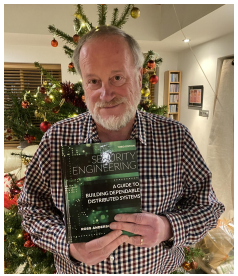
Machine Learning Needs Better Randomness Standards
Randomised Smoothing and PRNG-Based Attacks



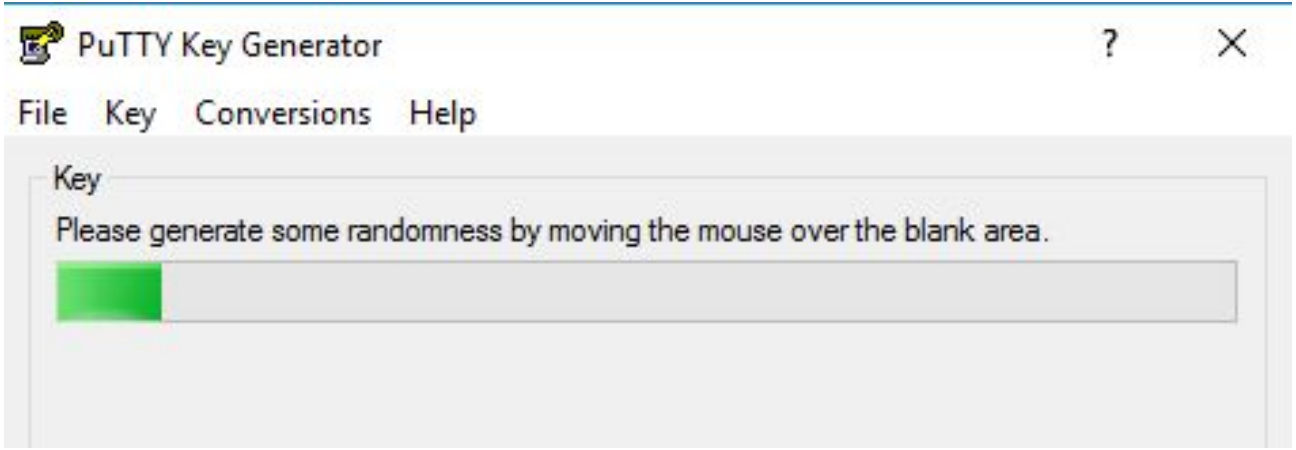


Pranav Dahiya, Ilia Shumailov, Ross Anderson

CHRIST CHURCH
UNIVERSITY OF OXFORD

UNIVERSITY OF OXFORD

UNIVERSITY OF CAMBRIDGE


PuTTY Key Generator
File
Key
Conversions
Help
Key
Please generate some randomness by moving the mouse over the blank area.

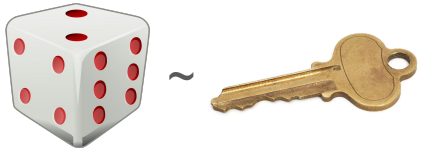
00111000010001101110100011000101001101011111100010100100100100111011
10000001110110111110011010010011111110100010000111000101101010110001
01111010011111101001000011110000001011001101000101101101000011100001
00110100111010101000101100000100111111001000011100110011010010010001
10101101000011110110111000000011110001100000111111010011110101000110
01011010110001100011110110010101111010100010011101001111000001111101
00011110100110000001111001001000001100010100101111010110001110001001
01101101000101111000011011010110010011100100001101101001010111001011
11101101000001010000001111100100100001010110101001100101111011000001
00111100011010101000111101001011001010110001101000011000100000000001
00101010110111101000000001010011101110010100111100111110101101100010
00110001000001100011000011111110111011010011101011110000111001010111
01111101111110001001110111101101000011110000011010110000010000001101
11110101100000001111110101111111111010101100101110001111001000110110
01110100001101000110100010101000010111011111000000011000000100100101 0
01100100001111110100000100011011110101111100100101111111110000000000
01110111101011100001000011111001011111001001101010000010110101011110
11101000100010111111101010101110001100111110

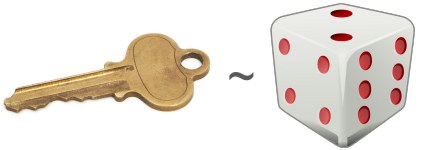
00111000010001101110100011000101001101011111100010100100100100111011
10000001110110111110011010010011111110100010000111000101101010110001
01111010011111101001000011110000001011001101000101101101000011100001
00110100111010101000101100000100111111001000011100110011010010010001
10101101000011110110111000000011110001100000111111010011110101000110
01011010110001100011110110010101111010100010011101001111000001111101
00011110100110000001111001001000001100010100101111010110001110001001
01101101000101111000011011010110010011100100001101101001010111001011
11101101000001010000001111100100100001010110101001100101111011000001
00111100011010101000111101001011001010110001101000011000100000000001
00101010110111101000000001010011101110010100111100111110101101100010
00110001000001100011000011111110111011010011101011110000111001010111
01111101111110001001110111101101000011110000011010110000010000001101
11110101100000001111110101111111110101011001011100011110010001101 10
01110100001101000110100010101000010111011110000000110000001001001010
01100100001111110100000100011011110101111100100101111111110000000000
01110111101011100001000011111001011111001001101010000010110101011110
11101000100010111111101010101110001100111110

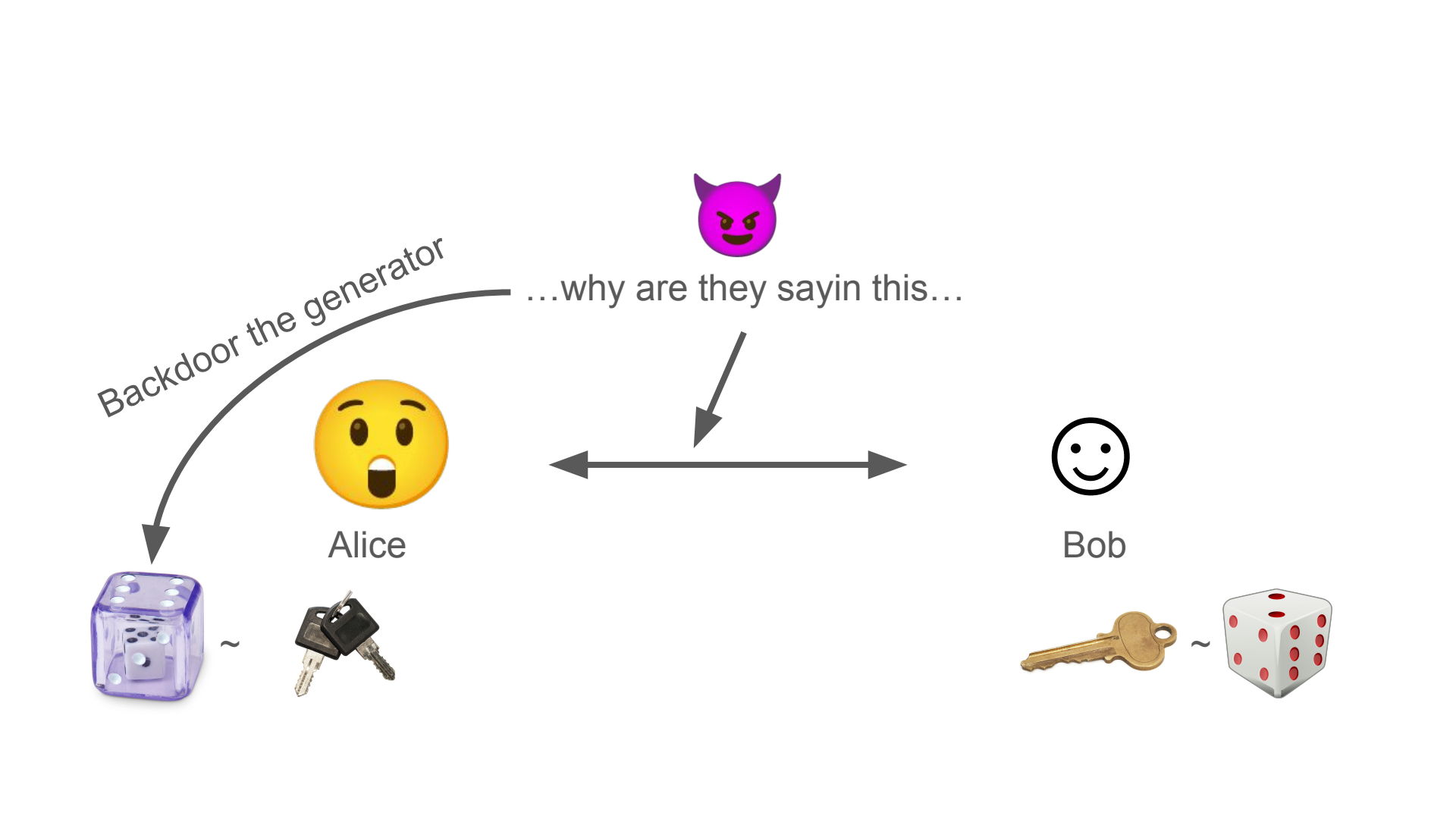
…what are they sayin…

Alice

Bob






Backdoor the generator
…why are they sayin this…
Alice
Bob

Randomness in Cryptography is important

- Standards for PRNG constructions
- Standards with tests
- Standards with sample size
- Importantly, all of the above are not perfect

Randomness in Cryptography is important

Sample Size

The issue of sample size is tied to the choice of the significance level. NIST recommends that, for these tests, the user should fix the significance level to be at least 0.001, but no larger than 0.01[8]. A sample size that is disproportional to the significance level may not be suitable. For example, if the significance level (α) is chosen to be 0.001, then it is expected that 1 out of every 1000 sequences will be rejected. If a sample of only 100 sequences is selected, it would be rare to observe a rejection. In this case, the conclusion may be drawn that a generator was producing random sequences, when in all likelihood a sufficiently large enough sample was not used. Thus, the sample should be on the order of the inverse of the significance level (α^{-1}). That is, for a level of 0.001, a sample should have at least 1000 sequences. Ideally, many distinct samples should be analyzed.

https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-22r1a.pdf

Randomness In Machine Learning

- Optimization – **Stochastic** Gradient Descent
- Data Sampling
- Flavors of Learning: Federated, Active, Curriculum
- Differential Privacy
- Uncertainty Estimation

and a lot more …

Randomness In Machine Learning

- Randomness for ML is held to the **same standard**, kinda…
- **Use is different:** uniform vs parametric
- **Sample size is different:** key material vs gigabytes
- **Latency is different:** when little is needed expensive is ok

Case study: Randomised Smoothing

Randomised Smoothing

- Why is randomised smoothing used?
  - Safety – uncertainty
  - Security – robustness

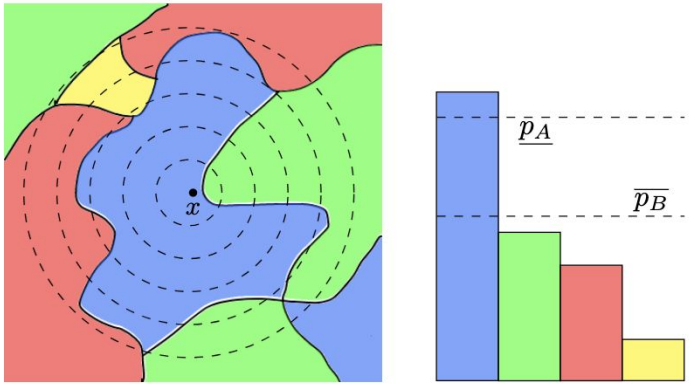


Figure 1. Evaluating the smoothed classifier at an input x. **Left**: the decision regions of the base classifier f are drawn in different colors. The dotted lines are the level sets of the distribution $\mathcal{N}(x, \sigma^2 I)$. **Right**: the distribution $f(\mathcal{N}(x, \sigma^2 I))$. As discussed below, $\underline{p_A}$ is a lower bound on the probability of the top class and $\overline{p_B}$ is an upper bound on the probability of each other class. Here, $g(x)$ is "blue."

- Find maximal radius around x such that adversarial examples do not exist with a high confidence.

- In the limit is an uncertainty estimator

https://arxiv.org/pdf/1902.02918

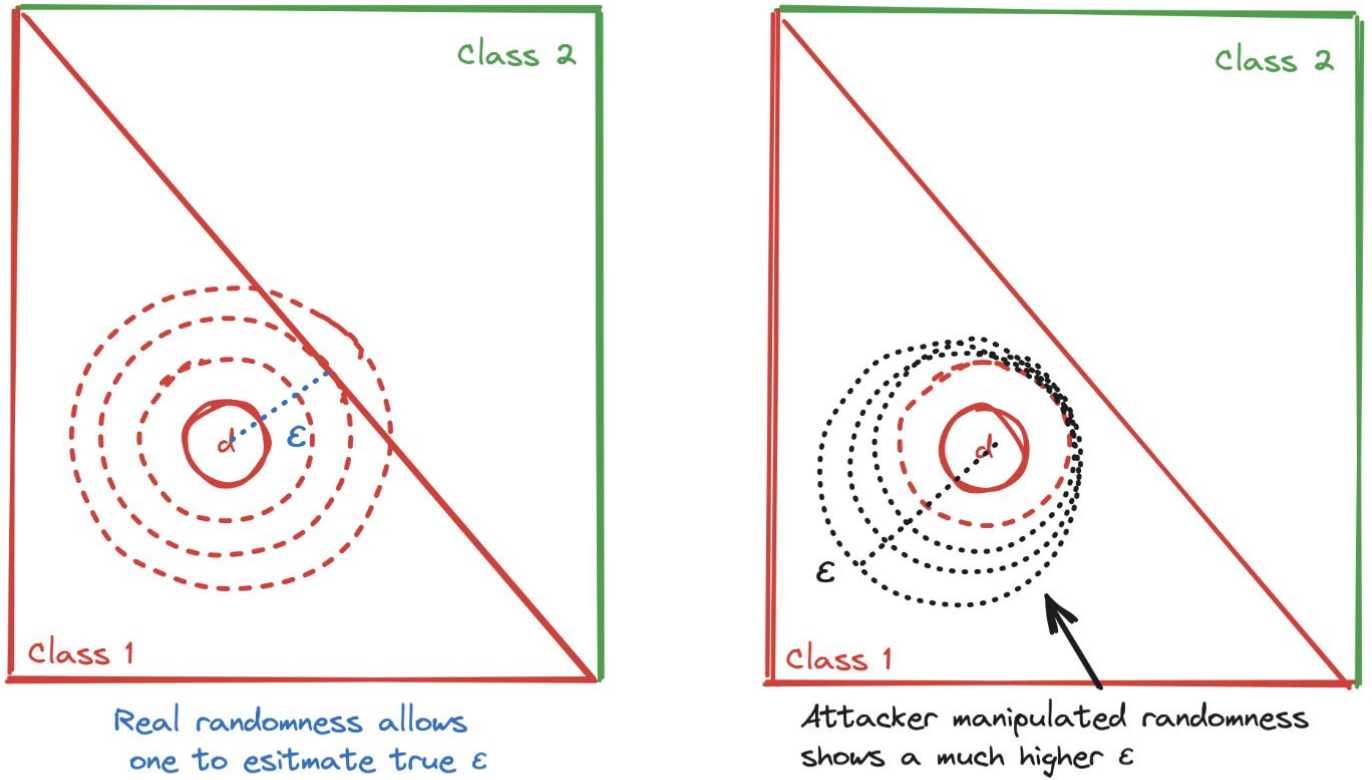


Figure 1: A visual example of how manipulated randomness can result in mis-perception of real model confidence. A normal Randomised Smoothing procedure is shown on the left – the original datapoint d is perturbed to get points in the dotted region, ultimately deriving a true ϵ confidence region. The attacker biases the sampled noise and changes the estimation of Randomised Smoothing, as shown on the right. This overestimates the confidence and leads to an incorrect prediction. Noise can also be biased to reduce the ϵ instead.

Naïve Attacker

- Base classifier trained with additive noise sampled from $N(0, 0.25^2)$
- Replace distribution for one of the following when a certification is performed:
  - Laplace distribution, $L(0, 0.25)$;
  - Absolute value of normal distribution, $|N(0, 0.25^2)|$;
  - Uniform distribution, $U(-0.25, 0.25)$;
  - Bernoulli distribution, $B(0.5)$

Naïve Attack Performance

$\mathcal{N}(0, 0.25^2)$ vs $\mathcal{L}(0, 0.25)$

	correct	incorrect	abstain
correct	**282**	48	44
incorrect	9	**52**	19
abstain	6	20	**20**

Laplace distribution

$\mathcal{N}(0, 0.25^2)$ vs $|\mathcal{N}(0, 0.25^2)|$

	correct	incorrect	abstain
correct	**287**	73	14
incorrect	12	**62**	6
abstain	15	24	**7**

Absolute normal distribution

$\mathcal{N}(0, 0.25^2)$ vs $\mathcal{U}(-0.25, 0.25)$

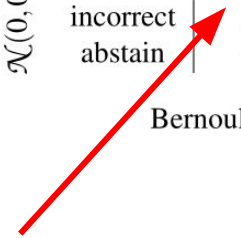
	correct	incorrect	abstain
correct	**128**	235	11
incorrect	9	**70**	1
abstain	3	31	**2**

Uniform distribution

$\mathcal{N}(0, 0.25^2)$ vs $\mathcal{B}(0.5)$

	correct	incorrect	abstain
correct	**154**	113	107
incorrect	5	**42**	33
abstain	4	25	**17**

Bernoulli distribution



Gaussian Random Number Generators

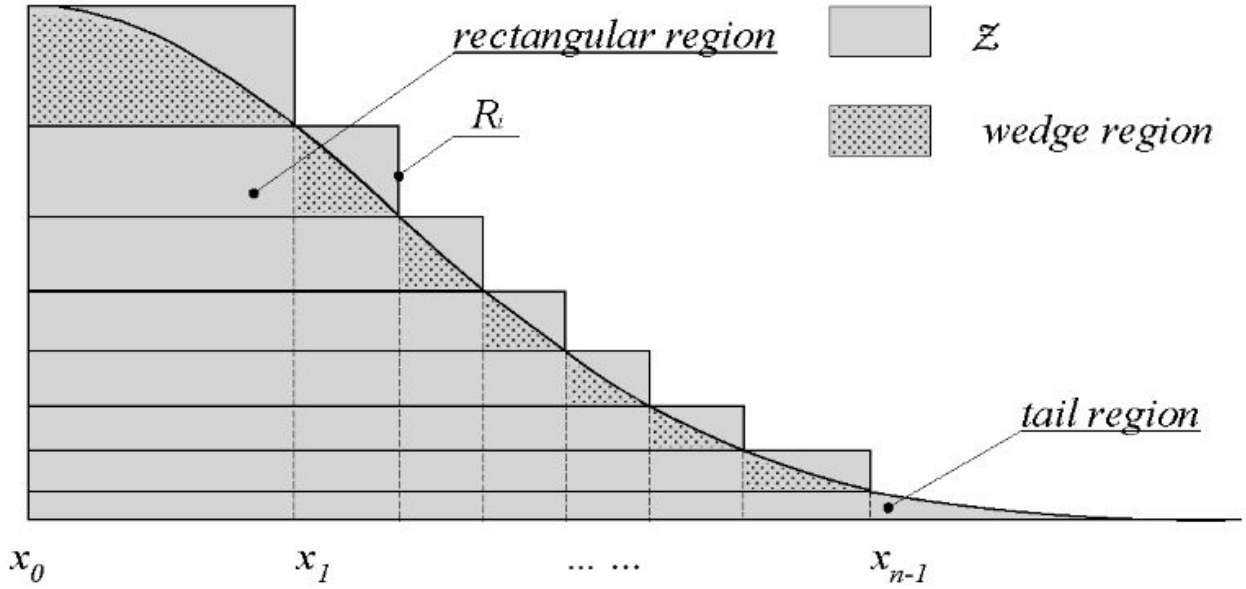
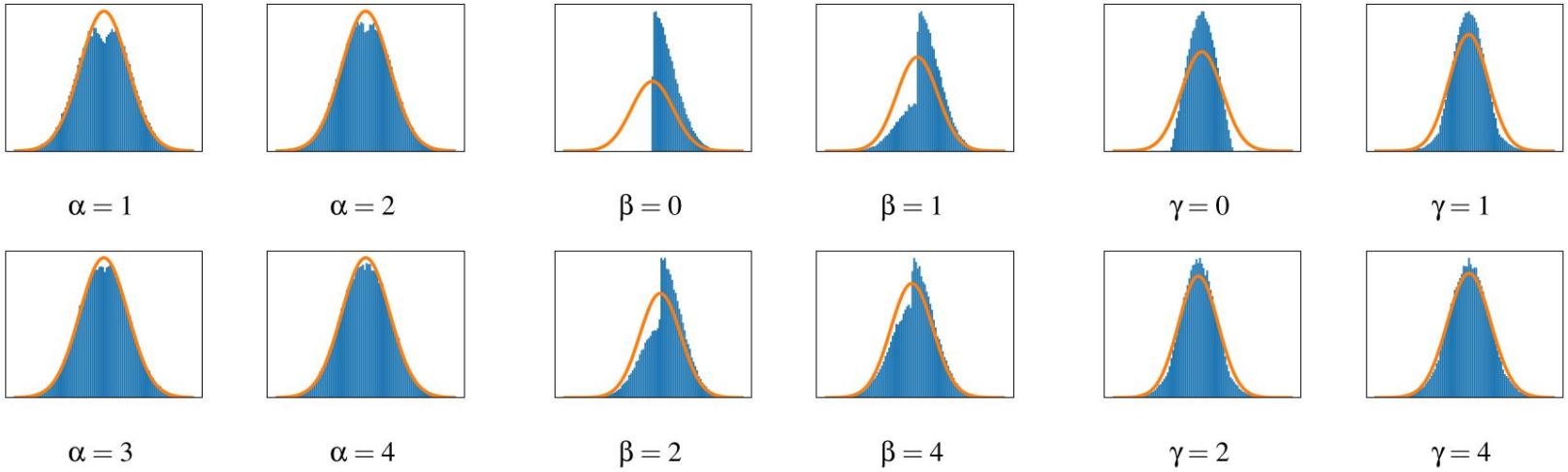


Fig. 6. Diagram showing the Gaussian distribution divided into rectangular, wedge, and tail regions in the Ziggurat method.

We change 1 bit in 64

https://www.doc.ic.ac.uk/~wl/papers/07/csur07dt.pdf by Thomas et al.

BitFlipping PRNG Attacker



(a) Negative kurtosis attack (b) Skewness attack (c) Positive Kurtosis attack

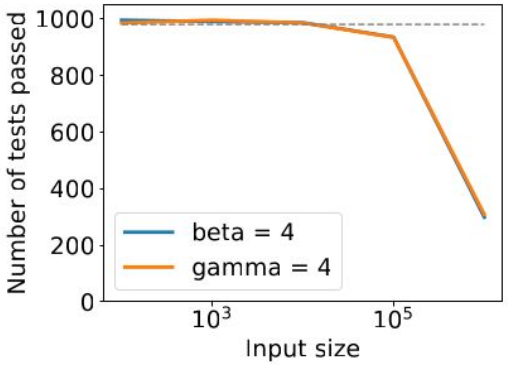
	$\frac{R'}{R} < 1$	$1.0 < \frac{R'}{R} < 1.1$	$1.1 < \frac{R'}{R} < 1.25$	$1.25 < \frac{R'}{R} < 1.5$	$1.5 < \frac{R'}{R} < 2.0$	$\frac{R'}{R} > 2.0$	$\max\left(\frac{R'}{R}\right)$
Naive noise distribution replacement attack							
$\mathcal{L}(0, 0.25)$	0.80	0.07	0.02	0.02	**0.02**	**0.07**	6.03
$\lvert\mathcal{N}(0, 0.25^2)\rvert$	0.24	0.31	0.09	0.08	**0.10**	**0.18**	**76.25**
$\mathcal{U}(-0.25, 0.25)$	0.32	0.22	0.03	0.08	**0.07**	**0.28**	**13.99**
$\mathcal{B}(0.5)$	0.88	0.02	0.01	0.02	**0.03**	**0.04**	9.58
Negative-Kurtosis attack							
$\alpha = 1$	0.58	0.31	0.05	0.03	**0.02**	**0.02**	2.10
$\alpha = 2$	0.57	0.35	0.04	0.02	**0.02**	0.00	1.38
$\alpha = 3$	0.52	0.41	0.04	0.01	**0.01**	0.00	1.06
$\alpha = 4$	0.55	0.40	0.03	0.01	0.00	0.00	1.95
Skewness attack							
$\beta = 0$	0.24	0.31	0.09	0.08	**0.10**	**0.18**	**81.24**
$\beta = 1$	0.24	0.35	0.10	0.08	**0.09**	**0.13**	**36.27**
$\beta = 2$	0.30	0.33	0.12	0.10	**0.06**	**0.08**	**27.43**
$\beta = 4$	0.34	0.39	0.12	0.06	**0.04**	**0.05**	**18.11**
Positive-Kurtosis attack							
$\gamma = 0$	0.14	0.32	0.14	0.14	**0.10**	**0.16**	**40.70**
$\gamma = 1$	0.18	0.38	0.17	0.13	**0.06**	**0.08**	**15.79**
$\gamma = 2$	0.20	0.42	0.20	0.09	**0.04**	**0.06**	**10.49**
$\gamma = 4$	0.22	0.51	0.16	0.04	**0.03**	**0.04**	6.05

Table 2: This table shows the relative certified radius, R'/R, where R' is the manipulated radius under attack, and R is the baseline radius. The fraction of images for which R'/R falls in different bins is shown for the naive attacks and all three of the bit-flipping `PRNG` attacks. The maximum value of the relative certified radius achieved for each attack is also reported. The values in red indicate instances when the attack managed to successfully manipulate the radius by a factor of at least 1.5. Out of the naive attacks, $\mathcal{U}(-0.25, 0.25)$ achieves the best performance, followed by $\lvert\mathcal{N}(0, 0.25^2)\rvert$, $\mathcal{L}(0, 0.25)$, and finally $\mathcal{B}(0.5)$. Among the bit-flipping `PRNG` attacks, the skewness performs the best, then the positive-kurtosis, followed by the negative kurtosis.

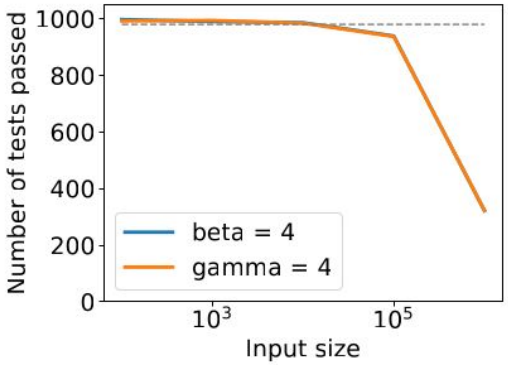
PRNG Attack vs NIST Tests

	MT19937	PCG64	Philox	SFC64	α				β				γ			
					1	2	3	4	0	1	2	4	0	1	2	4
Frequency	0	991	991	989	0	0	0	0	0	0	3	**299**	0	0	0	309
BlockFrequency	0	991	989	989	2	0	**667**	0	915	990	993	996	922	986	995	990
CumulativeSums	0	992	988	987	0	0	0	0	0	0	3	**322**	0	0	1	**323**
Runs	0	990	988	991	0	0	0	0	0	0	102	**805**	0	0	0	**233**
LongestRun	0	991	988	984	670	282	**984**	216	914	989	988	990	732	963	966	989
Rank	0	991	990	992	995	994	992	985	993	983	992	986	988	986	993	994
FFT	0	985	991	984	0	0	0	0	0	572	942	988	0	0	0	**125**
NonOverlappingTemplate	0	980	982	981	0	0	0	0	116	825	929	975	0	8	302	**787**
OverlappingTemplate	0	991	985	989	3	0	906	0	400	938	974	987	0	447	779	930
Universal	0	996	988	984	921	150	978	0	987	989	982	993	980	989	986	990
ApproximateEntropy	0	989	990	991	0	0	0	0	5	812	957	986	0	93	727	944
RandomExcursions	-	618/628	613/626	617/631	-	-	-	-	2/2	24/26	63/64	210/215	-	26/27	83/85	220/224
RandomExcursionsVariant	-	617/628	616/626	622/631	-	-	-	-	2/2	24/26	61/64	211/215	-	26/27	84/85	215/224
Serial	0	988	987	977	0	0	196	0	901	985	990	990	297	945	984	982
LinearComplexity	989	990	984	992	993	991	984	989	993	987	995	989	991	991	994	992

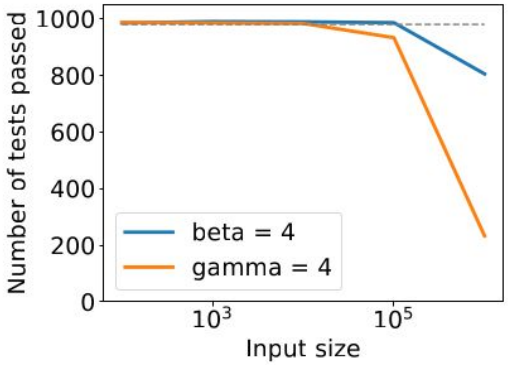
Sample Size vs Test Results



(a) Frequency



(b) Cumulative Sums



(c) Runs

PCG64	D'Agostino -	Kolmogorov-Smirnov -	Cramer-von Mises -	Jarque-Bera -	Shapiro -	Lilliefors 10^5
$\alpha = 1$	10^3	10^2	10^3	10^3	10^3	10^3
$\alpha = 2$	10^4	10^3	10^3	10^4	10^4	10^4
$\alpha = 3$	10^4	10^3	10^4	10^4	10^4	10^4
$\alpha = 4$	10^4	10^4	10^4	10^4	10^4	10^4
$\beta = 0$	10^2	10^2	10^2	10^2	10^2	10^2
$\beta = 1$	10^2	10^2	10^2	10^2	10^2	10^2
$\beta = 2$	10^2	10^2	10^2	10^2	10^2	10^2
$\beta = 4$	10^2	10^2	10^2	10^2	10^2	10^2
$\gamma = 0$	10^2	10^2	10^2	10^2	10^2	10^4
$\gamma = 1$	10^2	10^2	10^3	10^2	10^2	10^4
$\gamma = 2$	10^2	10^3	10^3	10^2	10^2	10^4
$\gamma = 4$	10^2	10^3	10^3	10^2	10^2	10^4

Table 7: This table reports the minimum sample size for which tests started to fail for each PRNG compared to PCG64. 1000 tests were run for increasing sample sizes from 10^2 to 10^6 following the same pass criteria as the NIST test suite in Table 3. A sample size of $\mathbf{10^5}$ was required to see every attacked PRNG conclusively fail all 6 tests.

Discussion

- How realistic is this?
- Technical Stack
- Protocols and Guidelines
- Development of Better Standards
- On Standards