Blockhains and Distributed Ledgers Lecture 01

Aggelos Kiayias

THE UNIVERSITY of EDINBURGH

Lecture 01

- Introduction to blockchain.
- What is a distributed ledger.
- Hash functions and digital signatures.
- Transactions in Bitcoin.

Understanding DL's

- Distributed ledgers use blockchain protocols as one main means of implementation.

- The blockchain is a distributed database that satisfies a unique set of safety and liveness properties.

- To understand it, we can focus to its first (and so far most successful) application.

Case study : Money

- What is money?

Money is useful

(1874) A man offering chicken
for a yearly newspaper subscription

THE COUNTRY EDITOR PAYING THE YEARLY SUBSCRIPTION.

Properties of Money

can be used as medium
for the exchange
of goods - no bartering

- a medium of exchange
- a unit of account
- a store of value

can be used for
pricing of all goods
and services, for
accounting purposes
and debt recording.

storing and retrieving it at a
point in the future maintains
its value.

Creating Money

Money 1.0 : using a trusted object

Analysis of Money 1.0

- a medium of exchange

mediocre
[ok for face to face transactions]

- a unit of account

mediocre fungible, but not divisible well. typically forgeable.

- a store of value

bad. some objects may deteriorate, others may have unknown hidden quantities.

Creating Money

Money 2.0 : using a trusted entity

Trusted entity issues “IOU”s

Analysis of Money 2.0

- a medium of exchange
- a unit of account
- a store of value

good
[for transactions within the domain of the trusted entity]

great!
fungible & divisible.

mediocre
[tied to the availability & reputation of the issuing entity]
[possible

Creating Money

Money 3.0 : Bitcoin

Enter Blockchains & Distributed Ledgers

The never-ending book parable

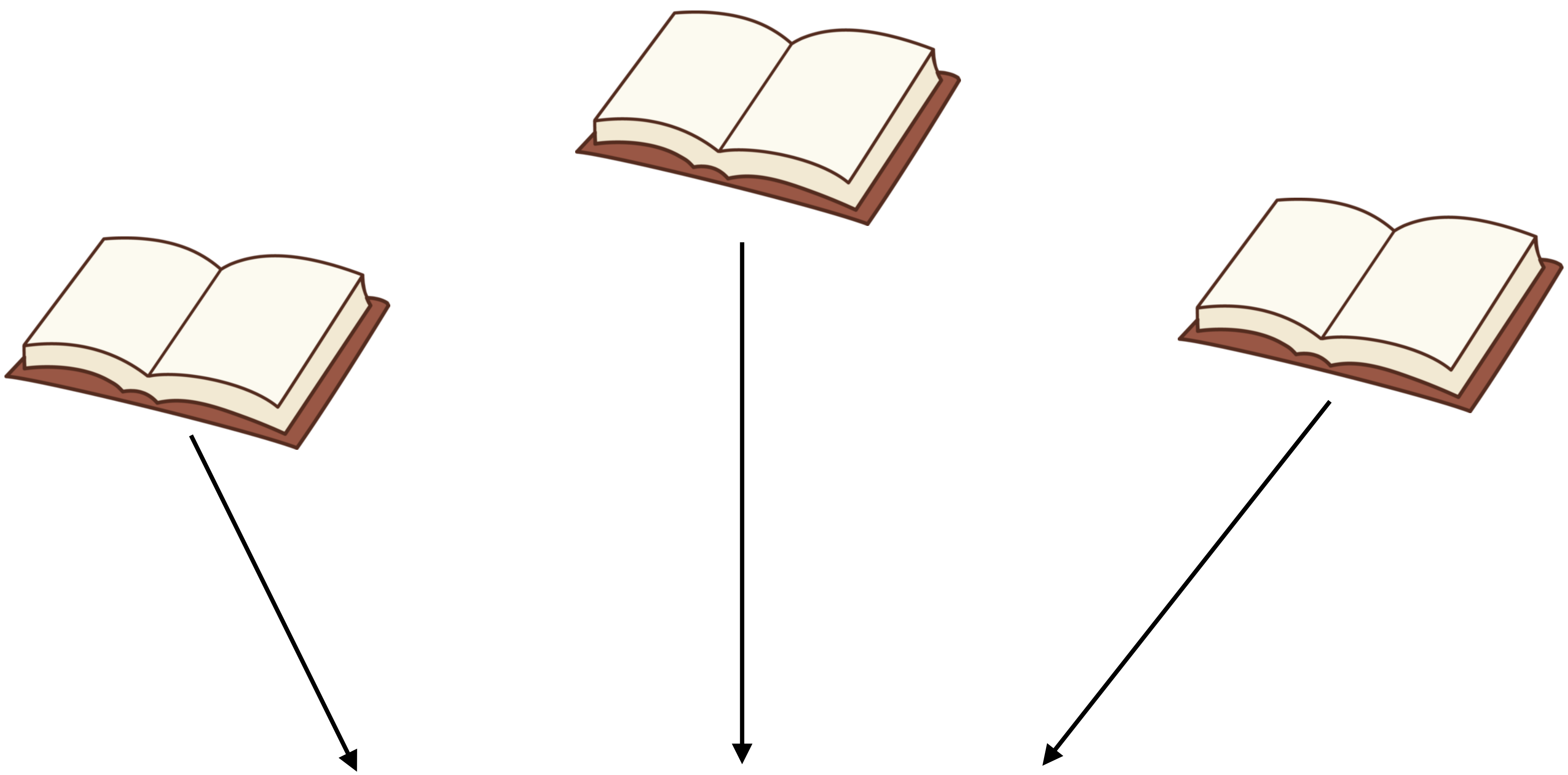
A "book" of transactions

- Each new page requires some effort to produce.
- Anyone can be a scribe and produce a page.
- New pages are produced indefinitely as long as scribes are interested in doing so.

Importance of Consensus

- If multiple conflicting books exist, which is the “right one”?

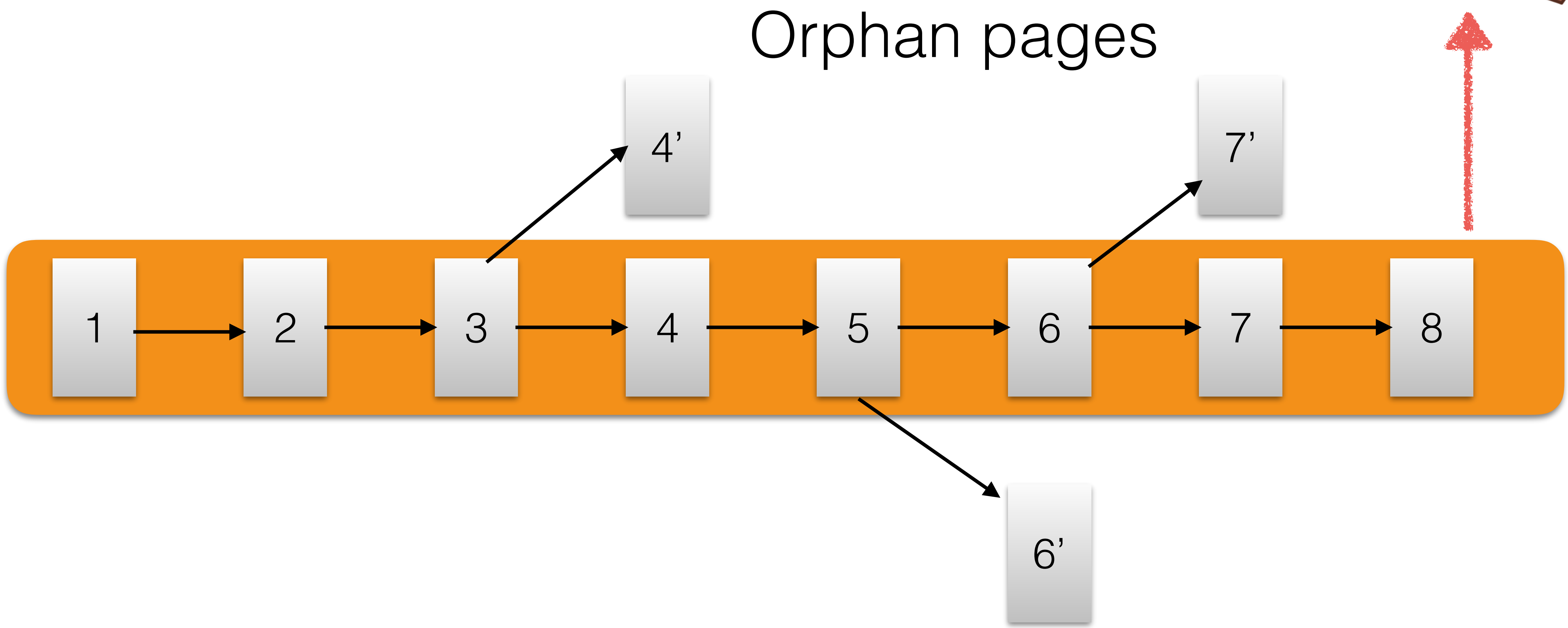
Choosing the correct book



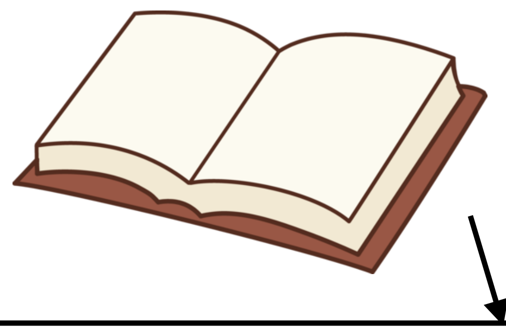
The **current book** to work on & refer to is the book with the most pages. if multiple exist, just pick one at random.

Assembling the current book

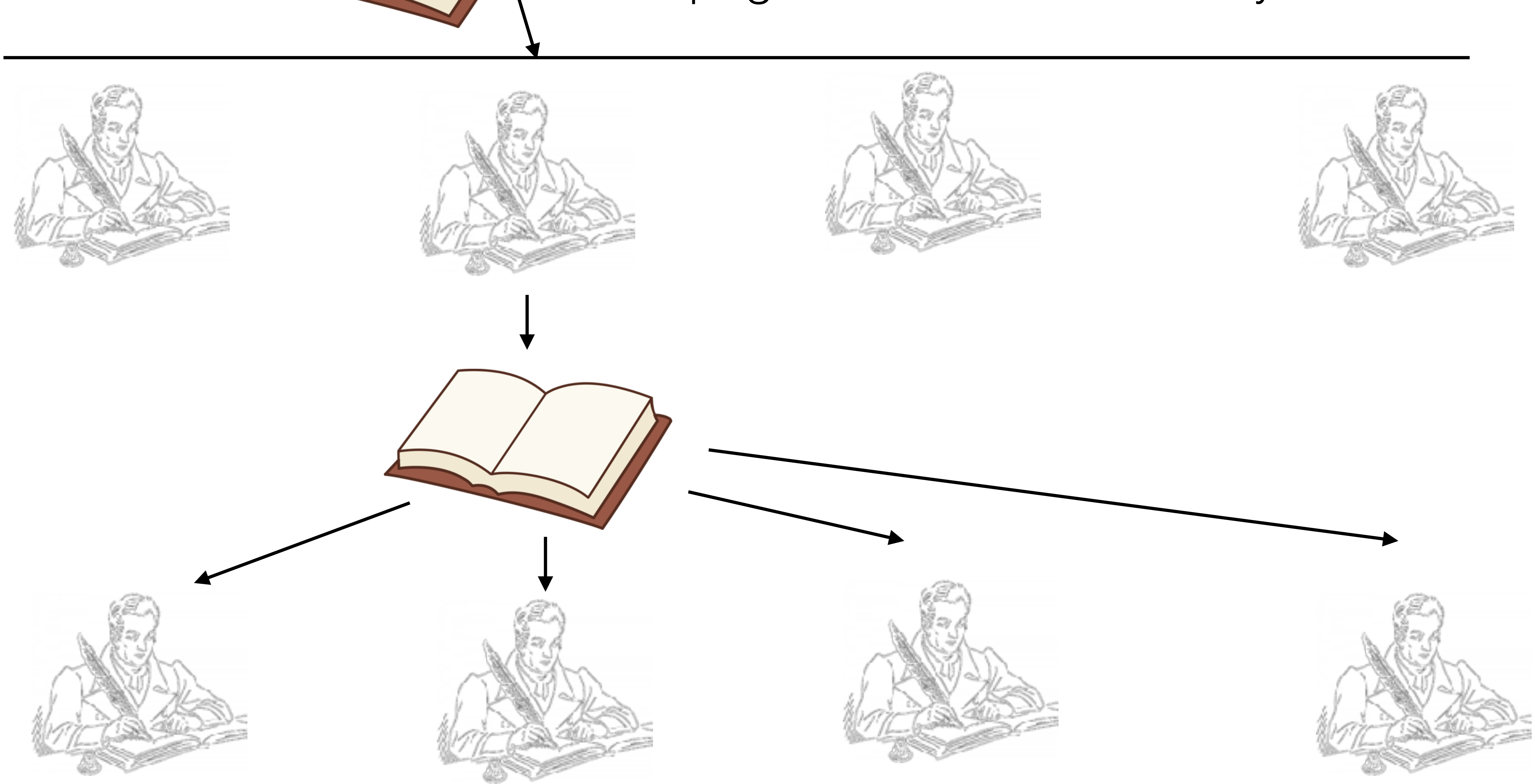
- each page refers only to the previous page
- current is assembled by stringing together the longest sequence of pages.



Rules of extending the book

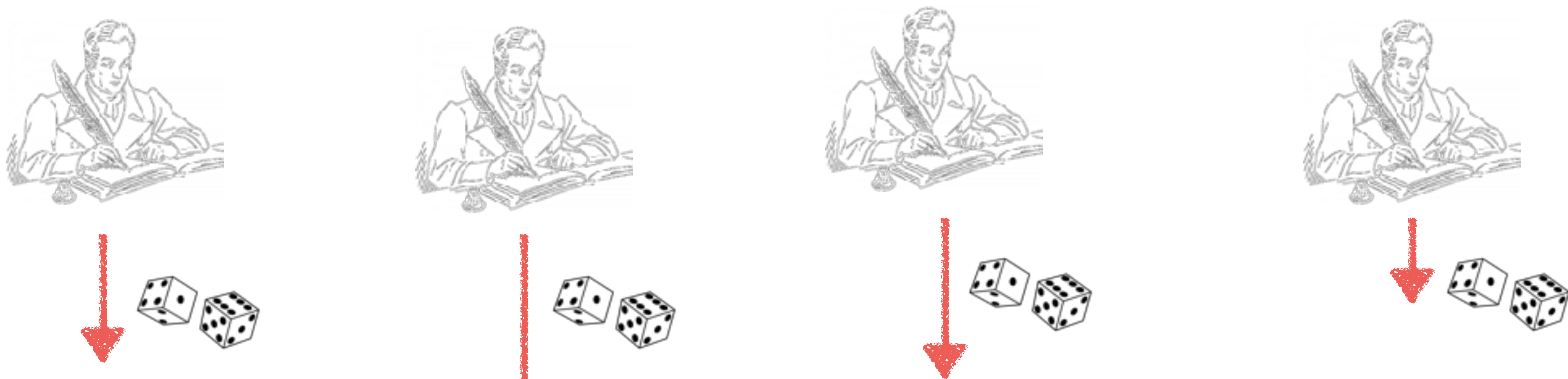


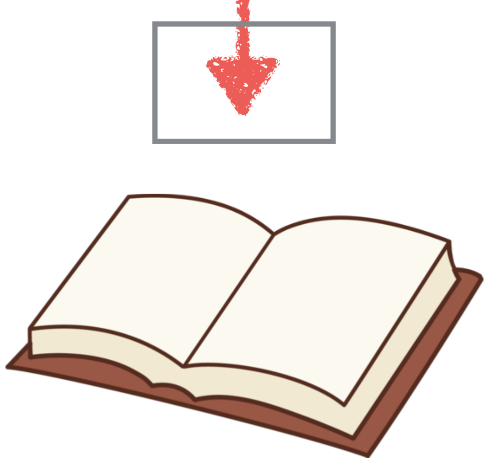
The first scribe that discovers
a page announces it to everyone else



Effort is needed to produce a page

equivalent to : each page needs a special combination from a set of dice to be rolled.

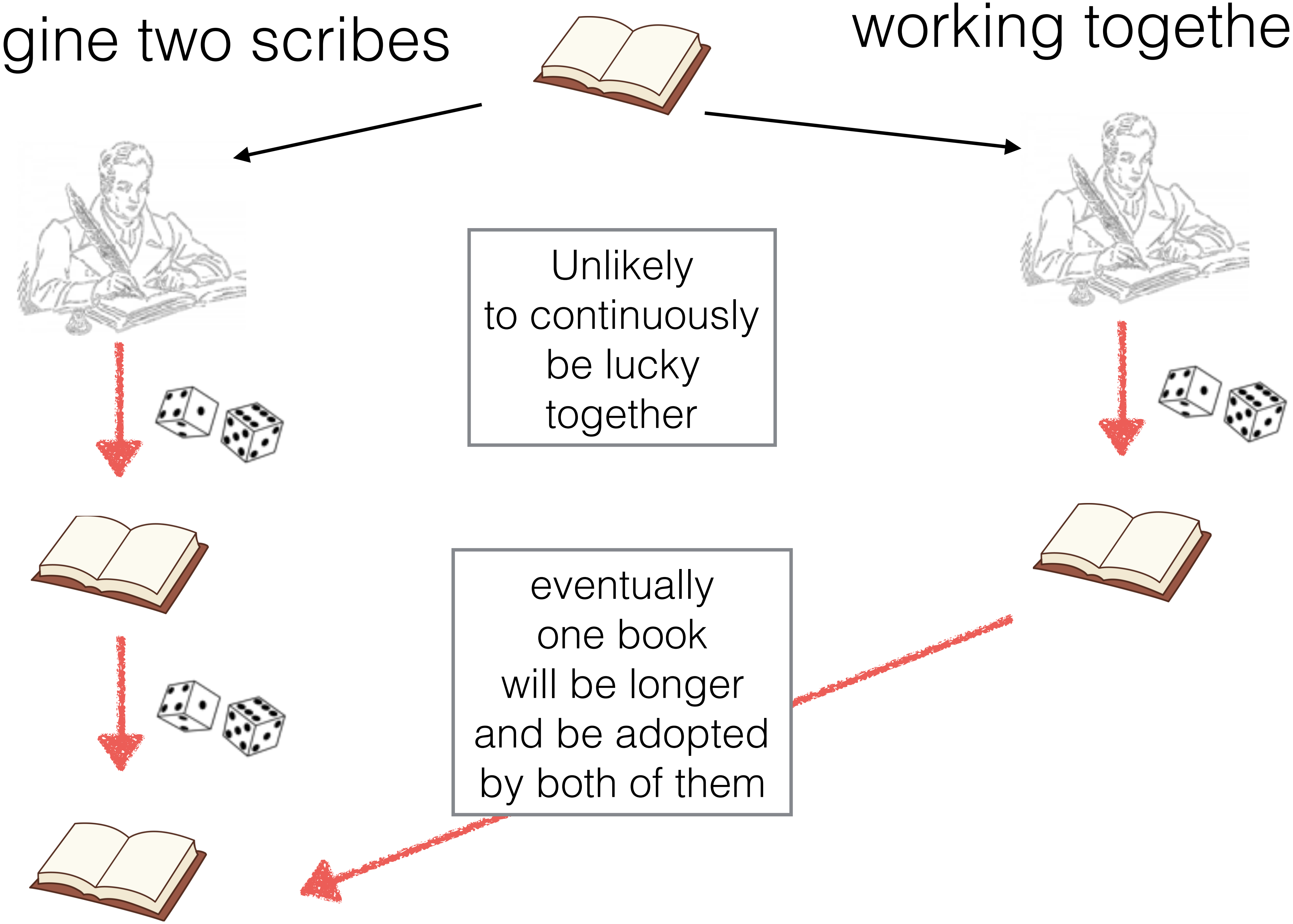




The probabilistic nature of the process is paramount to its security

The benefits of randomness

Imagine two scribes working together

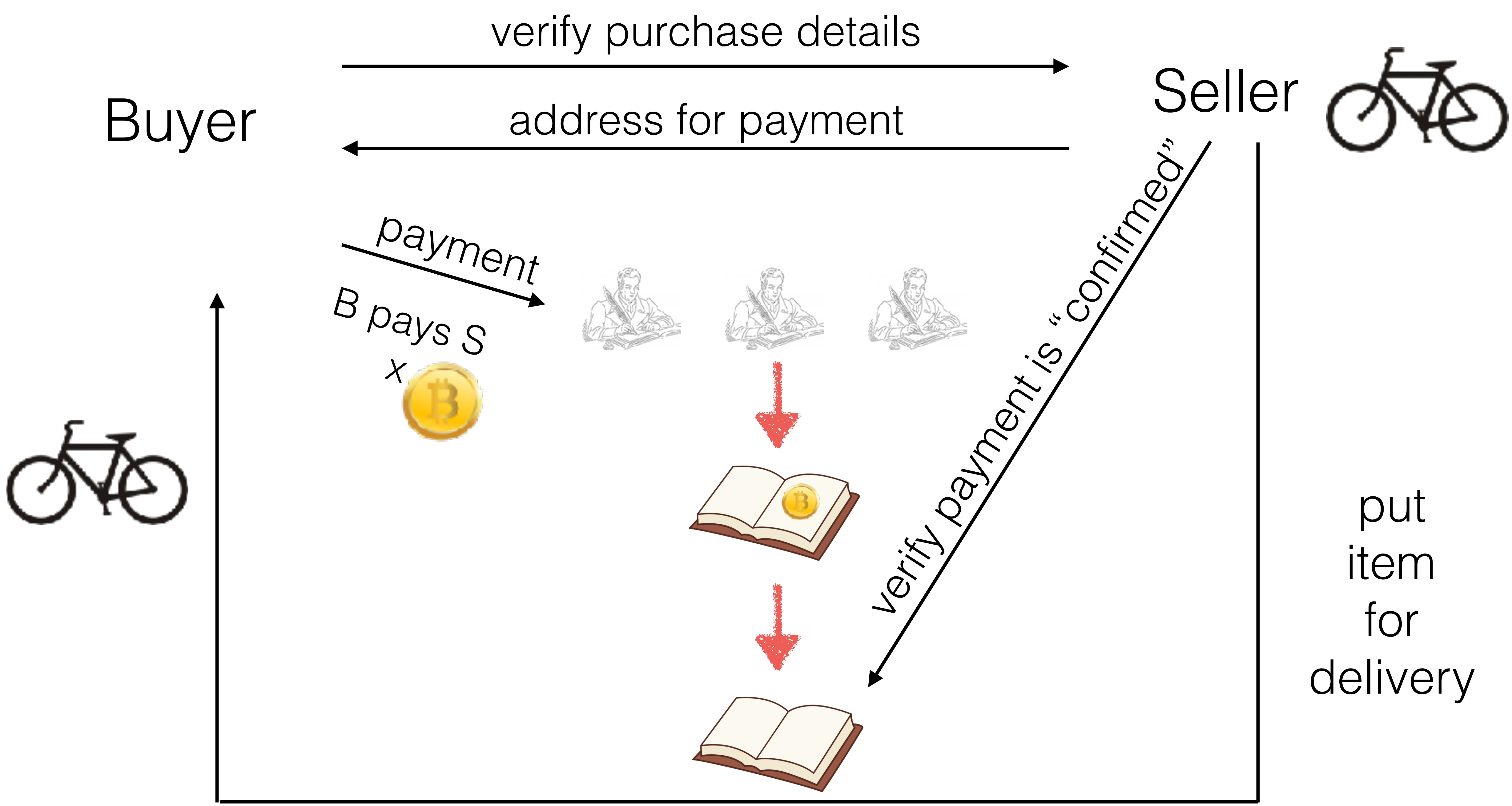


Symmetry Breaking

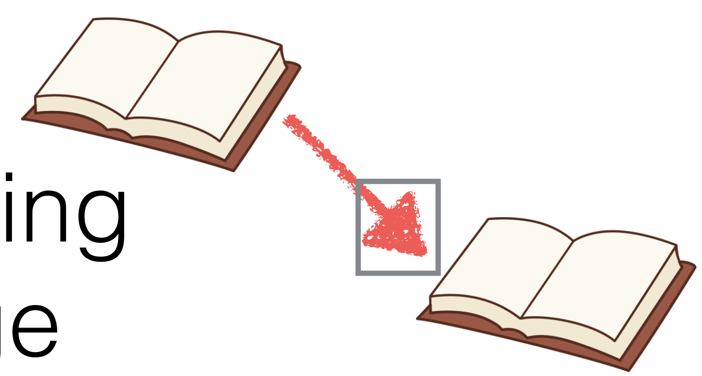
Being a scribe

- Anyone can be a scribe for the book.
- As long as one has a set of dice.
- The more dice one has, the higher the likelihood to produce the winning combination to make a page.

Using the book - Money 3.0



Parable & Reality

book		the “blockchain”
scribes		“Miners” / Computer systems that organize transactions in blocks
producing a page		Solving a **cryptographic puzzle** that is **moderately hard** to solve
rolling a set of dice		Using a computer to test for a solution from a large space of candidate solutions

Analysis of Money 3.0

- a medium of exchange
- a unit of account
- a store of value

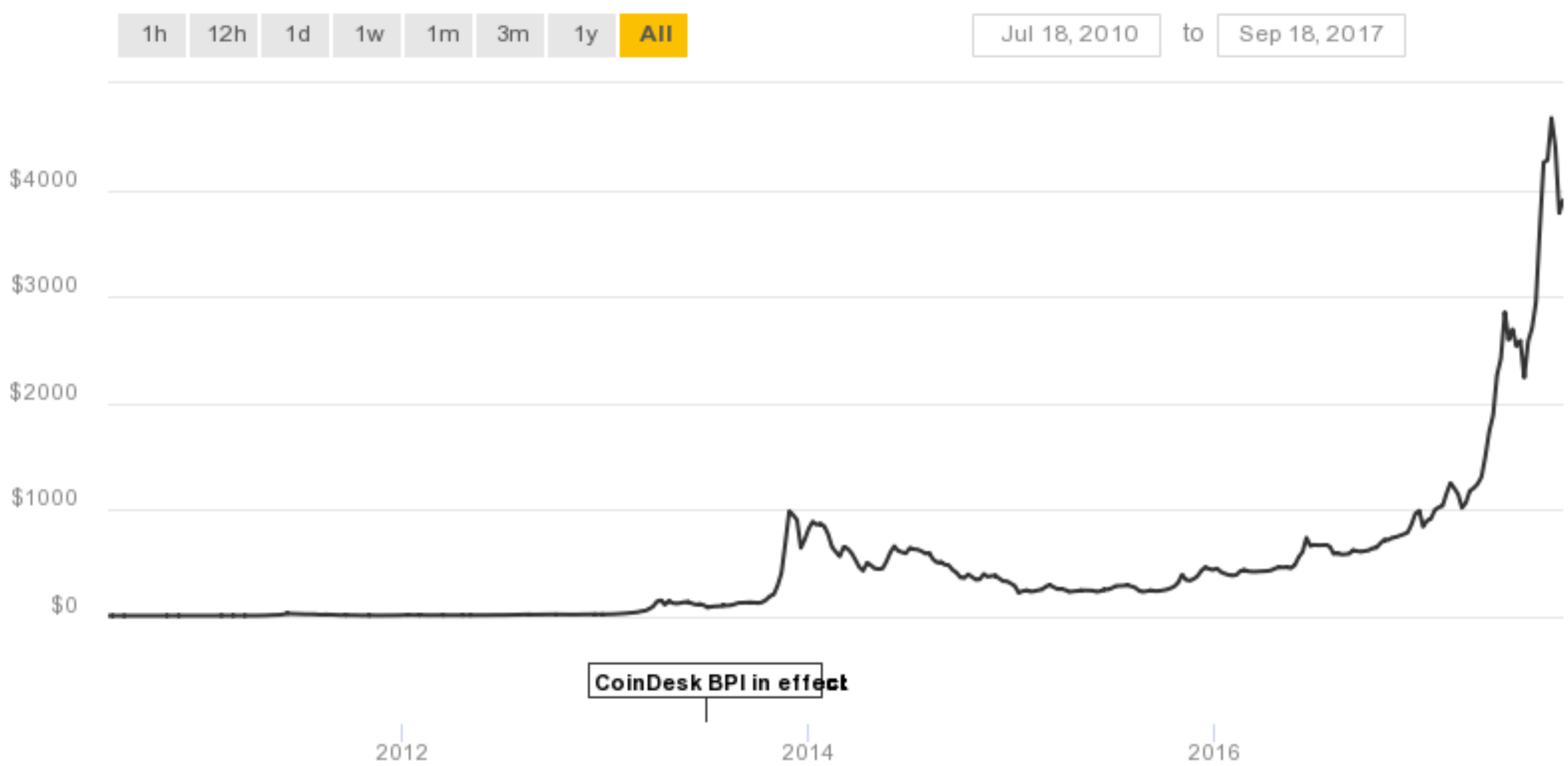
improving
[assuming internet connectivity / adoption]

great!
fungible & divisible.

good
[no trusted parties -
no natural deterioration]

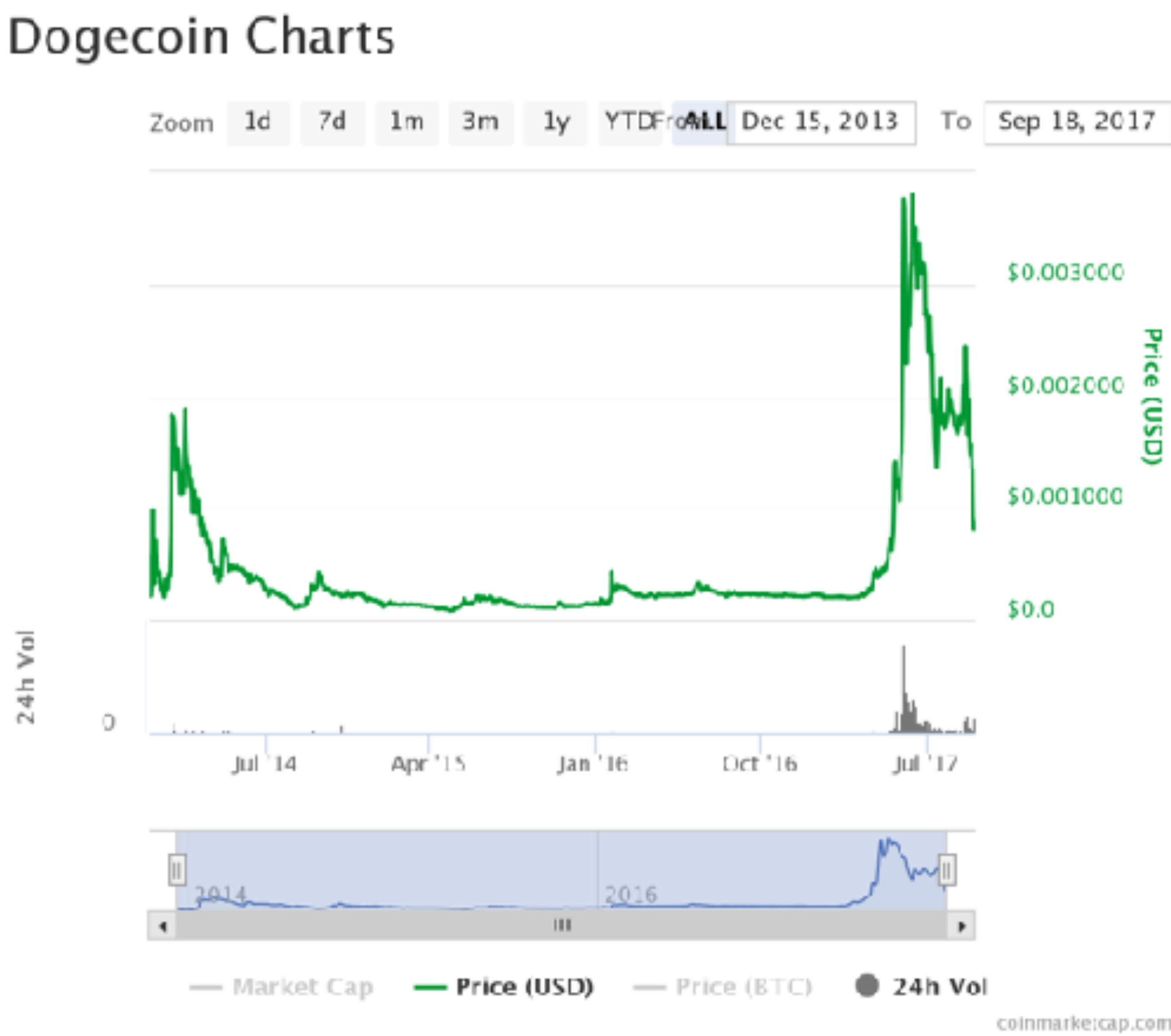
Word of Caution

- Just because something **can** be good as a store of value, it does **not** mean that it **will be** a good store of value in a real world deployment.



Word of Caution

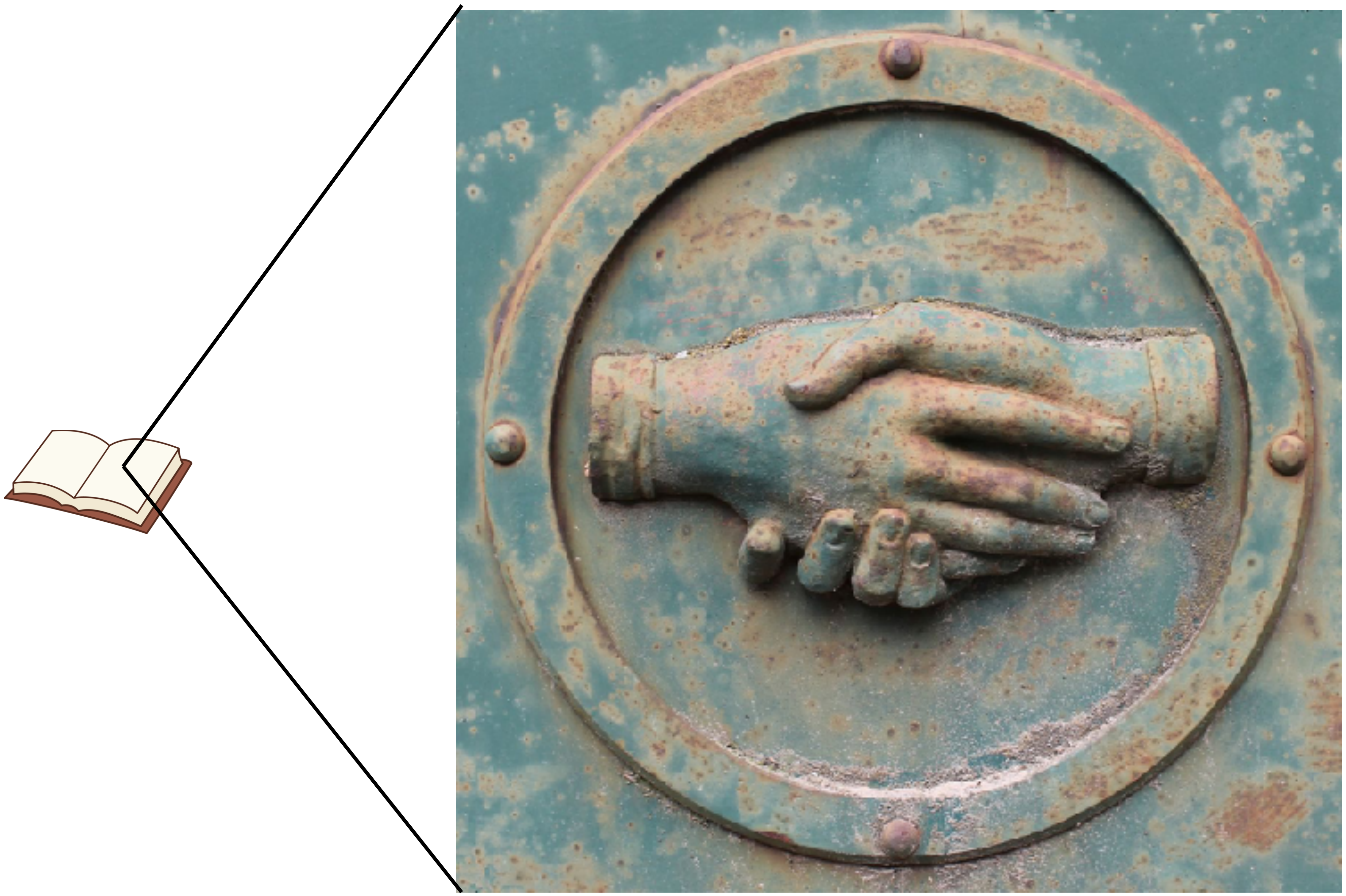
- Just because something **can** be good as a store of value, it does **not** mean that it **will be** a good store of value in a real world deployment.



From Money to Smart Contracts

- Since we have created **the book**, why stop at recording monetary transactions?
- We can encode in the book's pages **arbitrary relations** between persons.
- Furthermore, scribes, can perform tasks such as verifying that stakeholders **comply** to contractual obligations … and **take action** if they do not.

Smart Contract



Smart Contract Operation

- A smart contract is a **piece of code** written in a formal language that records all terms for a certain engagement between a set of persons, “stakeholders.”
- Stakeholders are identified by their **accounts**.
- The smart contract has a **public state**.
- The smart contract **self executes** each time a certain trigger condition is fulfilled.

Questions to Consider

- How are pages created? since the book is empty at the beginning, where do the money come from?

  proofs of work

- How is it possible to sign something digitally?

  digital signatures

- How does a page properly refer to the previous page?

  hash functions

Hash Functions

Hash Functions

- An algorithm that produces a fingerprint of a file.

$$\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^\lambda$$

- what are the required properties (traditionally):
  Efficiency.
  A good spread for various input distributions.

- What are Security/Cryptographic considerations?

Collisions

- Collision Attack: Find $x, y : \mathcal{H}(x) = \mathcal{H}(y)$
- Second pre-image Attack: Find $y : \mathcal{H}(x) = \mathcal{H}(y)$

  For given x

Birthday Paradox

- How many people should be in a room so that the probability that two of them share a birthday becomes larger than 50% ?

Paradox Explained

n possible dates $\qquad$ k people

$$\Pr[\neg Col] =$$

$$\frac{n}{n}\frac{n-1}{n}\frac{n-2}{n}\cdots\frac{n-k+1}{n} = \prod_{\ell=1}^{k}(1-\frac{\ell}{n})$$

$$\leq \exp(-\frac{1}{n}\sum_{\ell=1}^{k}\ell) = \exp(-k(k+1)/2n)$$

$$\Pr[Col] = \frac{1}{2} \Rightarrow k \approx 1.177\sqrt{n}$$

Finding Collisions via the BP

- Randomly sample pairs of the form $\langle x, \mathcal{H}(x) \rangle$
- Store in table and sort according to 2nd coordinate.
- Perform linear pass to see whether there are elements with an equal 2nd coordinate

Analysis

- For *k* elements.
- Running time is O(*k* log *k*)
- Choose *k* as $1.177\sqrt{2^{\lambda}}$

Pre-Image Attack

- Given $\mathcal{H}(m)$ $m \in \{0,1\}^t$

  Find an element of $\mathcal{H}^{-1}(\mathcal{H}(m))$

- Generic algorithms tries all possible values. How many?

  $O(2^t)$

Constructing Hash Functions

- Relates to the notion of one-way functions.

$$f : X \to Y$$

easy : given x find $f(x)$

hard : given $f(x)$ sample $f^{-1}(f(x))$

Word of Caution

Existence of one-way functions implies that :

$$P \neq NP$$

The single most important open question in Computer Science today

Hash Function Implementations

- **Retired.** MD5, SHA1.
- **Current.** SHA2, SHA3, available for 224,256,384,512 bits fingerprints.
- **Bitcoin.** Uses SHA2 with 256 bits output, SHA-256.

Digital Signatures

What is a Signature?

- Can be produced by one specified entity.
- Can be verified by anyone (that is suitably “equipped" and “initialized”).
- Cannot be forged on a new message even if multiple signatures have been transmitted.

Pen and Paper Signatures

What are the assumptions when you verify them?
When you produce them ?

Digital Signatures

Three algorithms (**KeyGen**, **Sign**, **Verify**)

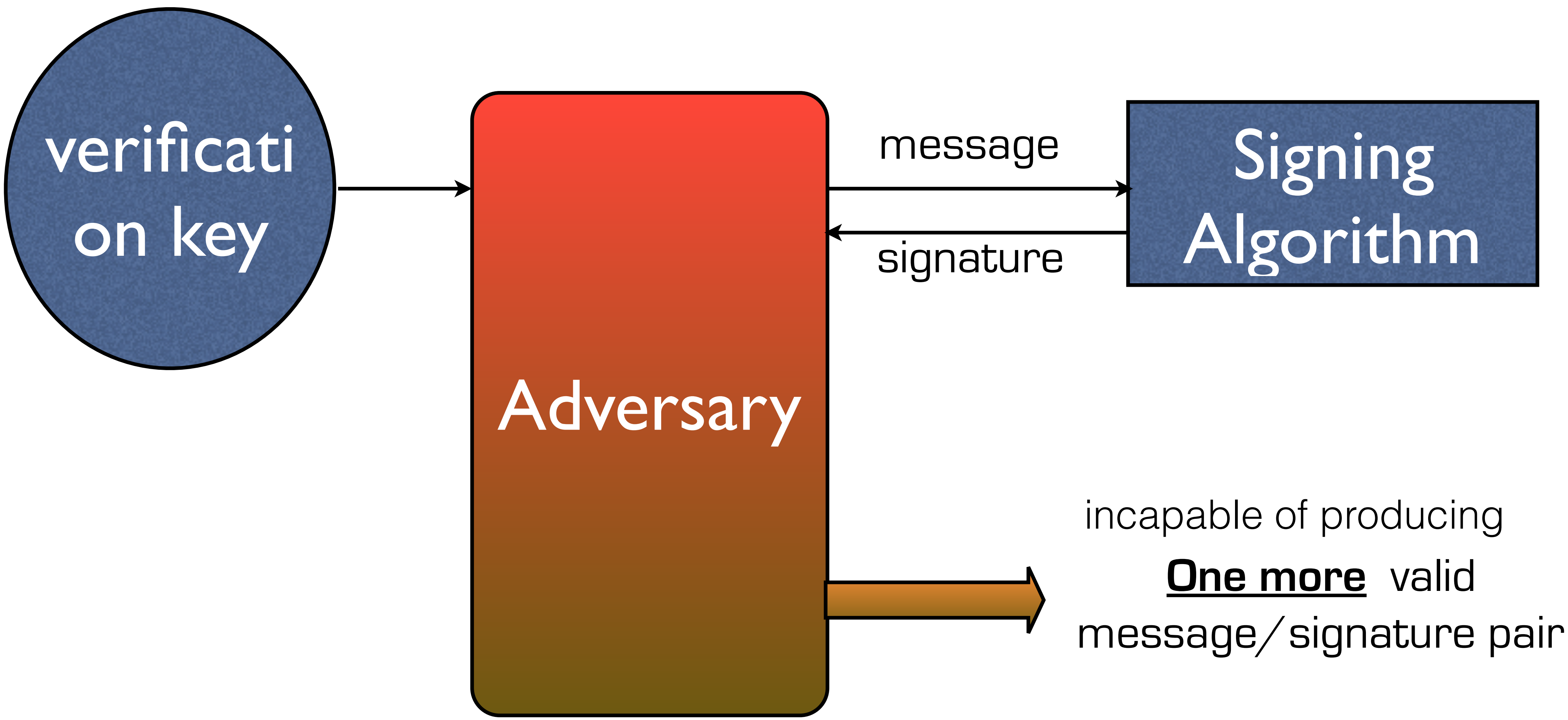
KeyGen : takes as input the *security parameter.* returns the signing-key and verification-key.

Sign : takes as input the *signing-key* and the *message* to be signed and returns a signature.

Verify : takes as input the *verification-key,* a *message* and a *signature* on the message and returns either True or False.

Digital Signature Security

Existential Unforgeability under a Chosen Message Attack (EU-CMA)



Constructing Digital Signatures

- Major challenge:
  - what prevents the adversary from learning how to *sign* messages by analyzing the *verification-key?*

Example Construction : using hash functions

One-Time Signatures

$$\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^\lambda$$

KeyGen

$$sk = \langle s_{i,b} \rangle_{i \in \{1,\ldots,\lambda\}, b \in \{0,1\}} \quad vk = \langle \mathcal{H}(s_{i,b}) \rangle_{i \in \{1,\ldots,\lambda\}, b \in \{0,1\}}$$

Sign

$$\mathcal{H}(m) = w = w_1 \ldots w_\lambda$$

$$\sigma = \langle s_{1,w_1}, \ldots, s_{\lambda,w_\lambda} \rangle$$

Verify Hash strings from σ and compare to the hashes stored in the vk at the positions dictated by $\mathcal{H}(m)$

Digital Signature Constructions

- Based on the RSA (Rivest Shamir Adleman), one way trapdoor function (with hardness that relates to the factoring problem).
  - The RSA algorithm
- Based on the discrete-logarithm problem.
  - the DSA algorithm
- **Bitcoin.** Uses ECDSA, a DSA variant over elliptic curve groups.

bitcoin transactions

- Typical transaction
  - input: contains a signature and public-key
  - output: contains a verification procedure

General Format

general format of a Bitcoin transaction (inside a block)

Field	Description	Size
Version no	currently 1	4 bytes
In-counter	positive integer VI = VarInt	1 - 9 bytes
list of inputs	the first input of the first transaction is also called "coinbase" (its content was ignored in earlier versions)	<in-counter>-many inputs
Out-counter	positive integer VI = VarInt	1 - 9 bytes
list of outputs	the outputs of the first transaction spend the mined bitcoins for the block	<out-counter>-many outputs
lock_time	if non-zero and sequence numbers are < 0xFFFFFFFF: block height or timestamp when transaction is final	4 bytes

Input and Output Scripts

```
Input:
Previous tx: f5d8ee39a430901c91a5917b9f2dc19d6d1a0e9cea205b009ca73dd04470b9a6
Index: 0
scriptSig:
304502206e21798a42fae0e854281abd38bacd1aeed3ee3738d9e1446618c4571d10
90db022100e2ac980643b0b82c0e88ffdfec6b64e3e6ba35e7ba5fdd7d5d6cc8d25c6b241501

Output:
Value: 5000000000
scriptPubKey: OP_DUP OP_HASH160 404371705fa9bd789a2fcd52d2c580b65d35549d
OP_EQUALVERIFY OP_CHECKSIG
```

The input imports 50 BTC from output #0 of tx f5d..
and sends them to a Bitcoin address 404…

Verifying a transaction

```
scriptPubKey: OP_DUP OP_HASH160 <pubKeyHash>
 OP_EQUALVERIFY OP_CHECKSIG

scriptSig: <sig> <pubKey>
```

Transaction Processing

- Uses a **stack** data structure:
  - items can be pushed to the stack and popped from the stack.
  - following LIFO (“last in first out”) order.

Transaction processing

Stack	Script	Description
Empty.	<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	scriptSig and scriptPubKey are combined.
<sig> <pubKey>	OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	Constants are added to the stack.
<sig> <pubKey> <pubKey>	OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	Top stack item is duplicated.
<sig> <pubKey> <pubHashA>	<pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	Top stack item is hashed.
<sig> <pubKey> <pubHashA> <pubKeyHash>	OP_EQUALVERIFY OP_CHECKSIG	Constant added.
<sig> <pubKey>	OP_CHECKSIG	Equality is checked between the top two stack items.
true	Empty.	Signature is checked for top two stack items.

https://en.bitcoin.it/wiki/Transaction

More general transactions

- Transactions can include arbitrary script operations.

e.g., pay to script hash

```
scriptPubKey: OP_HASH160 <scriptHash> OP_EQUAL
scriptSig: ..signatures... <serialized script>
```

(instead of):

```
scriptPubKey: OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG
scriptSig: <sig> <pubKey>
```

script operations

- Some operations that are allowed:
  - basic 32-bit arithmetic (addition subtraction)
  - computation of SHA256[.]
  - verify equality
  - calculate length of string
  - Verify signature (ECDSA)

End of Lecture 01

- Next time :
    - The consensus layer.
    - Basic Properties.
    - Proof of work.