Pseudo-Random Generators and Functions

Unveiling the Complexity and Elegance of Randomness

Ji Yong-Hyeon & Kim Dong-Hyeon

Department of Information Security, Cryptology, and Mathematics

hacker3740@gmail.com, bob@university.cs

I.1 Introduction of PRG

A deterministic function $G : \{0,1\}^{\lambda} \to \{0,1\}^{\lambda+l}$ with $l > 0$ is a **secure pseudorandom generator (PRG)** if $\mathcal{L}^{G}_{\mathsf{PRG-real}} \approx \mathcal{L}^{G}_{\mathsf{PRG-rand}}$, where:

$\mathcal{L}^{G}_{\mathsf{PRG-real}}$

Query():

$s \leftarrow \{0,1\}^{\lambda}$

return $G(s)$

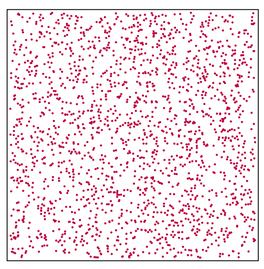
$\mathcal{L}^{G}_{\mathsf{PRG-rand}}$

Query():

$r \leftarrow \{0,1\}^{\lambda+l}$

return r

We illustrate the distributions, for a length doubling ($l = \lambda$) PRG (not drawn to scale):



Pseudorandom dist. ($\{0,1\}^{\lambda} \to \{0,1\}^{2\lambda}$)

Uniform dist. ($\{0,1\}^{2\lambda}$)

I.2 How NOT to Build a PRG

A straightforward approach for the PRG might be to duplicate its input string.

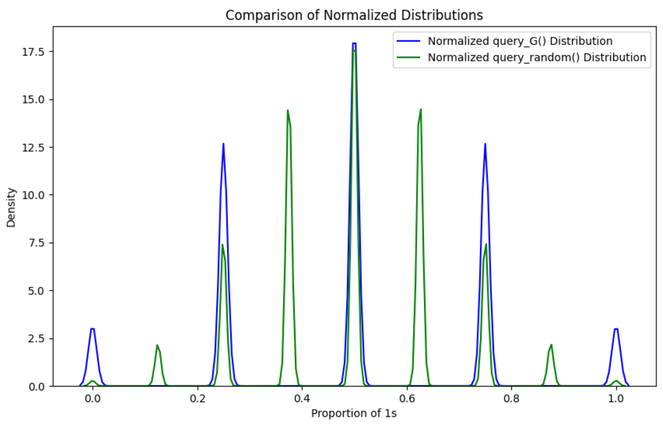
$G(s)$:

return $s \parallel s$

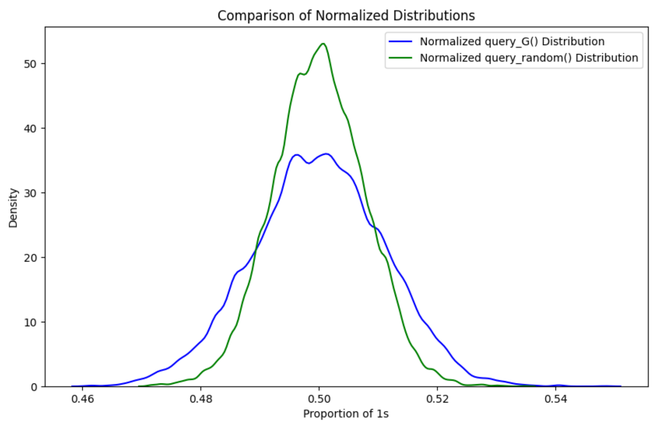
For example, the following strings look likely they were sampled uniformly from $\{0,1\}^{8}$:

$$11011101, 01110111, 01000100, \cdots$$

Comparison of Normalized Distributions:



$\lambda = 4$, i.e., $\{0,1\}^{8}$ with 1,000,000 experiments.

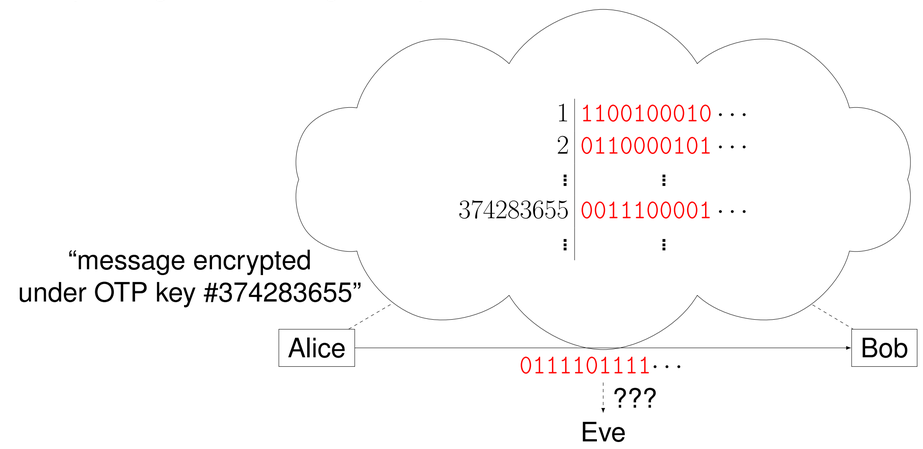


$\lambda = 1024$, i.e., $\{0,1\}^{2048}$ with 100,000 experiments

```python
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

def G(s):
    return s * 2
def query_G():
    lambda_length = 1024
    s = np.random.randint(0, 2, lambda_length).tolist()
    return G(s)

def query_random():
    lambda_length = 1024
    l_length = 1024
    r = np.random.randint(0, 2, lambda_length + l_length).tolist()
    return r

num_experiments = 100000
outputs_G = [query_G() for _ in range(num_experiments)]
outputs_random = [query_random() for _ in range(num_experiments)]
sums_G = [sum(output) for output in outputs_G]
sums_random = [sum(output) for output in outputs_random]
norm_sums_G = [s / 2048 for s in sums_G]
norm_sums_random = [s / 2048 for s in sums_random]
plt.figure(figsize=(10, 6))
sns.kdeplot(norm_sums_G, bw_adjust=0.5, label='Normalized query_G() Distribution', color='blue')
sns.kdeplot(norm_sums_random, bw_adjust=0.5, label='Normalized query_random() Distribution', color='green')

plt.legend()
plt.title('Comparison of Normalized Distributions')
plt.xlabel('Proportion of 1s')
plt.ylabel('Density')
plt.show()
```

II.1 Introduction of PRF

This research explores groundbreaking concepts...



The goal of a pseudorandom function is to "*look like*" a uniformly chosen array / lookup table.

Array T

for $x \in \{0,1\}^{\mathtt{Idx}}$

$T[x] \leftarrow \{0,1\}^{\mathtt{Out}}$

Lookup ($x \in \{0,1\}^{\mathtt{Idx}}$):

return $T[x]$

Index $x_{\mathtt{Idx}-1} \cdots x_1 x_0$	Output $T[x]$
000	$T[000]$
001	$T[001]$
010	$T[010]$
⋮	⋮
x	$T[x]$

Binary String Mapping
1100100010 ···
0110000101 ···
0011100001 ···

$k \leftarrow \{0,1\}^{\lambda}$

Lookup ($x \in \{0,1\}^{\mathtt{Idx}}$):

return $F(k, x)$

Pseudorandom Function F

Input ($x \in \{0,1\}^{\mathtt{Idx}}$)

Seed ($k \in \{0,1\}^{\lambda}$)

Output ($\{0,1\}^{\mathtt{Out}}$)

Objectives

The primary aim of this research is to... Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

References

[1] M. Rosulek, *The Joy of Cryptography*, [Online]. Available: https://joyofcryptography.com

[2] N. P. Smart, *Cryptography Made Simple*. 1st ed. Springer International Publishing, 2016.

[3] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*. 2nd ed. Chapman and Hall/CRC, 2014.