

Abstract

Conventional cryptography methods, such for AES (encryption), SHA-256 (hashing) and RSA/Elliptic Curve (signing), work well on systems which have reasonable processing power and memory capabilities, but these do not scale well into a world with embedded systems and sensor networks. Thus, lightweight cryptography methods are proposed to overcome many of the problems of conventional cryptography. This includes constraints related to physical size, processing requirements, memory limitation and energy drain. This paper outlines many of the techniques that are defined as replacements for conventional cryptography within an Internet of things space and discusses some trends in the design of lightweight algorithms.

While AES and SHA work well together within computer systems, they struggle in an Internet of things (IoT)/embedded world as they take up: too much processing power; too much physical space; and consume too much battery power. In the last decade, a large number of lightweight cryptography primitives have been proposed and used over resource-limited devices. Both the national (NIST) and international (ISO/IEC) organizations outline a number of methods which can be used for lightweight cryptography and which could be useful in IoT and RFID devices. They define the device spectrum as follows:

Conventional Cryptography. Servers and Desktops; Tablets and smartphones.

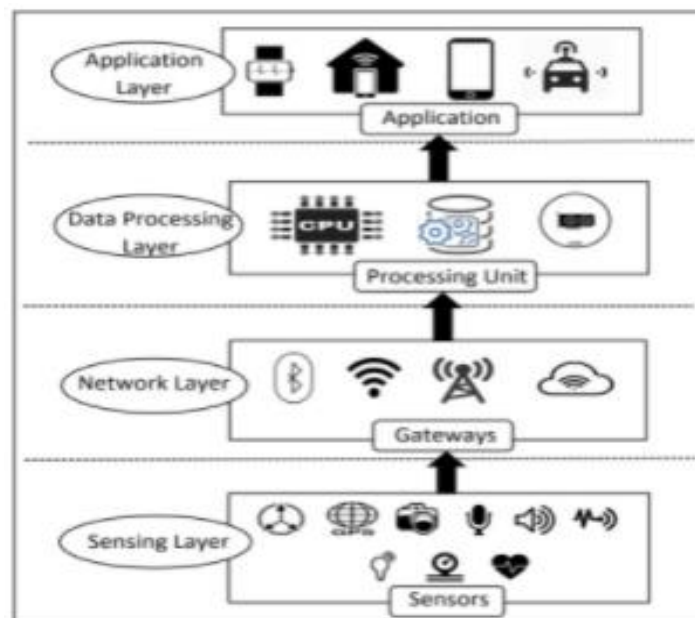
Lightweight Cryptography. Embedded Systems; RFID and Sensor Networks.

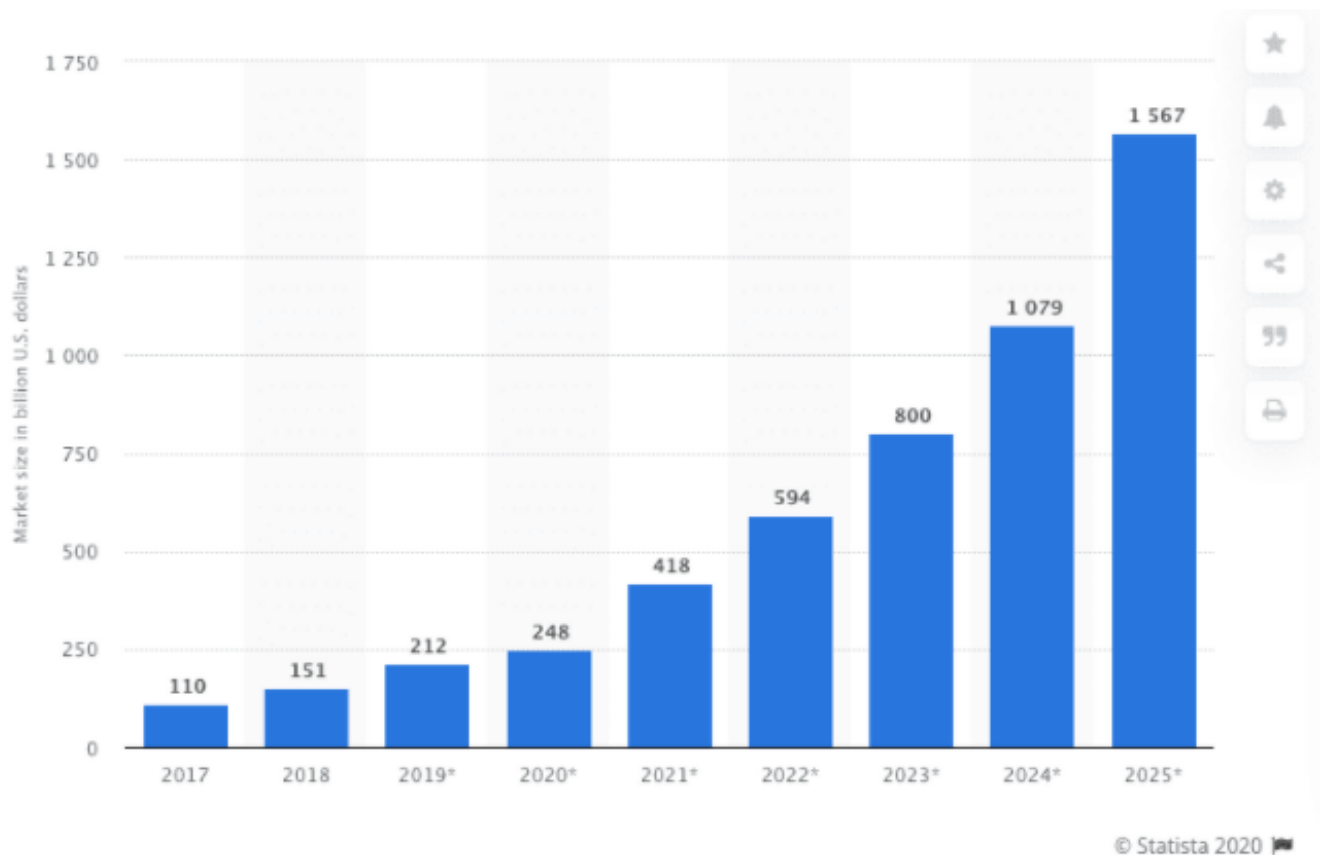
Internet of Things (IoT)

The “**IoT**” is a technique for interrelated figuring gadgets, mechanical and computerized machines that are given matchless identifiers (UIDs) and the capacity to move information over an organization without expecting human-to-human or human-to-PC cooperation.

Regularly IoT structures are partitioned into the three-layer design: -

1. Physical layer
2. Compensation layer
3. Application layer





The global market for the Internet of things (IoT) reached \$100 billion in revenue for the first time in 2017, and forecasts suggest that this figure will grow to around \$1.6 trillion by 2025. With such a prognosis, the technology is predicted to step far ahead than anyone can possibly imagine. But with the rise in popularity of IoT devices, there will be a rise in IoT app development as well as security challenges and issues.

Edge Computing

Computing that takes place at or near the physical location of either the user or the source of the data can be referred to as Edge Computing. By placing computing services closer to these locations, users benefit from faster, more reliable services while companies benefit from the flexibility of hybrid cloud computing. Edge computing is one way that a company can use and distribute a common pool of resources across a large number of locations.

Given the imminent development of the Internet of Things and its applications in real-time, Edge Computing has become a major interest as a solution to the process and rapid response, rather instantaneously, which means the main trigger for the development of technological trends that are around the corner. Edge computing can mean faster, more stable services at a lower cost. For users, edge computing means a faster, more consistent experience. For enterprises and service providers, edge means low-latency, highly available apps with real-time monitoring.

Security Concerns

The security issue is troublesome and an investigation subject everyone should be concerned about in IoT. Web of things has been widely applied for home, industry, and various applications.

As the IoT detecting gadgets are low controlled, heavyweight security calculations for information security are not plausible. Additionally, the detecting gadgets are not under management so they can be just intervened by gatecrashers and listening in can be conceivable. independently from that the validation, confirmation, information uprightness and different organization assaults, for example, satirizing, DDoS, blockage and protecting are a significant security concern. IoT application is helpful to individuals yet the IoT framework can't shield the client information from programmers, assaults, and weaknesses.

As the gadgets/individual can be handily followed by the detecting gadgets so the protection and privacy for an individual/object can't be safeguarded. the legitimate strategy must be concluded preceding the gadgets. And yet, it is hard to deal with the sensors that data ought to be gathered and by whom.

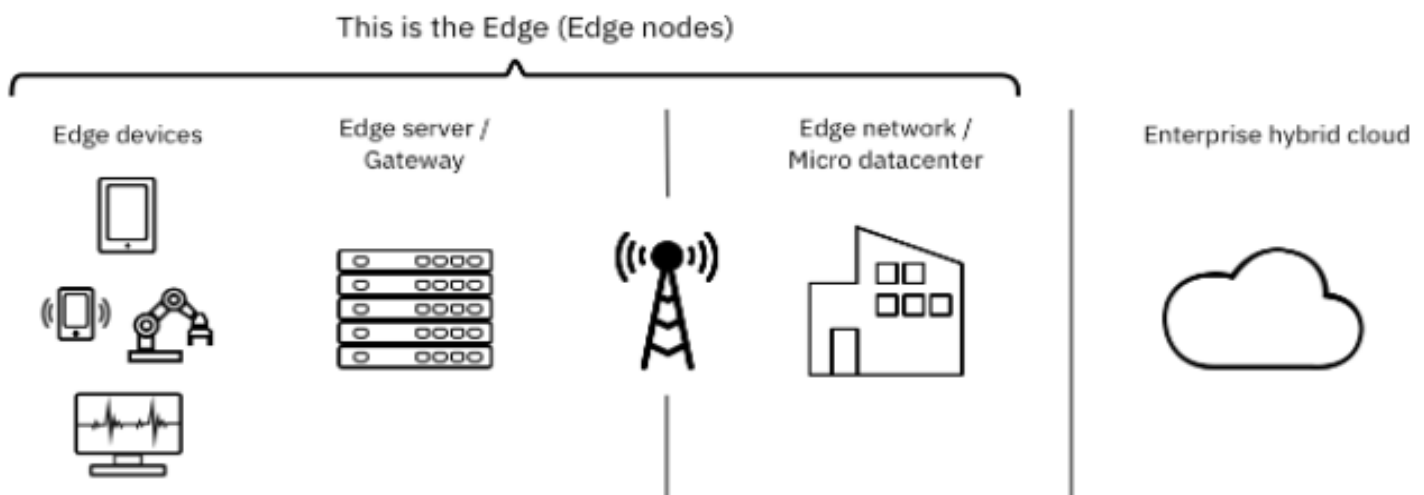
Computing Architecture

There are two categories of computing technology devices, including edge computing devices and edge computing-based IoT devices & technologies. The two categories vary in capabilities and services, but core features, e.g., processing, artificial intelligence model, complex event processing, and data management, differentiate these systems. The edge computing architecture has three levels.

(i) The first level holds IoT devices that can communicate with each other remotely using remote pooling techniques before being transferred to edge servers.

(ii) The second level is referred to as the edge server platform that acts as the access point of the network resources, e.g. computation and storage. These level reduces latency and improves load processing.

(iii) The third level is the infrastructure based cloud, where data computation and processing occur. The data generated by sensors and IoT devices are also stored in this level, and any other device of another level can virtually access the data.



Scenarios	Requirements				
	Scalability	Multi-tenancy	Privacy & Security	Latency	Extensibility (Open API)
Autonomus Vehicles	<i>Non critical</i>	<i>Non critical</i>	<i>Critical. Autonomous vehicles possess sensitive information about the user. Moreover, the constant need of sensors data for navigation make cars a primary target for malicious users.</i>	<i>Critical. Cars have strict real-time requirements</i>	<i>Non critical. Each car manufacturer will probably run exclusively their own software to ensure security and reliability.</i>
Augmented Reality	<i>Critical</i>	<i>Critical</i>	<i>Critical when processing sensitive multimedia streams.</i>	<i>Critical. AR applications require real-time information feed to ensure a smooth and acceptable experience.</i>	<i>Critical. Open API are important in this case to enable new services and features.</i>
Smart Sensors Networks	<i>Critical</i>	<i>Critical due to the number of potential users.</i>	<i>Depends on the specific Smart context (for Smart Health it's critical but not for Smart Environment). Strict control over which data can be public is required.</i>	<i>Depends. For example, in the case of Machine Type Communications (MTC) it's critical.</i>	<i>Critical to enable the creation of an 'IoT Marketplace' where developers can offer new and innovative application exploiting collected data.</i>
Smart Grid	<i>Non critical. Data and messages are exchanged at a fixed, predefined rate.</i>	<i>Non critical. The infrastructure is usually controlled by a single provider.</i>	<i>Critical. Disclosure and analysis of energy consumption information can lead to user profiling and tracking.</i>	<i>Critical especially for messages as Phasor Measurement Unit (PMU) or Advanced Metering Infrastructure (AMI).</i>	<i>Non critical</i>
E-Health	<i>Critical. IoT healthcare networks must be able to meet the growing demand of services from both individuals and health organizations.</i>	<i>Critical as multiple healthcare organizations and/or heterogeneous IoT medical devices could share the same network infrastructure.</i>	<i>Critical. IoT-edge medical devices deal with personal health data, which need to be securely stored. Integrity, privacy, and confidentiality must be kept.</i>	<i>Depends. It's critical in use-cases as remote surgery. Nevertheless, response time can be acceptable in other scenarios.</i>	<i>Critical to support new application able to offer a more accurate patients health condition monitoring.</i>
Distributed Surveillance	<i>Critical. Several control units are needed in order to grant the system of better usability and robustness.</i>	<i>Non-critical. A single provider usually controls the infrastructure.</i>	<i>Critical considering the sensitive information handled.</i>	<i>Critical to promptly identify suspects or recognize on-going crimes.</i>	<i>Non-critical. Same as Autonomous vehicles.</i>
Big Data Analytics	<i>Critical. A big data analytics system must be able to support very large datasets. All the components must be scalable to accommodate the constantly growing amount of data to be handled.</i>	<i>Critical. A single Big Data system has to be able to co-locate different use cases, applications, or data sets.</i>	<i>Critical. Users share large amount of personal data and sensitive content through their personal devices towards applications (e.g., social networks) and public clouds. Equipping Big Data systems of secure frameworks capable to store and manage user data with high sensitiveness represents a critical aspect.</i>	<i>Non critical</i>	<i>Critical to improve and deploy different algorithms and tools.</i>
Network Function Virtualization (NFV)	<i>Critical. Demand of new services is high and constantly growing.</i>	<i>Critical. Resources are shared among customers. A large number of multi-tenant networks run over a physical network.</i>	<i>Critical. The use of additional software (e.g., hypervisors, containers or unikernels) extends the chain of trust. Resource pooling and multi-tenancy bring further security/privacy threats.</i>	<i>Critical. NFV need to leverage real-time delivery services. NFV introduces additional sources of latency through the virtualization layer.</i>	<i>Non critical</i>

Novel Algorithm

In this project we will be proposing an algorithm for hybrid cryptography which serves as a reliable and secure way to keep the data uninformed to the intruder or any other outside middle man. The project aims to take the data as input and produce a meaning secure files which could be stored separately on two different geological locations and be safe. While these files are accessed by the concerned authority or system, calculations and manipulations are again performed with the help of a key from the file itself to produce the original data from normal use. The derivation of the equation and its Encryption and Decryption algorithms along with the numerical instance are mentioned as a hand written document below.

We have accumulated multiple newly published papers with new and improved cryptographic algorithms for different scenarios, which are usually comparing themselves with the ones that they have improved upon or only with the most famous ones. So in this project we will also try to compare all the aspects of these algorithms (all that we can) with one that we have prepared. The comparing scenarios will include performance, security and resource utilization of these particular algorithms.

Equation —

Diophantine equation — (linear) $ax + by = c$

$\text{Pell's equation — } x^2 - ny^2 = \pm 1$ quadratic

Cubic Pell's equation

(i) $x^3 - dy^3 = 1$ for some non cubic integer d

reference — Cubic Pell's Equation by Johannes Hedberg

Derivation & key generation

Embedding RSA for finding public and private keys

→ while considering two values of p and q , we would easily be able to calculate and find the values of $e, \phi(n), d, n$

Considering the function $f(x, y) = 1$

Now $f(x + \phi(n), y + e) = 1$

∴ The chosen equation of cubic pell's states

$$\rightarrow (x + \phi(n))^3 - R(y + e)^3 = \alpha$$

$$\rightarrow x^3 + (\phi(n))^3 + 3x^2\phi(n) + 3x\phi(n)^2 - (R(y))^3 - (R(e))^3 - 3Ry^2e - 3Rye^2 = \alpha \quad \text{--- (1)}$$

multiplying mod $\phi(n)$ to both sides of equation (1)

we have

$$\begin{aligned} x^3 \bmod \phi(n) + (\phi(n))^3 \% \phi(n) + 3x^2 \phi(n) \% \phi(n) + 3x \phi(n)^2 \% \phi(n) \\ - Ry^3 \bmod \phi(n) - Re^3 \bmod \phi(n) - 3Ry^2e \bmod \phi(n) - 3Rye^2 \bmod \phi(n) \\ = \alpha \bmod \phi(n) \end{aligned}$$

Since

$$\left. \begin{aligned} (\phi(n))^3 \% \phi(n) &= 0 \\ 3x^2 \phi(n) \% \phi(n) &= 0 \\ 3x^2 (\phi(n))^2 \% \phi(n) &= 0 \end{aligned} \right\}$$

So

$$(x^3 - Ry^3 - Re^3 - 3Ry^2e - 3Rye^2) \bmod \phi(n) = \alpha \bmod \phi(n)$$

Since

$$x^3 - Ry^3 = 1$$

$$\therefore (1 - Re^3 - 3Ry^2e - 3Rye^2) \bmod \phi(n) = \alpha \bmod \phi(n)$$

$$1 \bmod \phi(n) = (\alpha + Re^3 + 3Ry^2e + 3Rye^2) \bmod \phi(n)$$

hence the public keys = (α, R, n)
private keys = (y, e, n)

✓✓

3. Encryption using public key.

$$C \cdot T_1 = M^{a \bmod \phi(n)} \bmod n$$

$$C \cdot T_2 = M^{R \bmod \phi(n)} \bmod n$$

or

Decryption using the stated derivation

$$\text{or } (C \cdot T_1) \cdot (C \cdot T_2)^{e^3 \bmod \phi(n)} \cdot (C \cdot T_2)^{3y^2 e \bmod \phi(n)}$$

$$(C \cdot T_2)^{3ye^2 \bmod \phi(n)} \bmod n$$

$$\text{or } M^{a \bmod \phi(n)} \cdot M^{R e^3 \bmod \phi(n)} \cdot M^{3R y^2 e \bmod \phi(n)} \cdot M^{3R y e^2 \bmod \phi(n)} \bmod(n)$$

$$\text{or } \left[M^{(a + R e^3 + 3R y^2 e + 3y R e^2) \bmod \phi(n)} \bmod n \right] = \left[M \bmod(n) \right]$$

Numerical Instances

Let p and q be the two numbers $\boxed{p=5 \text{ and } q=11}$

$$n = pq = 55$$

$$\phi(n) = (p-1)(q-1)$$

$$\phi(n) = 4 \times 10 = 40$$

Now e should be coprime to $\phi(n)$ and should be $1 < e < \phi(n)$

\therefore taking $e=3$

Using extended Euclidean theorem

q	$\phi(n)$	e	r	t_0	t_1	$t = t_0 - q t_1$
13	40	3	1	0	1	-13
3	3	1	0	1	-13	40
0	1	0	1	-13	40	-13

$$\therefore d = -13 + \phi(n) = -13 + 40 = \underline{\underline{27}}$$

Now the induction to $x^3 - Ry^3 = 1$ should be integral

One set is $(x, y, r) = (9, 2, 91)$

$$9^3 - 91(2^3) = 1$$

$$729 - 728 = 1$$

$$\therefore 1 = 1$$

Now calculating α we get

$$\alpha = (n + \phi(n))^3 - R(y+e)^3$$

$$\alpha = (9+40)^3 - 91(2+3)^3$$

$$\alpha = 49^3 - 91(5)^3$$

$$\alpha = 117649 - 11375$$

$$\boxed{\alpha = 106274}$$

Let Message be $M=8$

Encryption — $C.T_1 = 8^{\alpha \bmod \phi(n)} \bmod 55$

$$C.T_1 = 8^{106274 \bmod 40} \bmod 55$$

$$C.T_1 = 8^{34} \bmod 55 = \underline{\underline{4}}$$

Similarly $C.T_2 = 8^{91 \bmod 40} \bmod 55$

$$C.T_2 = 8^{11} \bmod 55 = \underline{\underline{52}}$$

Decryption

putting values in equation (2) that we derived.

$$\cancel{M} \quad (C.T_1) (C.T_2)^{e^3 \bmod \phi(n)} \cdot (C.T_2)^{3y^2 \bmod \phi(n)} \cdot (C.T_2)^{3yc^2 \bmod \phi(n)}$$

$$\cancel{M} \quad 4 \cdot (52)^{27 \bmod 40} \cdot 52^{36 \bmod 40} \cdot 52^{84 \bmod 40} \pmod{55}$$

$$\cancel{M} \quad 4 \cdot 52^{(27+36+14)} \bmod 55$$

$$\cancel{M} \quad \boxed{\underline{\underline{8}} = M}$$

ALGORITHM

1. So as mentioned above in the derivation we have generated the formulae which we are supposed to take into consideration.
2. Take the initial file which is to be encrypted
3. Read the file
4. Now based on the formulae generated encrypt the file no. by no.
5. Two different files are generated which is to be taken care of.
6. A key is generated for carrying out symmetric encryption as by the formulae
 - a. $\text{key} = \text{int}((\max(L1) ** 2 + \max(L2) ** 2) / (\max(L1) + \max(L2)))$
 - b. where L1 and L2 are the CT1 and CT2 taken under consideration.
7. File is passed through this formula
 - a. `ct1x.append(ct1a[i]^key)`
8. Later when the files are required to be decrypted again they are fed by the formulae
 - a. `ct2a.append(int(L2[i]) + int(mine2))`
 - b. `ct2x.append(ct2a[i]^key)`

IMPLEMENTATION

After the chosen Diophantine equation, we have validated the result and developed a python code for the same to implement it in real world.

CODE:

```
# importing necessary libraries
import random
import os
import time

# Deleting the file if already exists
from typing import TextIO

def delete():
    if os.path.exists("file1.txt"):
        os.remove("file1.txt")
    if os.path.exists("ct1.txt"):
        os.remove("ct1.txt")
    if os.path.exists("ct2.txt"):
        os.remove("ct2.txt")
    if os.path.exists("dt.txt"):
        os.remove("dt.txt")
    if os.path.exists("ct1k.txt"):
        os.remove("ct1k.txt")
    if os.path.exists("ct1x.txt"):
        os.remove("ct1x.txt")
    if os.path.exists("ct2k.txt"):
        os.remove("ct2k.txt")
    if os.path.exists("ct2x.txt"):
        os.remove("ct2x.txt")

# predefined values for generation of RSA cryptography
x = 9
y = 2
```

```

r = 91
p = 7
q = 19
n = p * q
phi = (p - 1) * (q - 1)
e = 3

# Opening of the text files
h = open("ct1", "w+")
g = open("ct2", "w+")
j = open("dt", "w+")

# Function for encryption
def process(m):
    alpha = (x + phi) ** 3 - r * (y + e) ** 3
    M = m
    ct1 = M ** (alpha % phi) % n
    ct2 = M ** (r % phi) % n
    h.write("%d " % ct1)
    g.write("%d " % ct2)

# Function for decryption
def decrypt(a, b):
    a = int(a)
    b = int(b)
    dt1 = (a * b ** (((e ** 3) % phi + (3 * y ** 2 * e) % phi + (3 * y * e ** 2) % phi) % phi)) % n
    j.write("%d " % dt1)

# Creating list for generating random numbers
list = list(range(n))

# Generating File file1
W = 2700 # Initialising value to generate value of different file size
f = open("file1", "w+")
for i in range(W):
    f.write("%d " % random.choice(list))
f.close()

# File Accessing file1
access_file = open("file1", 'r')
yourResult = [line.split(' ') for line in access_file.readlines()]
L = yourResult[0][0:W]
# -----

# ENCRYPTING INITIALLY TAKEN FILE
start = time.time()
for i in range(W):
    process(int(L[i]))
h.close()
g.close()
print("Encryption Period : " + str(time.time() - start))

access_file1 = open("ct1", 'r')
yourResult1 = [line.split(' ') for line in access_file1.readlines()]
L1 = yourResult1[0][0:W]
access_file2 = open("ct2", 'r')
yourResult2 = [line.split(' ') for line in access_file2.readlines()]
L2 = yourResult2[0][0:W]

```



```

# -----
for i in range(W):
    L1[i] = int(L1[i])
    L2[i] = int(L2[i])
# GENERATION OF KEYS FOR SYMMETRIC SECTION
key = int((max(L1) ** 2 + max(L2) ** 2) / (max(L1) + max(L2)))
# GENERATED KEY

# -----
# SYMMETRIC ENCRYPTION FOR CT1
ct1a = []
ct1x = []
mine = max(L1)
for i in range(W):
    ct1a.append(int(L1[i]) + int(mine))
print("key", key)
for i in range(W):
    ct1x.append(ct1a[i] ^ key)
minx = int(mine) ^ key
m = open("ct1k", "w+")
m.write("%d" % int(minx))
m.close()
k = open("ct1x", "w+")
for i in ct1x:
    k.write("%d " % int(i))
k.close()
"""print("L1[0]",L1[0])
print("mine",mine)
print("ct1a[0]",ct1a[0])
print("ct1x",ct1x[0])"""
# -----

# -----
# SYMMETRIC ENCRYPTION FOR CT2
ct2a = []
ct2x = []
mine2 = max(L2)
for i in range(W):
    ct2a.append(int(L2[i]) + int(mine2))
for i in range(W):
    ct2x.append(ct2a[i] ^ key)
minx2 = int(mine2) ^ key
m = open("ct2k", "w+")
m.write("%d" % int(minx2))
m.close()
k = open("ct2x", "w+")
for i in ct2x:
    k.write("%d " % int(i))
k.close()
"""print("L2[0]",L2[0])
print("mine2",mine2)
print("ct2a[0]",ct2a[0])
print("ct2x",ct2x[0])"""
# -----

# DECRYPTION PART
mind = int(minx) ^ key
for i in range(W):
    ct1x[i] = (ct1x[i] ^ key) - mind
mind2 = int(minx2) ^ key
for i in range(W):
    ct2x[i] = (ct2x[i] ^ key) - mind2

```

```

start2 = time.time()
for i in range(W):
    decrypt(ct1x[i], ct2x[i])
j.close()
print("Decryption Period : " + str(time.time() - start2))
access_file3 = open("dt", 'r')
yourResult3 = [line.split(' ') for line in access_file3.readlines()]
DT = yourResult3[0][0:W]
# FINAL VALUES ARE STORED IN THE DECRYPTED TEXT FILE

```

We can clearly observe that the final file i.e. DT and initial file i.e. File1 are similar Hence, the algorithm worked perfectly. Taking into consideration the average time for encryption and decryption for the file with different sizes we will be able to compare the time complexity of the traditional RSA to that of our modified version



file1 - Notepad

File Edit Format View Help

```

403 35 319 17 379 30 29 162 304 5 169 414 134 335 214 125 197 393 50 59 224 350 411 64 176 360
74 113 240 107 226 431 46 376 231 393 395 306 352 378 379 128 233 126 213 337 405 178 152 430 3
372 243 311 13 197 321 129 245 72 279 53 186 70 6 344 263 167 102 426 116 414 304 360 46 254 36
83 291 49 198 164 320 328 323 145 227 181 245 425 153 6 349 376 149 216 381 329 94 254 200 102
66 139 170 392 234 137 311 392 5 159 36 405 262 38 108 346 204 55 125 10 317 135 415 373 180 61
1 161 228 412 299 429 340 86 181 413 5 25 403 80 308 313 317 212 184 383 383 184 207 216 211 85
94 350 137 414 105 203 30 388 195 316 402 109 182 93 340 187 183 365 61 46 415 0 336 217 184 25
426 280 428 5 301 421 164 82 218 309 276 373 370 211 313 208 7 316 296 233 146 288 67 255 40 38
320 120 107 112 421 62 250 350 175 56 251 141 186 429 384 131 89 242 371 242 223 64 328 356 316

```



dt - Notepad

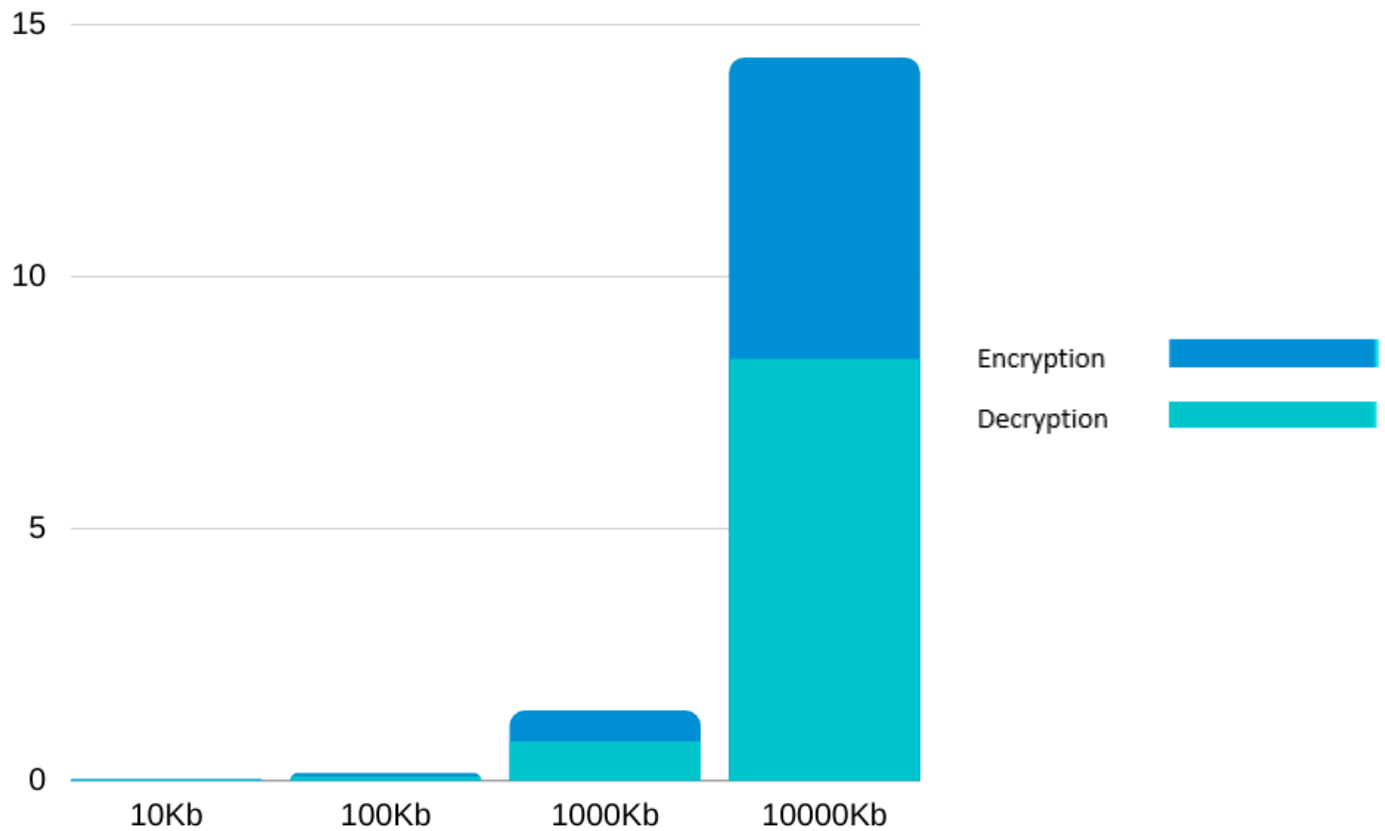
File Edit Format View Help

```

403 35 319 17 379 30 29 162 304 5 169 414 134 335 214 125 197 393 50 59 224 350 411 64 176 360
74 113 240 107 226 431 46 376 231 393 395 306 352 378 379 128 233 126 213 337 405 178 152 430 3
372 243 311 13 197 321 129 245 72 279 53 186 70 6 344 263 167 102 426 116 414 304 360 46 254 36
33 291 49 198 164 320 328 323 145 227 181 245 425 153 6 349 376 149 216 381 329 94 254 200 102
56 139 170 392 234 137 311 392 5 159 36 405 262 38 108 346 204 55 125 10 317 135 415 373 180 61
1 161 228 412 299 429 340 86 181 413 5 25 403 80 308 313 317 212 184 383 383 184 207 216 211 85
34 350 137 414 105 203 30 388 195 316 402 109 182 93 340 187 183 365 61 46 415 0 336 217 184 25
426 280 428 5 301 421 164 82 218 309 276 373 370 211 313 208 7 316 296 233 146 288 67 255 40 38
320 120 107 112 421 62 250 350 175 56 251 141 186 429 384 131 89 242 371 242 223 64 328 356 316

```

Size - Time Graph

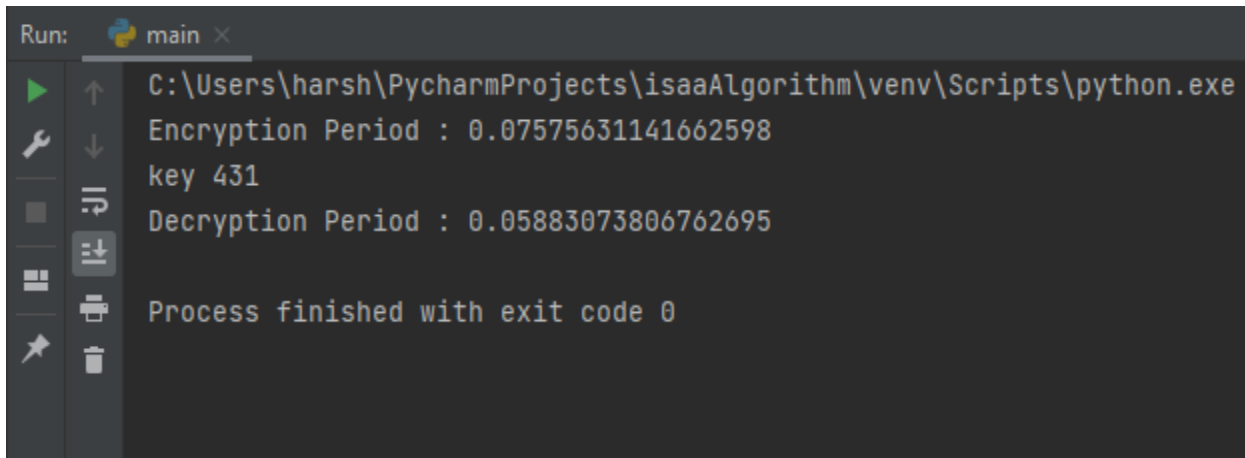


The estimated results for the time of encryption and decryption also includes the time for generating random numbers in our list to feed it to our algorithm. With predefined datasets the Data-Time graph would be much more accurate when compared to a randomly generated data set.

Screenshot When 10Kb file is generated and secured

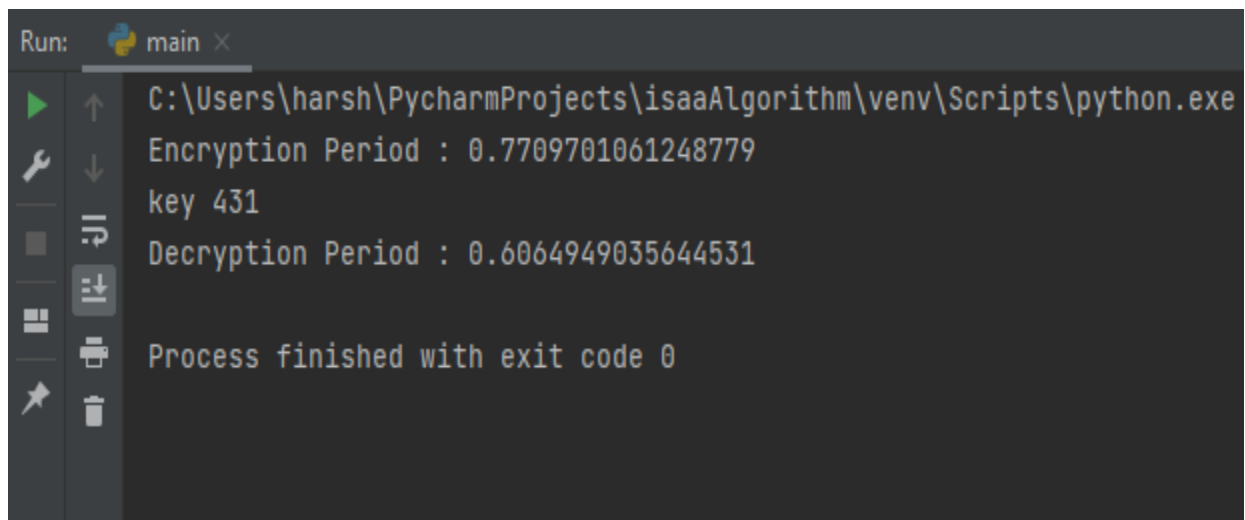
```
Run: main x
C:\Users\harsh\PycharmProjects\isaaAlgorithm\venv\Scripts\python.exe
Encryption Period : 0.008923053741455078
key 431
Decryption Period : 0.005967140197753906
Process finished with exit code 0
```


Screenshot When 100Kb file is generated and secured



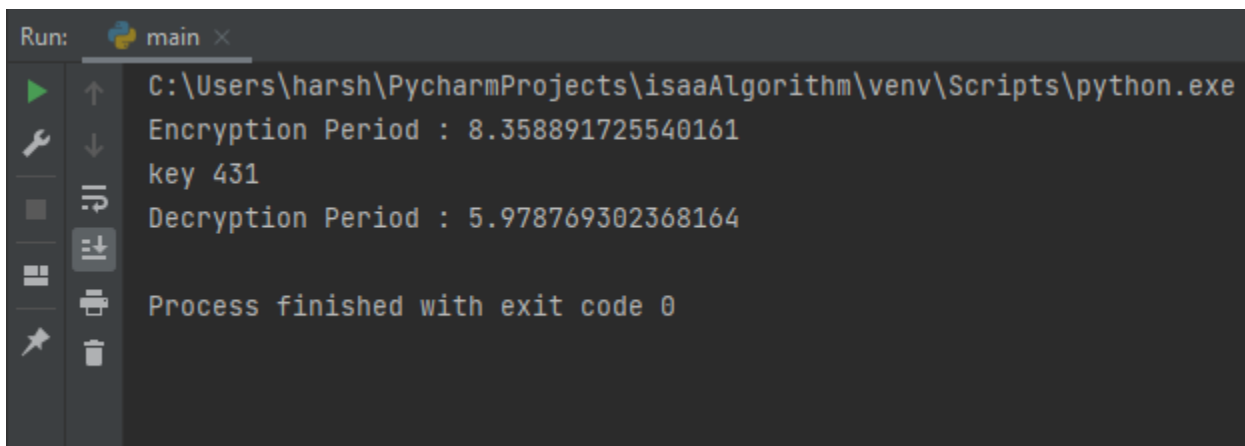
```
Run: main ×
C:\Users\harsh\PycharmProjects\isaaAlgorithm\venv\Scripts\python.exe
Encryption Period : 0.07575631141662598
key 431
Decryption Period : 0.05883073806762695
Process finished with exit code 0
```

Screenshot When 1Mb file is generated and secured



```
Run: main ×
C:\Users\harsh\PycharmProjects\isaaAlgorithm\venv\Scripts\python.exe
Encryption Period : 0.7709701061248779
key 431
Decryption Period : 0.6064949035644531
Process finished with exit code 0
```

Screenshot When 10Mb file is generated and secured



```
Run: main ×
C:\Users\harsh\PycharmProjects\isaaAlgorithm\venv\Scripts\python.exe
Encryption Period : 8.358891725540161
key 431
Decryption Period : 5.978769302368164
Process finished with exit code 0
```

ALGORITHM FOR TRADITIONAL RSA

1. Take the initial file which is to be encrypted
2. Calculate the value of n using the variables p and q where $n = p * q$
3. Calculate the value of Φ where $\Phi(n) = (p-1) * (q-1)$ (Euler's Totient Function)
4. Choose value of e where $1 < e < \Phi(n)$ and $\gcd(\Phi(n), e) = 1$
5. Calculate

$$d = \text{inverse}(e) \bmod(\Phi(n))$$

$$ed = \bmod(\Phi(n))$$

Finally, we need to find a value of d which gives $ed \bmod(\Phi(n)) = 1$. Can be found by hit and trial or extended Euclidean formula

Public key = $\{e, n\}$

Private key = $\{d, n\}$

6. Encryption $\rightarrow C = M^e \bmod(n)$

Decryption $\rightarrow M = C^d \bmod(n)$

CODE:

```
import random
import os
import time

def gcd(a, b):
    while b != 0:
        a, b = b, a % b
    return a

def delete():
    if os.path.exists("file1.txt"):
        os.remove("file1.txt")
    if os.path.exists("encrypted.txt"):
        os.remove("encrypted.txt")
    if os.path.exists("decrypted.txt"):
        os.remove("decrypted.txt")

def multiplicative_inverse(e, phi):
    d = 0
    x1 = 0
    x2 = 1
    y1 = 1
    temp_phi = phi

    while e > 0:
        temp1 = temp_phi // e
        temp2 = temp_phi - temp1 * e
        temp_phi = e
        e = temp2
```

```

    x = x2 - temp1 * x1
    y = d - temp1 * y1

    x2 = x1
    x1 = x
    d = y1
    y1 = y

if temp_phi == 1:
    return d + phi

def is_prime(num):
    if num == 2:
        return True
    if num < 2 or num % 2 == 0:
        return False
    for n in range(3, int(num**0.5)+2, 2):
        if num % n == 0:
            return False
    return True

def generate_key_pair(p, q):
    if not (is_prime(p) and is_prime(q)):
        raise ValueError('Both numbers must be prime.')
    elif p == q:
        raise ValueError('p and q cannot be equal')
    # n = pq
    n = p * q

    # Phi is the totient of n
    phi = (p-1) * (q-1)

    # Choose an integer e such that e and phi(n) are coprime
    e = random.randrange(1, phi)

    # Use Euclid's Algorithm to verify that e and phi(n) are coprime
    g = gcd(e, phi)
    while g != 1:
        e = random.randrange(1, phi)
        g = gcd(e, phi)

    # Use Extended Euclid's Algorithm to generate the private key
    d = multiplicative_inverse(e, phi)

    # Return public and private key_pair
    # Public key is (e, n) and private key is (d, n)
    return ((e, n), (d, n))

def encrypt(pk, plaintext):
    # Unpack the key into it's components
    key, n = pk
    # Convert each letter in the plaintext to numbers based on the character using a^b mod m
    cipher = [pow(ord(char), key, n) for char in plaintext]
    # Return the array of bytes
    return cipher

def decrypt(pk, ciphertext):

```



```

# Unpack the key into its components
key, n = pk
# Generate the plaintext based on the ciphertext and key using  $a^b \bmod m$ 
aux = [str(pow(char, key, n)) for char in ciphertext]
# Return the array of bytes as a string
plain = [chr(int(char2)) for char2 in aux]
return ''.join(plain)

if __name__ == '__main__':
    """
    Detect if the script is being run directly by the user
    """

    p = 23
    q = 19
    n = p*q

    public, private = generate_key_pair(p, q)

    # Creating list for generating random numbers
    list = list(range(n))

    # Generating File file1
    W = 27000 # Initialising value to generate value of different file size
    f = open("file1", "w+")
    for i in range(W):
        f.write("%d " % random.choice(list))
    f.close()

    # File Accessing file1
    access_file = open("file1", 'r')
    yourResult = [line.split(' ') for line in access_file.readlines()]
    L = yourResult[0][0:W]

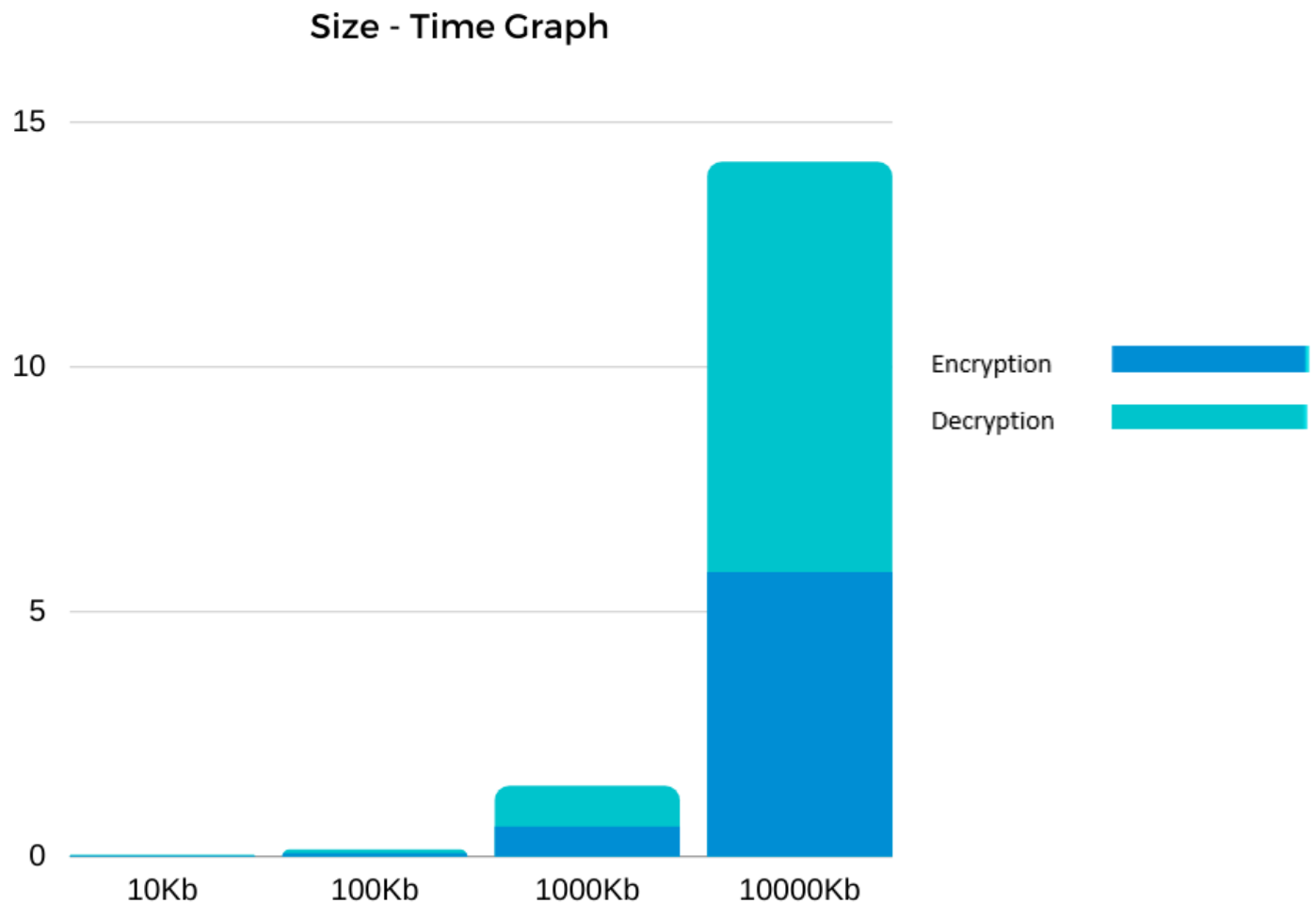
    text_file = open("file1", "r")
    data = text_file.read()
    text_file.close()

    start = time.time()
    encrypted_msg = encrypt(public, data)
    print("Encryption Period : " + str(time.time() - start))
    e = open("encrypted", "w+")
    for element in encrypted_msg:
        e.write(" " + str(element))
    e.close()

    print("key", n)

    start = time.time()
    decryped_msg = decrypt(private, encrypted_msg)
    print("Decryption Period : " + str(time.time() - start))
    d = open("decrypted", "w+");
    for element in decryped_msg:
        d.write(element)
    d.close()

```

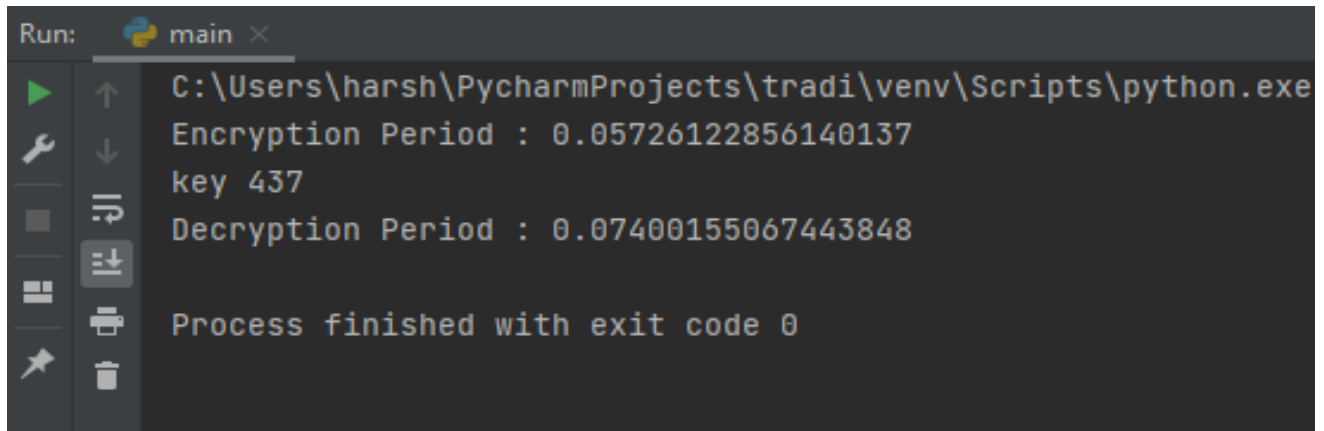


As we can clearly see that the time consumed in encryption is far less than the time for decryption which proves that our Novel Algorithm is somewhat less efficient than the traditional RSA as our algorithm has consumed added time for encryption and lesser time for decryption in all 4 size formats.

Screenshot When 10Kb file is generated and secured

```
Run: main x
C:\Users\harsh\PycharmProjects\tradi\venv\Scripts\python.exe
Encryption Period : 0.005930423736572266
key 437
Decryption Period : 0.007995843887329102
Process finished with exit code 0
```

Screenshot When 100Kb file is generated and secured

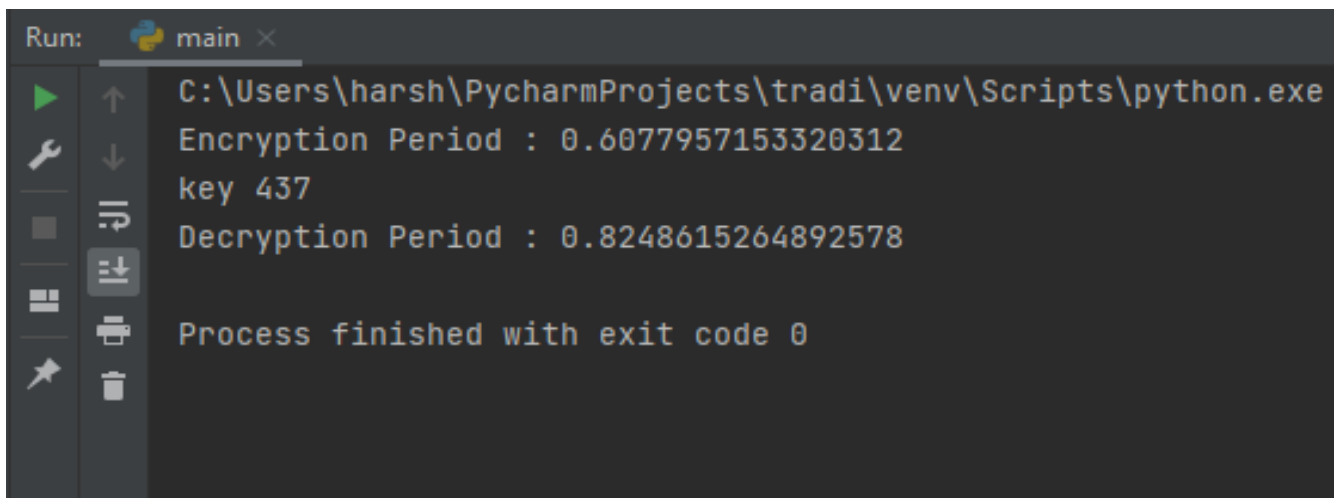


A screenshot of the PyCharm Run console. The top bar shows 'Run:' and a tab labeled 'main'. The console output is as follows:

```
C:\Users\harsh\PycharmProjects\tradi\venv\Scripts\python.exe
Encryption Period : 0.05726122856140137
key 437
Decryption Period : 0.07400155067443848

Process finished with exit code 0
```

Screenshot When 1Mb file is generated and secured

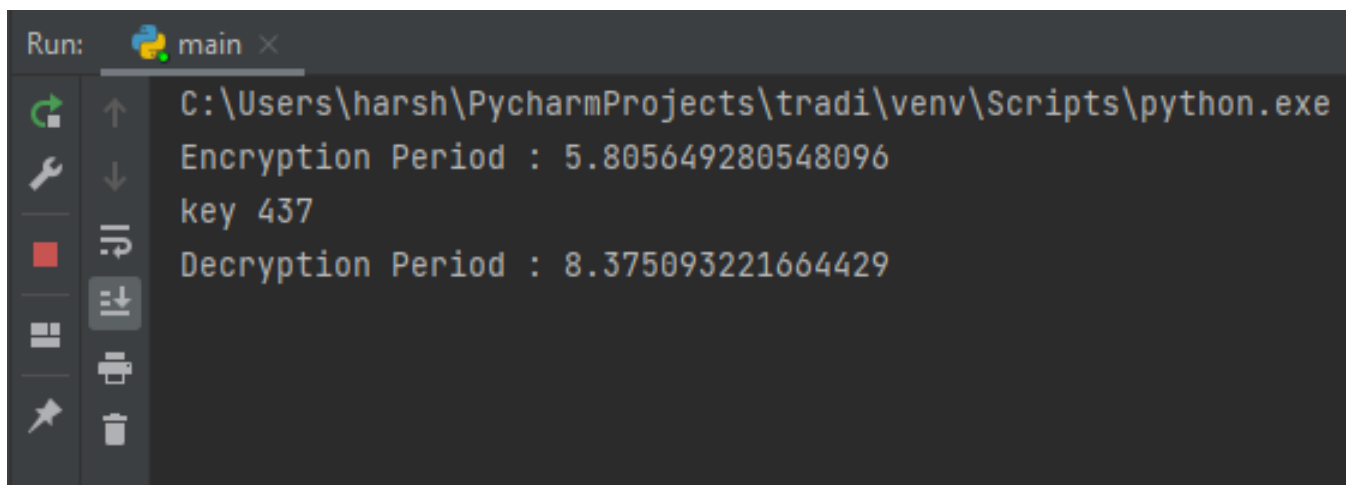


A screenshot of the PyCharm Run console. The top bar shows 'Run:' and a tab labeled 'main'. The console output is as follows:

```
C:\Users\harsh\PycharmProjects\tradi\venv\Scripts\python.exe
Encryption Period : 0.6077957153320312
key 437
Decryption Period : 0.8248615264892578

Process finished with exit code 0
```

Screenshot When 10Mb file is generated and secured



A screenshot of the PyCharm Run console. The top bar shows 'Run:' and a tab labeled 'main'. The console output is as follows:

```
C:\Users\harsh\PycharmProjects\tradi\venv\Scripts\python.exe
Encryption Period : 5.805649280548096
key 437
Decryption Period : 8.375093221664429
```

Conclusion

As we can observe from the above visualizations that if we divide the file into smaller pieces or sub files, clearly, we would be able to compute relatively faster when the size of the file also is comparatively larger. Our Algorithm had greater encryption timings and lesser decryption timings and hence cannot be used for lightweight where latency is a major consideration for many of the major sectors. On the other hand, further modifications to the algorithm can be made to make use of its complex and dynamic behavior which can be fruitful for specific situations. This method can also be used as Hybrid cryptosystem and various other derivatives can be taken into consideration for generation of various cryptographic algorithms and solutions.

Sno	Research Paper	Description
1.	A. Banafa, "Three major challenges facing IoT," IEEE IoT Newsletter, 2017.	An extensive research that talked about various challenges that the Iot and Edge System faces while maintaining secure communication from user to storage and processing. Various technological challenges covered in this research were security, connectivity, processing limitation, intelligent analysis and actions and much more. Privacy being a main issue with the devices, this newsletter highlights the significance of maintaining a secure connection. In order to realize the opportunities of the IoT, strategies will need to be developed to respect individual privacy choices across a broad spectrum of expectations, while still fostering innovation in new technologies and services.
2.	R H Aswathy, N Malarvizhi, "An investigation on cryptographic algorithms usage in IoT/Edge contexts", 2018.	A thorough explanation and comparison between most of the mainstream cryptographic algorithms on the basis of parameters such as data size, power consumption, memory requirements, computation potential, and much more. The algorithms that were compared were mainly conventional cryptography which is not advisable for Iot and Edge computation as these do not scale well into a world with embedded systems and sensor networks.
3.	Roberto Morabito, Vittorio Cozzolino, Aaron Yi Ding, Nicklas Beijar, and Jörg Ott "Consolidate IoT Edge Computing with Lightweight Virtualization", 2018.	Focuses on the requirements of the Edge computation in various scenarios such as E-HealthCare, Distributed Surveillance, Smart Grids, Autonomous Vehicles, and Smart Sensor Networks. The differentiation was performed on the basis of specifications such as Scalability, Multi-tenancy, Privacy & Security Latency, Extensibility (Open API) etc. Through their research they presented three different IoT use cases, namely Vehicular Edge Computing, Edge Computing for Smart City and Augmented Reality, in which LV solutions can bring a set of benefits and a desirable design flexibility.
4.	Norah Alassaf & Adnan Gutub & Shabir A. Parah & Manal Al Ghamdi "Enhancing speed of SIMON: A light-weight-cryptographic algorithm for IoT applications", 2019.	SIMON is a family of lightweight block ciphers publicly released by the National Security Agency (NSA) in June 2013. SIMON being a lightweight crypto algorithm was an appropriate choice for edge computation, This research proposed a modified version of the SIMON which focused particularly on the IoT systems and simultaneously preserved the main structure of the algorithm. An algorithm named SPECK was used to modify the original work. Minor modification led to massive changes making the modified SIMON with block sizes 32, 48, 64, 96 bits win among AES 128 in multiple cases. The implementation study examined both sides of running the algorithm on software, i.e. execution time as well as memory consumption.

5.	Nair Arun Mohandas, Adinath Swathi, Abhijith R, Ajmal Nazar and Greeshma Sharath "A4: A Lightweight Stream Cipher", 2020.	Similar to our Enhanced SIMON, A4 cryptography was one of the popular proceedings in Fifth International Conference on Communication and Electronics Systems (ICCES 2020). A4 is a newly found lightweight stream cipher which works on Linear Feedback Shift Register and One Feedback with Carry Shift Register as proposed in the documentation. It is highly efficient in resisting and counter-acting some of the most common attacks such as Meet in the middle Attack, Algebraic attack, Exhaustive Attack, Differential Attack, Correlation Attack. A4 is a hardware dependent mechanism due to which it may result in increased implementation complexity.
6.	Oscar Novo, "Blockchain Meets IoT: an Architecture for Scalable Access Management in IoT", 2018	This research work addresses the scalability and access management to billions of constrained devices in the IoT. A lightweight scalable Blockchain is one of many proposed solutions to this situation as centralized access control systems lack the ability to deal with increased load efficiently. Since the majority of IoT devices are largely constrained to support blockchain technology directly, the IoT devices in the design do not belong to the blockchain network which makes the integration of any IoT device easier to adapt to the whole system.
7.	Johannes Hedberggymnasiet "Formulation and Analysis Cubic Pell's Equation", 2012.	Pell's equation is proven to be an extension of the diophantine equation, usually involving two or more unknowns, such that the only solutions of interest are the integer ones (an integer solution is such that all the unknowns take integer values).The focus of this report is to investigate two possible cubic versions of the diophantine equation. The cubic Pell's equation has possible applications within Approximation Theory and Cryptography. The implementation of our proposed hybrid cryptographic system is based upon the conclusions of this thesis
8.	P. Chinnasamy, S. Padmavathi, R. Swathy, and S. Rakesh, "Efficient Data Security Using Hybrid Cryptography on Cloud Computing", 2021.	This paper provides an efficient solution to Storage issues that are prominent in cloud computing.The Blowfish and ECC algorithms are used for the processes of key generation, encryption and decryption. Elliptic curve cryptography (ECC) is implemented to achieve an enhanced level of security in cloud computing. ECC provides a more robust and secure model for developing and deploying a secure application in the cloud. To solve the key distribution they incorporated it with a steganography method to hide the keys.
9.	D.Chandravathi , P.V.Lakshmi "Privacy Preserving Using Extended Euclidean Algorithm Applied To RSA-Homomorphic Encryption Technique", 2019.	The Technique proposed by the authors used Extended Euclidean Algorithm to provide enhanced security for the private key. RSA Homomorphic encryption using Extended Euclidean algorithm (RSA-HEEEA) is secure when compared to RSA as it is based on the generation of private keys which makes the algorithm complex. The disadvantage of this methodology is the increased computational and processing requirements that may make the device slow and makes it inappropriate for many scenarios in IoT and Edge implementations.
10.	Guest Editorial - IEEE INTERNET OF THINGS JOURNAL, VOL. 8, NO. 4, "Special Issue on Blockchain and Edge Computing Techniques for Emerging IoT Applications", 2021	Highlights some of the major concerns with blockchain and its implementation on Iot/Edge devices. Proposes a blockchain-based containerized edge computing platform called CUTE, which is designed to provide low latency computing services over the Internet of Vehicles, and blockchain technology is used to improve network security. Instead of securing the channel with optimized blockchain methodologies, the emphasis is put into the protection of keys to design a transparent and optimal system for the workflow.

11.	Asif, Md. Rashid Al & Mahfuz, Nagib & Momin, Md & Hossain, Md. Alam & Roy, Saumendu. "Prototype Implementation of Edge Encryption in IoT Architecture", 2019.	The research implements edge encryption support in IoT architecture. In which data is encrypted first before sending to the cloud from the encryption service enabled IoT node. Moreover, the corresponding IoT node owner and sharer can only access that data. The IoT node owner can share, unshare, activate, and deactivate encryption service anytime. Meanwhile, the cryptographic key is stored in the local database as well as in a key-server without the intervention of CSPs. As sensitive data encrypted from the edge site and cryptographic key kept in an independent key-server, the less possibility of revealing information to intruders. Thus, enabling better data security over sensitive information in IoT architecture by edge encryption support.
12.	Mullapudi Chaitanya Krishna, Arjun Varma, Ashwath A, Vishnuvardhan A,"Comparison of Encryption Techniques in Internet of Things", 2020.	This paper proposes a hybrid cryptography method using two symmetric cryptography techniques Data Encryption Standard(DES) and International Data Encryption Algorithm (IDEA) for security of sensitive data in military data, Banking trades, etc. It has used AES and Blowfish algorithms for application in military, bank, network companies, big websites that control big data bases,etc. In this, they have proposed a hybrid encryption system with Blowfish and RSA for secure storage for healthcare data on cloud because it minimizes the time for encryption and decryption compared to other symmetric techniques.
13.	Xuesong Xu, Zhi Zeng, Shengjie Yang and Hongyan Shao "A Novel Blockchain Framework for Industrial IoT Edge Computing",2020.	Proposes a LLBF for IIoT edge computing application. The corresponding blockchain operation is designed for the edge devices with different resource capabilities. In RCL,resource constrained nodes are subdivided into clusters to maximize the scalability. In order to reduce the asynchronism of block operation in network delay, a time consistency algorithm is designed, which limits the number of different blocks generated in the consensus cycle. To improve the throughput of blockchain, a dynamic trust right confirmation algorithm is designed. A high throughput management mechanism is proposed to determine blockchain efficiency. Finally, an operation example of locally resource constrained devices on data transaction simulation experiment is presented. It has implemented data trust mechanisms in RCL blockchain performance analysis.
14.	Mohammed Omar Awadh Al-Shatari, Fawnizu Azmadi Hussin , Azrina Abd Aziz; Gunawan Witjaksono, Xuan-Tu Tran, "FPGA-Based Lightweight Hardware Architecture of the PHOTON Hash Function for IoT Edge Devices", 2020	An iterative architecture of PHOTON-80/20/16 lightweight hash function is implemented on several Altera and Xilinx FPGA devices. It is a round-based architecture where all the permutation operations are executed in one round. The proposed design achieves better area-performance trade-offs than the existing designs as the architecture of the MixColumns module is designed using look-up tables to avoid the intensive computations of the multipliers. The round constants are also utilized as a round counter to reduce the logic resources. It consumes less logic resources and achieves higher performance resulting in higher efficiency.

15.	Muhammad Usman, Irfan Ahmedy, M. Imran Aslamy, Shujaat Khan and Usman Ali Shahy “SIT: A Lightweight Encryption Algorithm for Secure Internet of Things”,2017	A lightweight encryption algorithm named as SecureIoT (SIT) is proposed by the authors .It is a 64-bit block cipher and requires 64-bit key to encrypt the data.The architecture of the algorithm is a mixture of feistel and a uniform substitution-permutation network which is followed by Crypto however functions of each component is simplified to enhance its performance for the constrained hardware. In the sussessor Hummingbird-1 is proposed as Hummingbird-2(HB-2). With 128 bits of key and a 64 bit initialization vector Hummingbird-2 is tested to stay unaffected by all the previously known attacks. However the cryptanalysis of HB-2 highlights the weaknesses of the algorithm and that the initial key can be recovered. They analysed different legacy encryption algorithms including RC4, IDEA and RC5 and measured their energy consumption. Also calculated the computational cost of these ciphers on different platforms.
16.	Mohammed Tarique, Rohan Raul, Khalil Pinjari, Pranay Patil, Rohit Shinde,“A Lightweight Encryption Scheme for Network-Coded Mobile Ad Hoc Networks ”,2016	This paper proposes a new encryption scheme that fully adventure the security property of net-work coding. The coding vectors and the message content are both necessary for decoding, arbitrarily reordering/mixing they will generate significant confusion to the eavesdropping challenger. In specific, we propose P-Coding which is a lightweight encryption scheme to fight against eavesdroppers in network-coded MANETs. In a nutshell, with the help of permutation encryption, P-Coding randomly mixed symbols of each packed which is a coded pocket (packet prefixed with its coding vector), so as to make it harder for eavesdroppers to locate coding vectors for package decoding. P-Coding, a lightweight encryption scheme on top of network coding, reduces energy consumption in MANETs by decreasing the security cost. P-Coding events the intrinsic security stuff of network coding, and uses simple permutation encryptions to generate great mistake to eavesdropping adversaries.P-Coding is efficient in calculation, and incurs less energy consumption for encryptions/decryptions.
17.	Panasayya Yalla, Jens-Peter Kaps,“Lightweight Cryptography for FPGAs”,2010	This paper proposes lightweight cryptography for FPGAs by introducing block cipher independent optimization techniques for Xilinx Spartan3 FPGAs and applying them to the lightweight cryptographic algorithms HIGHT and Present. Our implementations are the first reported of these block ciphers on FPGAs. Furthermore, they are the smallest block cipher implementations on FPGAs using only 117 and 91 slices respectively, which makes them comparable in size to stream cipher implementations. Both are less than half the size of the AES implementation by Chodowiec and Gaj without using block RAMs.
18.	Neha Bisht, Joel Thomas,Dr. Thanikaiselvan V , ”Implementation of security algorithm for wireless sensor networks over multimedia images”, 2016	A Hybrid encryption algorithm which includes two phases of work. In Phase- 1, it takes the benefit of both symmetric and asymmetric cryptographic techniques using both AES and RSA algorithms. The initial key for AES is provided by the Arnold transformation. In Phase-2, the entrusted data obtained from the latter algorithms is given to the RDH system which provides data integrity by using the difference expansion.The combination of AES – RSA with RDH (Difference Expansion) is done to have double layer of security and to provide high operation speed, high security performance and strong usability.

19.	Shahriar Ebrahimi, Siavash Bayat-Sarmadi, Hatameh Mosanaei-Boorani, "Post-Quantum Crypto Processors Optimized for Edge and Resource-Constrained Devices in IoT", 2019	An optimized version of RingBinLWE, namely InvRBLWE, which is highly efficient for hardware implementation. It proposes two architectures for InvRBLWE scheme targeting different IoT devices with alternative capabilities varying from edge and powerful devices to resource-constrained and tiny end-nodes. FPGA implementations improve time and AT compared to the best previous RingLWE implementations. These implementations also improve AT by at least 92% compared to the best of ECC implementations. They are the first to propose a practical ASIC implementation of LWE-based cryptosystems for IoT devices. Moreover, lightweight ASIC implementation has power consumption as low as 0.18 mW, which can be supplied by low-power energy harvesting devices such as Vibration Piezo or EM. In other words, the lightweight architecture is the first public key ASIC implementation that satisfies NIST report criteria and can be practically exploited in IoT end-nodes.
20.	Riadh Ayachi, Ayoub Mhaouch, Abdesslem Ben Abdelali, "Lightweight Cryptography for Network-on-Chip Data Encryption", 2021	This paper proposes: (i) an NI for NoC with high performance and security level (ii) Proposing a lightweight encryption algorithm for NoC based on the LED64 algorithm (iii) Proposing a bigger input/output FIFO to avoid data waiting and accelerate the processing time (iv) Synthesize the proposed solution on the Xilinx Virtex 5 XC5VFX200T (v) High performance achieved with a small footprint and high processing speed (vi) Guaranteeing the use of the proposed NI with limited resources devices. The LED block cipher was used to encrypt data due to its low implementation area, fast processing speed, and high-security level. The reported results show that the proposed NI design outperforms existing works based on the AES block cipher with a wide range in terms of implementation area and working frequency.
21.	Md. Navid Bin Anwar, Maherin Mizan Maha, "AMPC: A Lightweight Hybrid Cryptographic Algorithm for Wireless Sensor Networks", 2020	This paper proposes a hybrid cryptographic algorithm, AMPC, strengthen the security of WSN as there are two cryptographic algorithms, AES and modified PlayFair, are used for data security and another algorithm, Diffie-Hellman, is used for secure the key exchange process. In addition, AMPC requires less operational time compared to other existing hybrid algorithms, therefore, it facilitates fast data transmission with lower energy consumption. To recapitulate, AMPC scheme is an optimal solution to overcome the limitations of WSN which provides more security with minimum computational delay.
22.	Hichem Mrabet, Sana Belguith, Adeeb Alhomoud, Abderrazak Jemai, "IoT Security Based on a Layered Architecture of Sensing and Data Analysis", 2020	In this paper, an IoT five-layer architecture is proposed based on potential security threats and countermeasures. This architecture is based on a modification of OSI architecture, considering the security vulnerabilities and threats. In addition to existing OSI layers, we define a cloud and data layer, which involves several publicly available IoT frameworks providing IoT data storage, processing, and analysis. This architecture is extended to involve machine-learning applications that process data and protect IoT components. Furthermore, we present a discussion of current challenges facing IoT security solutions, such as the lack of standard encryption algorithms adapted for IoT devices. We also explore the application of novel techniques to secure IoT, such as the use of Blockchain in IoT and machine-learning models, as well as reviewing the potential of 5G network applications, and their reliance on IoT.

23.	Mario Frustaci, Pasquale Pace, Gianluca Aloï, Giancarlo Fortino, "Evaluating Critical Security Issues of the IoT World: Present and Future Challenges", 2018	This paper tries to bring order on the IoT security panorama providing a taxonomic analysis from the perspective of the three main key layers of the IoT system model: 1) perception; 2) transportation; and 3) application levels. It has shown that the IoT system model has many security issues among which threats that can exploit some possible weaknesses. For these reasons, it is necessary to appropriately enforce trust management and security in the IoT world starting from the characterization of the different threats related to each specific level of the general IoT system model. As a result of the analysis, it will highlight the most critical issues with the aim of guiding future research directions.
24.	Nilupulee A. Gunathilake, William J. Buchanan, Rameez Asif, "Next Generation Lightweight Cryptography for Smart IoT Devices: Implementation, Challenges and Applications", 2019	The implementation, challenges and futuristic applications of LWC algorithms for smart IoT devices have been discussed, especially the performance of Long-Range Wide Area Network (LoRaWAN) which is an open standard that defines the communication protocol for Low-Power Wide Area Network (LPWAN) technology. This paper has covered the necessity of LWC, its current status, compatible technologies and protocols, i.e., LoRaWAN, and also challenges in the present situation by evaluating existing theoretical and practical studies. The overall analysis indicates promising capabilities in the direction of successful implementation of LWC and its performance towards 5GN smart cities.
25.	Chandrasegar Thirumalai, Senthilkumar Mohan, Gautam Srivastava (SITE-VIT), "An efficient public key secure scheme for cloud and IoT security", 2019	In this paper, It presented an ENPKESS method which applied three stage of encryption and two stage of decryption through Diophantine equation and RSA public keys. Moreover, it showed the method with Knapsack to secure the ENPKESS public key from timing attack. From this mixture, one can attain high strength and security to protect their confidential data with an adequate key size. The experimental results demonstrate that the ENPKESS scheme provides proper time computation by utilizing around 0.24 periods of ESRKGS and 0.82 periods of Standard. RSA.
26.	Dennis Agyemanh Nana Gookyi, Guard Kanda, Kwangki Ryoo, "NIST Lightweight Cryptography Standardization Process: Classification of Second Round Candidates, Open Challenges, and Recommendations", 2021	Provides a comparative analysis of the 32 candidates' algorithms that have made it to the second round of the process of setting up a lightweight AEAD. The 32 candidates for the second round are categorized according to their basic building methods including block cipher-based, tweakable block cipher-based, stream cipher-based, and permutation-based. security parameters, operating system, features, and use of hardware/software resources. The work goes on to discuss specific challenges with the process of setting up cryptography standardization and providing some useful recommendations.
27.	Daniel Engels, Markku-Juhani O. Saarinen, Peter Schweitzer, Eric M. Smith, "The Hummingbird-2 Lightweight Authenticated Encryption Algorithm", 2011	Hummingbird-2 is an encryption algorithm with a 128-bit secret key and a 64-bit initialization vector. Hummingbird-2 optionally produces an authentication tag for each message processed. In this paper it presents the Hummingbird-2 algorithm, its design and security arguments, performance analysis on both software and hardware platforms, and timing analysis in relation to the ISO 18000-6C protocol. It has also presented results of software and hardware implementations of Hummingbird-2. Hummingbird-2 can be implemented with little more than 2000 gate equivalents, making it well suited for ubiquitous devices such as RFID tags and sensors. Hummingbird-2 has the additional advantage over other lightweight encryption primitives that it produces a message authentication code.

28.	Gaurav Bansod, Nishchal Raval, Narayan Pisharoty, "Implementation of a New Lightweight Encryption Design for Embedded Security",2013	Paper presents the design of a new lightweight compact encryption system based on bit permutation instruction GRP (group operation) which is widely studied and extensively researched. By using the S-box of PRESENT, with addition of the confusion property for GRP. By comparing the existing S-boxes of compact algorithms and its cryptanalysis, this paper proposes a new hybrid system that provides more compact results in terms of both memory space and gate equivalents. A hybrid cryptosystem which consists of GRP and S-box of PRESENT is designed and implemented on a 32 bit processor. This fusion has resulted in a lightweight cipher that is the most compact implementation, till now, in terms of memory requirement.
29.	Alex Biryukov, Léo Perrin,"State of the Art in Lightweight Symmetric Cryptography",2017	This paper surveys Ultra-Lightweight Cryptography corresponds to algorithm that fit in very pacific areas of the design space such as "stream cipher with low gate area in ASICs" or "block cipher with high speed on 32-bit micro-controllers". As performance strongly dictates the use of such algorithms, making bolder choices in the performance versus security trade-off makes sense.The AES, while very versatile both in software and in hardware, cannot address all design constraints. The need for lightweight algorithms is well established, as is evidenced by the NIST and CRYPTREC work to select and standardize such algorithms.
30.	Indu Joseph,Dr. S. D. Sawarkar , "Study on Integration of Blockchain and IoT in Smart City Applications",2020	The blockchain technology, smart contracts and its integration for IoT data is analysed. This paper focuses on understanding blockchain, its features, challenges, types, and how its integration can be an advantage in IoT. The challenges faced in the integration of Blockchain and IoT is summarized and a study on the existing IoT-blockchain applications for smart city is done.The various challenges faced in IoT-blockchain integration are discussed and potential solutions proposed are also reviewed. The various existing applications for the IoT-Blockchain convergence in Smart City use cases are surveyed and studied to understand the uniqueness covered in each implementation.