IoT and Blockchain

Engineering and Economics

Phillip G. Bradford (University of Connecticut, Stamford CT. **USA)**

phillip.bradford@uconn.edu

Outline

What data are we transferring?

    Sustaining blockchains

Proof-of-work

Ledgers

Generating Blocks on RPIs

ClockChains

IoT and Properties of Blockchain

Autonomous nodes

Immutability

Verifiability

→ Consensus based

Data

Account numbers: public keys

Anyone can send you a message securely
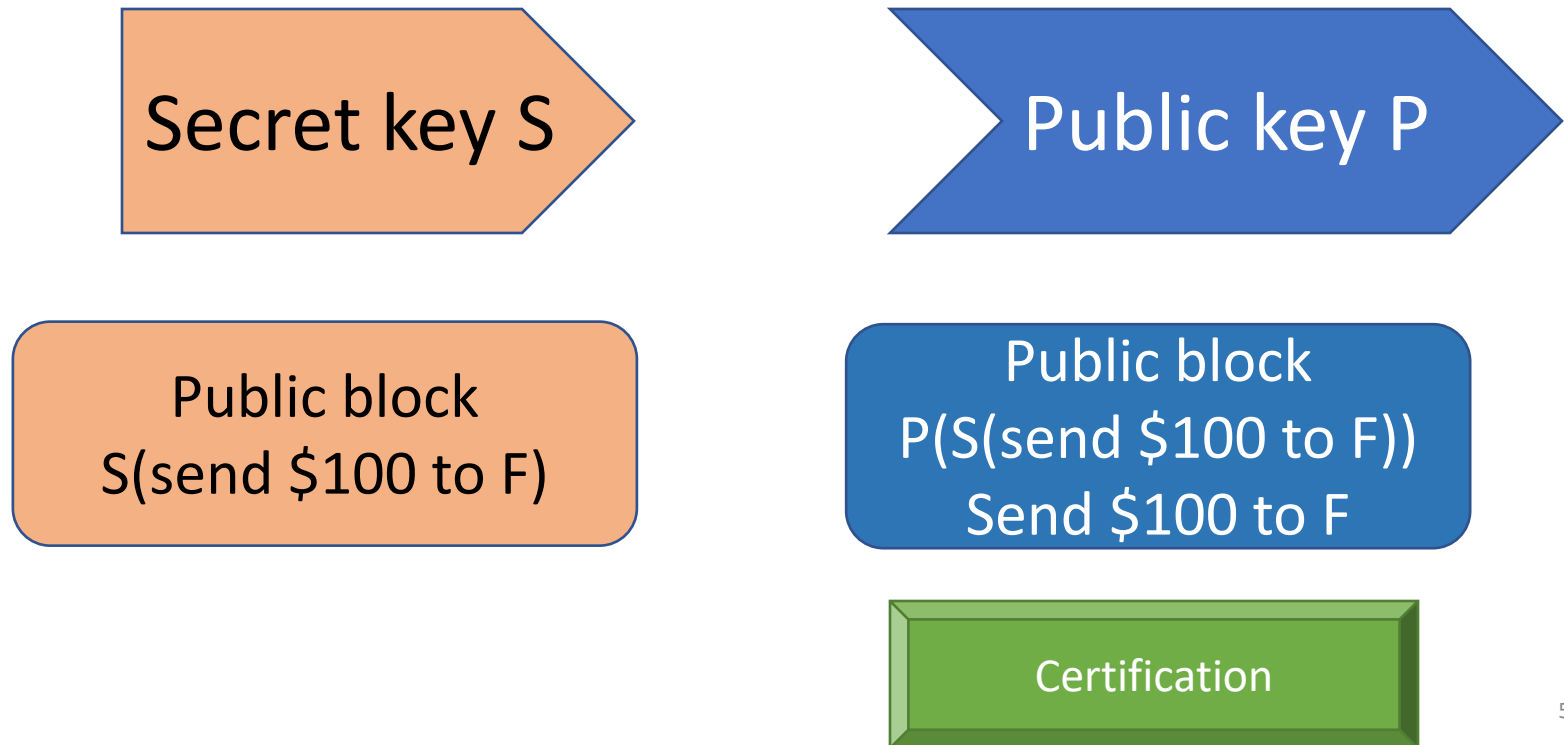
Transfer resources

Intractable to get your secret key

Signatures by private keys

Verify signature using public key

Data

If Alan-T wants to send you digital money how do they do it?

Secret key S

Public key P

Public block
S(send $100 to F)

Public block
P(S(send $100 to F))
Send $100 to F

Certification

Data

Alan-T

1. Determines the amount to send you

2. Takes to-public-key

3. Compute S(amount, to-public-key)

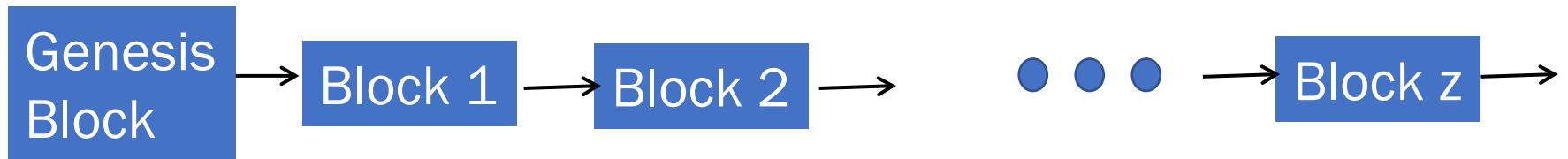[S(amount, to-public-key), from-public-key]

Verification?

Pay yourself first

Each miner inserts a payment to themselves
    This self-pay is pre-agreed upon

xfers {
[S(amount, to-public-key), from-public-key ;
 . . .
 ]
}

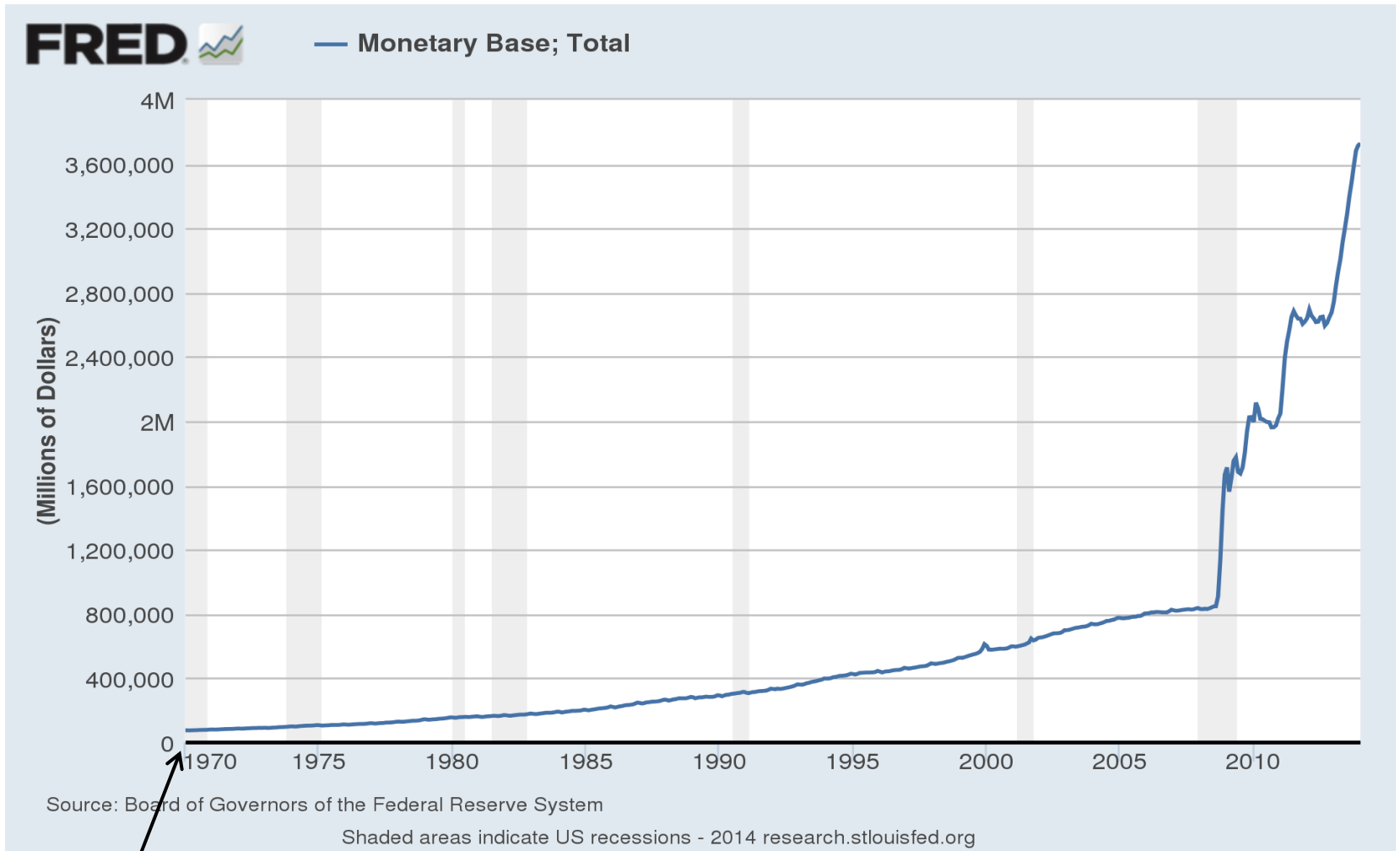Also, can require a transaction fee!

Blockchain: public ledger

Genesis Block → Block 1 → Block 2 → ● ● ● → Block z →

All transactions in blocks
New block about every X seconds
Can trade all money from the start

https://blockchain.info/

US Money Supply

First Unix-day

Bitcoin example

Max of about 21 million total Bitcoins by 2140

Geometric series
  50 + 25 + 12.5 + 6.25 + 3.125 + 1.5625 + ...

Last term about $\dfrac{1}{100,000,000}$!

Last term takes about 33*4 = 132 years

New Bitcoins

About ever 10 minutes

Halve amount added ever 4 years
   2009-2013: add 50 BTC every ~10 minutes
   2014-2017: add 25 BTC every ~10 minutes
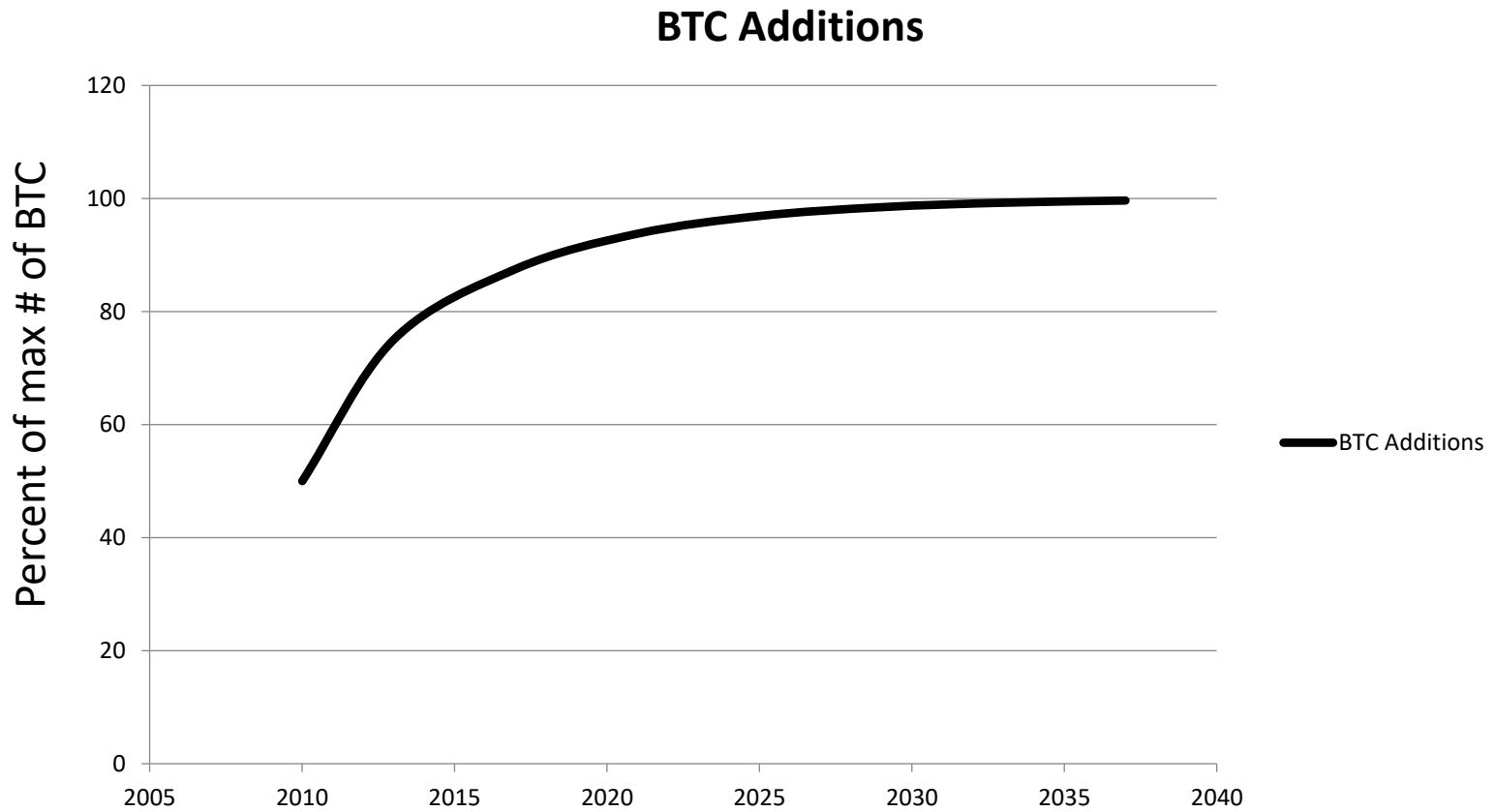   2018-2021: add 12.5 BTC every ~10 minutes
   2022-2025: add 6.25 BTC every ~10 minutes
   .

Last year for new BTC is 2140
   Almost none added after 2030

Bitcoin Supply

BTC Additions

Money Purpose & Properties

Purpose
  Medium of exchange
  Unit of account
  Store of value

Properties
  Opportunities for earning & commerce
  Safety, stability, supply, storage
  Fungibility, divisibility, liquidity, exchange & transfer
  Opportunity for investment

Money - Technology Dependent

5,000 BC – metal money

760 BC – coins

960 AD – paper money

2009 – operating digital currency Bitcoin
        Satoshi Nakamoto
            See also Chaum 1982

Blockchain Currencies

No central authority

Seignorage – difference in cost of making money and its value

    Seignorage to miners not a central bank

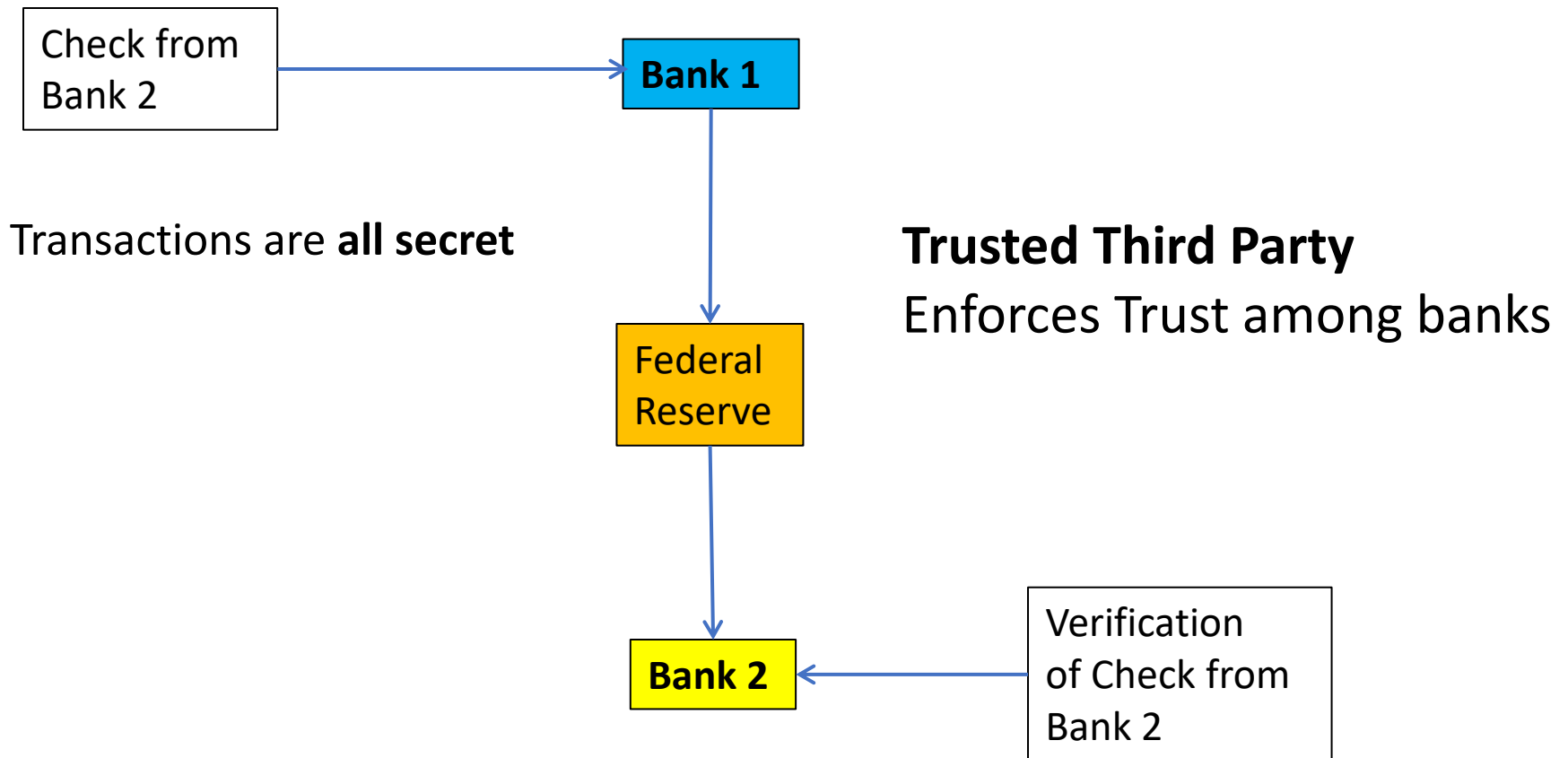   Bottom up rather than current top-down

    Transaction fees

Legal tender issues

Jurisdiction issue

Privacy model: Classic

The user & firms are responsible for the secret keys

Check from Bank 2 → **Bank 1**

Transactions are **all secret**

Bank 1 → Federal Reserve

Trusted Third Party
Enforces Trust among banks

Federal Reserve → **Bank 2**

Bank 2 ← Verification of Check from Bank 2

Privacy model: Bitcoin

The user is responsible for the secret keys & security

Entity 1
Wallet

All Transactions public
Anonymity is up to
individuals
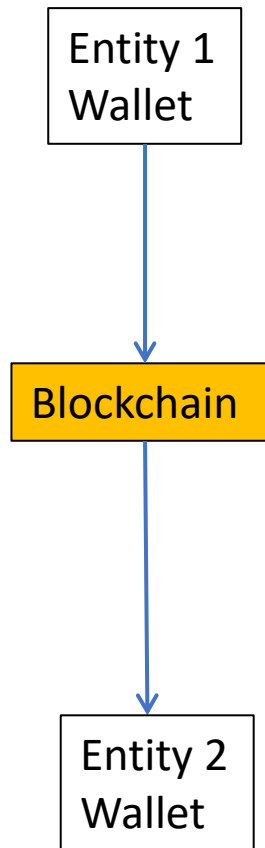
Blockchain

Verified Trust Enforces
Trust among banks

Entity 2
Wallet

Large-scale IoT Economics

Distributed ledger
  Autonomous

Paying nodes to support the network – mining

"Want weak currency now, strong currency in the future"

Synchronous Clockchains

Clockchains are not Blockchains

https://github.com/wonder-phil/2022Mining

Mine on RPI_1

```python
import paramiko

RPI_1 = paramiko.SSHClient()

RPI_1.set_missing_host_key_policy(paramiko.AutoAddPolicy())
RPI_1.connect(hostname='localhost',username='pi',port=5022)

stdout=RPI_1.exec_command('python3 testMine.py')
output = stdout.readlines()
for items in output:
    print("RPI_1:" + items)

RPI_1.close()
```

Two RPI synchronous test

```python
RPI_1.connect(hostname='localhost',username='pi',port=5022,password=
RPI_2.connect(hostname='localhost',username='pi',port=5023,password=

stdin,stdout,stderr=RPI_2.exec_command('python3 testMine.py ' + "3")
output = stdout.readlines()
for items in output:
    print("RPI_2:" + items)

stdin,stdout,stderr=RPI_1.exec_command('python3 testMine.py ' + "2")
output = stdout.readlines()
for items in output:
    print("RPI_1:" + items)
```

Two non-RPI Almost-Clock-Chain

```python
class AlmostClockChain(threading.Thread):
    def __init__(self, name, difficulty):
        threading.Thread.__init__(self)
        self.name = name
        self.difficulty = difficulty

    def run(self):
        print("Starting thread "+self.name+"\n")
        tm = TestMine(self.difficulty)

t1=AlmostClockChain("Non-RPI_1",5)
t2=AlmostClockChain("Non-RPI_2",3)
t1.start()
t2.start()
```

Two RPI ClockChains

See code
  ClockChain.py

Where to from here?