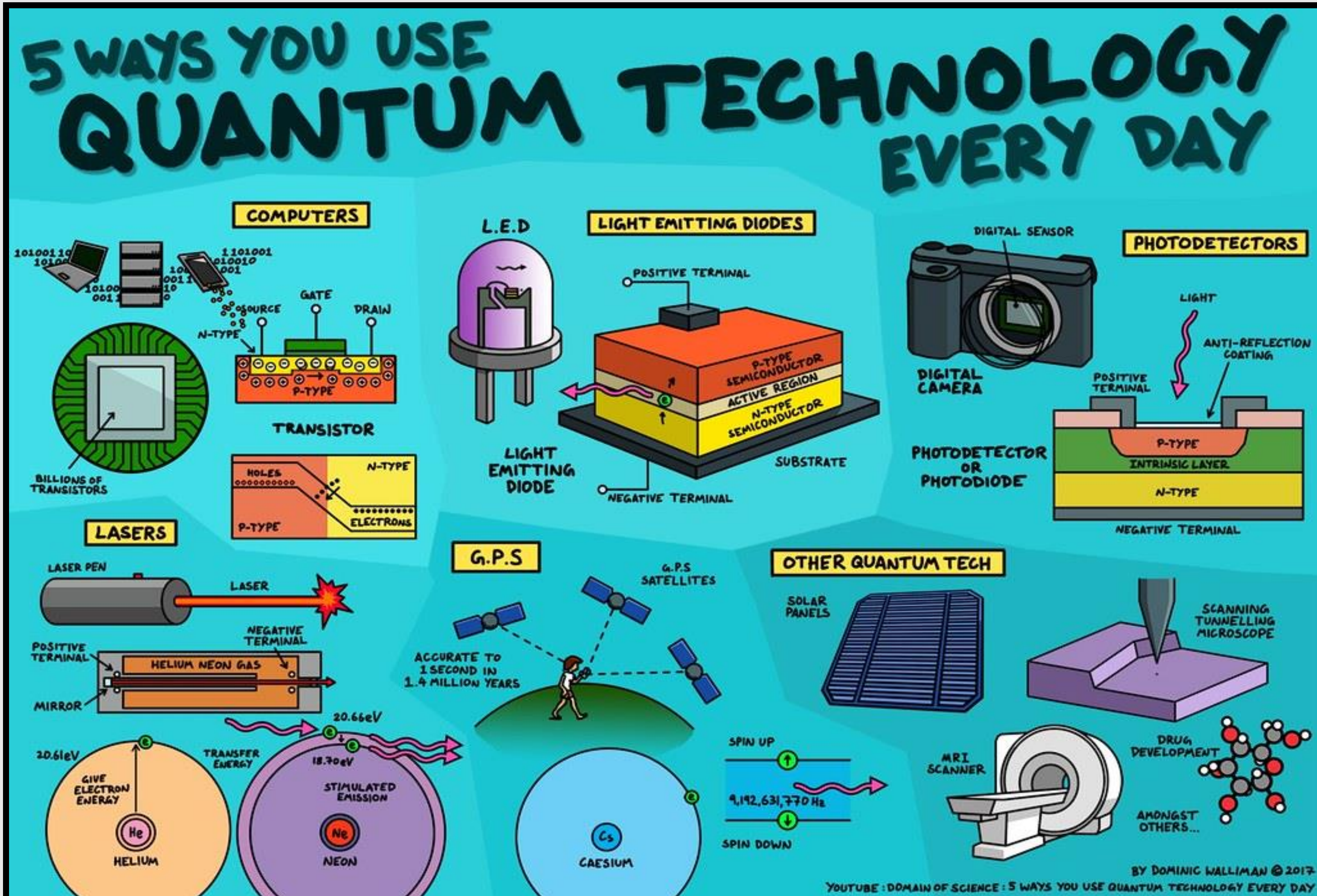
Introduction to Quantum Cryptography

Shashank Gupta
(Research Lead)
shashank@qnulabs.com

Q→NU

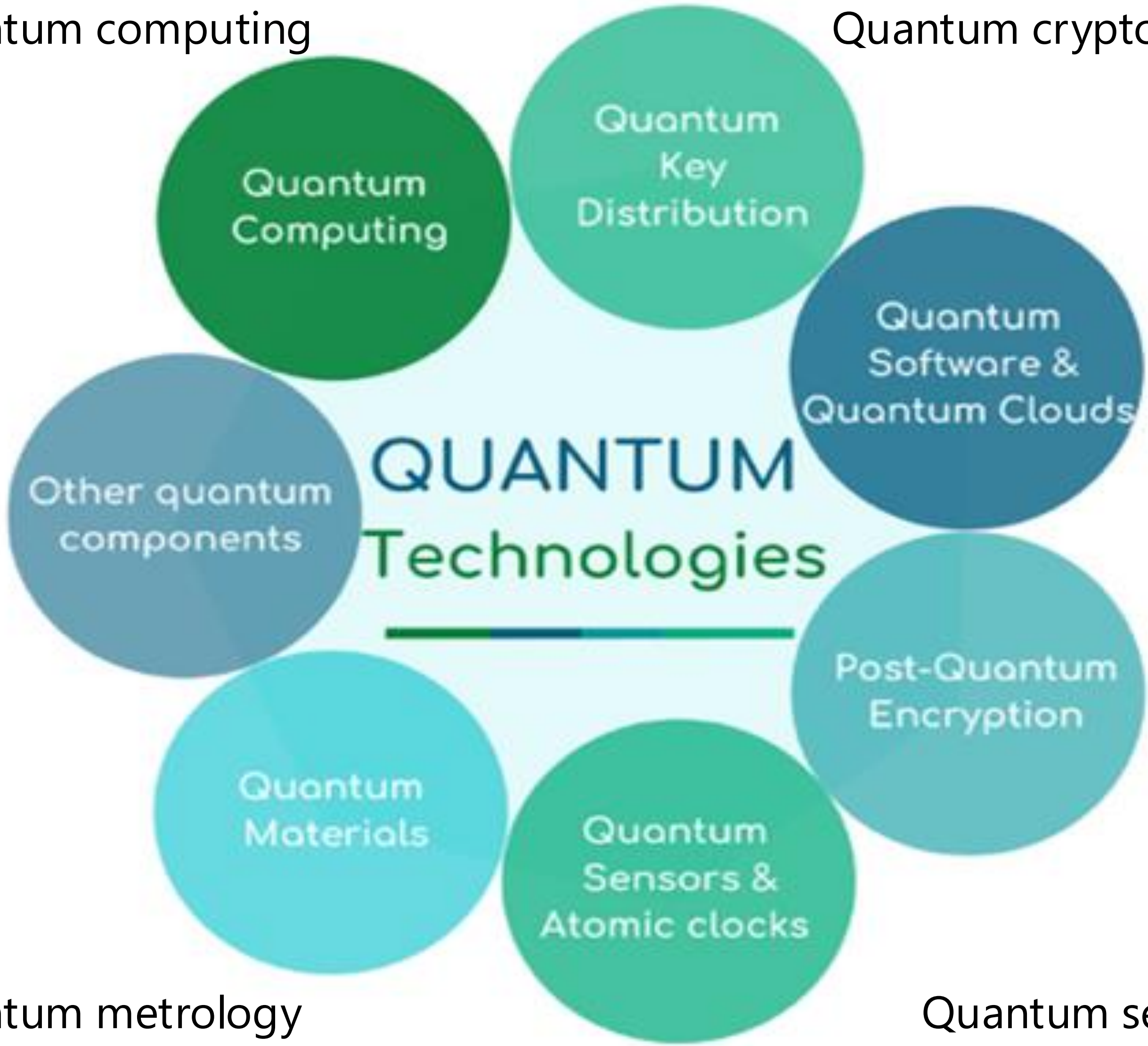


https://www.flickr.com/photos/95869671@N08/35480763464

Second Quantum Revolution
(using non-classical resources fundamentally)

Quantum computing

Quantum cryptography



Quantum metrology

Quantum sensing

https://thequantuminsider.com/2021/05/03/quantum-for-everyone-the-online-course-to-learn-a-practical-approach-to-quantum-technologies-for-you-and-your-business/

Cryptography

It is a technique of securing information and communications using codes to ensure confidentiality, integrity and authentication.

Q→NU

Features

Confidentiality

Ensures data is accessible only to intended recipients.

Integrity

Guarantees data remains unaltered during storage and transmission.

Non-repudiation

Prevents senders from denying their actions.

Authentication

Verifies the identities of senders and receivers.

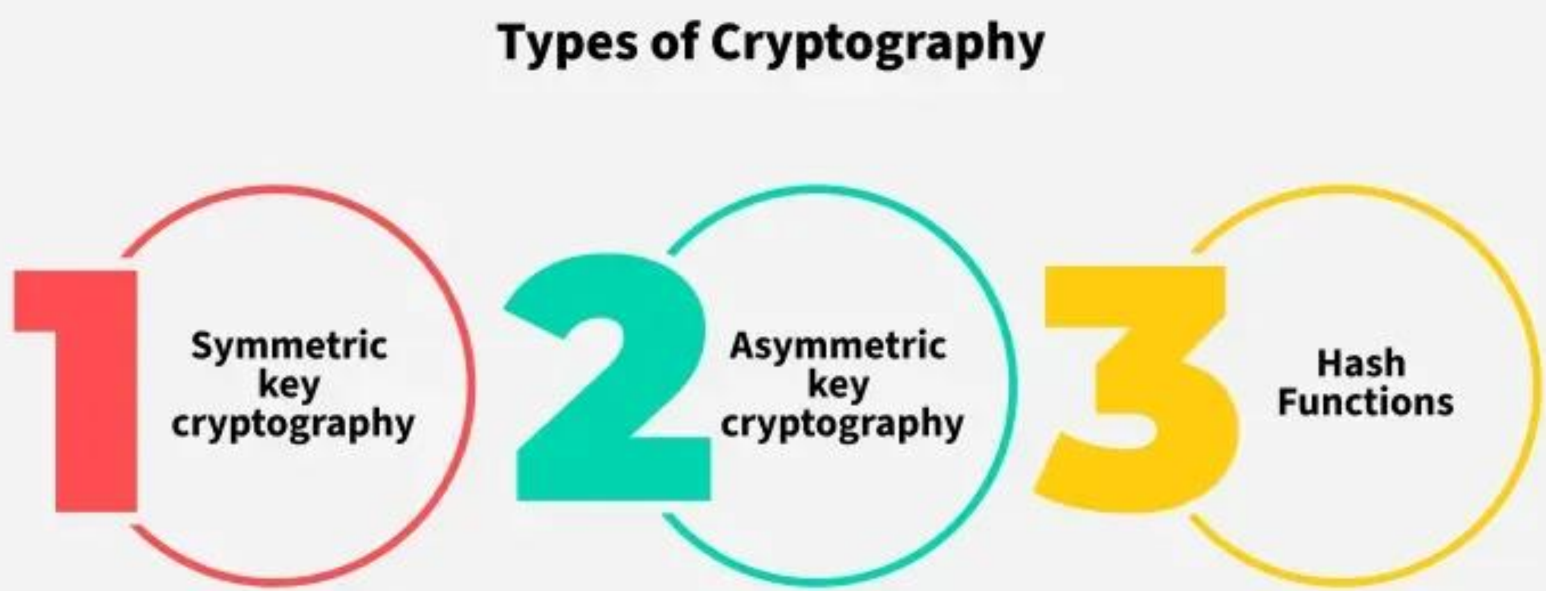
Interoperability

Enables secure communication across diverse systems.

Adaptability

Continuously evolves to counter emerging threats.

Types


Types of Cryptography
1
Symmetric key cryptography
2
Asymmetric key cryptography
3
Hash Functions

Symmetric key crypt

Example - 1

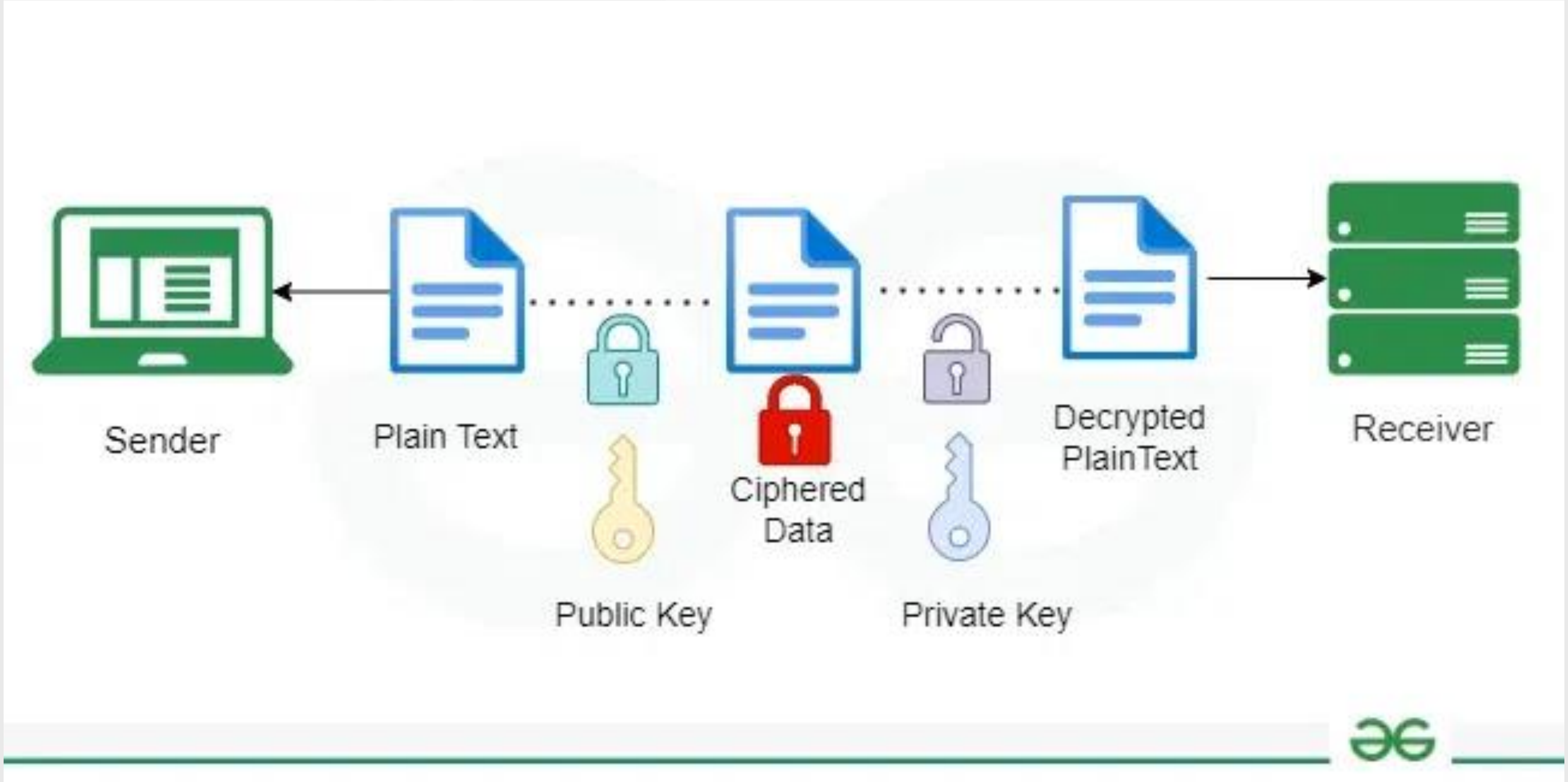
```python
# Generate a key (keep this secret!)
key = Fernet.generate_key()
cipher_suite = Fernet(key)
print(f"Key: {key.decode()}")

# Our secret message
message = b"Secret message for symmetric encryption"
print(f"Message: {message.decode()}")

# Encrypt
cipher_text = cipher_suite.encrypt(message)
print(f"Encrypted: {cipher_text.decode()}")

# Decrypt
plain_text = cipher_suite.decrypt(cipher_text)
print(f"Decrypted: {plain_text.decode()}")
```

✓ 0.0s

```
Key: b_9zSriOg1BekMgYl2s5DpKNxRYU0NaENXsqtqYN_m4=
Message: Secret message for symmetric encryption
Encrypted: gAAAAABogt44fp_Omd-T4YF7h4InhZb8OqXAzby7SmYT-gXmXY4CwWBU65jY7BnNeafjusFuAixqSeupte
Decrypted: Secret message for symmetric encryption
```

Q→NU

Asymmetric key crypt



Example - 2

```python
# Generate private key
private_key = rsa.generate_private_key(
    public_exponent=65537,
    key_size=2048,
)
# print(f"Private key: {private_key.private_bytes(
#     encoding=serialization.Encoding.PEM,
#     format=serialization.PrivateFormat.TraditionalOpenSSL,
#     encryption_algorithm=serialization.NoEncryption()
# ).decode()}")

# Get public key
public_key = private_key.public_key()

# Message to encrypt
message = b"Secret message for asymmetric encryption"

# Encrypt with public key
cipher_text = public_key.encrypt(
    message,
    padding.OAEP(
        mgf=padding.MGF1(algorithm=hashes.SHA256()),
        algorithm=hashes.SHA256(),
        label=None
    )
)
print(f"Encrypted: {base64.b64encode(cipher_text).decode()}")

# Decrypt with private key
plain_text = private_key.decrypt(
    cipher_text,
    padding.OAEP(
        mgf=padding.MGF1(algorithm=hashes.SHA256()),
        algorithm=hashes.SHA256(),
        label=None
    )
)
print(f"Decrypted: {plain_text.decode()}")
```

✓ 0.0s

```
Encrypted: dgjd/T1JdrRqeNeRABxUYLMarpwQHTLklak98ty/RDL6Z1k36ZWodJOHT+mFgx2lWlId7un7RkJGGv
Decrypted: Secret message for asymmetric encryption
```

Hash Functions

Example - 3

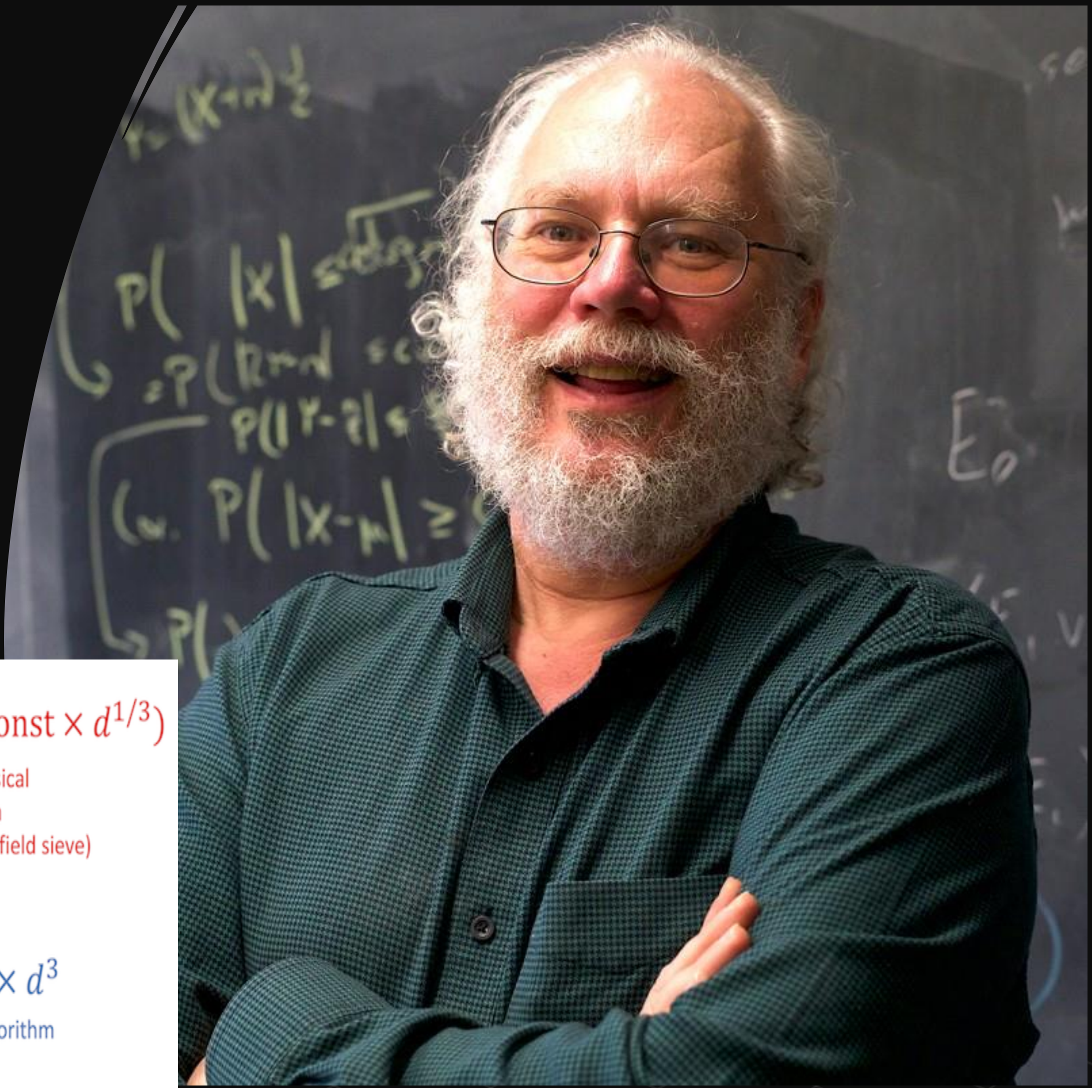
```python
# Create a hash object
digest = hashes.Hash(hashes.SHA256(), backend=default_backend())

# Add data
data = b"Data to hash"
digest.update(data)

# Get the hash value
hash_value = digest.finalize()
print(f"SHA-256 hash of '{data.decode()}': {hash_value.hex()}")
```

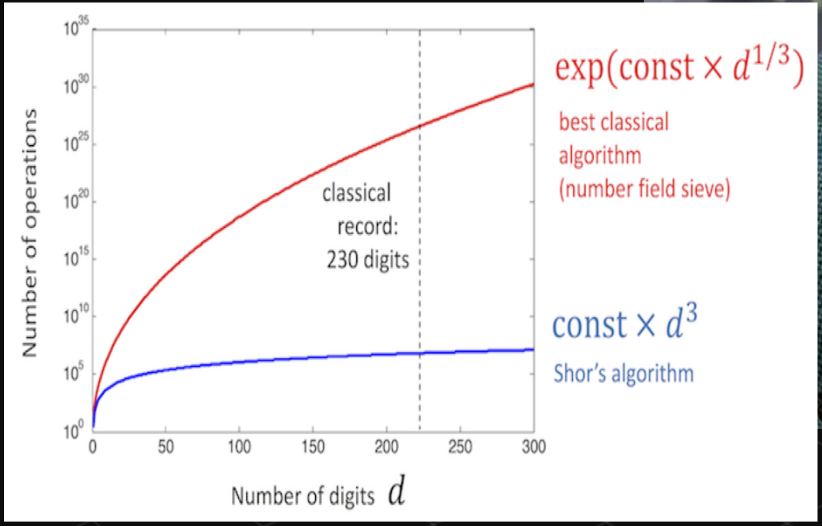
✓ 0.0s

```
SHA-256 hash of 'Data to hash': edd63071745ec4beafe0b3a80ba2c009c73e042e863003268680ce278f8b4b97
```

Is Classical Cryptography secure?

Shor's Algorithm





https://quantum-computing.ibm.com/composer/docs/iqx/guide/shors-algorithm

https://www.nature.com/articles/d41586-020-03068-9

Ideal Security Requirements

Q→NU

Challenges with current Cryptography

Today, Digital India is entirely based on computational security which will fall like a pack of cards:

Due to Exponential increase in computational resources available due to Quantum computers and increased computational power for Crypto Analysis

Vulnerabilities on the physical link like Eaves-dropping and other attacks through the physical architecture cannot be detected.

Backdoors are present in the classical algorithms which can be easily cracked with increasing computational power. Many backdoors will open once Quantum computers are here.

Ideal security scenario

"Security is as good as the key"

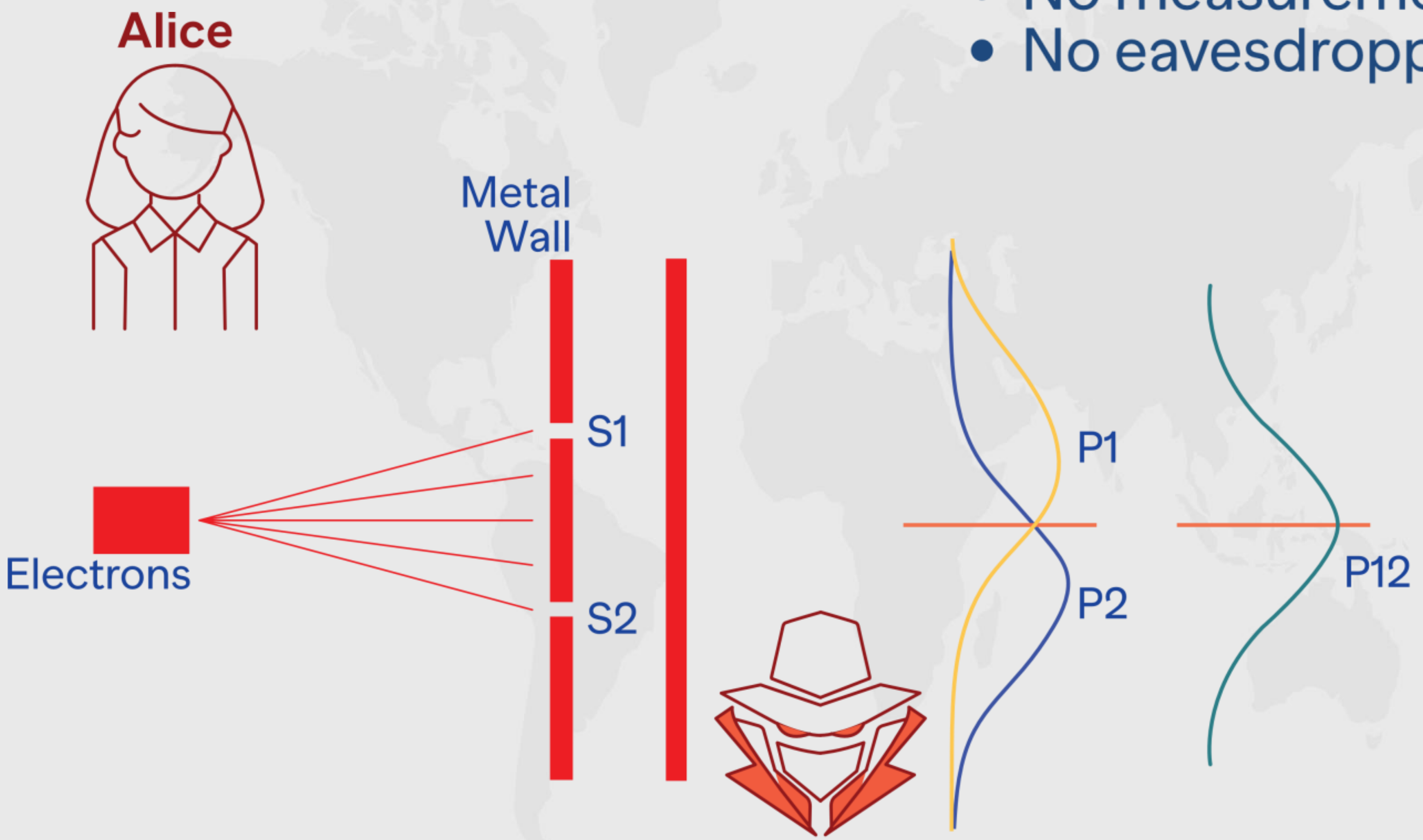
The key should not be copied

The keys should be truly random

Key tampering should be detectable

Quantum principle guaranteeing Ideal security

- Measurement perturbs the state
- No peturbation implies
  - No measurement
  - No eavesdropping



Q→NU

No-cloning theorem

A unknown quantum state cannot be cloned

Consider an unknown state: $|\alpha\rangle$

Quantum cloning machine: $|\alpha\rangle\,|0\rangle \mapsto |\alpha\rangle\,|\alpha\rangle$

Suppose such a machine exists then the following would happen

Orthogonal state: $|0\rangle\,|0\rangle \mapsto |0\rangle\,|0\rangle$

Orthogonal state: $|1\rangle\,|0\rangle \mapsto |1\rangle\,|1\rangle$

General state:

$$|\alpha\rangle\,|0\rangle \mapsto (a\,|0\rangle + b\,|1\rangle)\,|0\rangle$$

$$|\alpha\rangle\,|0\rangle \mapsto a\,|0\rangle\,|0\rangle + b\,|1\rangle\,|1\rangle \qquad (1)$$

But for cloning a general state would mean

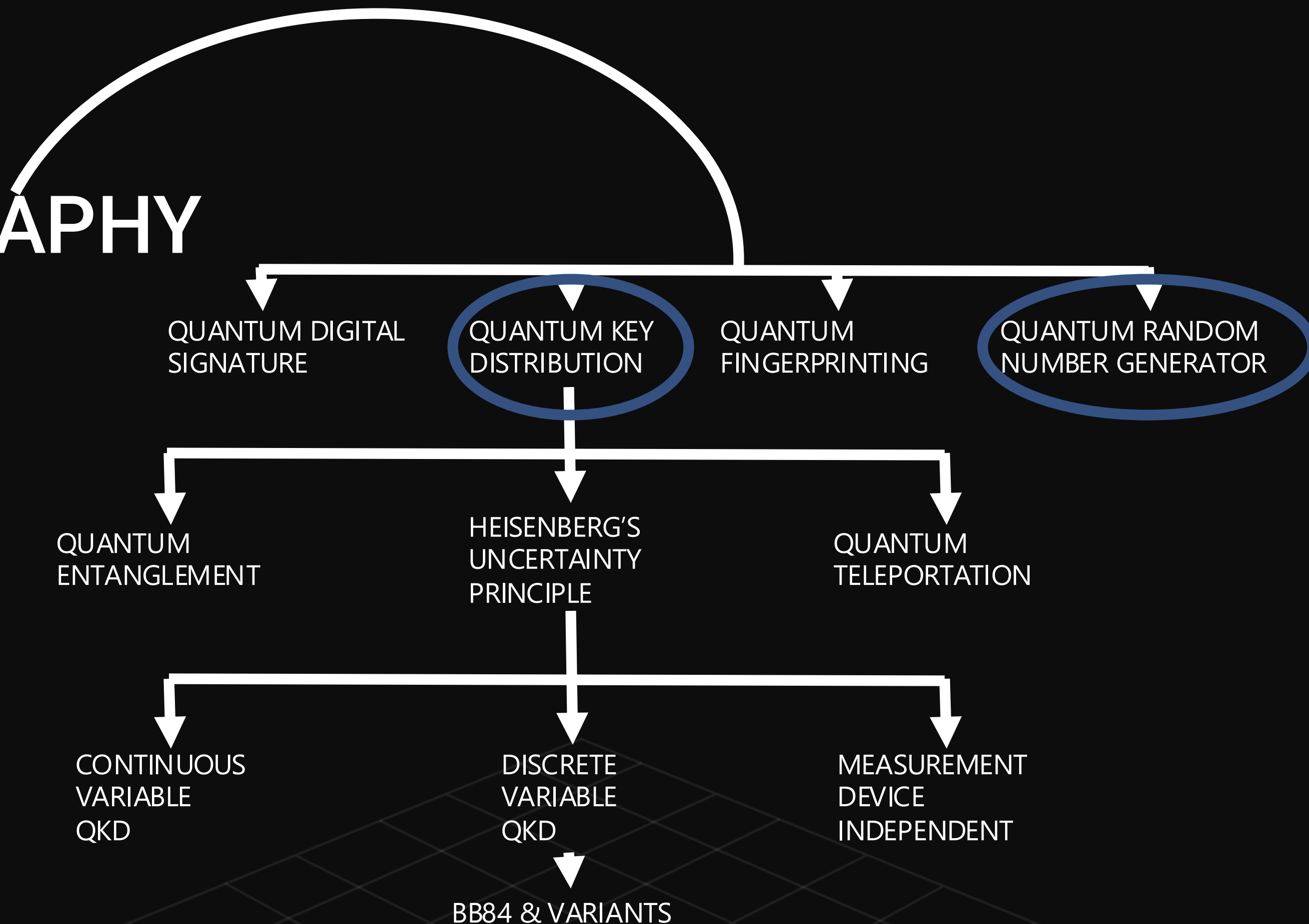
$$|\alpha\rangle\,|0\rangle \mapsto (a\,|0\rangle + b\,|1\rangle)\,(a\,|0\rangle + b\,|1\rangle)$$

$$|\alpha\rangle\,|0\rangle \mapsto a^2\,|0\rangle\,|0\rangle + b\,|^2 1\rangle\,|1\rangle + ab\,|0\rangle\,|1\rangle + ab\,|1\rangle\,|0\rangle \qquad (2)$$

If $ab \neq 0$ then (1) and (2) are different

Quantum cloning not possible

QUANTUM CRYPTOGRAPHY



Q→NU

Randomness

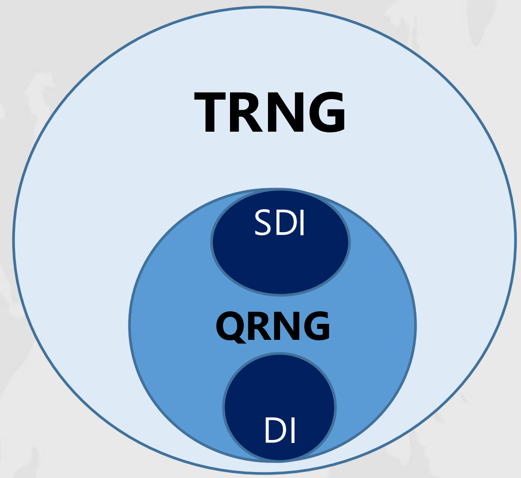
I.I.D	Unpredictable	Uniform
Unbiased	Asymptomatic	Uncompressable

The impossibility of compression, independently of the algorithm used for compression, can actually be considered the defining trait of randomness, according to the approach to complexity by Gregory Chaitin and Andrey Kolmogorov

Random number generator (RNG)

Algorithmic

Physical process



PRNG: Pseudo random number generator

TRNG: True random number generator
QRNG: Quantum random number generator

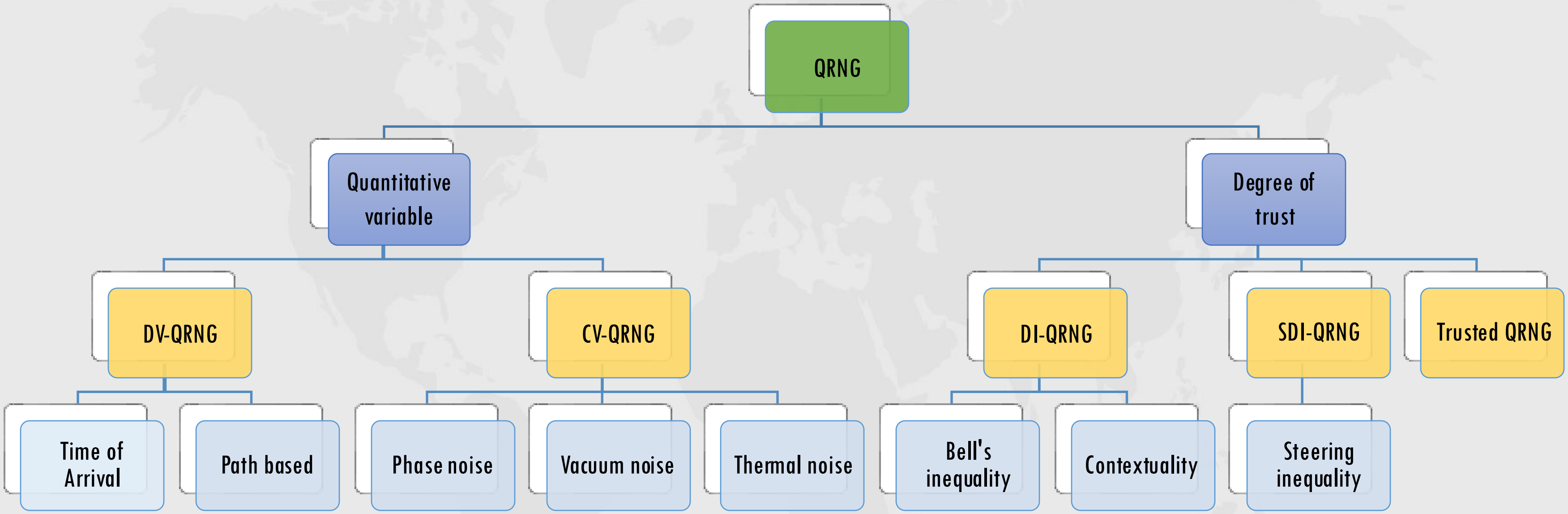
Q→NU

Classical TRNG vs QRNG

Property	Classical TRNG	Quantum RNG
Entropy source	Complexity of physical processes and partial ignorance.	Fundamental and intrinsic randomness of quantum mechanical events and measurements.
Quality of entropy	Various degrees. The underlying process used as entropy source may work in a physical regime where there are large bias and relatively high correlations (i.e., small entropy).	High entropy from the start based on the simple design of the source.
Generation of true random numbers	No or unknown (only appear to be random due to ignorance)	Yes
Validation of the randomness	One can never fully monitor the physical process, nor prove that it is secure.	Live monitoring of the entropy source is very effective and is provably secure.
Presence of bias	Prone to bias and has to be corrected with post-processing algorithms.	The bias, if any, is negligible.
Robustness	Some ability to run health check on entropy source.	Built-in check based on the simplicity of the process and more sensitive to tampering.
Vulnerable to quantum computers	Yes	No
Ease of certification	Limited ability to certify the underlying physical process, which is inherently a complex one. Certification of the quality of the output based on standard tests.	Can validate the underlying physical processes. Certification of the quality of the output based on standard tests.

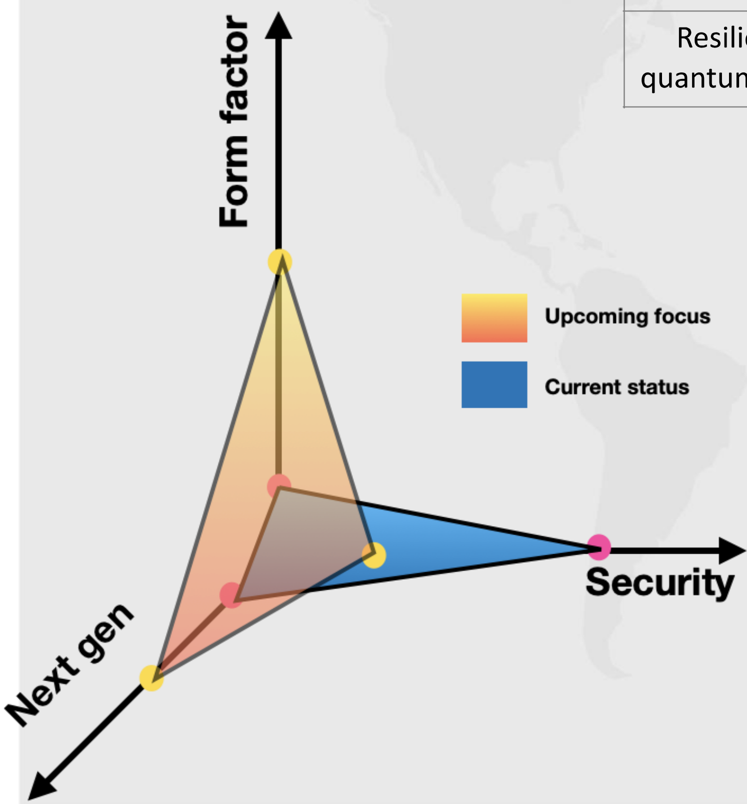
Hence, QRNG is better approach

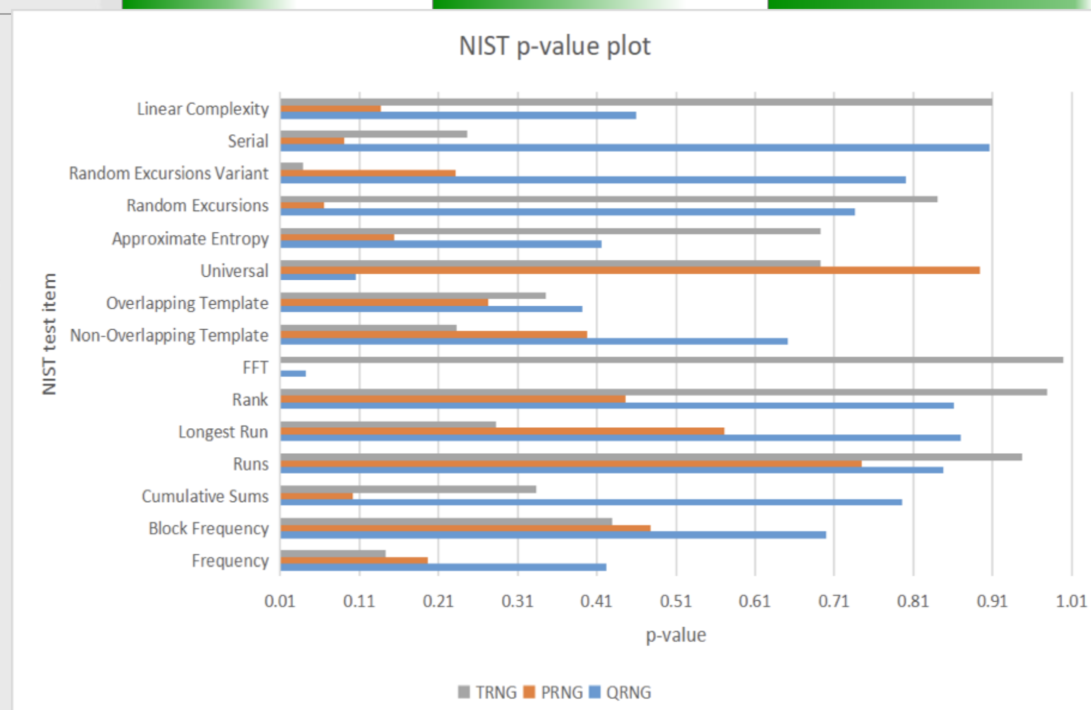
Types of QRNG



RNG comparison

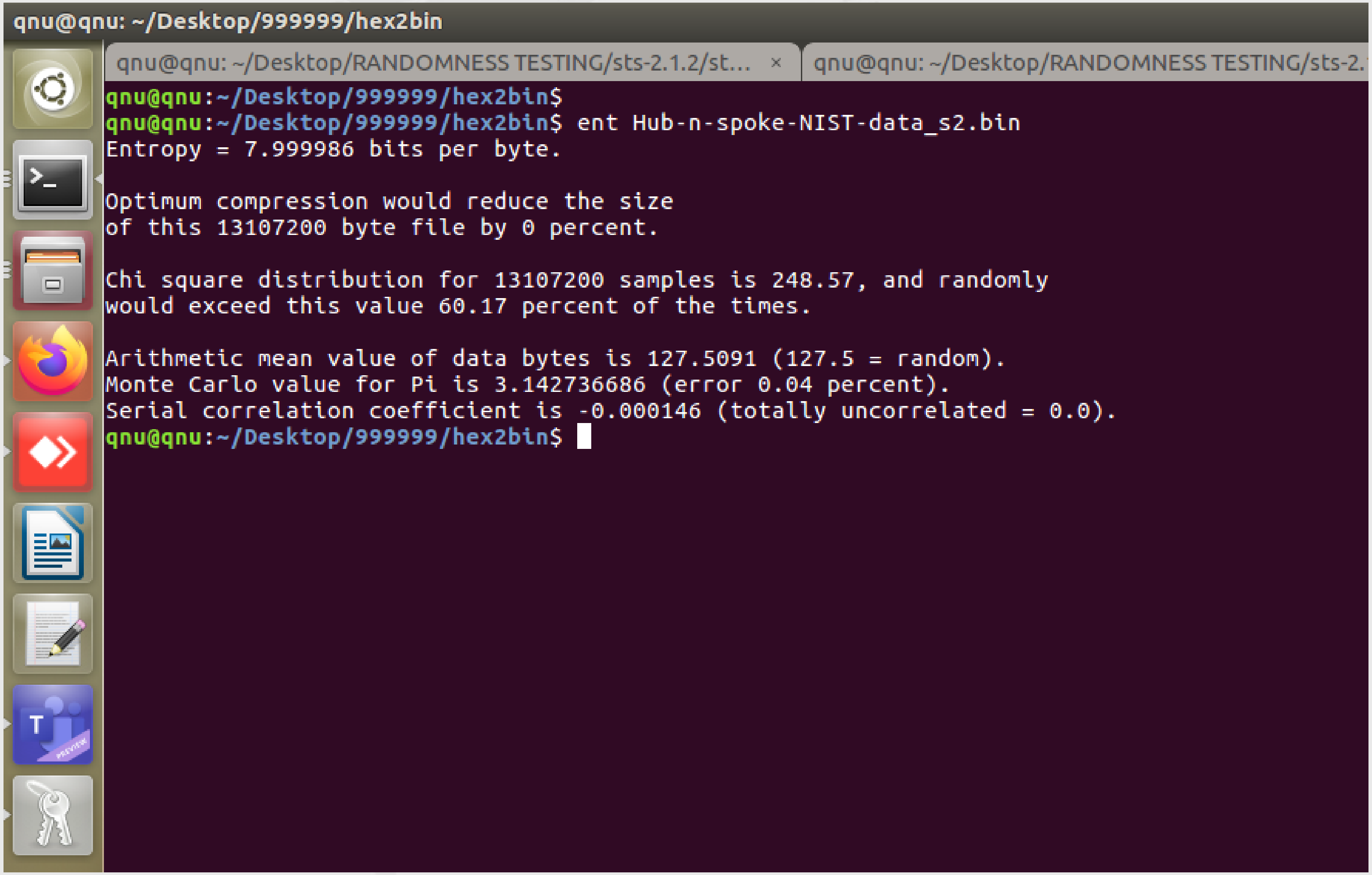
Characteristic	PRNG	TRNG	QRNG
Entropy source	NA		
Randomness Test-suites			
Health status	NA		
Robustness (external noise)	NA		
Resilient to quantum attacks			





Randomness tests

ENT testsuite



Q→NU

NIST 800 series

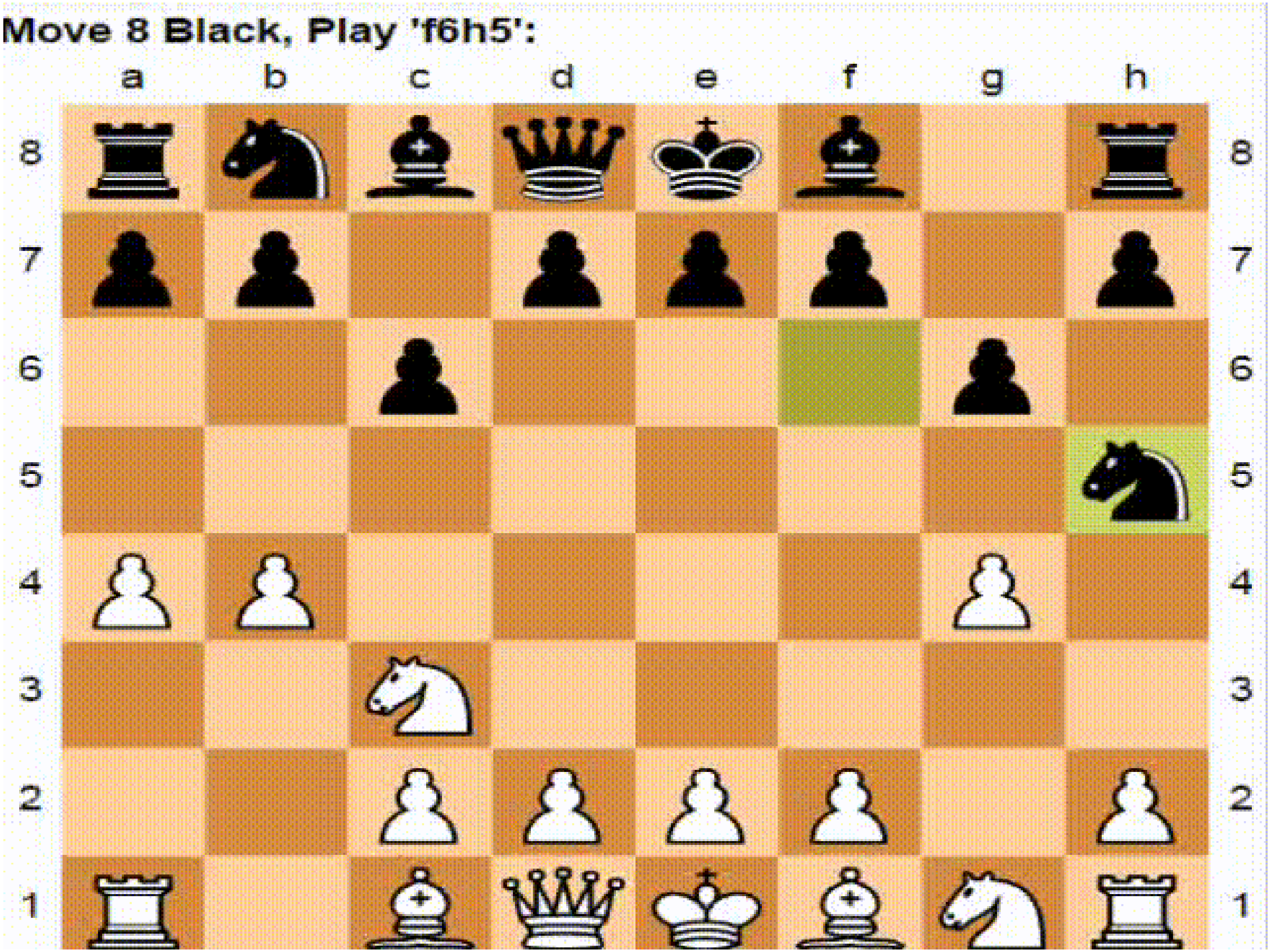
RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES

generator is <Hub-n-spoke-NIST-data_s2.txt>

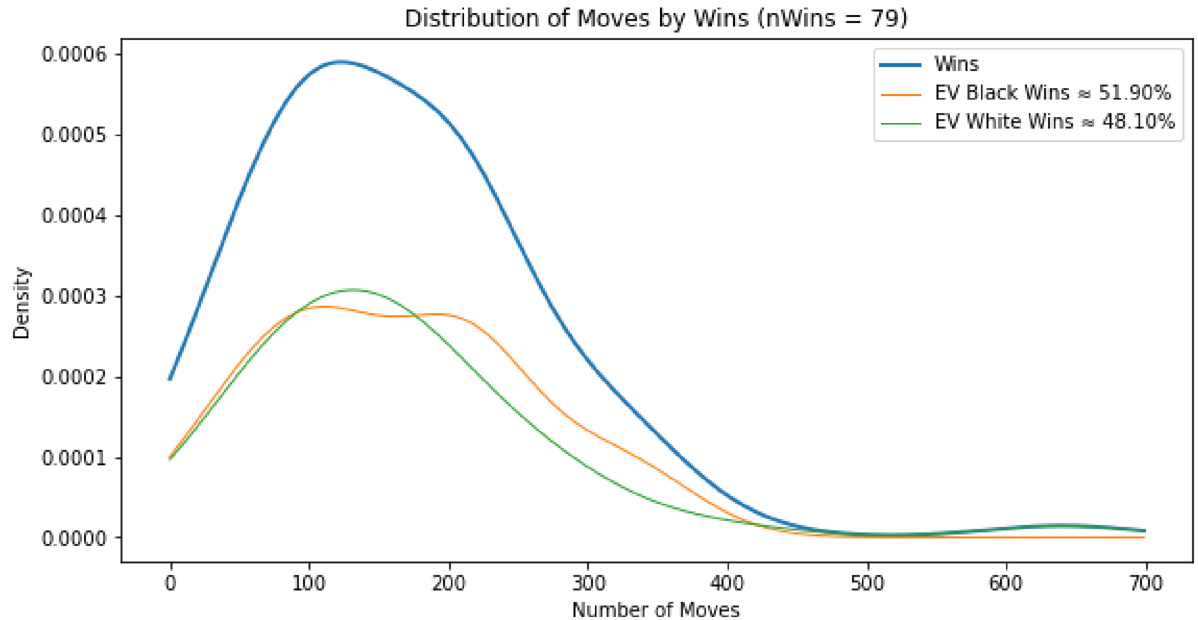
C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST
13	7	9	15	6	8	11	14	7	10	0.437274	100/100	Frequency
7	7	10	9	10	10	5	11	13	18	0.224821	97/100	BlockFrequency
15	11	7	14	10	7	14	5	10	7	0.275709	100/100	CumulativeSums
17	8	8	9	7	9	13	10	6	13	0.334538	99/100	CumulativeSums
11	10	3	10	19	8	12	9	6	12	0.066882	100/100	Runs
10	12	14	13	13	5	7	7	10	9	0.514124	98/100	LongestRun
14	9	9	10	11	5	6	10	15	11	0.474986	100/100	Rank
7	14	6	10	11	10	8	13	10	11	0.779188	99/100	FFT
8	20	7	10	13	9	13	3	11	6	0.019188	100/100	NonOverlappingTemplate
13	4	7	12	11	6	18	9	10	10	0.122325	99/100	NonOverlappingTemplate
10	6	10	14	6	9	10	12	13	10	0.719747	99/100	NonOverlappingTemplate
11	10	10	14	5	10	10	5	11	14	0.494392	100/100	NonOverlappingTemplate
8	7	11	14	11	7	9	16	8	9	0.514124	99/100	NonOverlappingTemplate
12	6	10	13	7	12	8	11	12	9	0.816537	100/100	NonOverlappingTemplate
11	11	12	8	13	9	9	7	11	9	0.955835	99/100	NonOverlappingTemplate
12	10	6	8	6	10	9	16	10	13	0.474986	98/100	NonOverlappingTemplate
10	9	9	11	10	10	5	12	12	12	0.911413	98/100	NonOverlappingTemplate
10	12	11	10	5	8	13	8	13	10	0.779188	98/100	NonOverlappingTemplate
8	10	11	9	9	9	7	12	12	13	0.946308	100/100	NonOverlappingTemplate
8	10	14	4	13	7	15	8	10	11	0.319084	99/100	NonOverlappingTemplate
13	12	13	7	8	11	11	8	9	8	0.867692	97/100	NonOverlappingTemplate
8	15	8	9	9	10	15	10	8	8	0.657933	99/100	NonOverlappingTemplate
13	18	2	7	17	6	8	13	5	11	0.002971	98/100	NonOverlappingTemplate
11	10	8	7	9	10	8	13	13	11	0.924076	99/100	NonOverlappingTemplate
12	10	7	12	10	15	15	6	7	6	0.289667	99/100	NonOverlappingTemplate
13	7	8	9	11	10	13	7	9	13	0.816537	96/100	NonOverlappingTemplate
6	13	17	11	8	11	9	9	9	7	0.419021	100/100	NonOverlappingTemplate

Visualising randomness

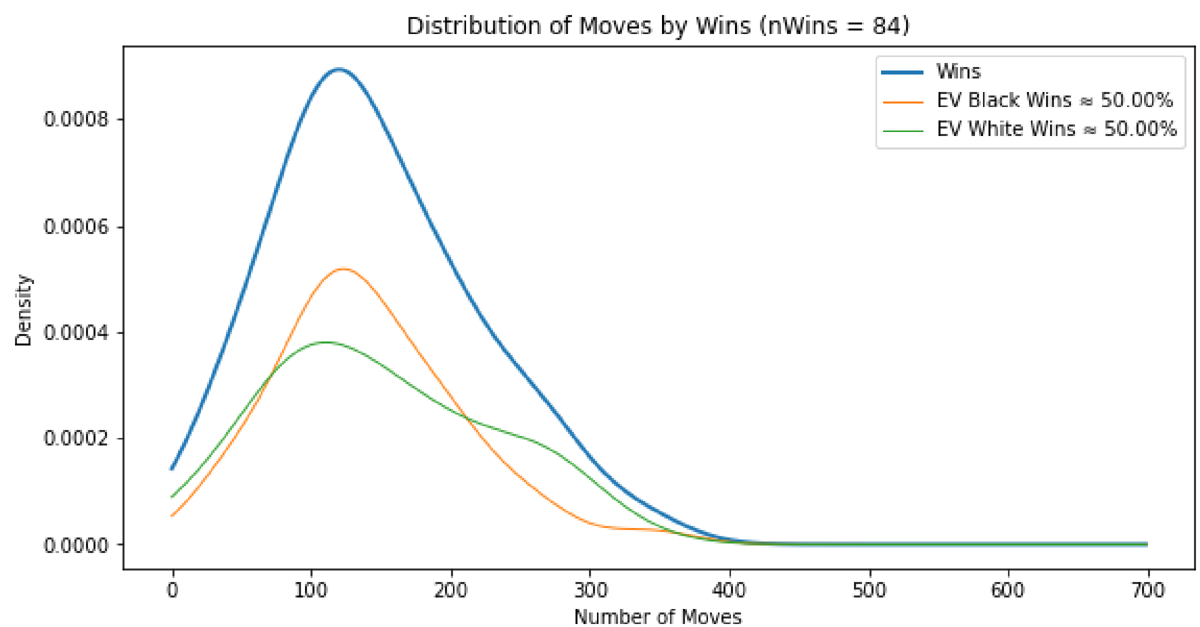
Random Moves vs Random Moves:



PRNG


Distribution of Moves by Wins (nWins = 79)
Wins
EV Black Wins ≈ 51.90%
EV White Wins ≈ 48.10%
Density
0.0000
0.0001
0.0002
0.0003
0.0004
0.0005
0.0006
0
100
200
300
400
500
600
700
Number of Moves

QRNG


Distribution of Moves by Wins (nWins = 84)
Wins
EV Black Wins ≈ 50.00%
EV White Wins ≈ 50.00%
Density
0.0000
0.0002
0.0004
0.0006
0.0008
0
100
200
300
400
500
600
700
Number of Moves

Quantum Key Distribution Demystified (to be continued..)