2021 S2

Research on using blockchains to improve data security

Project 2:
Final Presentation

The University of Adelaide
Computer Science (Advanced)

Jiajun Yu (Jason)
a1806320

CONTENTS

PART 01

Topic Introduction

Interesting Data

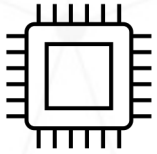
Blockchain & IPFS

Interesting Data

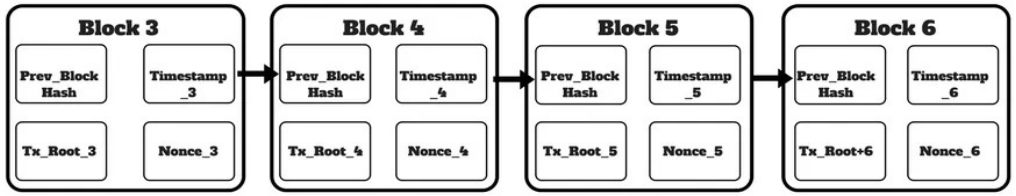
In 2020, roughly **306.4 billion** emails were sent and received each day.

Google processes more than ***20 petabytes*** of data every day. This includes around 3.5 billion search queries. (*1 petabyte = 2^{50}bytes*)

Abbreviation	Unit	Value	Size (in bytes)
b	bit	0 or 1	1/8 of a byte
B	bytes	8 bits	1 byte
KB	kilobytes	1,000 bytes	1,000 bytes
MB	megabyte	$1{,}000^2$ bytes	1,000,000 bytes
GB	gigabyte	$1{,}000^3$ bytes	1,000,000,000 bytes
TB	terabyte	$1{,}000^4$ bytes	1,000,000,000,000 bytes
PB	petabyte	$1{,}000^5$ bytes	1,000,000,000,000,000 bytes
EB	exabyte	$1{,}000^6$ bytes	1,000,000,000,000,000,000 bytes
ZB	zettabyte	$1{,}000^7$ bytes	1,000,000,000,000,000,000,000 bytes
YB	yottabyte	$1{,}000^8$ bytes	1,000,000,000,000,000,000,000,000 bytes



BLOCKCHAIN



A blockchain --> growing list of records, called blocks, that are linked together.

Nodes collectively adhere to a protocol to communicate and validate new blocks.

If a member's ledger is altered or corrupted in any way, it will be rejected by most of the members in the network.

(Wikipedia 2021)

IPFS

The Inter Planetary File System is a protocol and peer-to-peer network for storing and sharing data in a distributed file system.

The storage method of IPFS is fragmented distributed storage.

(Desai et.al 2021)

PART 02

Challenge Motivation and Introduction

Challenge motivation

High-speed Brute-force attack

Challenge Motivation and Introduction

In June 2021, Data associated with 700 million LinkedIn users was posted for sale in a Dark Web forum. This exposure impacted 92% of the total LinkedIn user base of 756 million users. (Tunggal 2021)

The data security model plays a very important role in the society.

High-speed Brute-force attack

A brute force attack uses trial-and-error to guess information. Hackers work through all possible combinations hoping to guess something correctly.

PART 03

Paper Review

A Blockchain-based Decentralized Data Storage and Access Framework for PingER

A Secure Data Sharing Platform Using Blockchain and Interplanetary File System

A Blockchain-based Decentralized Data Storage and Access Framework for PingER

Saqib Ali†, Guojun Wang*
School of Computer Science and Technology, Guangzhou University, Guangzhou, P. R. China, 510006
Emails: {saqibali,csgjwang}@gzhu.edu.cn

Bebo White
Stanford Linear Accelerator Center
P.O. Box 4349,
Palo Alto, California 94309-4349
Email: bebo@slac.stanford.edu

Roger Leslie Cottrell
Stanford Linear Accelerator Center
P.O. Box 4349,
Palo Alto, California 94309-4349
Email: cottrell@slac.stanford.edu

- We design a PingER data storage and access framework by storing the metadata of daily PingER files on the permissioned blockchain whereas the actual data is stored off-chain on multiple MAs.
- We design an off-chain storage of PingER data files using distributed hash tables on multiple MAs in which contents are accessed through hashes with high throughput and low latency.
- We outline the implementation requirements for the development of the blockchain-based PingER framework.
- Finally, we explore the capabilities that the framework can provide aside from the PingER decentralized data storage.

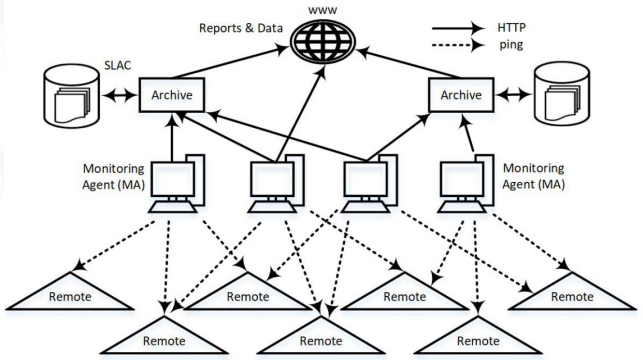
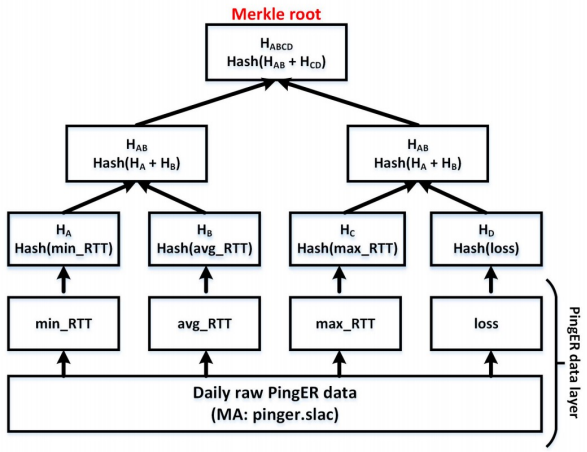


Fig. 1. PingER: Data storage & access architecture



can run over the Inter Planetary File System (IPFS) [18], [29] which is a peer-to-peer hypermedia protocol using distributed hash tables and a self-certifying namespace. An important feature of IPFS is that the objects can be traversed with a string path. Paths work as they do in traditional UNIX filesystems and the Web. For example, `/ipfs/BLYkgq61ZYaQ8NhkcqyU7rLcnSa7dSHQ16x/min_rtt-100-pinger.slac.edu-2018-01-28.txt.gz`. Thus, all data files are always accessible via their hash. This approach still works even if a few MAs are offline as the files are located in multiple locations for redundancy. Thus, IPFS provides smarter, faster and permanent web services to

A blockchain based data storage and access framework for PingER to remove its total dependence on a centralized repository.

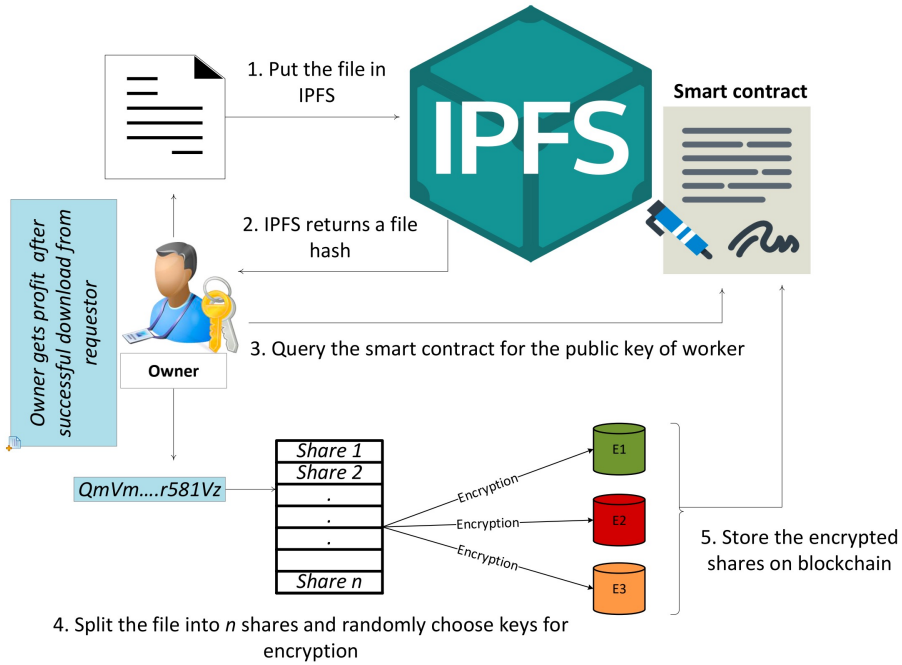
Permissioned Blockchain (managed by industry leaders) / Merkle Root (check immutability)

Inter Planetary File System (IPFS)
Off-chain storage strategy
Store fragmented large data in other computers

Article

A Secure Data Sharing Platform Using Blockchain and Interplanetary File System

Muqaddas Naz [1], Fahad A. Al-zahrani [2], Rabiya Khalid [1], Nadeem Javaid [1,*], Ali Mustafa Qamar [3,4], Muhammad Khalil Afzal [5] and Muhammad Shafiq [6,*]



6. Implementation Details

In this section, the implementation details are provid
of Ethereum blockchain. Ethereum is open source distr
solidity. A programmable language that allows to write s

6.1. Simulation Setup

The specifications for implementation setup are: In
GB RAM, 64 bit operating system and X64-based proces
to write smart contract. Front-end web GUI is developec
interactive forms. The primary tools used to develop this

Visual Studio Code

A lightweight cross-platform code editor designed by
allows powerful debugging with a variety of tools, highli
build in Git control and code refactoring [34].

Only that worker can provide the services, who is selected by the owner. When the owner receives the hash, SSS algorithm is used to split the IPFS hash of file into *k* number of shares. In response to these shares, owner decides the *n* number of random keys to be used for encryption. Once all the shares are encrypted, they are stored in the blockchain along with other important information such as; authorized recipient for the file. Data security is ensured by encrypting the hashes. The reason

A blockchain-based secure data sharing platform by leveraging the benefits of interplanetary file system (IPFS)

The mechanism of this model, such as its 'SSS' algorithm which is used to split the file hash.

Implementation details, such as some code implementations and analysis.

PART 04

Identify drawbacks

Lack of deployment analysis

Difficult to manage keys

The way of slicing file hash is risky

Critique of literature

01

■ Lack of migration analysis

No study about the feasibility of migrating the system from the local repository to the designed model (Cost / Company wishes).

02

■ Difficult to manage keys

Owner decides the *n* number of random keys to be used for encryption. (n ∝ file hash slices)

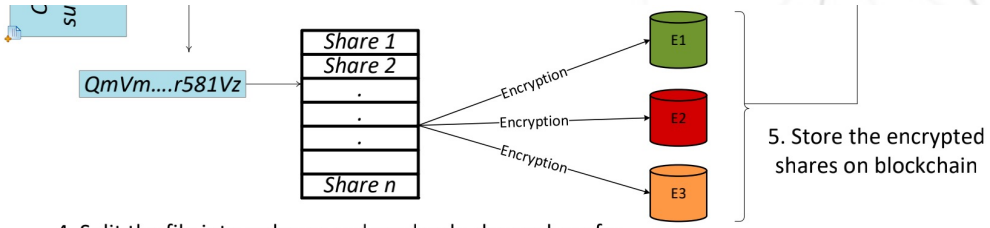
03

■ The way of slicing file hash is risky

File hash can retrieve data back from IPFS. Linearly slicing file hash is dangerous.

PingER (Ping End-to-end Reporting)[4] is a worldwide end-to-end Internet performance measurement framework developed and managed by the SLAC National Accelerator Laboratory USA [13]. The client-server architecture of PingER consists of 50 active Monitoring Agents (MAs) in 20 countries of the world. These MAs probe 700 remote sites located in 170

t worker can provide the services, who is selected by the owner. When the ov
SSS algorithm is used to split the IPFS hash of file into *k* number of shares. I
res, owner decides the *n* number of random keys to be used for encryption.
e encrypted, they are stored in the blockchain along with other important info
rized recipient for the file. Data security is ensured by encrypting the hashes
is is that, hash itself is not secure. It just represents a unique finger print for som
ized customer, who has not submitted deposit for digital content gets access



4. Split the file into *n* shares and randomly choose keys for encryption

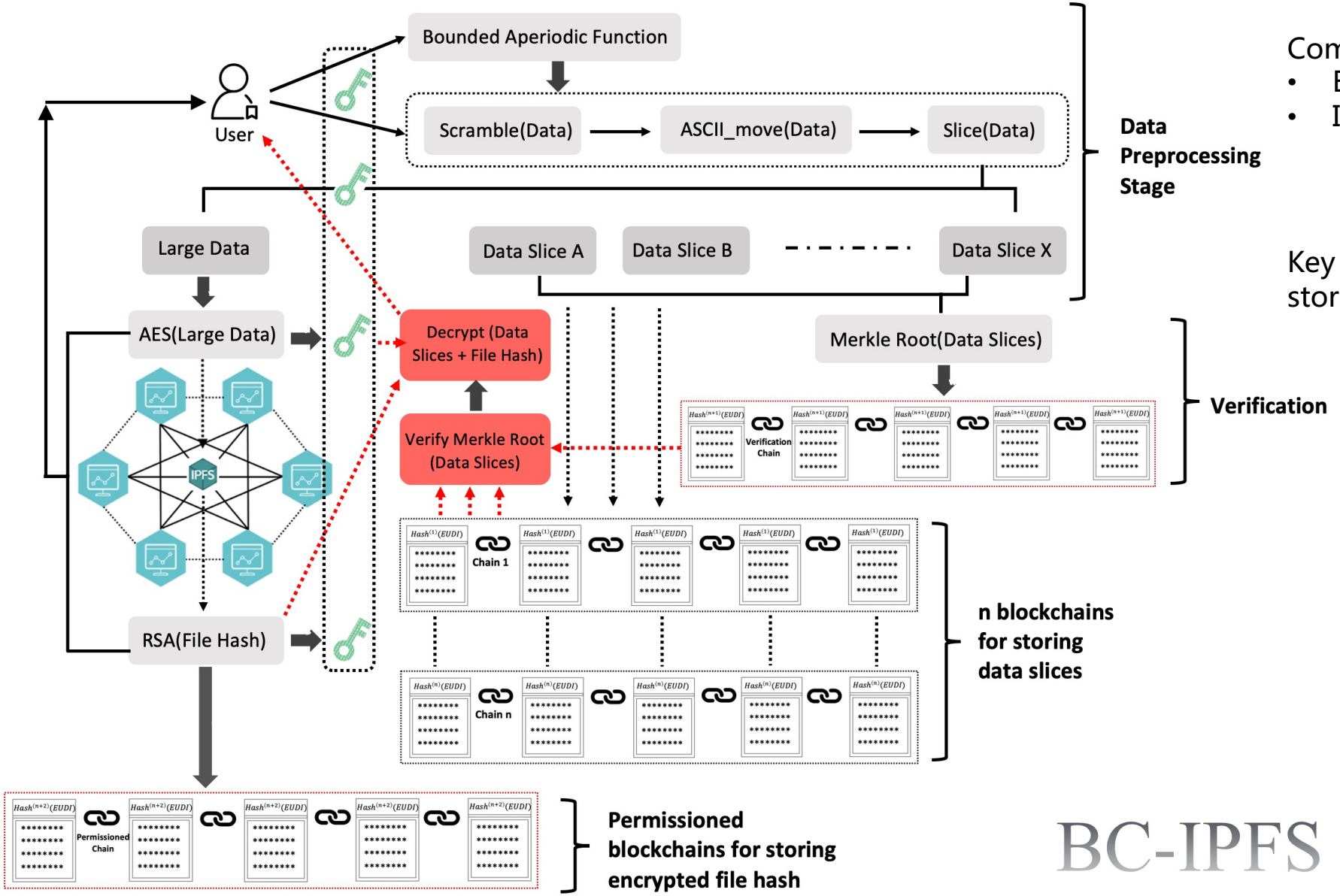
PART 05

Solution (Designed Model)

Overall structure of the designed model (BC-IPFS)

Implementation of each component

Overall structure of the designed model



Combined with blockchain & IPFS

- Blockchain → small data
- IPFS → large data → file hash

Key concept: Slicing data & Distributed storage (Blockchain & IPFS)

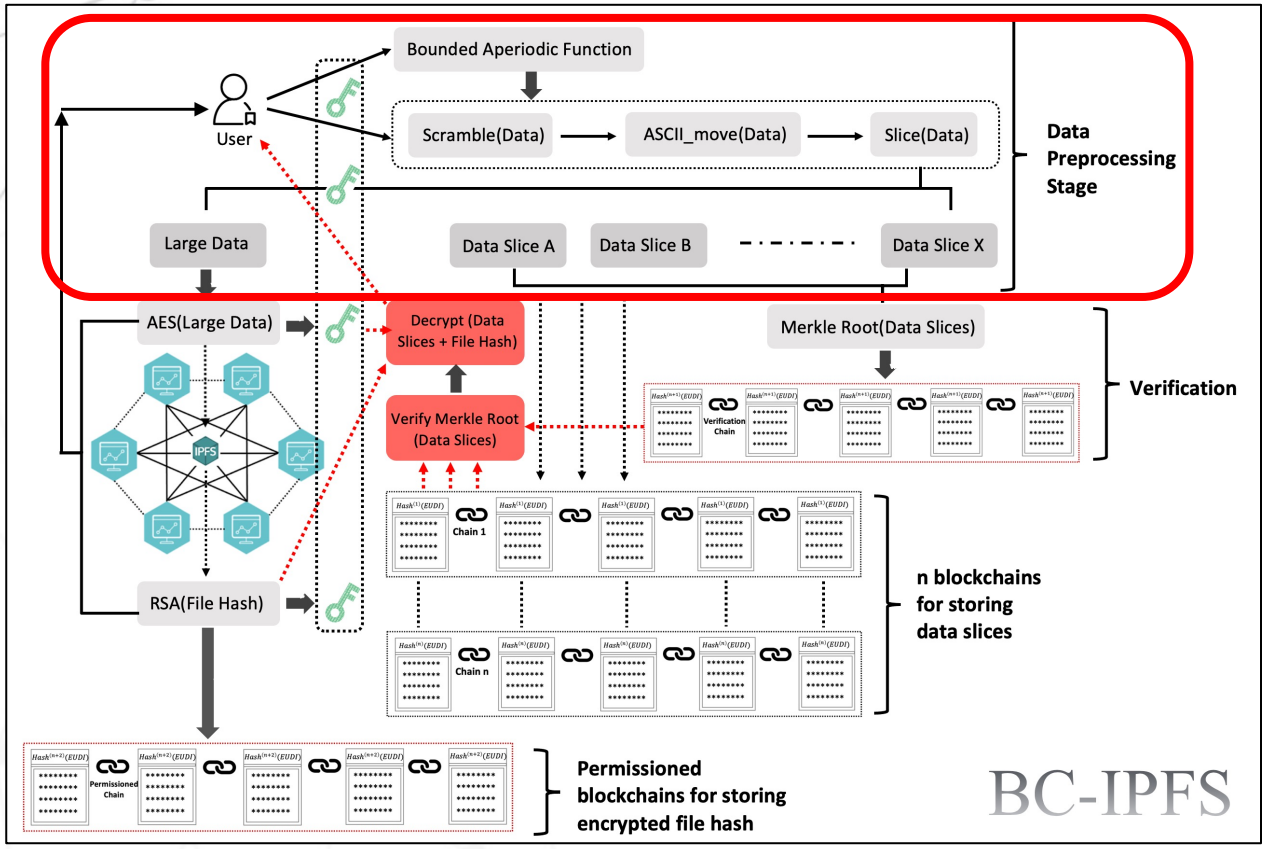
Process:

- Data presprocess
- Get Merkle root
- Data storage
- Data decryption

BC-IPFS

Data Preprocess

```
value=10*math.sin(17*math.cos(100*i)-3*math.log(abs(math.cos((190)*i)*150)))
```



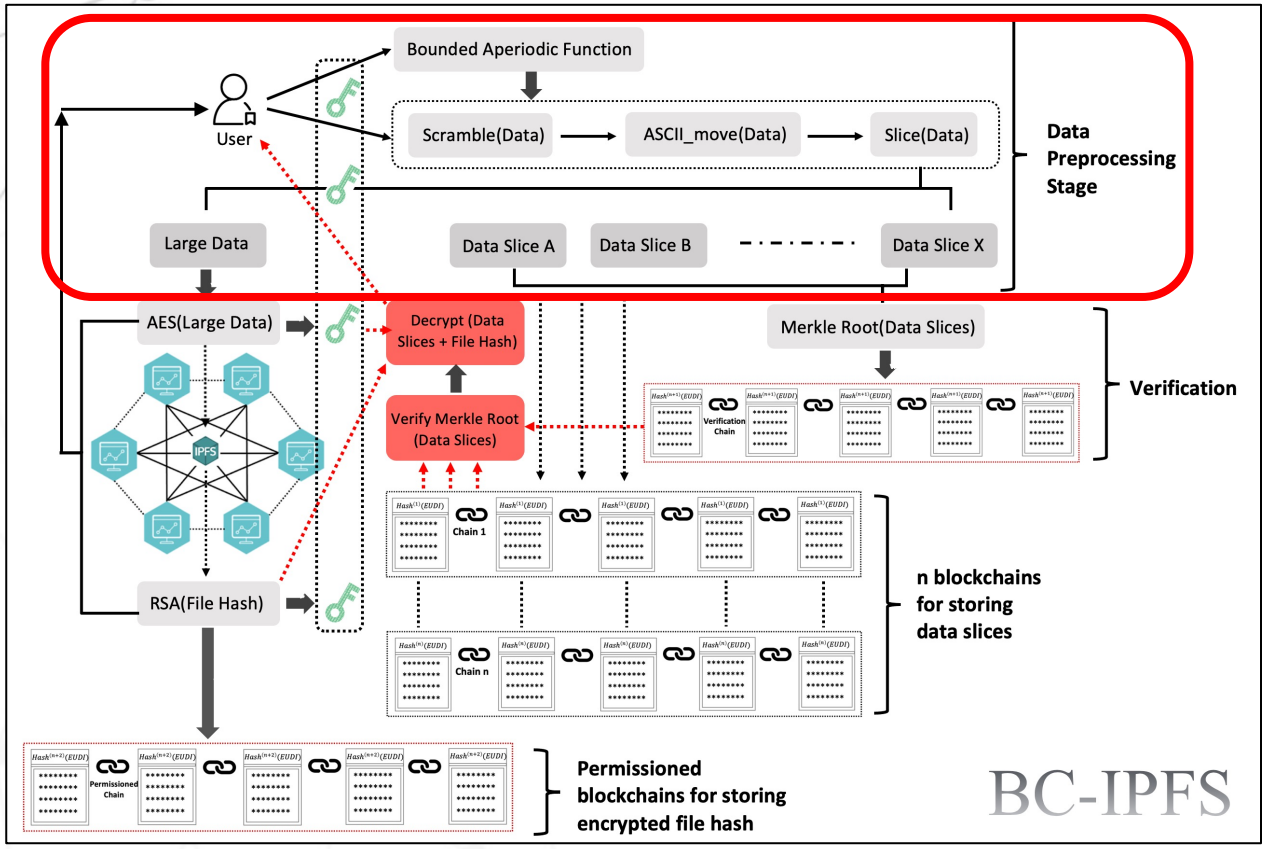

```
Data Slice 1:  []
Data Slice 2:  ['y', 'k', 'a']
Data Slice 3:  ['i', 'n']
Data Slice 4:  ['(', 'j', 'x', 'w']
Data Slice 5:  ['z']
Data Slice 6:  ['b', 'r', '|', '\x7f', 'c']
Data Slice 7:  ['q', 't', '*', 'x']
Data Slice 8:  ['~', '&', 'f']
Data Slice 9:  ['q', 'w', 'g', 'h', 'm', 'W']
Data Slice 10:  [' ', 'p', 'k', '"', 'v', 'o', '*', 'k']
Data in IPFS: [rjset Io cwih  nse  tlc pp aytruil arp
tccilunotm lr t ccee ,  iucy nt,pisdnt f e ndiisarp u
lulaari eyrtesg ictnttlnh   tewhneieato  ce ahi  rioidp
vco, npe l spdolt yufot od  erns i u   .Yard f eee o
ssuec   sy ttlnac iupy wntor ,o b ooi wl  uyni td onc
coieurv enros n  aaur lylt    cy t tdhreh or t
pdnehettcci heenleer elna  t e ailcng igog dcn hd
feanten ialeeat.to]
```

A bounded aperiodic function → Scramble data → change character's ASCII → Slice data

Different functions will preprocess data in a complelely different way.

Data Preprocess

```
value=10*math.sin(16*math.cos(100*i)-3*math.log(abs(math.cos((190)*i)*150)))
```



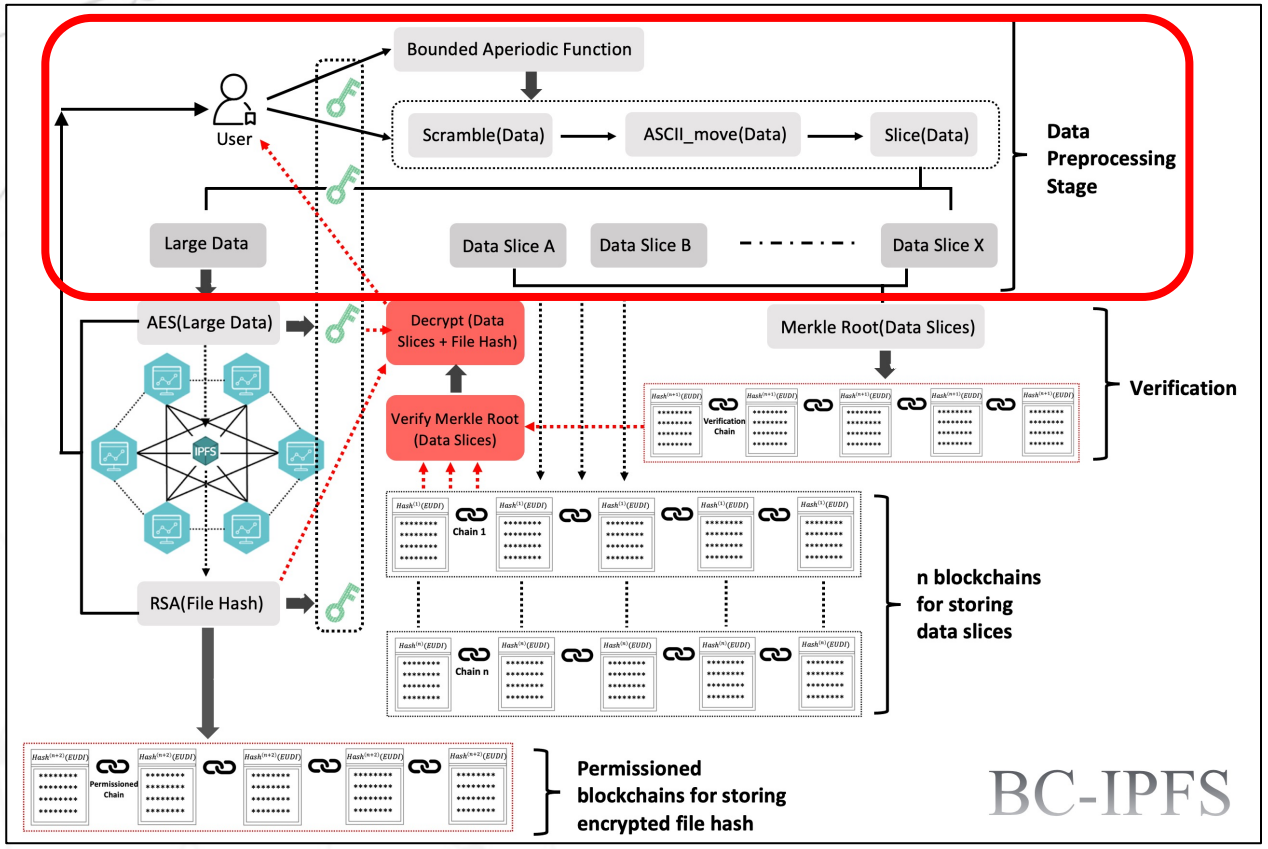

```
Data Slice 1:  ['m']
Data Slice 2:  ['\x81']
Data Slice 3:  [',', 'l', 'q']
Data Slice 4:  [' ', 'a', 'i']
Data Slice 5:  ['x']
Data Slice 6:  ['m', 'm', '&', 'h', 'x', 'p']
Data Slice 7:  ['j', 'a']
Data Slice 8:  ['\x18', 'w', '{', 'x', 'k']
Data Slice 9:  ['y', 'j', 'k', '|', 'r', '\x1a', 'x',
'}']
Data Slice 10:  ['~', 'i', 'm', '~', '_']
Data in IPFS: [p oetjt  suoilw necieraly a cc ula tor ic
nIiclts tr ps r nceei s c uodti dni ttm fneo,ereyi a
iylctrereli sg ctli etlaw  heiuh het  ipon  n,dgp
ircdita  uctn v f lplet  uto od aeos ti ioseu o r nfe.
tY s o en teyto nc airwlsy p,u  yo wtul  euon td
icatnneb yo sru esi pae  uo rol otreucrtesttyv nr ioe an
tcde ca  ae ge  hl ihleacf a nae nh  ngnailpn hda
```

A bounded aperiodic function → Scramble data → change character's ASCII → Slice data

Different functions will preprocess data in a complelely different way.

Data Preprocess

```
value=10*math.sin(15*math.cos(100*i)-3*math.log(abs(math.cos((190)*i)*150)))
```



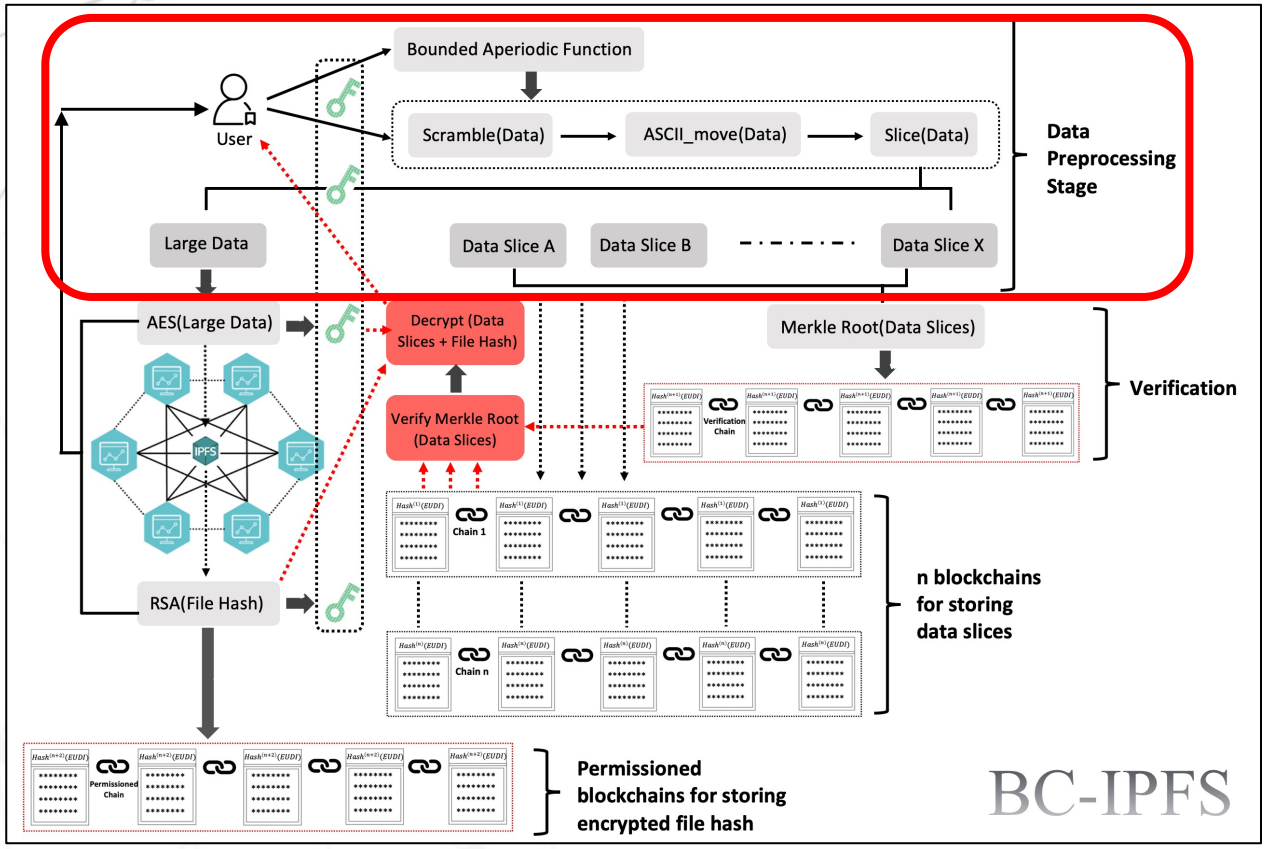

```
Data Slice 1:  ['q', '\\', 'n']
Data Slice 2:  ['l', 'n']
Data Slice 3:  ['d', 'W', 'j']
Data Slice 4:  ['o', 'm', 'd', 'c']
Data Slice 5:  ['\\', 'j', '\x16']
Data Slice 6:  ['\x17']
Data Slice 7:  ['q', '[']
Data Slice 8:  ['y', 'h', 'q', 'i']
Data Slice 9:  ['s', 'j', 'o', 'e']
Data Slice 10:  ['|', 'j', 'Z', 'i', 'h', 'u', 'x', '|',
'k']
Data in IPFS: [Ite j yptni wll orecsc iptral uear cs
cti ano pmuo  sp eenu ire dcu    td,iiihncy oe  fsicr
lnru ytiec sltat ci hylnreg nnhiga thpl , t,e adiotoprn
iin  cup  w   vsolooion e at eessui .t o ous  adr  ltea
eerlt   s acopynyo ietw ,n  tbyo   dll nwce  eouc vi fnt
y ouice roasuey  o u nd  nertchals  thteeocpanu  ue  ht
aellg ene c  rfatcdhn h aget  iia n lgne ttnd oeecannda
```

A bounded aperiodic function → Scramble data → change character's ASCII → Slice data

Different functions will preprocess data in a complelely different way.

Data Preprocess

```
value=10*math.sin(15*math.cos(10*i)-3*math.log(abs(math.cos((190)*i)*150)))
```

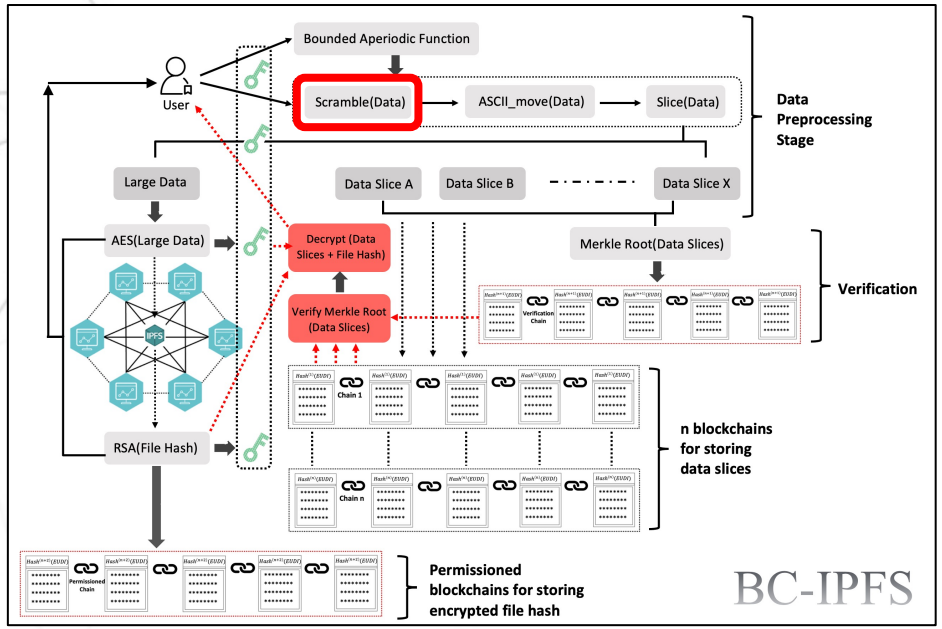



```
Data Slice 1:  ['r', '|', 'm']
Data Slice 2:  ['r', 'y', '\x18']
Data Slice 3:  ['\x16', '%']
Data Slice 4:  ['x']
Data Slice 5:  ['z', 'f', 'a', 'd', '\x18', 'f']
Data Slice 6:  ['t', 'w']
Data Slice 7:  ['w', 'c', '~']
Data Slice 8:  ['z', 'x', '$', 'Z', 'k']
Data Slice 9:  ['\x1e', '\x16']
Data Slice 10:  ['v', 'v', 'l', "'", 'k', 'y', 'k']
Data in IPFS: [I ncspt huwylii  jecot   er lt relauoto t
caioamun c reccpeei  cps ii y  seitntdyoie , tircfnprld
tul,iree i siat nnyg lhehticng   oleep  ncattiv
dhi,aceuo  dr   ilu oo oso n  tpdseletfs saY ur eiai  d
reole  efty taro i oy canp,uncow eo  yuilwb    t  oevei
cneoyrd nu o tn rnosl ye tere e saahtrhle  c dt unodaih
tpr e ne hgel  ca ca l  tnanrici ae h g daaelnteg itd
ntnlenn.]
```

A bounded aperiodic function → Scramble data → change character's ASCII → Slice data

Different functions will preprocess data in a complelely different way.

Data Preprocess ----- Data Scramble



Function	$f(x) = 10\sin(16\cos 100x - 3ln\lvert\cos(190x) * 150\rvert)$																							
Data	'H'	'i'	','	space	'e'	'v'	'e'	'r'	'y'	'o'	'n'	'e'	','	space	'I'	space	'a'	'm'	space	'J'	'a'	's'	'o'	'n'
$f(x)$	$f(0)$	$f(1)$	$f(2)$	$f(3)$	$f(4)$	$f(5)$	$f(6)$	$f(7)$	$f(8)$	$f(9)$	$f(10)$	$f(11)$	$f(12)$	$f(13)$	$f(14)$	$f(15)$	$f(16)$	$f(17)$	$f(18)$	$f(19)$	$f(20)$	$f(21)$	$f(22)$	$f(23)$
Value	8.2	5.8	-7.9	8.9	9.7	-6.2	6.0	-7.9	-2.3	3.2	8.2	5.9	9.2	-9.2	-6.3	5.3	4.4	9.3	-3.8	-7.0	8.1	0.5	9.9	-0.2
Move Value	8	6	8	9	10	6	6	8	2	3	8	6	9	9	6	5	4	9	4	7	8	0	10	0
Swap Index	0+8	1+6	2+8	3+9	4+10	5+6	6+6	7+8	8+2	9+3	10+8	11+6	12-9	13-9	14-6	15-5	16-4	17-9	18-4	19-7	20-8	21-0	22-10	23-0
Scrambled Data	'y'	'r'	'n'	'o'	spac e	'e'	spac e	spac e	'v'	'e'	'i'	'm'	'o'	'l'	'H'	spac e	','	'e'	','	'a'	'J'	's'	'a'	'n'

```
text_list=[]
# text=input("Please enter data: ")
text="Hi, everyone, I am Jason"
text.split()
for i in text:
    text_list.append(i)

print("The data that needs to be scrambled is: \n")
for i in range(len(text_list)):
    print(text_list[i],end="")

text_len=len(text)
print("\n\nThe length of the data is: ",text_len)
```

```
The data that needs to be scrambled is:

Hi, everyone, I am Jason

The length of the data is:  24
```

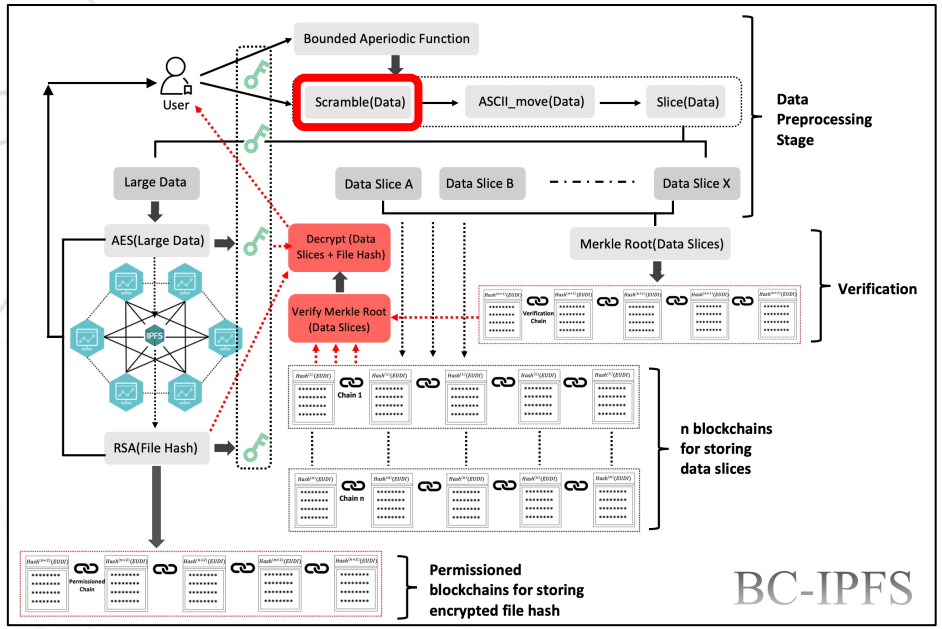
```
    else:
        for i in range(int(text_len/2)):
            temp=text_list[i]
            text_list[i]=text_list[i+move_valu
            text_list[i+move_value[i]]=temp
        for i in range(int(text_len/2)+1,text_
            temp=text_list[i]
            text_list[i]=text_list[i-move_valu
            text_list[i-move_value[i]]=temp

    print("The data that scrambled is: \n")
    for i in range(len(text_list)):
        print(text_list[i],end="")
```

✓ 0.2s

```
The data that scrambled is:

nrv ,,eye HmoIe ai saonJ
```

- Put the letters on the x-axis
- Move values depend on $f_{(x)}$
- Swap data: (index)
  - First part: i ⟺ i + Move Value
  - Second part: i ⟺ i – Move Value

Data Preprocess ----- Data Scramble



Function	$f(x) = 10\sin(16\cos 100x - 3ln\lvert\cos(190x) * 150\rvert)$																							
Data	'H'	'i'	','	space	'e'	'v'	'e'	'r'	'y'	'o'	'n'	'e'	','	space	'I'	space	'a'	'm'	space	'J'	'a'	's'	'o'	'n'
$f(x)$	$f(0)$	$f(1)$	$f(2)$	$f(3)$	$f(4)$	$f(5)$	$f(6)$	$f(7)$	$f(8)$	$f(9)$	$f(10)$	$f(11)$	$f(12)$	$f(13)$	$f(14)$	$f(15)$	$f(16)$	$f(17)$	$f(18)$	$f(19)$	$f(20)$	$f(21)$	$f(22)$	$f(23)$
Value	8.2	5.8	-7.9	8.9	9.7	-6.2	6.0	-7.9	-2.3	3.2	8.2	5.9	9.2	-9.2	-6.3	5.3	4.4	9.3	-3.8	-7.0	8.1	0.5	9.9	-0.2
Move Value	8	6	8	9	10	6	6	8	2	3	8	6	9	9	6	5	4	9	4	7	8	0	10	0
Swap Index	0+8	1+6	2+8	3+9	4+10	5+6	6+6	7+8	8+2	9+3	10+8	11+6	12-9	13-9	14-6	15-5	16-4	17-9	18-4	19-7	20-8	21-0	22-10	23-0
Scrambled Data	'y'	'r'	'n'	'o'	spac e	'e'	spac e	spac e	'v'	'e'	'i'	'm'	'o'	'l'	'H'	spac e	','	'e'	','	'a'	'J'	's'	'a'	'n'

```
text_list=[]
# text=input("Please enter data: ")
text="Hi, everyone, I am Jason"
text.split()
for i in text:
    text_list.append(i)

print("The data that needs to be scrambled is: \n")
for i in range(len(text_list)):
    print(text_list[i],end="")

text_len=len(text)
print("\n\nThe length of the data is: ",text_len)
```

```
The data that needs to be scrambled is:

Hi, everyone, I am Jason

The length of the data is:  24
```

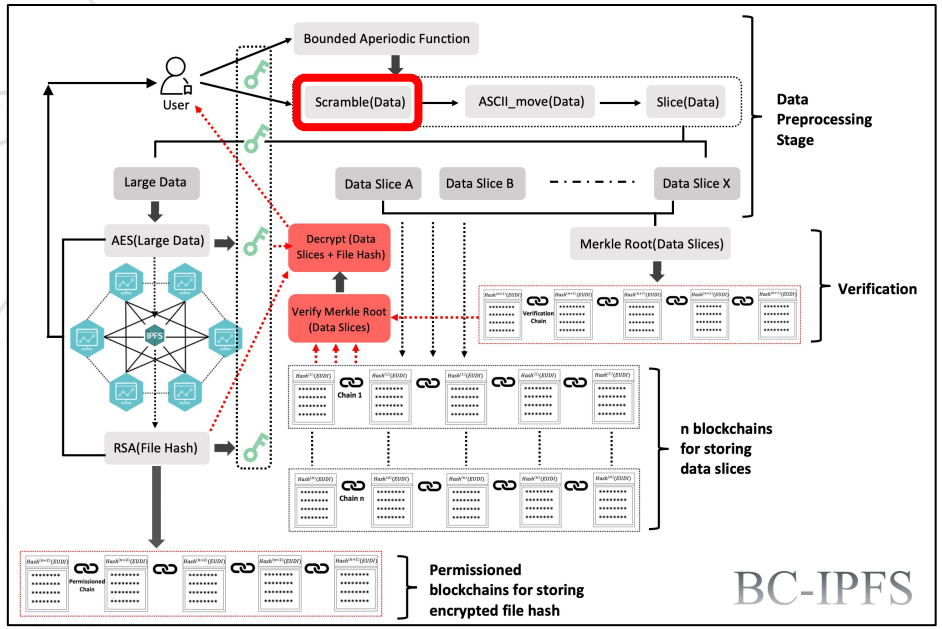
```
    else:
        for i in range(int(text_len/2)):
            temp=text_list[i]
            text_list[i]=text_list[i+move_valu
            text_list[i+move_value[i]]=temp
        for i in range(int(text_len/2)+1,text_
            temp=text_list[i]
            text_list[i]=text_list[i-move_valu
            text_list[i-move_value[i]]=temp

    print("The data that scrambled is: \n")
    for i in range(len(text_list)):
        print(text_list[i],end="")
```

✓ 0.2s

```
The data that scrambled is:

nrv ,,eye HmoIe ai saonJ
```

- Put the letters on the x-axis
- Move values depend on $f_{(x)}$
- Swap data: (index)
  - First part: i ⟺ i + Move Value
  - Second part: i ⟺ i – Move Value

Data Preprocess ----- Data Scramble



Function	$f(x) = 10\sin(16\cos 100x - 3ln\lvert\cos(190x) * 150\rvert)$																							
Data	'H'	'i'	','	space	'e'	'v'	'e'	'r'	'y'	'o'	'n'	'e'	','	space	'I'	space	'a'	'm'	space	'J'	'a'	's'	'o'	'n'
$f(x)$	$f(0)$	$f(1)$	$f(2)$	$f(3)$	$f(4)$	$f(5)$	$f(6)$	$f(7)$	$f(8)$	$f(9)$	$f(10)$	$f(11)$	$f(12)$	$f(13)$	$f(14)$	$f(15)$	$f(16)$	$f(17)$	$f(18)$	$f(19)$	$f(20)$	$f(21)$	$f(22)$	$f(23)$
Value	8.2	5.8	-7.9	8.9	9.7	-6.2	6.0	-7.9	-2.3	3.2	8.2	5.9	9.2	-9.2	-6.3	5.3	4.4	9.3	-3.8	-7.0	8.1	0.5	9.9	-0.2
Move Value	8	6	8	9	10	6	6	8	2	3	8	6	9	9	6	5	4	9	4	7	8	0	10	0
Swap Index	0+8	1+6	2+8	3+9	4+10	5+6	6+6	7+8	8+2	9+3	10+8	11+6	12-9	13-9	14-6	15-5	16-4	17-9	18-4	19-7	20-8	21-0	22-10	23-0
Scrambled Data	'y'	'r'	'n'	'o'	space	'e'	space	space	'v'	'e'	'i'	'm'	'o'	'l'	'H'	space	','	'e'	','	'a'	'J'	's'	'a'	'n'

```
text_list=[]
# text=input("Please enter data: ")
text="Hi, everyone, I am Jason"
text.split()
for i in text:
    text_list.append(i)

print("The data that needs to be scrambled is: \n")
for i in range(len(text_list)):
    print(text_list[i],end="")

text_len=len(text)
print("\n\nThe length of the data is: ",text_len)
```

```
The data that needs to be scrambled is:

Hi, everyone, I am Jason

The length of the data is:  24
```

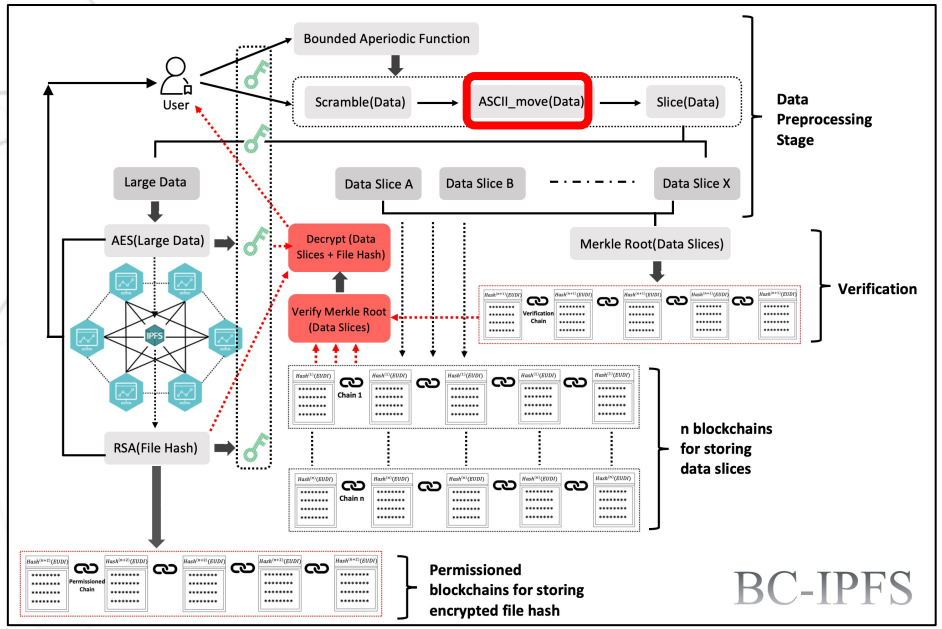
```
    else:
        for i in range(int(text_len/2)):
            temp=text_list[i]
            text_list[i]=text_list[i+move_valu
            text_list[i+move_value[i]]=temp
        for i in range(int(text_len/2)+1,text_
            temp=text_list[i]
            text_list[i]=text_list[i-move_valu
            text_list[i-move_value[i]]=temp

    print("The data that scrambled is: \n")
    for i in range(len(text_list)):
        print(text_list[i],end="")
```

✓ 0.2s

```
The data that scrambled is:

nrv ,,eye HmoIe ai saonJ
```

- Put the letters on the x-axis
- Move values depend on $f_{(x)}$
- Swap data: (index)
  - First part: i ⟺ i + Move Value
  - Second part: i ⟺ i – Move Value

Data Preprocess ----- Change character's ASCII



Function	$f(x) = 10\sin(16\cos 100x - 3ln\lvert\cos(190x) * 150\rvert)$																						
Scrambled Data	'y'	'r'	'n'	'o'	space	'e'	space	space	'v'	'e'	'i'	'm'	'o'	'l'	'H'	space	','	'e'	','	'a'	'J'	's'	'a'	'n'
$f(x)$	$f(0)$	$f(1)$	$f(2)$	$f(3)$	$f(4)$	$f(5)$	$f(6)$	$f(7)$	$f(8)$	$f(9)$	$f(10)$	$f(11)$	$f(12)$	$f(13)$	$f(14)$	$f(15)$	$f(16)$	$f(17)$	$f(18)$	$f(19)$	$f(20)$	$f(21)$	$f(22)$	$f(23)$
Value	8.2	5.8	-7.9	8.9	9.7	-6.2	6.0	-7.9	-2.3	3.2	8.2	5.9	9.2	-9.2	-6.3	5.3	4.4	9.3	-3.8	-7.0	8.1	0.5	9.9	-0.2
Change Value	8	6	-8	9	10	-6	6	-8	-2	3	8	6	9	-9	-6	5	4	9	-4	-7	8	0	10	0
Original ASCII	121	114	110	111	32	101	32	32	118	101	105	109	111	73	72	32	44	101	44	97	74	115	97	110
Changed ASCII	129	120	102	120	42	95	38	24	116	104	113	115	120	64	66	37	48	110	40	90	82	115	107	110
ASCII Changed Data	'·'	'x'	'f'	'x'	'*'	'_'	'&'	'\x18'	't'	'h'	'q'	's'	'x'	'@'	'B'	'%'	'0'	'n'	'('	'Z'	'R'	's'	'k'	'n'

- Put the letters on the x-axis
- Change values depend on $f_{(x)}$
- Changed character ASCII
  - Original ASCII + Change Value

```
            temp=text_list[i]
            text_list[i]=text_list[i-s
            text_list[i-scramble_value

    print("The data that scrambled is:
    for i in range(len(text_list)):
        print(text_list[i],end="")
```

✓ 0.1s

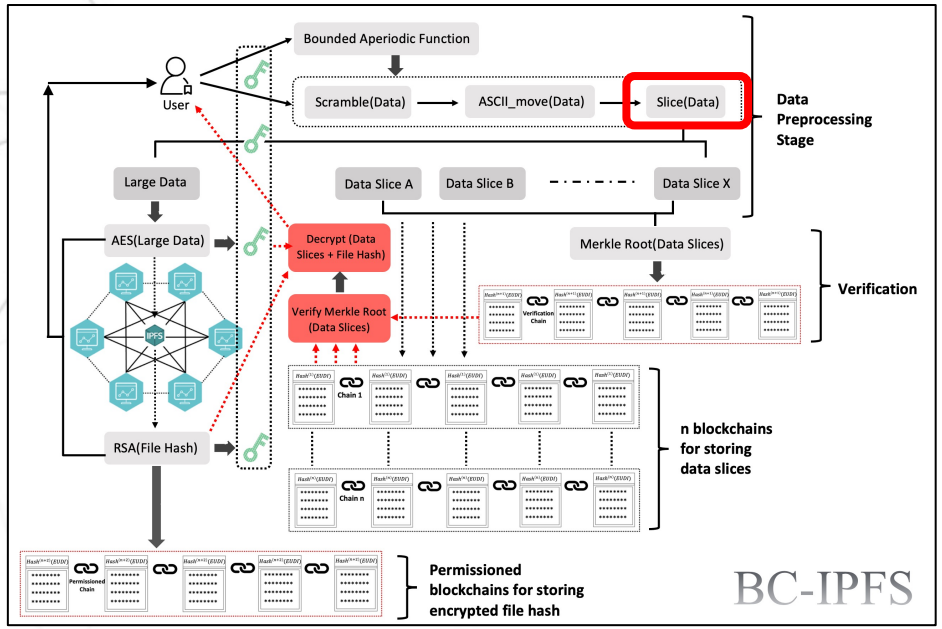
```
The data that scrambled is:

yrno e  veimoIH ,e,aJsan
```

```
    # Move data (by ASCII value)
    moved_data=[]
    for i in range(text_len):
            moved_data.append(chr(ord(

    print("The ASCII changed data is:
    for i in range(text_len):
        print(moved_data[i],end="")
```

✓ 0.2s

```
The ASCII changed data is:

·xfx*_& thqsx@B%0n(ZRskn
```

Data Preprocess -----Slice Data



Function	$f(x) = 10 * \sin(16\cos 100x - 3ln\lvert\cos(190x) * 150\rvert)$																						
ASCII Changed Data	'·'	'x'	'f'	'x'	'*'	'_'	'&'	'\x18'	't'	'h'	'q'	's'	'x'	'@'	'B'	'%'	'0'	'n'	'('	'Z'	'R'	's'	'k'	'n'
$f(x)$	$f(x) = \lvert f(0)\rvert = 8.2$ ---> 8										$f(x) = \lvert f(1)\rvert = 5.8$ ---> 6										$f(x) = \lvert f(2)\rvert = 7.9$ ---> 8			
Data Selected	The 8th character: '\x18'										The 6th letter character: '%'										The 8th character			
Data Slice	Data Slice 8 (Data Slice J)										Data Slice 6 (Data Slice F)										Data Slice 8 (theoretically)			

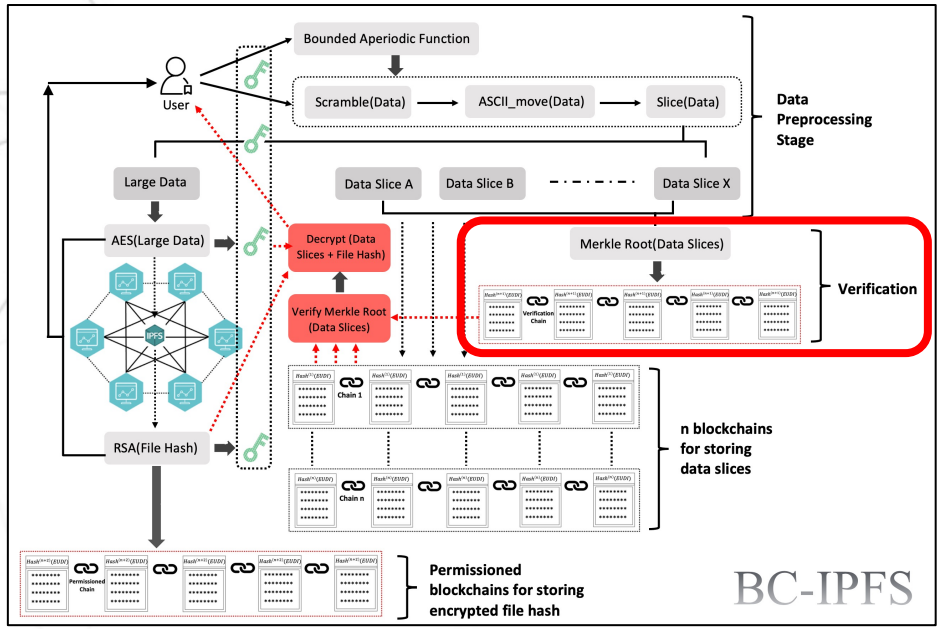
Data Segment 1
- The 8^{th} character will be put in Data Slice 8

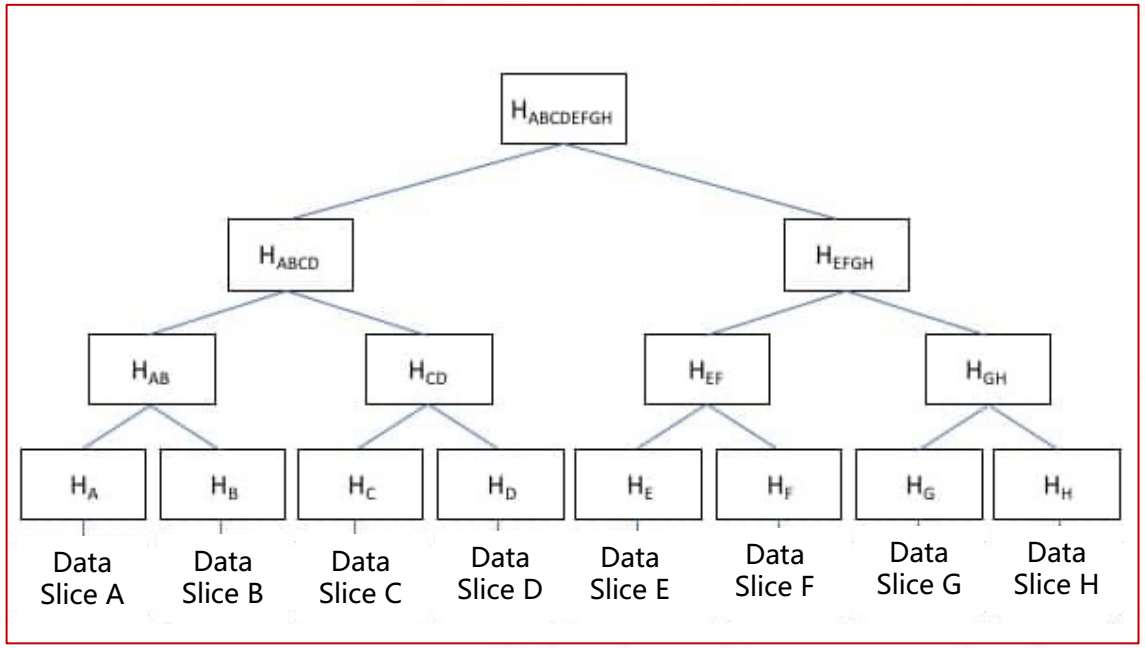
Data Segment 2
- The 6^{th} character will be put in Data Slice 6

- The data will be divided into segments of 10 characters.
- For each segment, the absolute values of the bounded aperiodic function will determine which character in the segment will be stored in which data slice.
- The rest characters will be stored by IPFS. (selected characters will be replaced by space)

```
Data Slice 1:  []
Data Slice 2:  []
Data Slice 3:  []
Data Slice 4:  []
Data Slice 5:  []
Data Slice 6:  ['%']
Data Slice 7:  []
Data Slice 8:  ['\x18']
Data Slice 9:  []
Data Slice 10:  []
Data in IPFS: [·xfx*_& thqsx@B 0n(ZRskn]
```

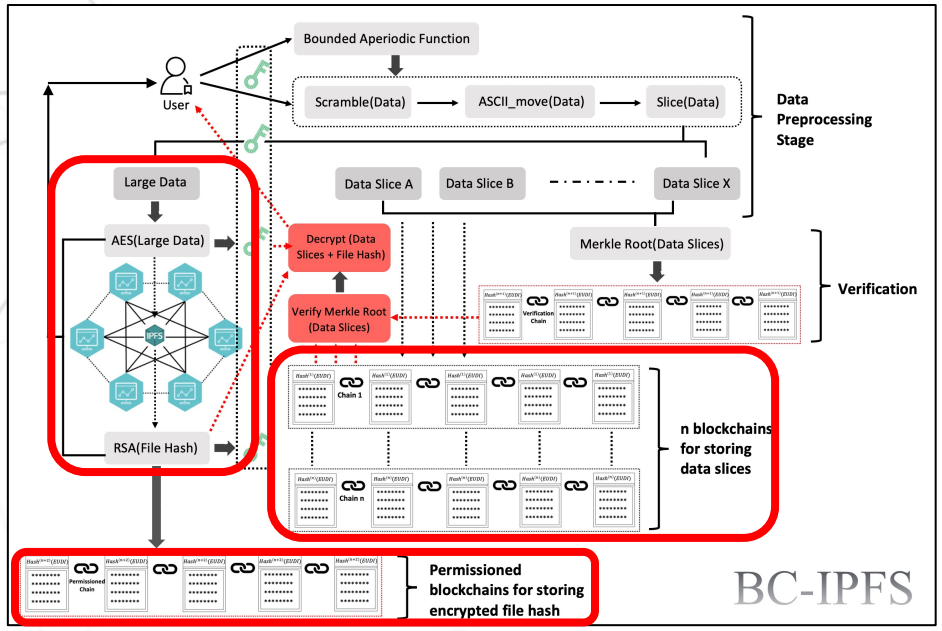
Data Storage -----Generate Merkle Tree





- In this model, data slices will be stored in random blockchains, so when it comes to decoding data, the correct order of data slices must be verified.
- The root hash ($H_{ABCDEFGH}$) will be obtained through the hash calculation of every two sequential adjacent data slices.
- Permutating obtained data slices can check whether the obtained data slices are valid, and in the correct order.
- Meanwhile, the root hash ($H_{ABCDEFGH}$) will be stored in the ‘Verification Chain’.

Data Storage



- **Data slices will be stored in random blockchains.**
  - Data Slice A → Blockchain 2
  - Data Slice B → Blockchain 10
  -
  - Data Slice m → Blockchain n
- Same user use different recursive hashtags to label their data.
  - $Hash^{(n)}(UDI) = Hash(Hash^{(n-1)}(UDI) + UDI)$
  - UDI = User's Digital Identity
- **Large data will be stored by IPFS**
  - AES → Encrypt large data
  - RSA → Encrypt File hash → Permissioned Blockchain

PART 06

Conclusion

Main Challenge

Potential Solutions

Model Evolution

Conclusion

Data Security

In the computer world, the data security model plays a very important role.

Main Challenge

High-speed Brute-force attack
(Quantum Computing)

Potential Solution

- BC-IPFS → Increase the difficulty of getting distributed data
- AES-RSA → Increase the IPFS file security
- Pre-process stage → Increase the difficulty of encryption

Model Evaluation

- Multiple blockchains are utilized to store fragments of information
- Recursive hash calculations of user's digital identity for blockchain
- Only need to keep RSA private key, AES encryption key, BA function

References:

Hill, M & Swinhoe, D 2021, *The 15 biggest data breaches of the 21st century,* CSO, viewed 28 September 2021, <https://www.csoonline.com/article/2130877/the-biggest-data-breaches-of-the-21st-century.html>.

Wikipedia 2021, *Blockchain*, Encyclopaedia, Wikipedia, viewed 24 September 2021, < https://en.wikipedia.org/wiki/Blockchain >.

Santoso, P et. al 2021, 'Systematic literature review: comparison study of symmetric key and asymmetric key algorithm', *IOP*, vol. 1, no. 420, viewed 24 September 2021, < https://iopscience.iop.org/article/10.1088/1757-899X/420/1/012111 >.

Tunggal, A 2021, *The 59 Biggest Data Breaches (Updated for October 2021)*, UpGuard, viewed 23 October 2021, < https://www.upguard.com/blog/biggest-data-breaches>.

Vuleta, B 2021, *How Much Data Is Created Every Day*, blog, viewed 24 September 2021, < https://seedscientific.com/how-much-data-is-created-every-day/ >.

Code Implementation (Data Pre-Process)

```
import random
import math
import matplotlib.pyplot as plt
```
✓ 0.8s

```
# Give the row data
text_list=[]
# text=input("Please enter data: ")
text="Hi, everyone, I am Jason"
# text="In this project you will select a particular topic i
text.split()
for i in text:
    text_list.append(i)

print("The data that needs to be preprocessed is: \n")
for i in range(len(text_list)):
    print(text_list[i],end="")

text_len=len(text)
print("\n\nThe length of the data is: ",text_len)
```
✓ 0.4s

```
The data that needs to be preprocessed is:

Hi, everyone, I am Jason

The length of the data is:  24
```

```
# Generate scramble_value, move_value, select_value
x_value=[]
y_value=[]
scramble_value=[]
change_value=[]
select_value=[]

for i in range(24):
    value=10*math.sin(16*math.cos(100*i)-3*math.log(abs(r
    x_value.append(i)
    y_value.append(value)
    scramble_value.append(abs(round(value)))
    change_value.append(round(value))
    select_value.append(abs(round(value)))

# print(x_value);
# print(y_value)
# print(scramble_value)
print(change_value)

# Plot
plt.figure(figsize=(30,10))
plt.scatter(x_value,change_value)
plt.style.use("dark_background")
```
✓ 0.2s

```
[8, 6, -8, 9, 10, -6, 6, -8, -2, 3, 8, 6, 9, -9, -6, 5, 4, 9
```

```
# Scramble data
if text_len%2==0:
    for i in range(int(text_len/2)):
        temp=text_list[i]
        text_list[i]=text_list[i+scramble_value[i]]
        text_list[i+scramble_value[i]]=temp
    for i in range(int(text_len/2),text_len):
        temp=text_list[i]
        text_list[i]=text_list[i-scramble_value[i]]
        text_list[i-scramble_value[i]]=temp
else:
    for i in range(int(text_len/2)):
        temp=text_list[i]
        text_list[i]=text_list[i+scramble_value[i]]
        text_list[i+scramble_value[i]]=temp
    for i in range(int(text_len/2)+1,text_len):
        temp=text_list[i]
        text_list[i]=text_list[i-scramble_value[i]];
        text_list[i-scramble_value[i]]=temp

print("The data that scrambled is: \n")
for i in range(len(text_list)):
    print(text_list[i],end="")
```
✓ 0.1s

```
The data that scrambled is:

yrno e  veimoIH ,e,aJsan
```

```
# Move data (by Ascii value)
moved_data=[]
for i in range(text_len):
        moved_data.append(chr(ord(text_list[i])+change_value[i]))

print("The ASCII changed data is: \n")
for i in range(text_len):
    print(moved_data[i],end="")
```
✓ 0.1s

```
The ASCII changed data is:

·xfx*_& thqsx@B%0n(ZRskn
```

```
hide_value=[]
for j in range(group_number):
    index=select_value[j]
    value=j*10+index-1
    hide_value.append(value)
for i in range(text_len):
        if(i not in hide_value):
            data_IPFS.append(moved_data[i])
        else:
            data_IPFS.append(" ")

print("Data Slice 1: ",data_slice1)
print("Data Slice 2: ",data_slice2)
print("Data Slice 3: ",data_slice3)
print("Data Slice 4: ",data_slice4)
print("Data Slice 5: ",data_slice5)
print("Data Slice 6: ",data_slice6)
print("Data Slice 7: ",data_slice7)
print("Data Slice 8: ",data_slice8)
print("Data Slice 9: ",data_slice9)
print("Data Slice 10: ",data_slice10)
print("Data in IPFS: [",end="")
for i in range(len(data_IPFS)):
    print(data_IPFS[i],end="")
print("]")
```
✓ 0.2s

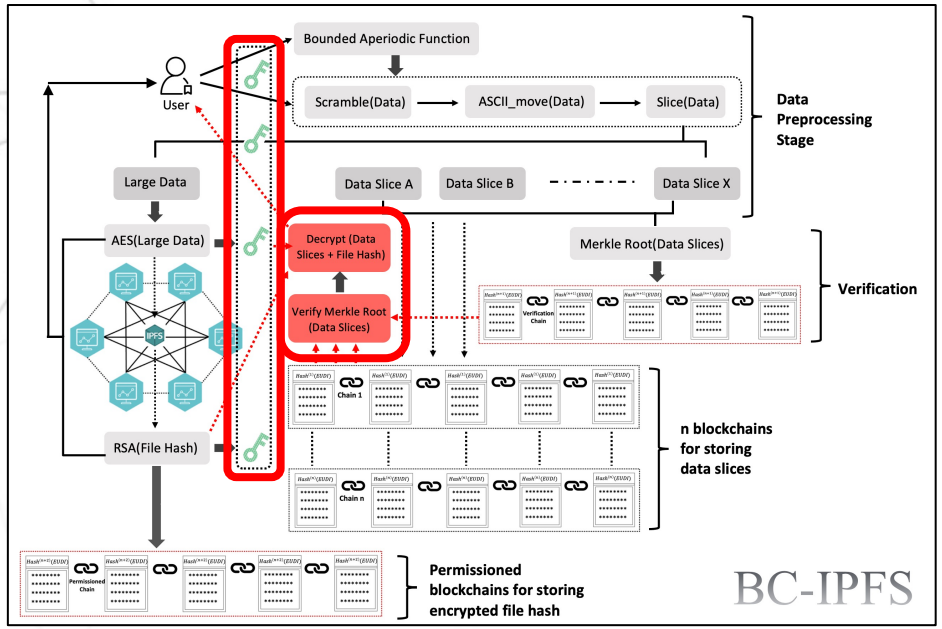
```
# Slice data
data_slice1=[]
data_slice2=[]
data_slice3=[]
data_slice4=[]
data_slice5=[]
data_slice6=[]
data_slice7=[]
data_slice8=[]
data_slice9=[]
data_slice10=[]
data_IPFS=[]

group_number=text_len//10;
for i in range(group_number):
    index=select_value[i]
    if index==1:
        data_slice1.append(moved_data[10*(i)+index-1])
    elif index==2:
        data_slice2.append(moved_data[10*(i)+index-1])
    elif index==3:
        data_slice3.append(moved_data[10*(i)+index-1])
    elif index==4:
        data_slice4.append(moved_data[10*(i)+index-1])
    elif index==5:
        data_slice5.append(moved_data[10*(i)+index-1])
    elif index==6:
        data_slice6.append(moved_data[10*(i)+index-1])
    elif index==7:
        data_slice7.append(moved_data[10*(i)+index-1])
    elif index==8:
        data_slice8.append(moved_data[10*(i)+index-1])
    elif index==9:
        data_slice9.append(moved_data[10*(i)+index-1])
    elif index==10:
        data_slice10.append(moved_data[10*(i)+index-1])
```

```
Data Slice 1:  []
Data Slice 2:  []
Data Slice 3:  []
Data Slice 4:  []
Data Slice 5:  []
Data Slice 6:  ['%']
Data Slice 7:  []
Data Slice 8:  ['\x18']
Data Slice 9:  []
Data Slice 10:  []
Data in IPFS: [·xfx*_& thqsx@B 0n(ZRskn]
```


Thanks for listening


Thanks for listening

Data Decryption



- Get data slices from blockchains (User's Digital Identity)
  - Find the correct sequence of data slices (Merkle Root)

- Get large data from IPFS
  - File hash (Permissioned Blockchain)
  - RSA, AES decryption keys (User)

- Post process data (Bounded Aperiodic Function)
  - Data combination
  - Data ASCII move back
  - Data scramble back