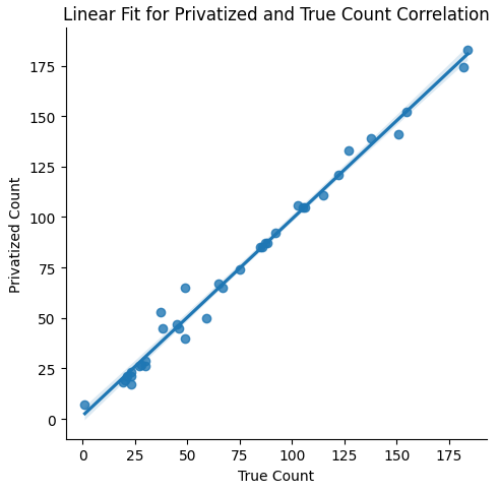


Figure 2: Census Data Correlation

strong linear relationship between the original and privatized datasets (depicted in figure 2). This suggests that while individual values are obfuscated, the overall structure and utility of the data are well preserved.

5.3 Limitations

It is important to note that obtaining the testing noise in section 5.1 through the QRNG took significantly longer than with the TRNG. The QRNG-based process required over two hours to complete, whereas the TRNG-based noise generation was completed in a matter of minutes. For applications involving highly sensitive data, the strong unpredictability and physical guarantees of the QRNG may justify its use despite the increased latency. However, in time-constrained scenarios, the TRNG remains a practical compromise, offering reasonable randomness with considerably lower overhead.