Share Your Data and Control It

An Open Solution to the Post-Corona Data Dilemma

> *2020-09-25*

Ken Takaki, Yussef Milburn, & Yuxi Liu

What we want:

- Public Health
    - collect a lot of data
    - share with a lot of people
    - use in a lot of ways
- Privacy & Security
    - produce little data
    - share with few people
    - restrict usage

It is well known that

You can't have your cake and eat it.

But is it possible to

Share your data for public health and **protect your privacy** at the same time?

The key is *control*.

We use a curtain to protect our privacy.

... but that doesn't mean we close the curtain all the time!

We just decides when to pull it up.

But how does that translate into data privacy/control?

How it’s done now?

Say a user wants to go abroad.

They need to submit

1. an anti-body test result
2. travel history

to

1. embassy (for visa application)
2. departure airport
3. entry airport

COVID-19に関する検査証明
Certificate of Testing for COVID-19

Date of issue_____
交付年月日

氏　名　　パスポート番号
Name _____,　　Passport No._____,
国籍　　生年月日　　性別
Nationality_____,　Date of Birth_____, Sex_____,

上記の者のCOVID-19に関する検査を行った結果、その結果は下記のとおりである。よって、この証明を交付する。
This is to certify the following results which have been confirmed by testing for COVID-19 conducted with the sample taken from the above-mentioned person.

採取検体 Sample （下記いずれかをチェック/Check one of the boxes below)	検査法 Testing for COVID-19 （下記いずれかをチェック/Check one of the boxes below)	結果 Result	①決定年月日 Result Date ②検体採取日時 Sampling Date and Time	備考 Remarks
□鼻咽頭ぬぐい液 Nasopharyngeal Swab □唾液　Saliva	□核酸増幅検査（real time RT-PCR法) nucleic acid amplification test (real time RT-PCR) □核酸増幅検査（LAMP法)		① ②	

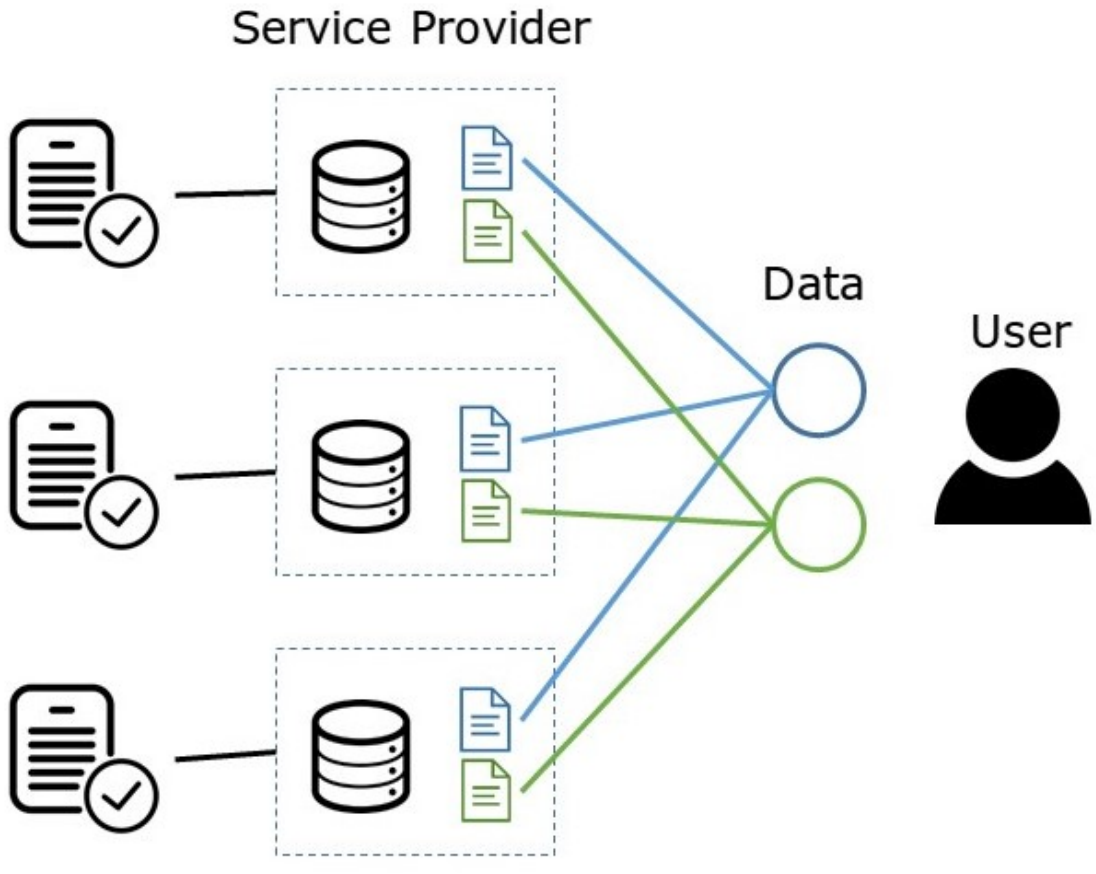
	nucleic acid amplification test (LAMP) □抗原定量検査 antigen test (CLEIA)			

医療機関名 Medical institution _____

住所 Address of the institution _____

医師名 Signature by doctor _____

An imprint of a seal 印影


Service Provider
Data
User

Problems:

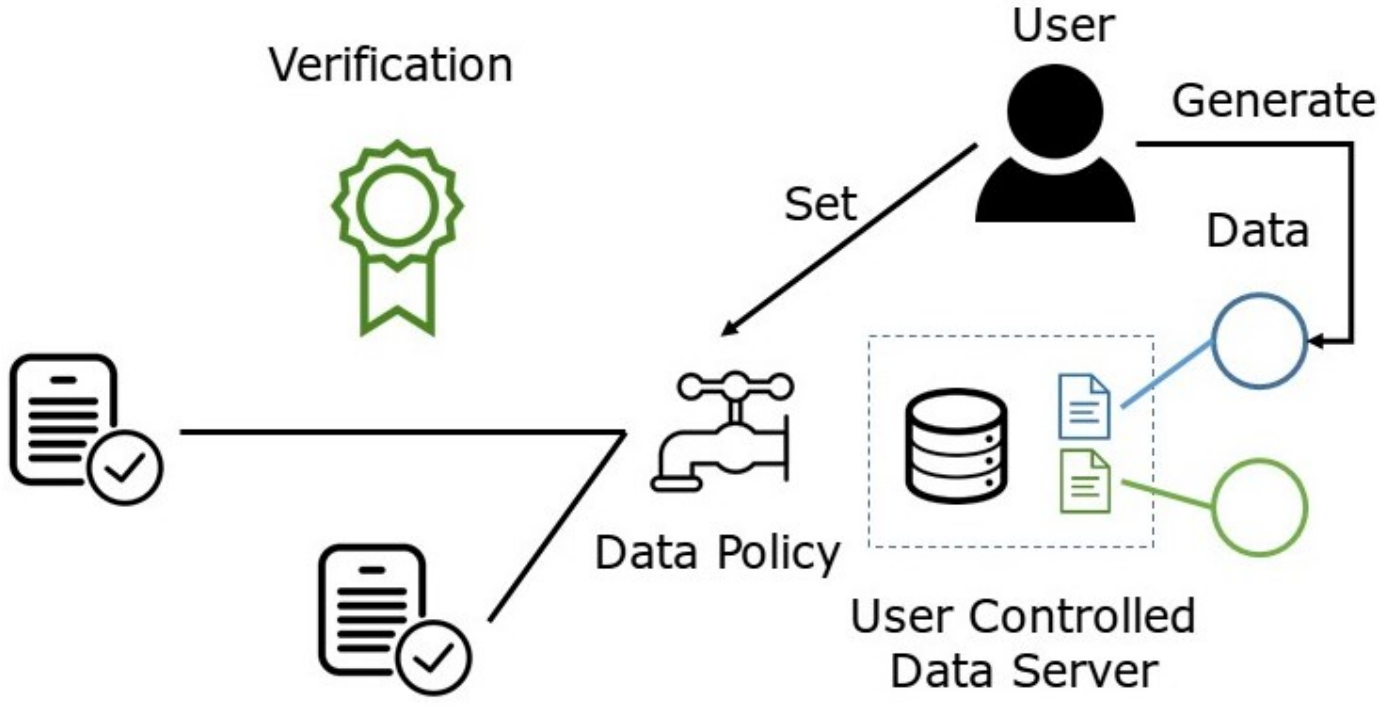
- m documents to n facilities = $m \times n$ forms!
- Handwriting: errors? forge?
- Is the **โรงพยาบาลพริ้นซ์** hospital qualified to issue a certificate?
- Misuse?
- Leak?

Idea:

- Let user control their own data
- Service providers use the same open format

Further problem:

- Integrity?


Verification
User
Generate
Set
Data
Data Policy
User Controlled
Data Server

Feature: Interoperability

The data format is open and well-known.

So any service provider can use the same format.

E.g. the user can transfer the data to embassy & airports without any conversion.

In [2]:

```
# Taking a test

# the medical facility issues the result
testing_res1 = { 'time': '2020-09-21'
               , 'facility': 'two-point hospital'
               , 'is_negative': True
               }
verified_medical_results.append(testing_res1)

# user sends the result to server (can be done automatically)
req = { 'type':  REQ_DATA_UPDATE
      , REQ_DATA_UPDATE: DATA_TEST_RES
      , DATA_TEST_RES: [testing_res1]
      }
res = send(req)
print(res)
```

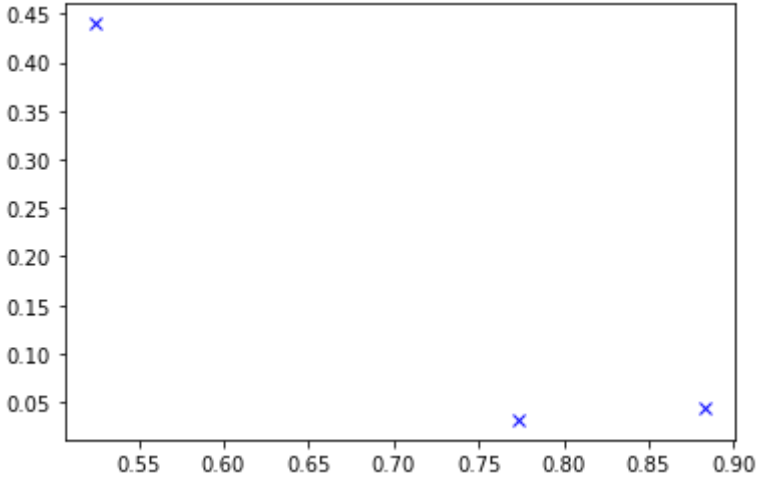
```
('update_success', None)
```

In [3]:

```
# Travelling to places

travelling_history = [(random(), random()) for _ in range(3)]
print(travelling_history)
plot_dots(travelling_history, 'bx')
```

```
[(0.883393606892517, 0.04510941498829235), (0.5241552541074321, 0.4397465117818138), (0.7733363225742038, 0.03196467410673054)]
```



In [5]:

```
# Send travel history to data server

req = { 'type': REQ_DATA_UPDATE
      , REQ_DATA_UPDATE: DATA_TRAVEL
      , DATA_TRAVEL: travelling_history
      }
res = send(req)
print(res)
```

```
('update_success', None)
```

In [6]:

```
# User allows embassy to read data

req = { 'type': REQ_POLICY_UPDATE
      , POLICY_KEY: 'embassy'
      , REQ_POLICY_UPDATE: { POLICY_VIEW: [DATA_TEST_RES, DATA_TRAVEL]
                           , POLICY_UPDATE: []
                           , POLICY_COMPUTE: []
                           }
      }
res = send(req)
print(res)
```

```
('update_success', None)
```

In [7]:

```
# Embassy gets data from the server

req = { 'type': REQ_SP
      , POLICY_KEY: 'embassy'
      , REQ_SP_ACT: POLICY_VIEW
      , REQ_SP_PARAM: DATA_TEST_RES
      }
res = send(req)
print(res)

status, data = res
```

```
('service_provider_success', [{'time': '2020-09-21', 'facility': 'two-point hospi
tal', 'is_negative': True}])
```

In [8]:

```
# Embassy checks the test result

today = date.today()

is_recent = lambda x: x >= 0 and x <= 5
is_qualified = lambda h: h in ['two-point hospital', 'another good hospital']

recent_tests = [test for test in data if is_recent((today -
date.fromisoformat(test['time'])).days) and is_qualified(test['facility'])]
if recent_tests and all(test['is_negative'] for test in recent_tests):
    print('PASSED')
else:
    print('DENIED')
```

```
PASSED
```

Automatic Handling: The user only needs to set the policy.

The data server will handle the data.

In [9]:

```
# User sets policy to allow departure airport to read

req = { 'type': REQ_POLICY_UPDATE
      , POLICY_KEY: 'departure airport'
      , REQ_POLICY_UPDATE: { POLICY_VIEW: [DATA_TEST_RES, DATA_TRAVEL]
                           , POLICY_UPDATE: []
                           , POLICY_COMPUTE: []
                           }
      }
res = send(req)
print(res)
```

```
('update_success', None)
```

In [10]:

```
# Departure Airport requests both data, and use a different requirement
# But user doesn't need to do anything

req = { 'type': REQ_SP
      , POLICY_KEY: 'departure airport'
      , REQ_SP_ACT: POLICY_VIEW
      , REQ_SP_PARAM: DATA_TEST_RES
      }
res = send(req)
print(res)

status, tests = res
all_neg = all(test['is_negative'] for test in tests)
```

```
('service_provider_success', [{'time': '2020-09-21', 'facility': 'two-point hospi
tal', 'is_negative': True}])
```

In [11]:

```
# Airport gets travelling history
req = { 'type': REQ_SP
      , POLICY_KEY: 'departure airport'
      , REQ_SP_ACT: POLICY_VIEW
      , REQ_SP_PARAM: DATA_TRAVEL
      }
res = send(req)
print(res)
```

```
('service_provider_success', [(0.883393606892517, 0.04510941498829235), (0.524155
2541074321, 0.4397465117818138), (0.7733363225742038, 0.03196467410673054)])
```

In [12]:

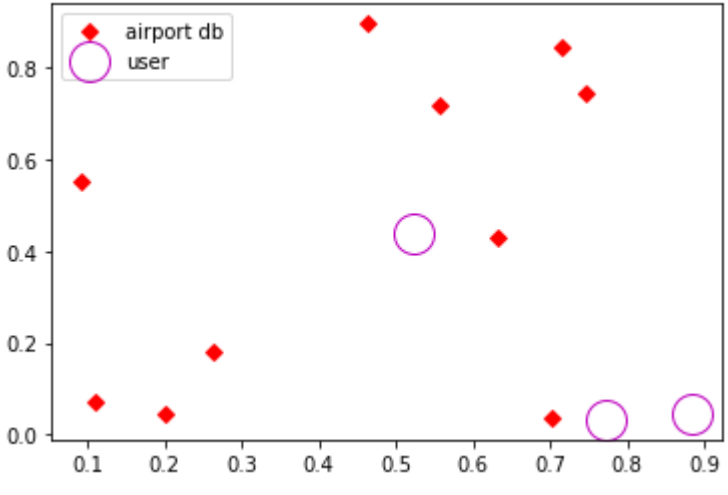
```
# Airport checks travelling history
status, history = res

airport_db = deepcopy(fast_database)
no_contact = not(any(close(p, case) for p in history for case in airport_db))

plot_dots(airport_db, 'rD', label='airport db')
plot_dots(travelling_history, 'mo', markersize=20, fillstyle='none', label='user')
plt.legend()
```

Out[12]:

```
<matplotlib.legend.Legend at 0x1bbc0e66370>
```



In [13]:

```
# if both succeeds

if all_neg and no_contact:
    print('Good')
else:
    print('Please return')
```

```
Good
```

Interoperability: The departure airports uses a different standard from embassy.

But they share the same data format.

User only needs to set the policy.

After departure, don't want the airport to read more data.

In [14]:

```
# User updates policy to stop further sharing

req = { 'type': REQ_POLICY_UPDATE
      , POLICY_KEY: 'departure airport'
      , REQ_POLICY_UPDATE: { POLICY_VIEW: []
                           , POLICY_UPDATE: []
                           , POLICY_COMPUTE: []
                           }
      }
res = send(req)
print(res)
```

```
('update_success', None)
```

In [15]:

```
# If the airport tries to read

req = { 'type': REQ_SP
      , POLICY_KEY: 'departure airport'
      , REQ_SP_ACT: POLICY_VIEW
      , REQ_SP_PARAM: DATA_TEST_RES
      }
res = send(req)
print(res)
```

```
('not_conform_policy', None)
```

Exiting: The airport gets rejected.

User can stop giving data whenever it's not necessary.

They have total control over their data.

If a user can

- see what's been collected
- know where it's stored
- control their data in anyway, anytime

... maybe they want to share more!

Feature: Data Verification

In [18]:

```
# the medical facility issues the result

testing_res2 = { 'time': '2020-09-22'
               , 'facility': 'another good hospital'
               , 'is_negative': False # !NOTICE HERE!
               }

# it also records the data to a reliable database (can be provided by a blockchain)
verified_medical_results.append(testing_res2)

# user sends the result to server, but modifies the data
modified = deepcopy(testing_res2)
modified['is_negative'] = True

req = { 'type':  REQ_DATA_UPDATE
      , REQ_DATA_UPDATE: DATA_TEST_RES
      , DATA_TEST_RES: [modified]
      }
res = send(req)
print(res)
```

```
('update_success', None)
```

In [21]:

```
# embassy uses a new check

status, data = res

is_verified = lambda x: verify(x) #NOTIC: new check#

recent_tests = [test for test in data if is_recent((today -
date.fromisoformat(test['time'])).days) and is_qualified(test['facility'])]

if any(not(is_verified(test)) for test in recent_tests):
    print('WARNING: Using qualified facility, but data unverifiable')
elif recent_tests and all(test['is_negative'] for test in recent_tests):
    print('PASSED')
else:
    print('DENIED')
```

```
WARNING: Using qualified facility, but data unverifiable
```

Quick question

Why not just use the reliable database?

Because...

If verification is

- centralized, then decentralized data is meaningless.
- decentralized -> blockchain, not efficient to store data

In the reliable database, we only store proofs (certifications).

The proof

- is shorter
- doesn't reveal the content (c.f. zero-knowledge proof).

Limitation

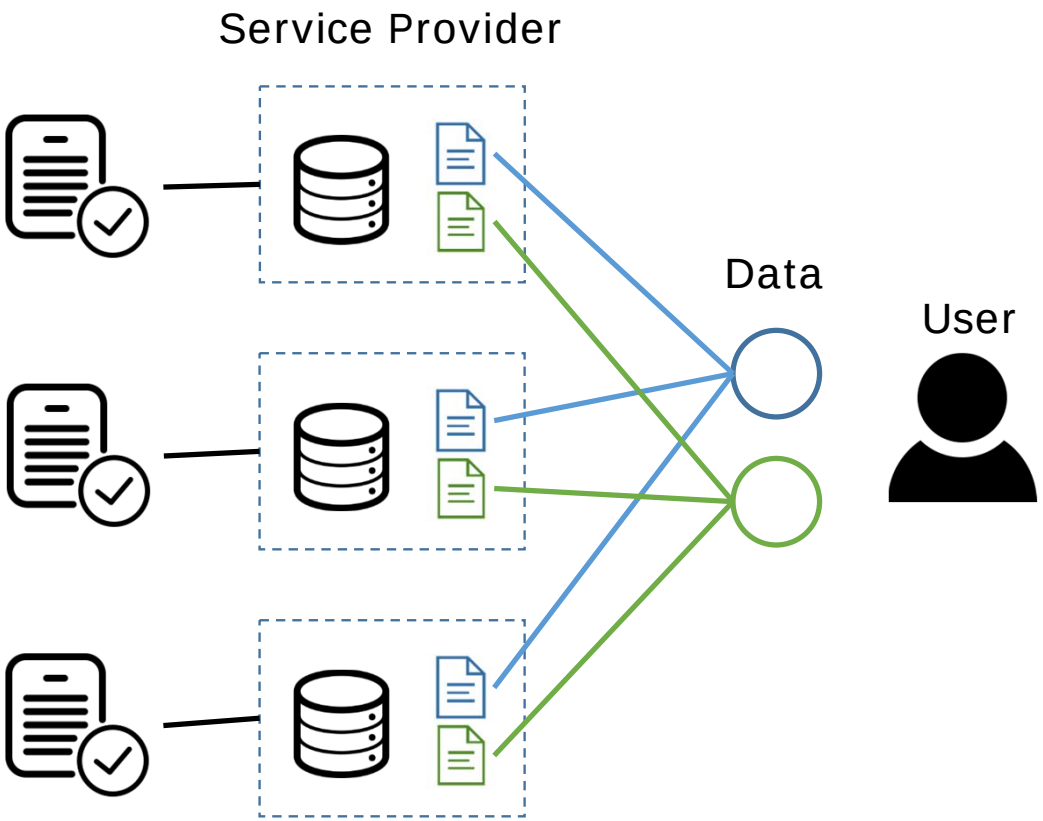
- User: only simple datatype
- Format: not open, ad hoc
- Data server: simple tcp server
- Verification: no real certification

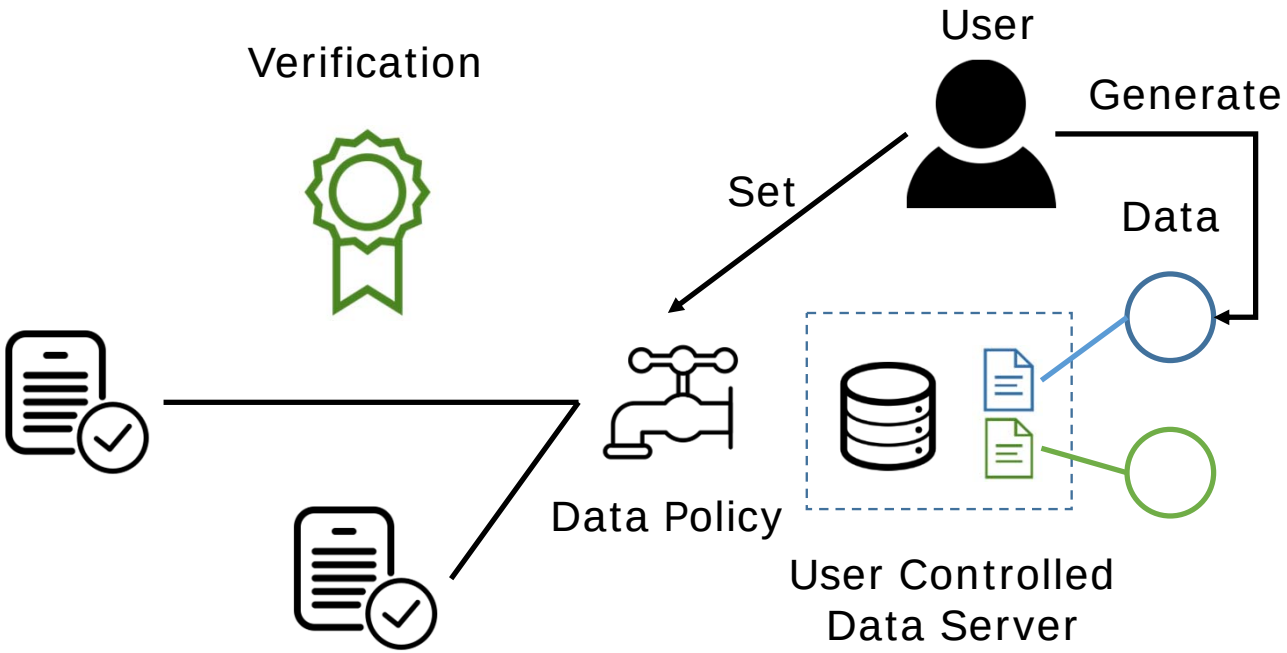
However,

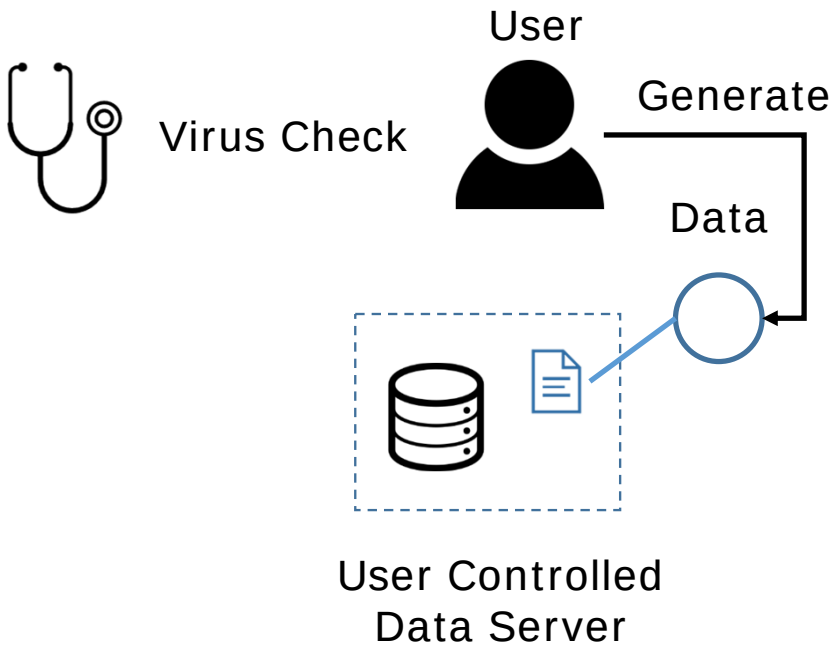
- Lots of research on data format
    - Resource Description Framework
    - Linked Data Fragments
- Verification API is similar

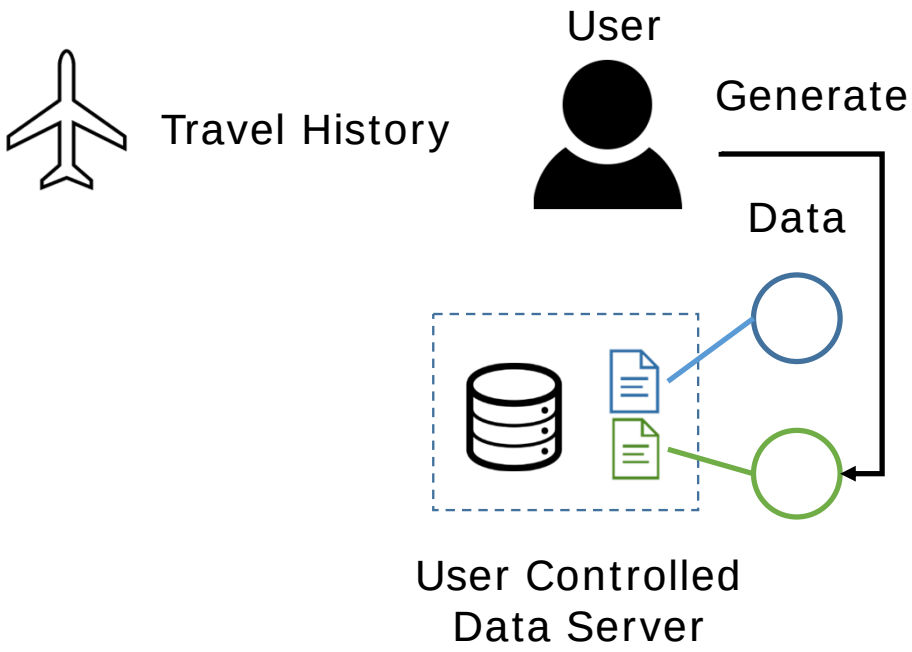
Future Work

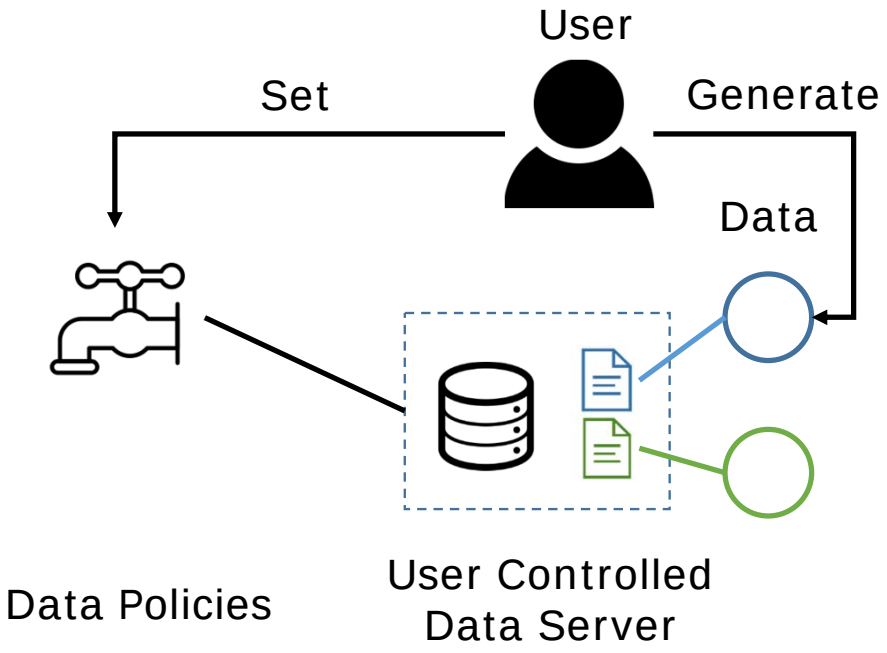
- Design a COVID-19 related open data format
- Use real verification API
- Advocate the idea to a broader audience

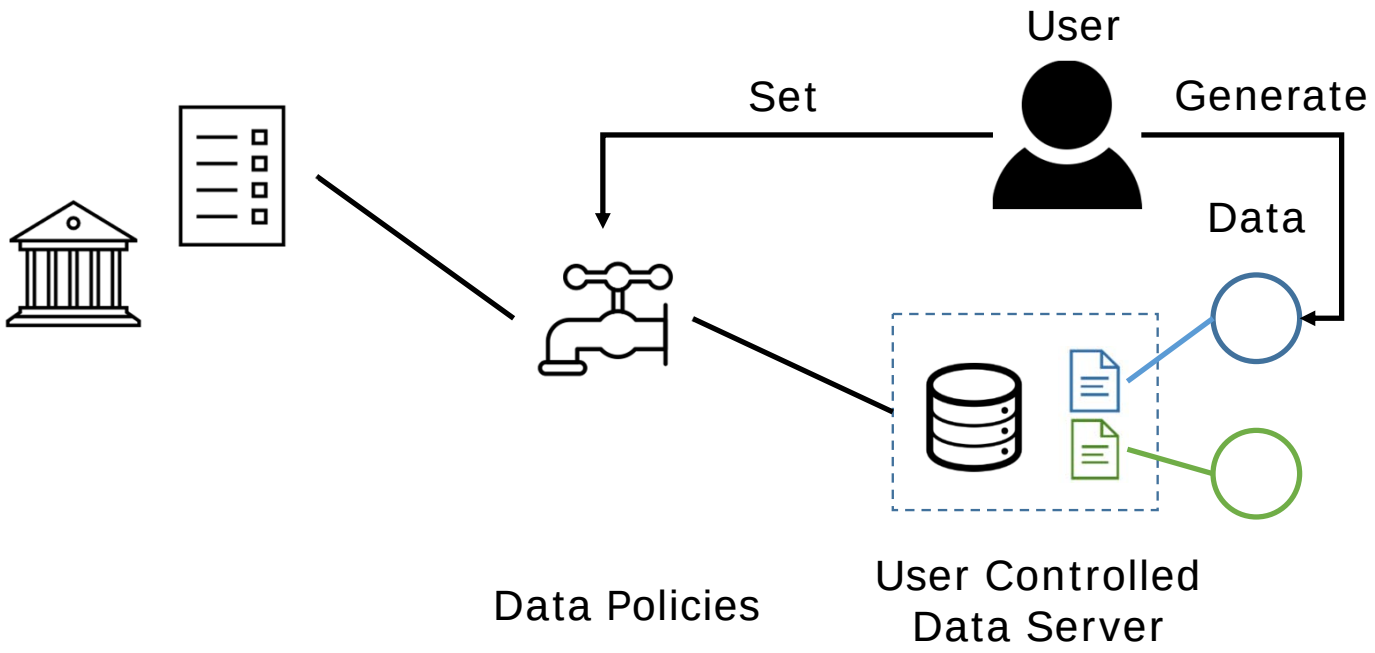

Service Provider
Data
User

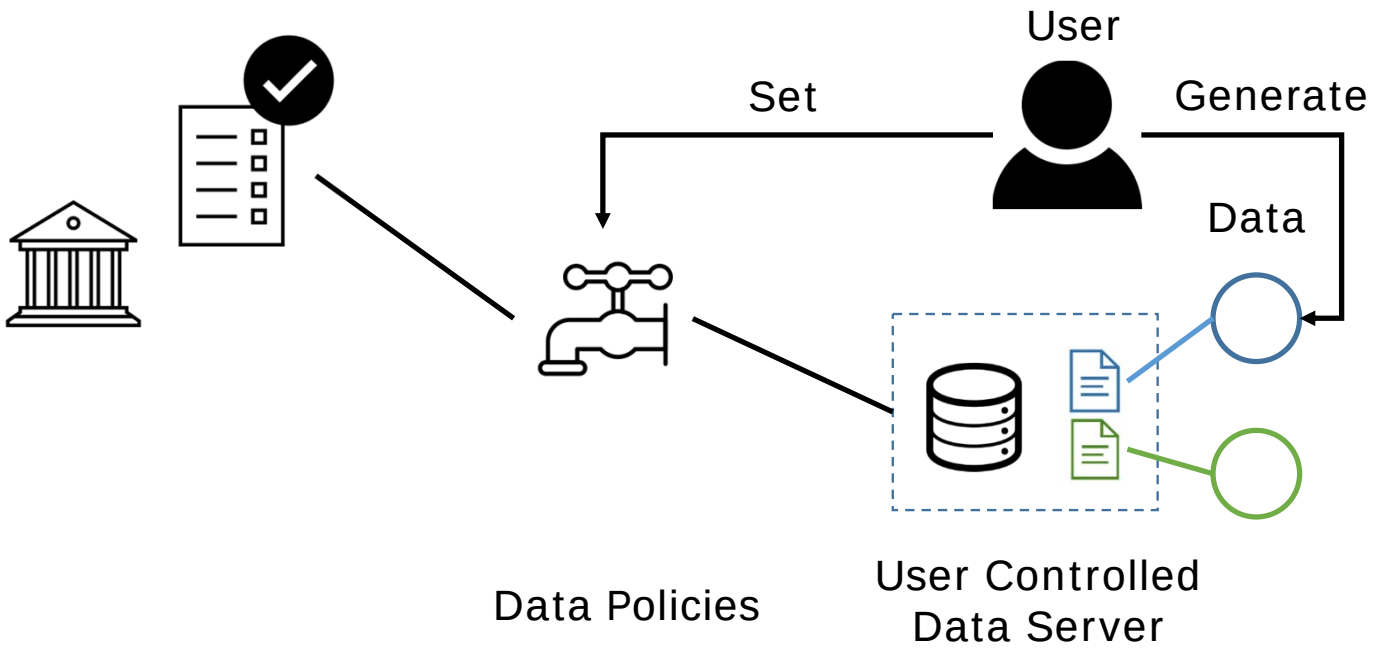

Verification
User
Generate
Set
Data
Data Policy
User Controlled
Data Server

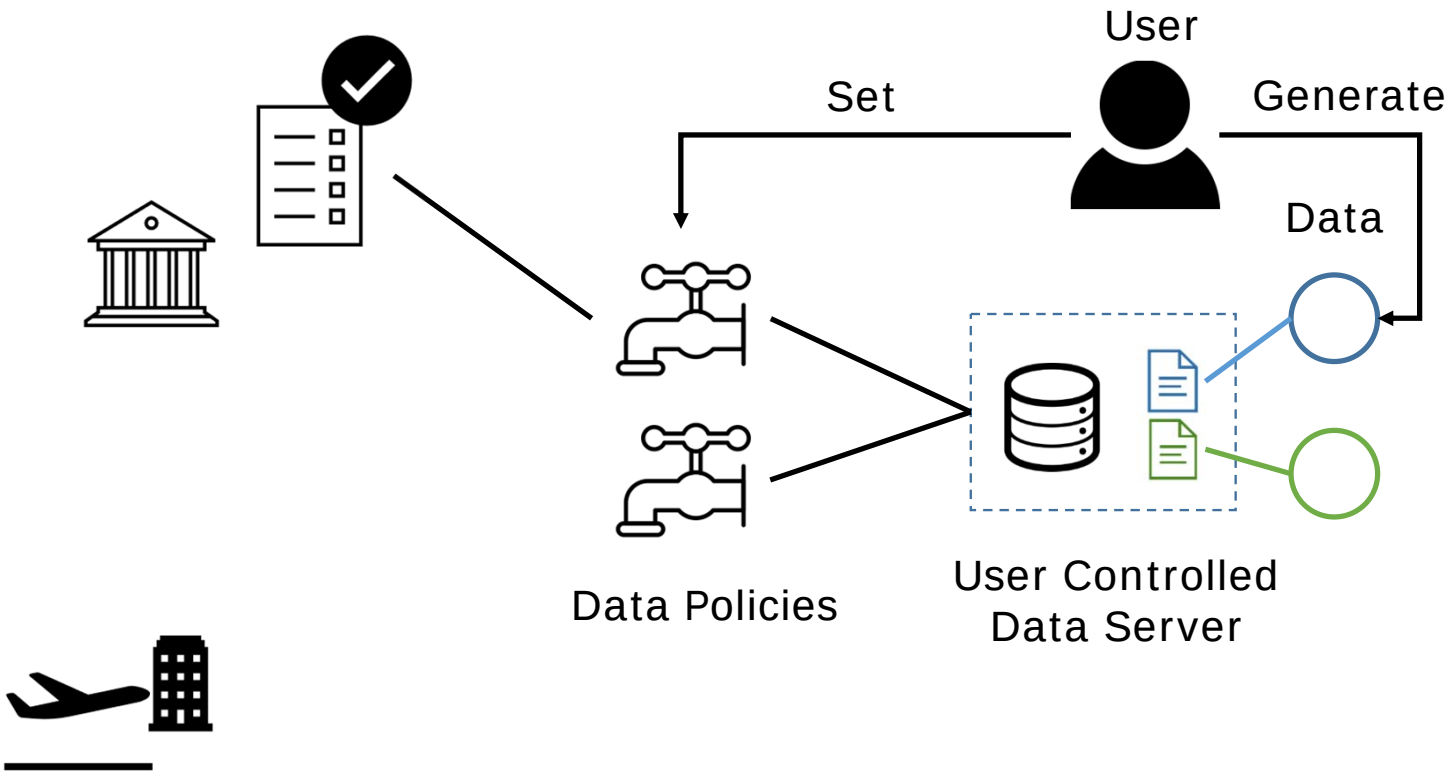

User
Virus Check
Generate
Data
User Controlled
Data Server

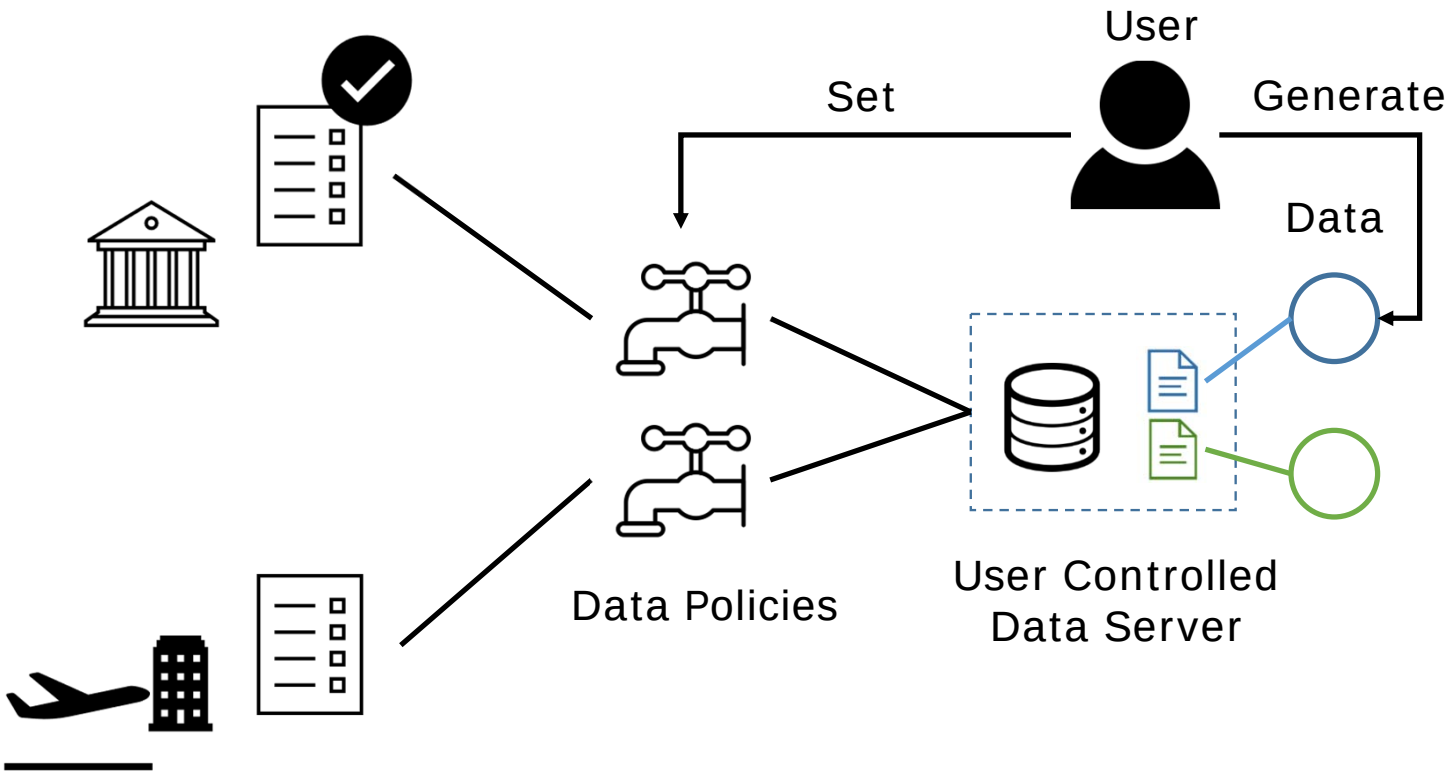

Travel History
User
Generate
Data
User Controlled
Data Server

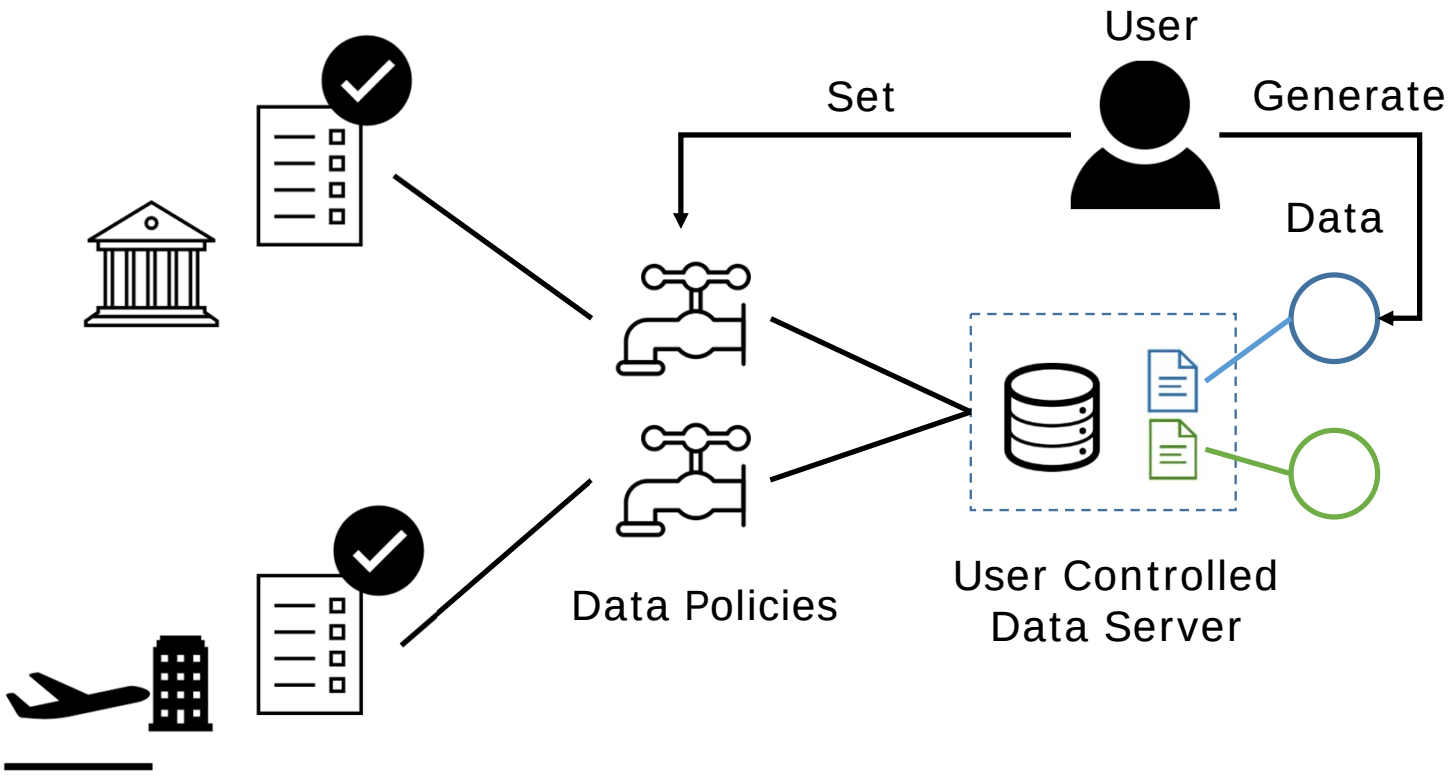

User
Set
Generate
Data
Data Policies
User Controlled
Data Server

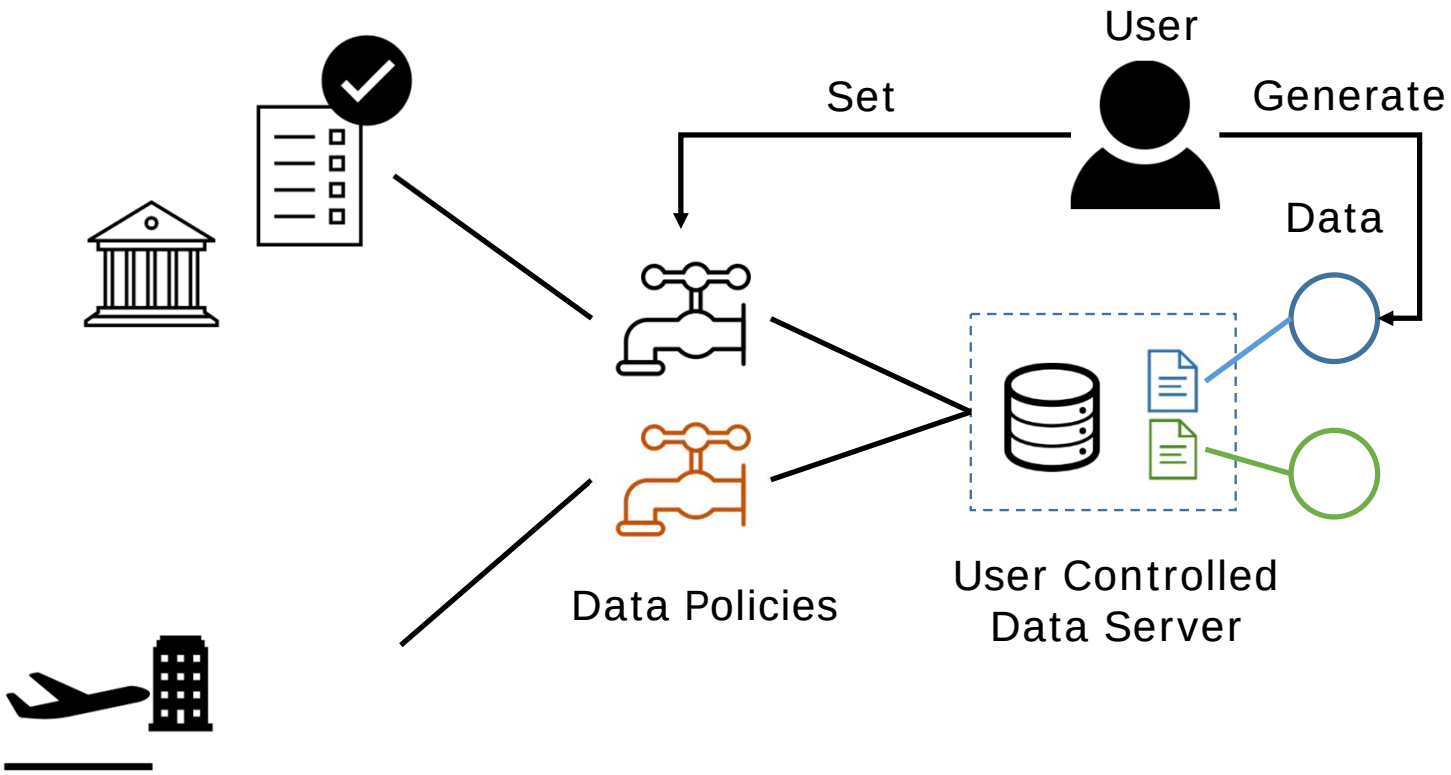

User
Set
Generate
Data
Data Policies
User Controlled
Data Server

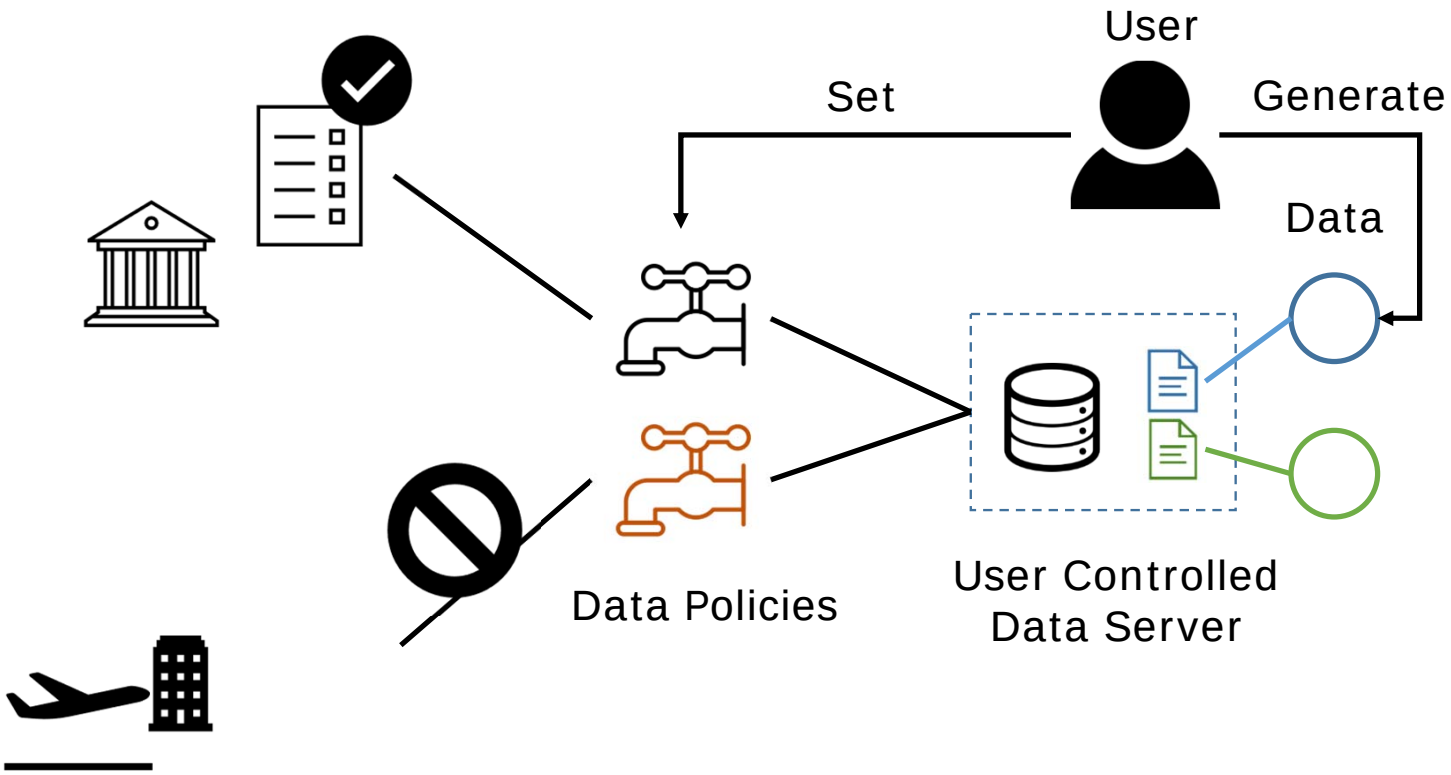

User
Set
Generate
Data
Data Policies
User Controlled
Data Server

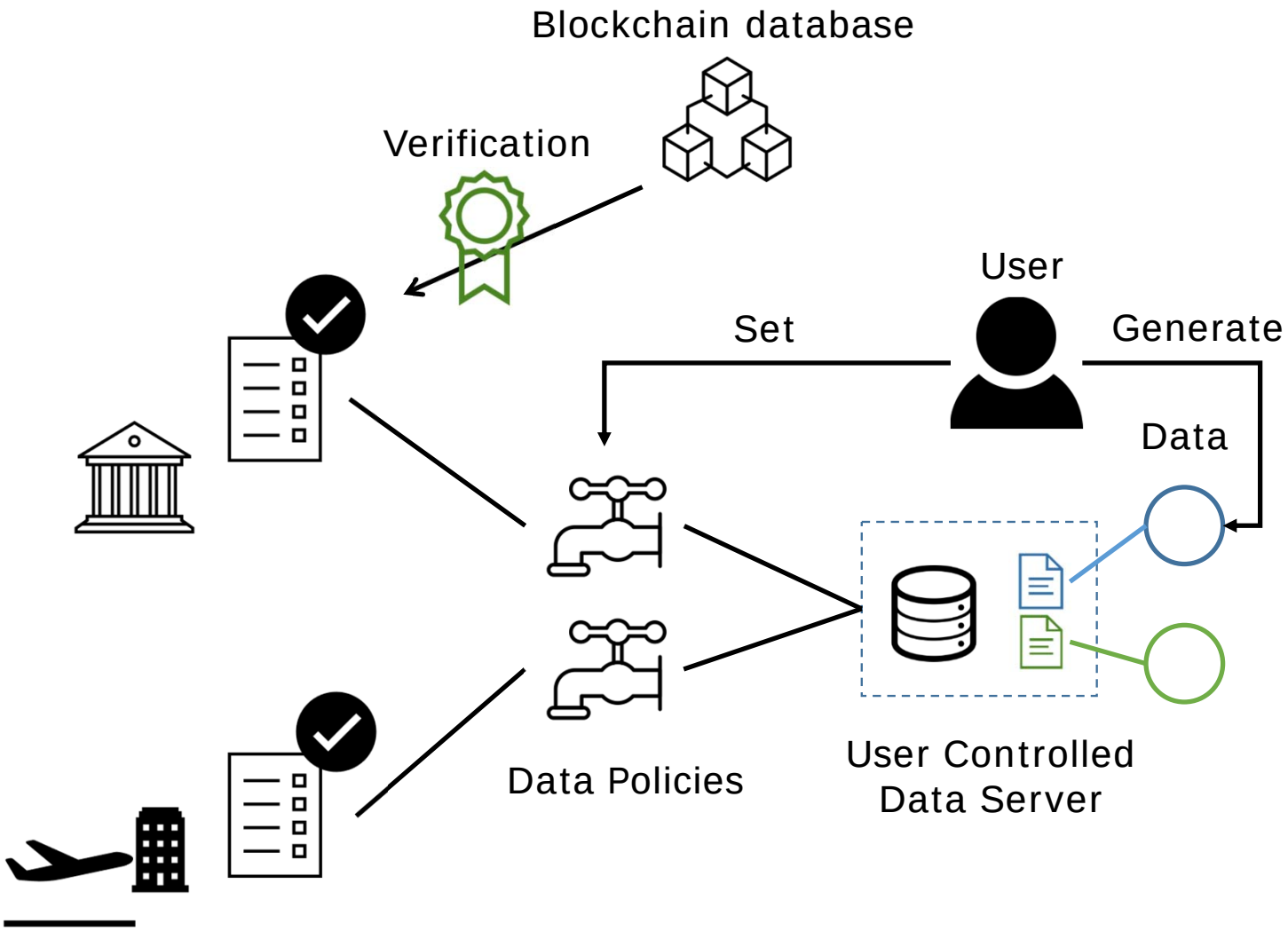

User
Set
Generate
Data
Data Policies
User Controlled
Data Server


User
Set
Generate
Data
Data Policies
User Controlled
Data Server


User
Set
Generate
Data
Data Policies
User Controlled
Data Server


User
Set
Generate
Data
Data Policies
User Controlled
Data Server


User
Set
Generate
Data
Data Policies
User Controlled
Data Server


Blockchain database
Verification
User
Set
Generate
Data
Data Policies
User Controlled
Data Server