Blockchain technologies: a primer

Giuseppe Bianchi
University of Roma Tor Vergata / CNIT

Early warning #1

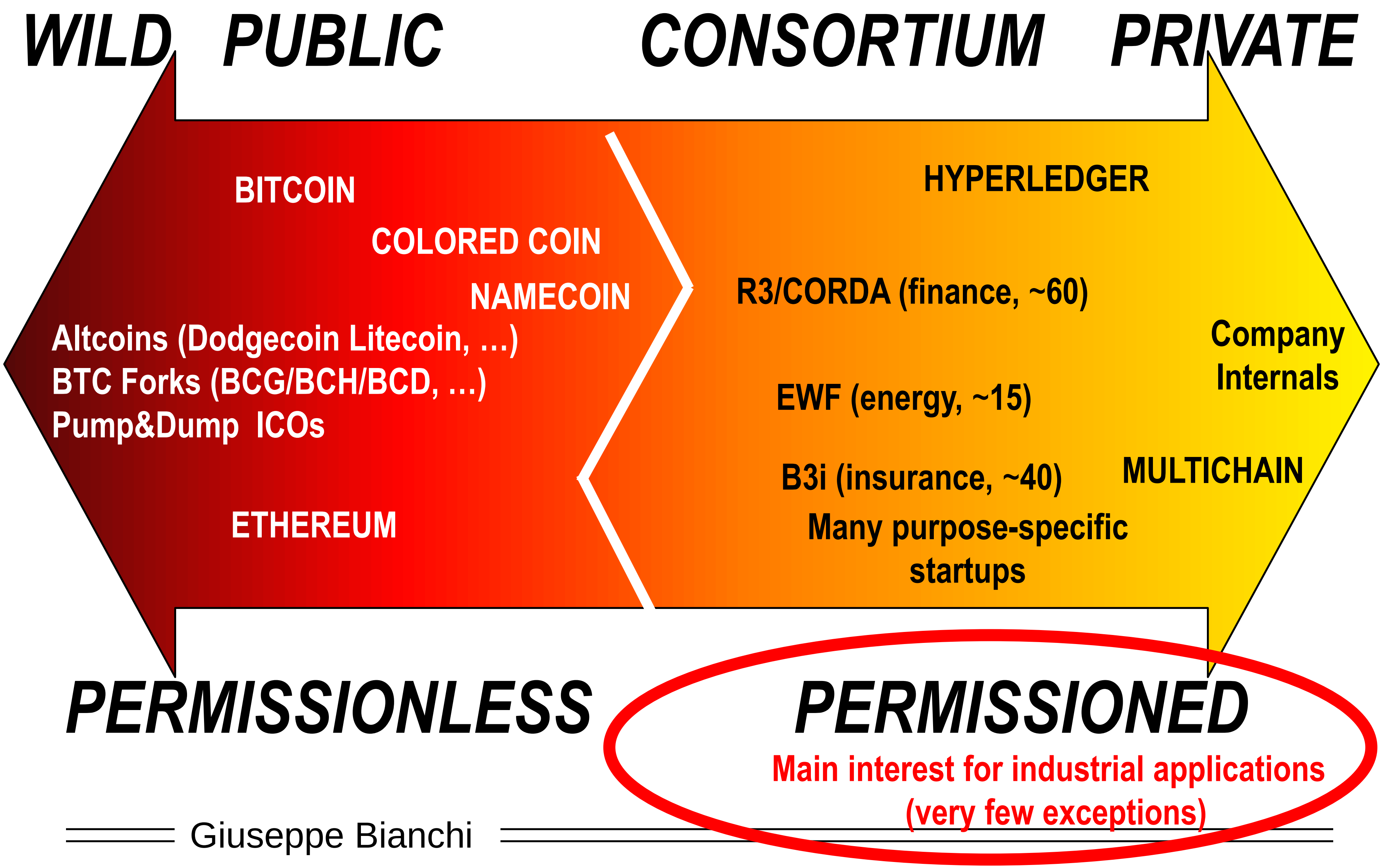
do you *really* need blockchains?

a.k.a. the blockchain… overhype ☺

*If your requirements are fulfilled by today's relational **databases**, you'd be **insane** to use a blockchain (*)*

(*) quote from Gideon Greenspan, Multichain founder

Early warning #2
blockchain ≠ bitcoin



WILD PUBLIC CONSORTIUM PRIVATE

BITCOIN

COLORED COIN

NAMECOIN

Altcoins (Dodgecoin Litecoin, …)
BTC Forks (BCG/BCH/BCD, …)
Pump&Dump ICOs

ETHEREUM

HYPERLEDGER

R3/CORDA (finance, ~60)

Company Internals

EWF (energy, ~15)

B3i (insurance, ~40)

MULTICHAIN

Many purpose-specific startups

PERMISSIONLESS

PERMISSIONED

Main interest for industrial applications
(very few exceptions)

Today: 3+1 goals

1. **Do you really need blockchains?**

2. **Which blockchain «type»?**

3. **Which possible applications?**

4. **Simplified blockchain primer**

   ⇨No time for anything more meaningful

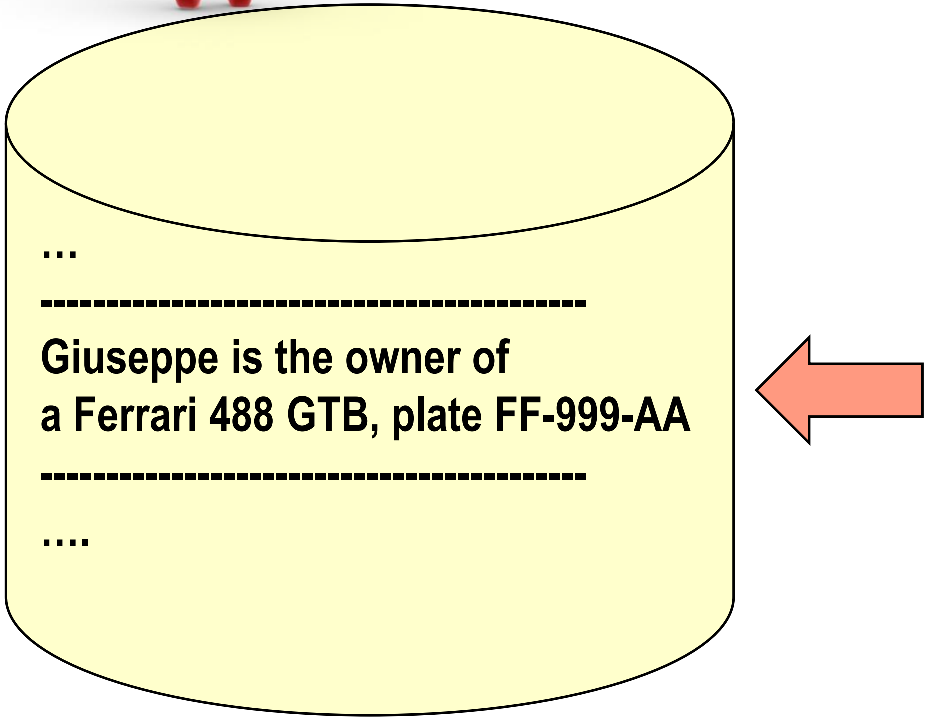
Intro: understanding blockchains

A layman/conceptual perspective

Blockchains in a nutshell: a tentative black-box definition

authoritative *log of*
validated *transactions*
without a ***trusted*** *intermediary*

With a trusted authority…
…a DB is all you need



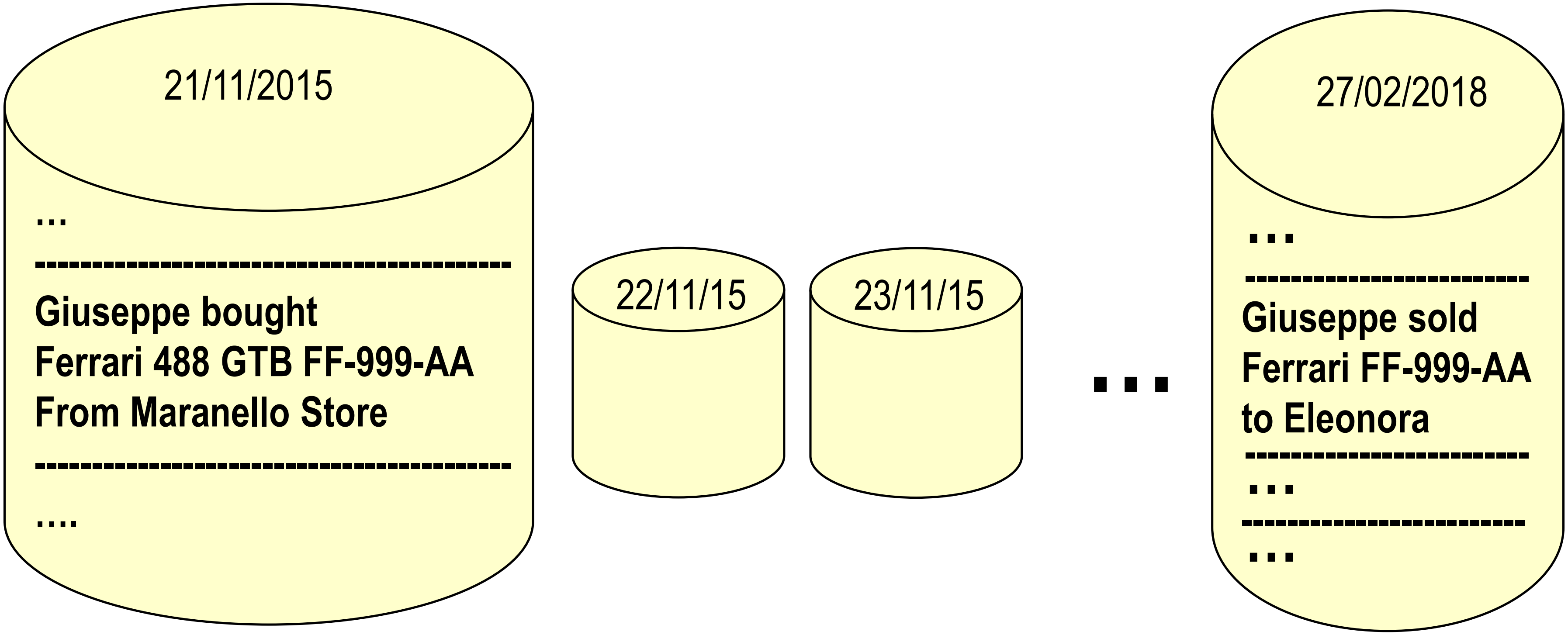
…

Giuseppe is the owner of
a Ferrari 488 GTB, plate FF-999-AA

….

*You can trust that
what you read here
is TRUE…*

A DB can be organized as a ledger (i.e. blocks logging transactions)



21/11/2015

…

Giuseppe bought
Ferrari 488 GTB FF-999-AA
From Maranello Store

….

22/11/15

23/11/15

…

27/02/2018

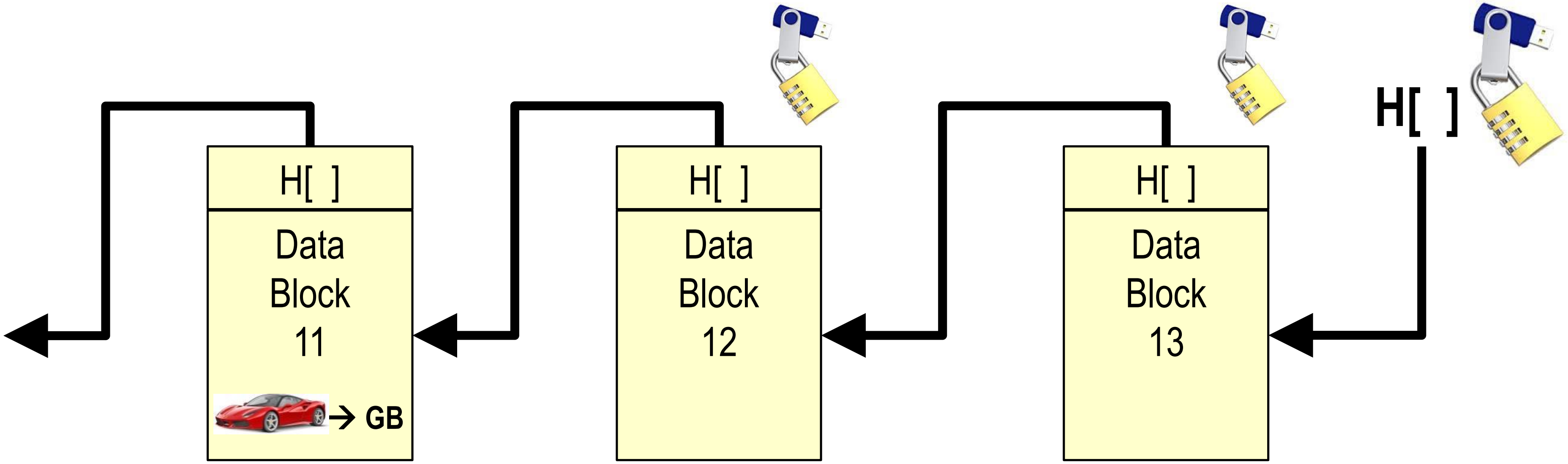
…

Giuseppe sold
Ferrari FF-999-AA
to Eleonora

…

…

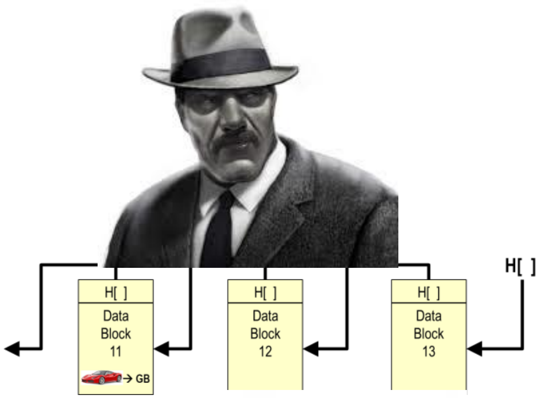
A ledger can be append-only & deployed over unsecure storage

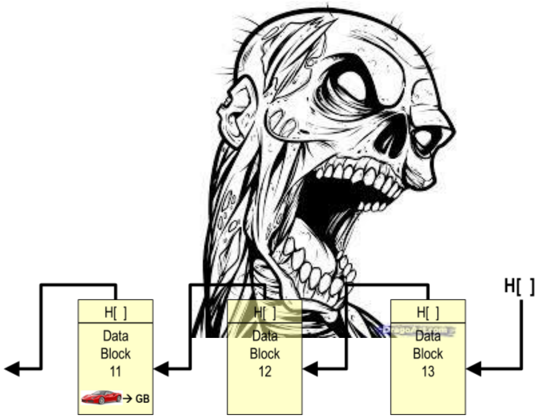


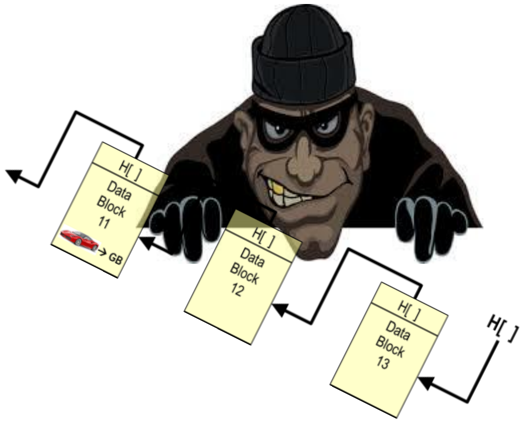
e.g. via Hash Pointer data structures (& Merkle Trees) → since the 70ies

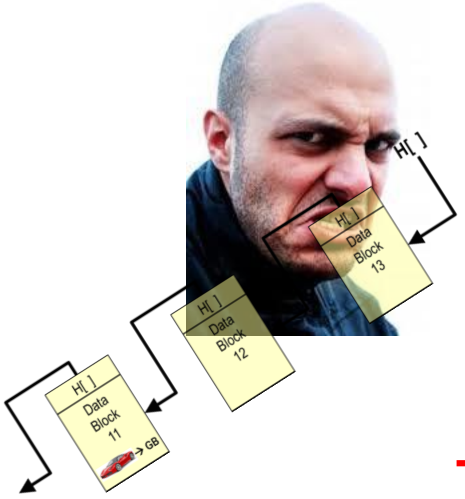
Data Integrity Guaranteed even on unsecure storage (more later)

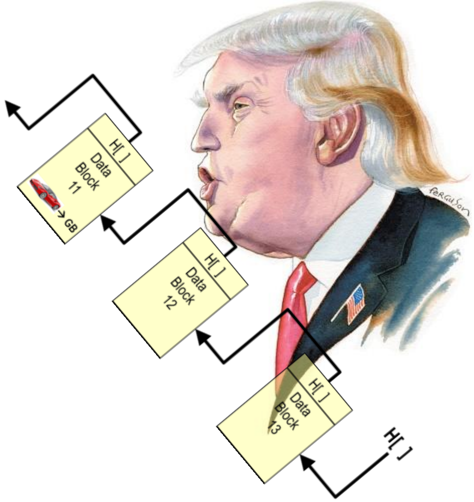
… and can be even replicated among multiple non-mutually-trusting parties











Consensus protocols:

reach shared agreement among a group of participants

→ since early 80ies (Lamport etc)

(more later)

So far, so good

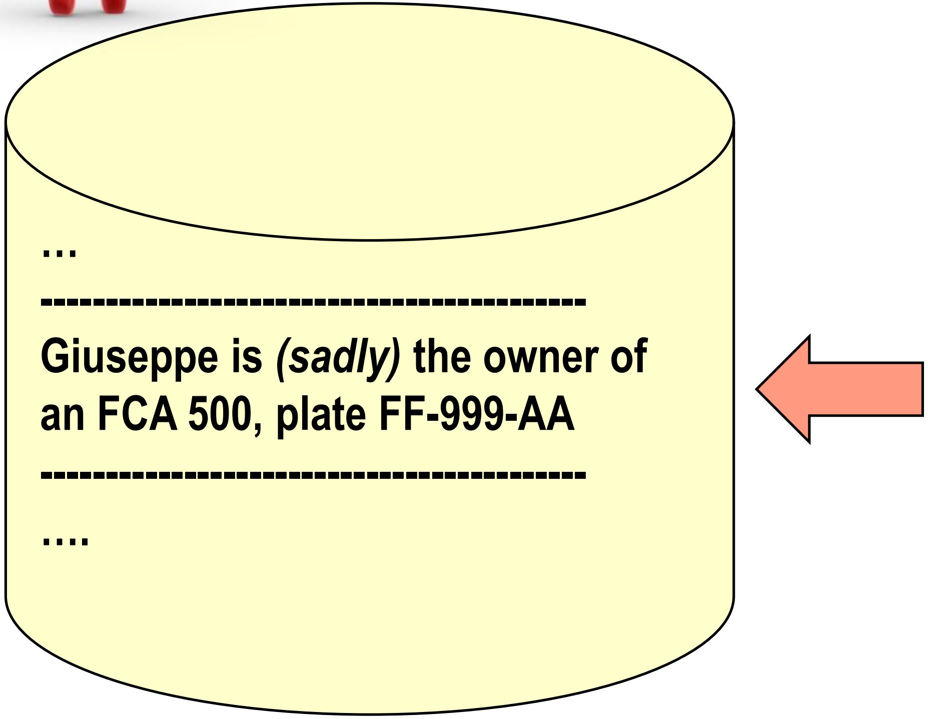
➔Besides an unfortunate small detail…

➔Giuseppe has NOT bought a Ferrari…

➔… but just a Fiat 500

We have eventually reached distributed secure storage and consensus on a FALSE statement!

Truthfulness: easy with a trusted authority!



...

Giuseppe is *(sadly)* the owner of an FCA 500, plate FF-999-AA

….

You trust that what you read here is TRUE…

… not ONLY because storage is secure… (this is just data integrity!)…

… but because the authority does not lie to you!

Truthfulness without a trusted authority: consensus only?

THE KEY QUESTION

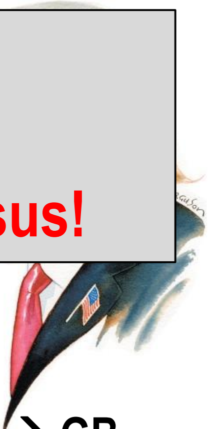
(foundational to properly understand blockchains!)

How a party NOT involved in the specific business can state something about the truthfulness of your logged transaction?

→ GB

Either all parties understand about car property…

… or there is MORE behind here, than «just» consensus!



→ GB

Majority =

→ GB

→ GB

Back to the start: our tentative black-box definition

THE property that makes a blockchain different from a DB

***authoritative** log of*

***validated** transactions*

*without a **trusted** intermediary*

Validation < Truth (remember Godel's theorem…)

but still a huge step beyond plain data-logging-only DBs!

→ **Block miners = (application-unaware?!) validators!** ←

Do YOU need blockchains? Checklist!

(the «AND» of what follows, NOT the «OR»! ☺)

- ➔ **Need a shared (append-only) database, with multiple writers which do NOT trust each other**
  - ⇨ *What I "see" about you is true*
  - ⇨ *What I «own» can be changed only by me*
- ➔ **We cannot rely on trusted intermediaries**
  - ⇨ *No authorities, banks, trusted mediators. …*
- ➔ **Transactions "interact" among them**
  - ⇨ *Order, dependencies, etc*
    - →*B pays C only after A pays B (and more interesting interactions!)*
- ➔ **Transactions must be validated**
  - ⇨ *E.g. cannot sell more than what I own, cannot double spend, etc*
  - ⇨ *No trusted intermediary can validate!*

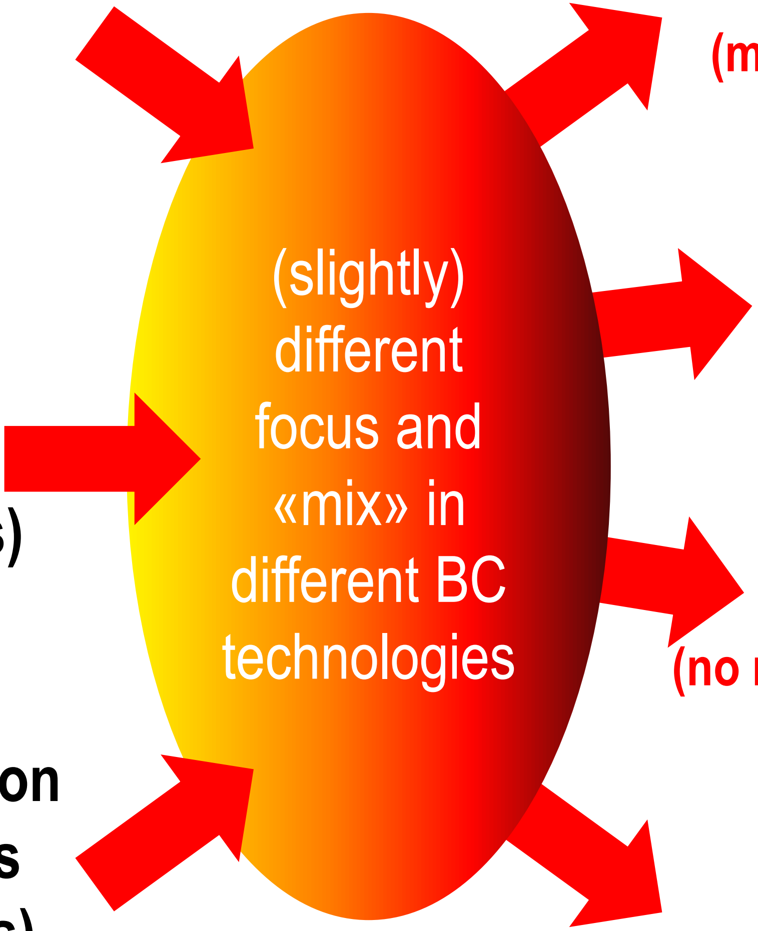
Taking stocks: blockchains in a slide

Technical asset

Append-only
secure storage
(hash-based ledger)

Trust without
single trusted party
(consensus protocols)

Transactions’ validation
and smart contracts
(scripting languages)



(slightly)
different
focus and
«mix» in
different BC
technologies

Outcome, impact

Transparency
(many societal implications)

Indelibility
(notary services)

Shareability across
boundaries of trust
(no need for single trust anchor)

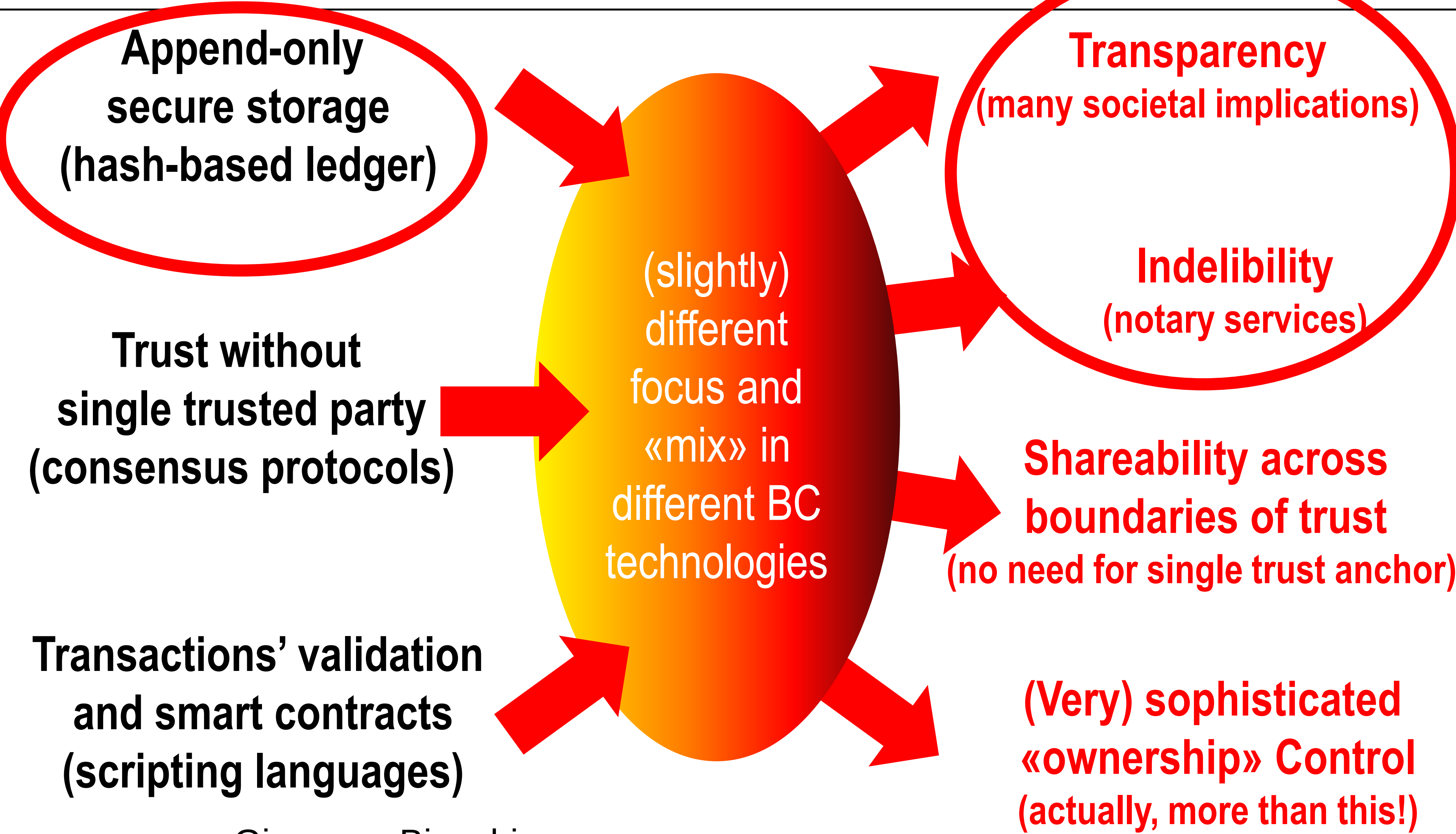
(Very) sophisticated
«ownership» Control
(actually, more than this!)

1° blockchain dimension: The ledger

The ledger

Technical asset

Outcome, impact



Append-only secure storage (hash-based ledger)

Trust without single trusted party (consensus protocols)

Transactions’ validation and smart contracts (scripting languages)

(slightly) different focus and «mix» in different BC technologies

Transparency (many societal implications)

Indelibility (notary services)

Shareability across boundaries of trust (no need for single trust anchor)

(Very) sophisticated «ownership» Control (actually, more than this!)

Background: cryptographic (one-way) hash functions

Hic ego: laudare igitur eloquentiam et quanta vis sit eius expromere quantamque eis, qui sint eam consecuti, dignitatem afferat, neque propositum nobis est hoc loco neque necessariu**m**. hoc vero sine ulla dubitatione confirmaverim, sive illa arte pariatur aliqua sive exercitatione quadam sive natura, rem unam esse omnium difficillumam. quibus enim ex quinque rebus constare dicitur, earum una quaeque est ars ipsa magna per sese. quare quinque artium concursus maxumarum quantam vim quantamque difficultatem habeat existimari potest.

H[ ]

c6c8258947bffe06ea4a0c8132af337a3c74ec
81d754a96d5a29e3ca7d8ce49d

Hic ego: laudare igitur eloquentiam et quanta vis sit eius expromere quantamque eis, qui sint eam consecuti, dignitatem afferat, neque propositum nobis est hoc loco neque necessariu**s**. hoc vero sine ulla dubitatione confirmaverim, sive illa arte pariatur aliqua sive exercitatione quadam sive natura, rem unam esse omnium difficillumam. quibus enim ex quinque rebus constare dicitur, earum una quaeque est ars ipsa magna per sese. quare quinque artium concursus maxumarum quantam vim quantamque difficultatem habeat existimari potest.

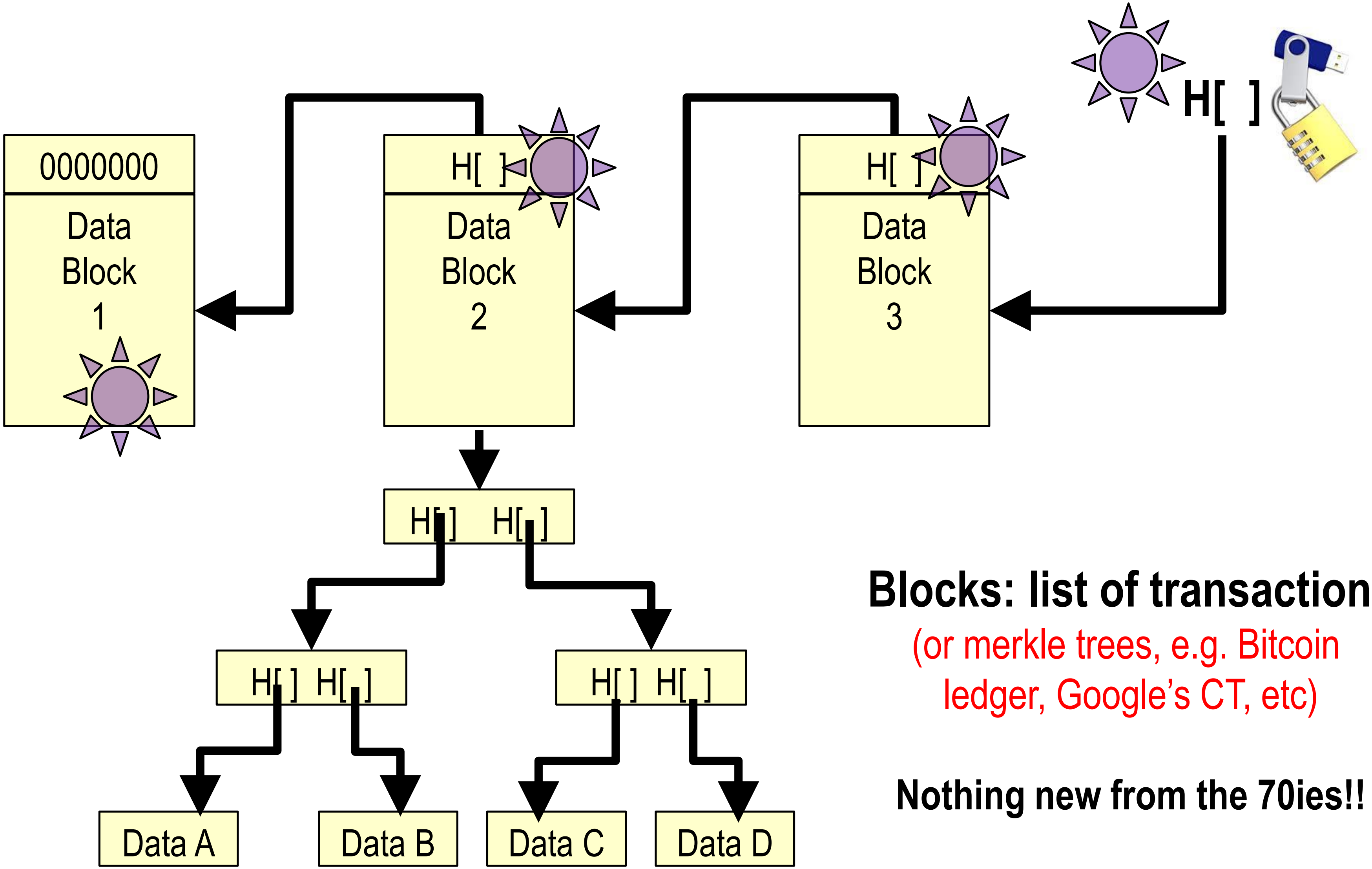
H[ ]

3238ead7fb611463703c47adc4215aa245a1f
1a4a0cea4c11296b466a76bbac4

Fixed size digest
(e.g. SHA-256: 64 hex)

No way for an attacker to purposedly modify/extend/replace initial text so as to obtain original digest!!

Hash pointers: append-only secure log over unsecure support!



Blocks: list of transactions

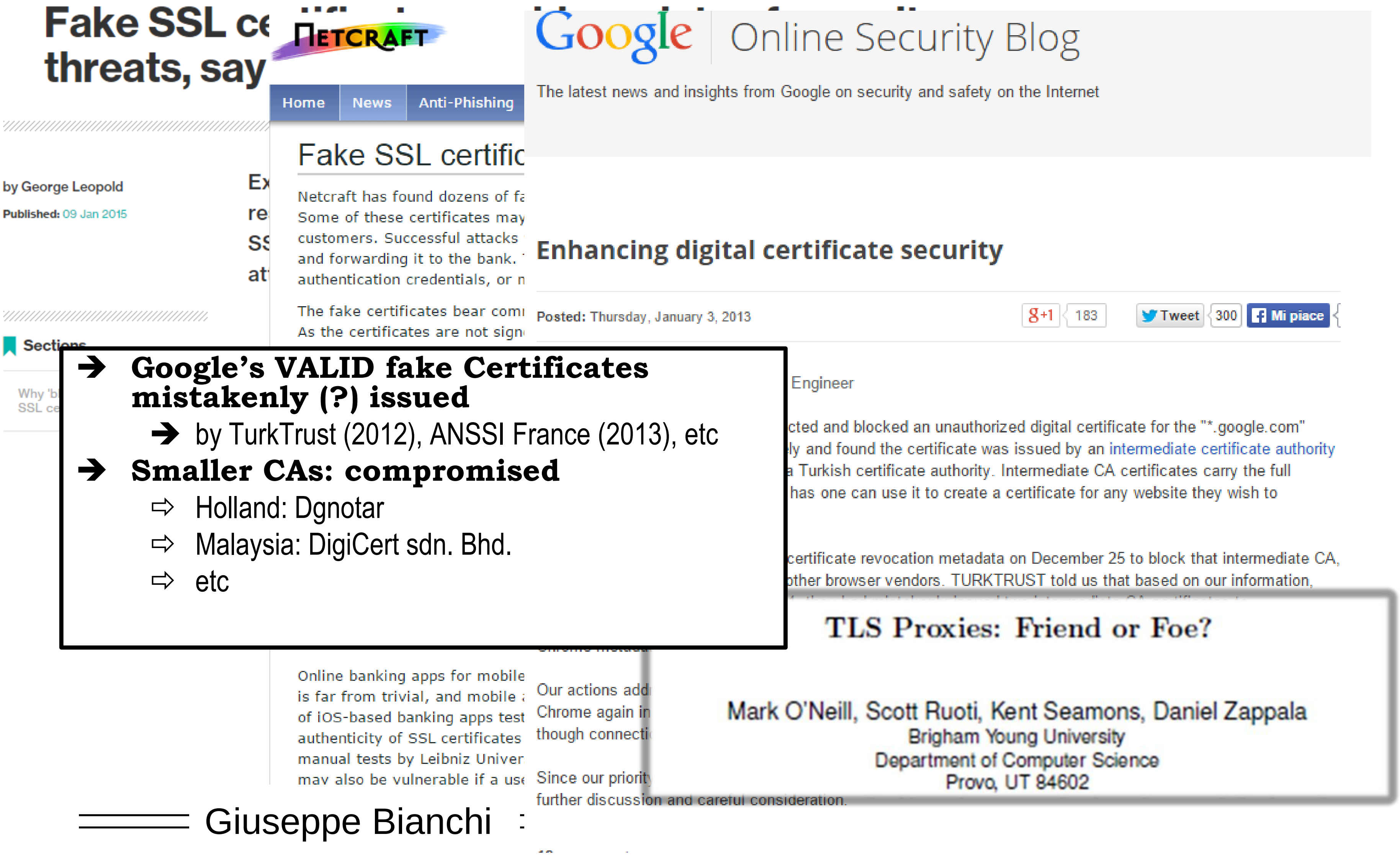
(or merkle trees, e.g. Bitcoin ledger, Google’s CT, etc)

Nothing new from the 70ies!!

Technical Interlude 1: Google's certificate transparency as a «quasi»-blockchain

A real world example of a standard (though cleverly organized) DB which most would today call «blockchain», but which is NOT.

Fact: trusted CA assumption at stake



Fake SSL ce
threats, say

by George Leopold

Published: 09 Jan 2015

Ex
re
SS
at

Sections

Why 'b
SSL ce

NETCRAFT

Home News Anti-Phishing

Fake SSL certific

Netcraft has found dozens of fa
Some of these certificates may
customers. Successful attacks
and forwarding it to the bank.
authentication credentials, or n

The fake certificates bear comm
As the certificates are not sign

Online banking apps for mobile
is far from trivial, and mobile
of iOS-based banking apps test
authenticity of SSL certificates
manual tests by Leibniz Univer
may also be vulnerable if a use

Google Online Security Blog

The latest news and insights from Google on security and safety on the Internet

Enhancing digital certificate security

Posted: Thursday, January 3, 2013

g+1 183 Tweet 300 Mi piace

Engineer

cted and blocked an unauthorized digital certificate for the "*.google.com"
ly and found the certificate was issued by an intermediate certificate authority
a Turkish certificate authority. Intermediate CA certificates carry the full
has one can use it to create a certificate for any website they wish to

certificate revocation metadata on December 25 to block that intermediate CA,
other browser vendors. TURKTRUST told us that based on our information,

Our actions add
Chrome again in
though connecti

Since our priority
further discussion and careful consideration.

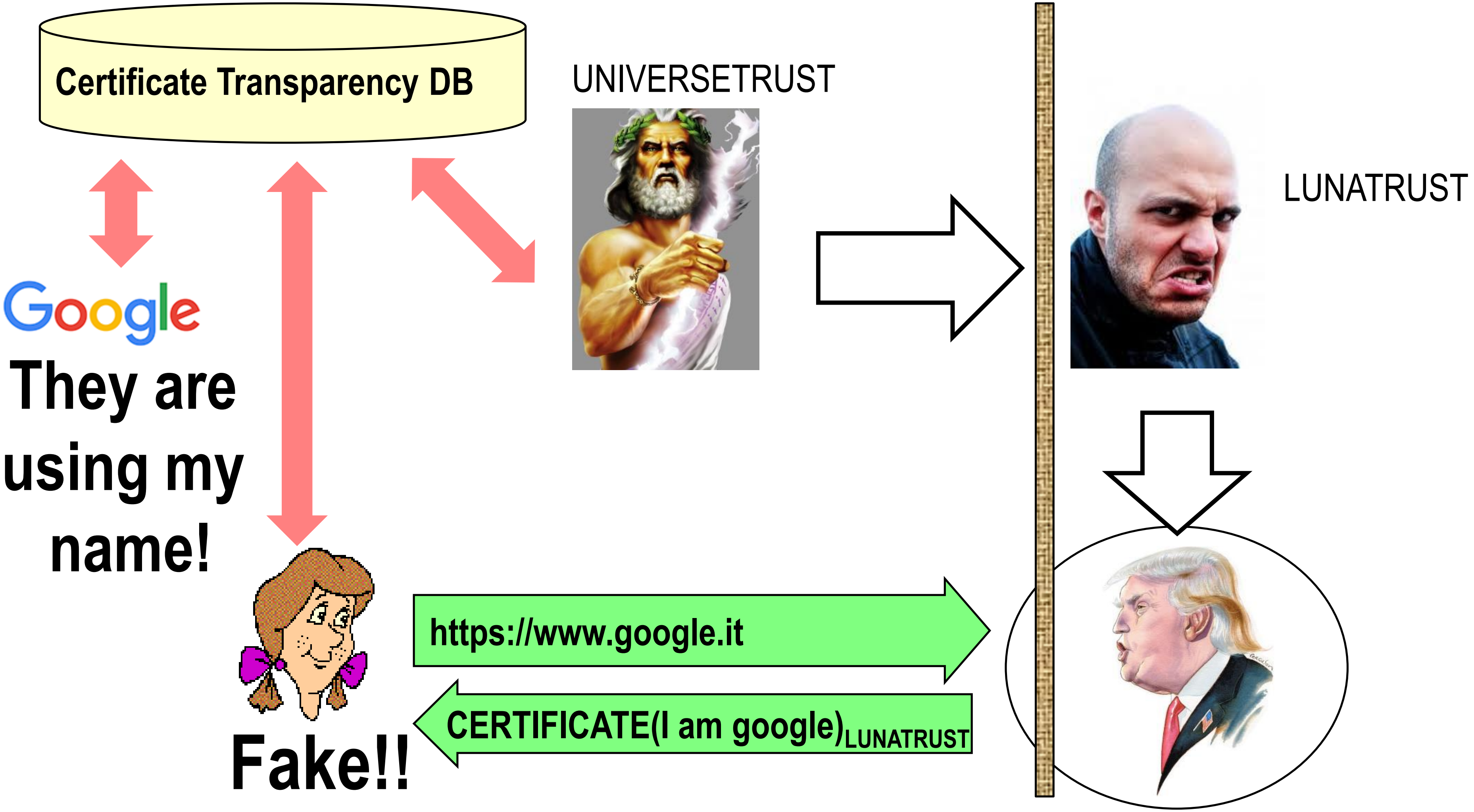
- ➔ **Google's VALID fake Certificates mistakenly (?) issued**
  - ➔ by TurkTrust (2012), ANSSI France (2013), etc
- ➔ **Smaller CAs: compromised**
  - ⇨ Holland: Dgnotar
  - ⇨ Malaysia: DigiCert sdn. Bhd.
  - ⇨ etc

TLS Proxies: Friend or Foe?

Mark O'Neill, Scott Ruoti, Kent Seamons, Daniel Zappala
Brigham Young University
Department of Computer Science
Provo, UT 84602

How to cope with malicious CAs?

Idea: gigantic worldwide DB which anyone can check!



Certificate Transparency DB

UNIVERSETRUST

LUNATRUST

Google

They are using my name!

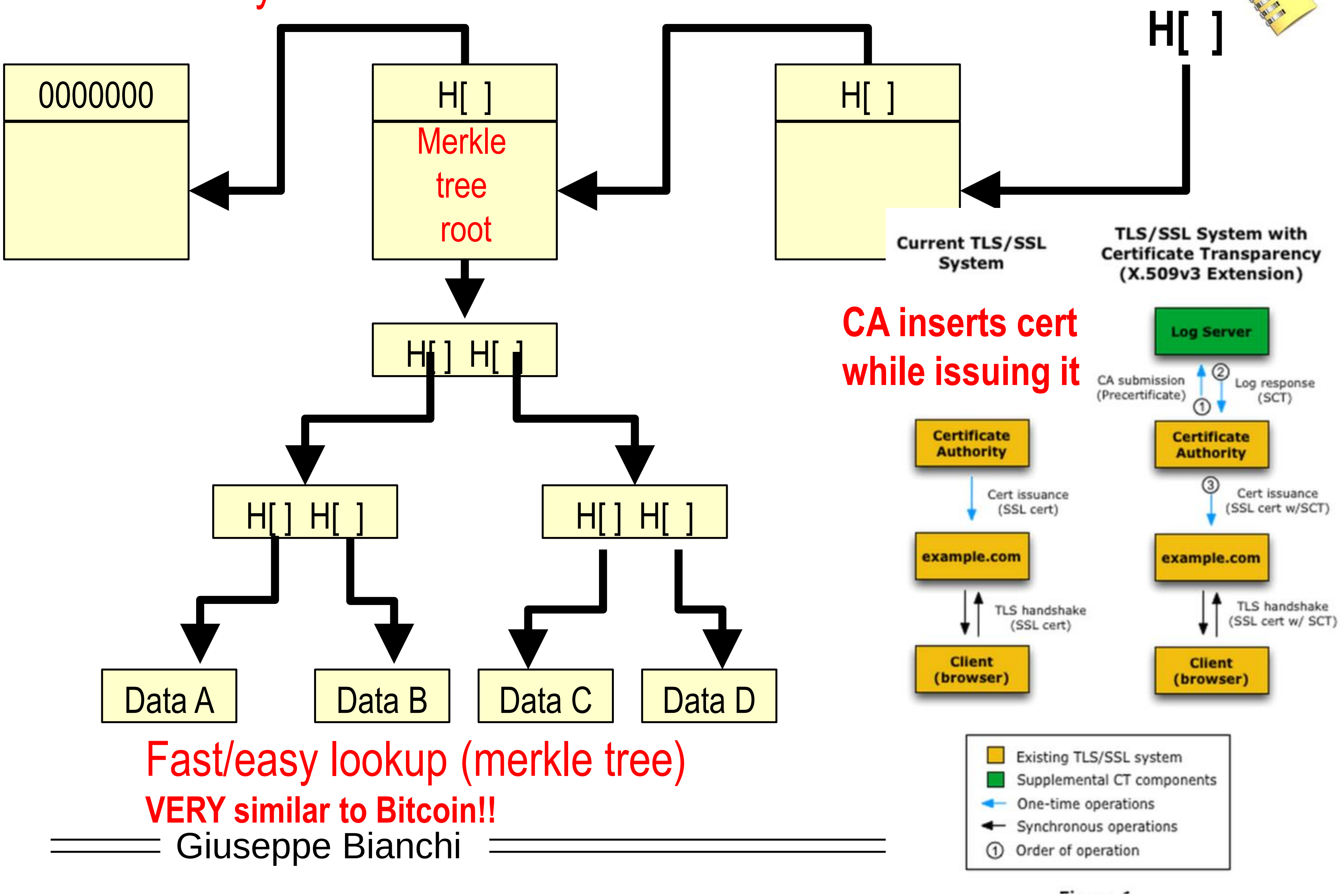
https://www.google.it

CERTIFICATE(I am google)$_{\text{LUNATRUST}}$

Fake!!

Done! (2013+, by google+)

1 block every 24 hours



CA inserts cert while issuing it

Fast/easy lookup (merkle tree)

VERY similar to Bitcoin!!

Looks like a blockchain…

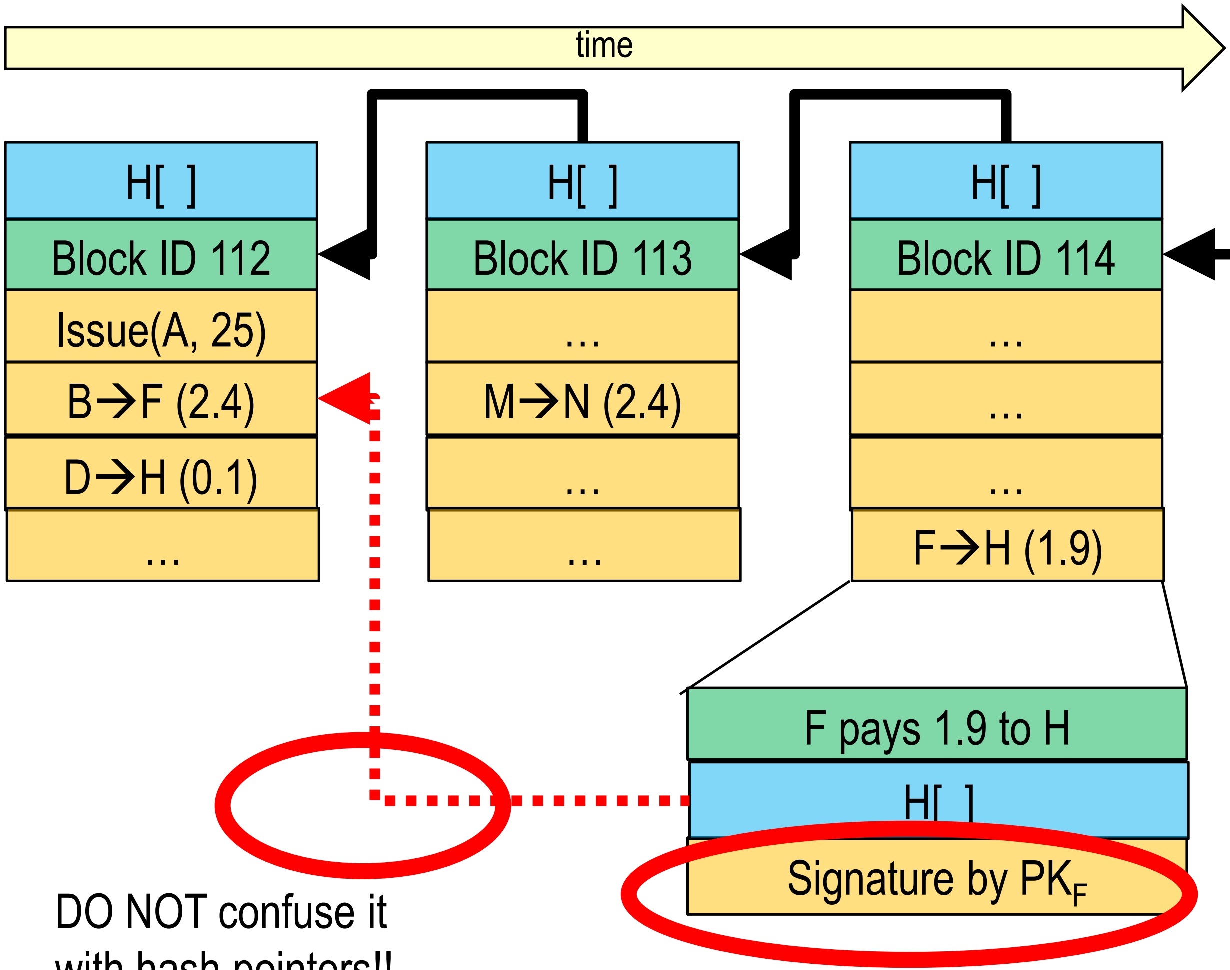
- ➔ **Hash pointer (block-based) data structure**
- ➔ **Potentially multiple log servers**
  - ⇨ Actually, not only Google’s log server
  - ⇨ Not synchronized but could have been (via consensus protocols)

But it is not… why?

- ➔ **No validation for inserted data!!**
  - ⇨ at least, no thorough validation; writers (CA) are (assumed) trusted
- ➔ **Log servers implement the application**
  - ⇨ Compare with bitcoin miners who don’t care at all about transactions!
- ➔ **Goal is (only) trasparency**
  - ⇨ Blockchain goal is **much** broader: **trustfulness**!!

Back to the ledger..

Ledger (multiple transactions into blocks)

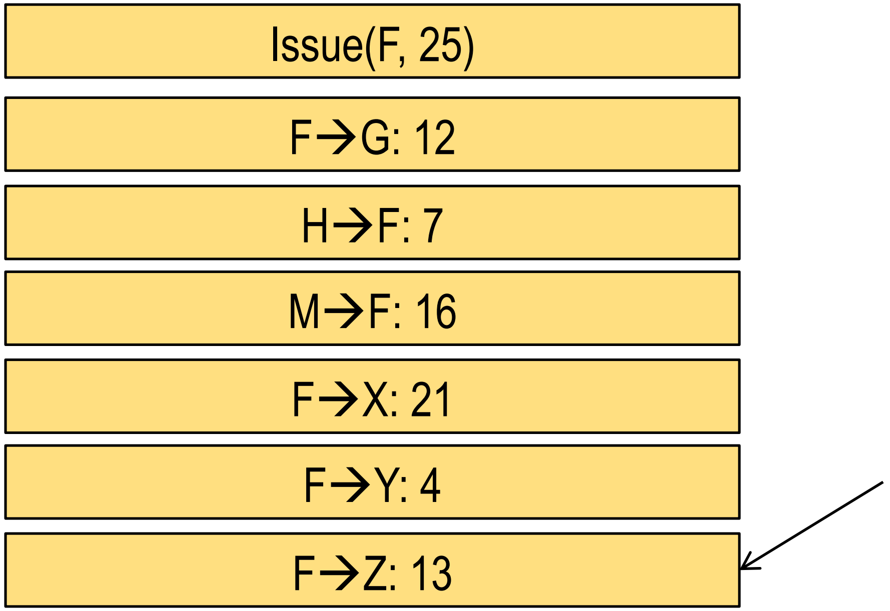


account balance securely logged in blockchain…

DO NOT confuse it with hash pointers!!

VERY IMPORTANT!!! Transactions controlled by end users, not by ledger!! (ledger «just» verifies that they are valid ones)

Account reconstruction: back to the genesis block



Issue(F, 25)

F→G: 12

H→F: 7

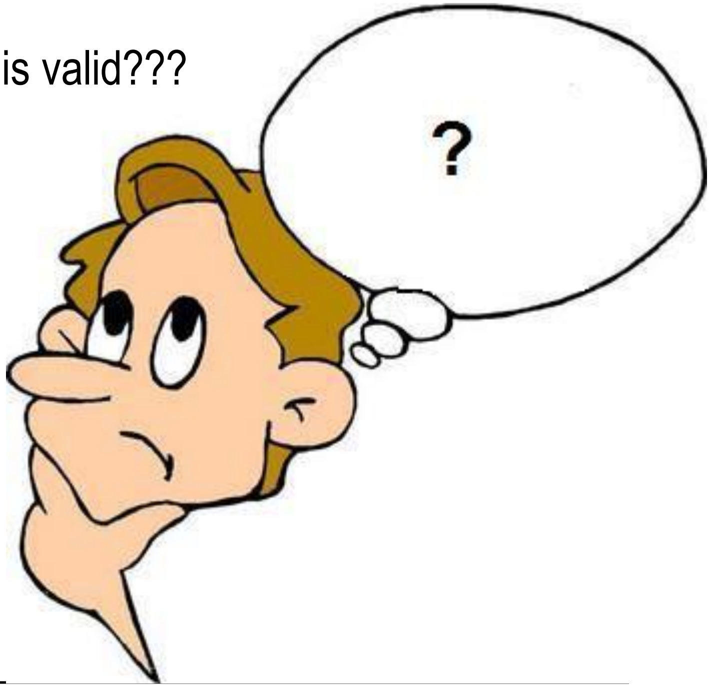
M→F: 16

F→X: 21

F→Y: 4

F→Z: 13

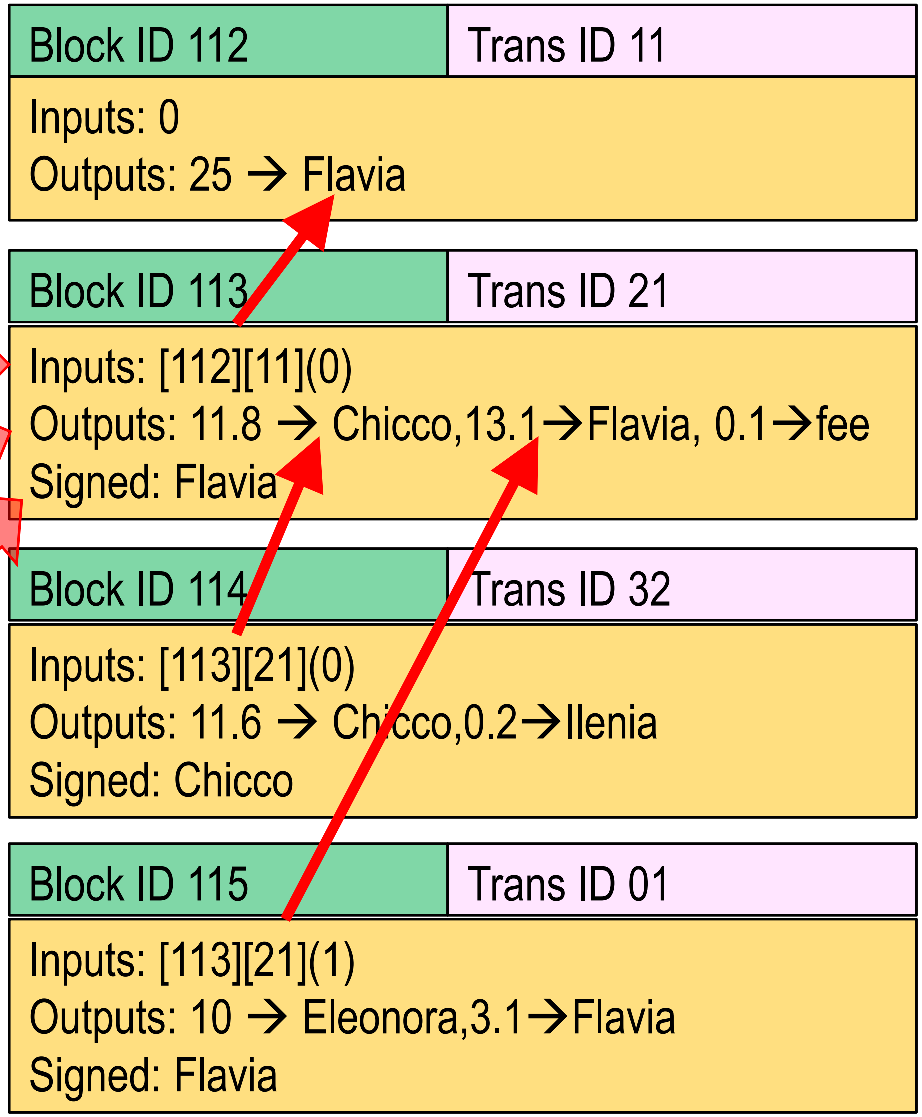
Is this valid???



The actual bitcoin transaction-based ledger (simplified example: one transaction per block)

- ➔ **Idea: each transaction has inputs**
  - ⇨ Except generation transaction

- ➔ **ALL inputs transformed into output, zero-sum**
  - ⇨ If sum != 0 transaction invalid
  - ⇨ If not signed by input owner, transaction invalid



Flavia has 25 BTC

Block ID 112 | Trans ID 11
Inputs: 0
Outputs: 25 → Flavia

Block ID 113 | Trans ID 21
Inputs: [112][11](0)
Outputs: 11.8 → Chicco,13.1→Flavia, 0.1→fee
Signed: Flavia

Block ID 114 | Trans ID 32
Inputs: [113][21](0)
Outputs: 11.6 → Chicco,0.2→Ilenia
Signed: Chicco

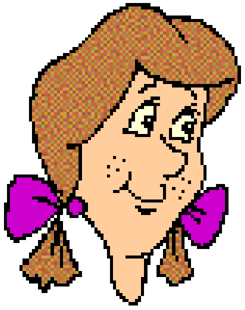
Block ID 115 | Trans ID 01
Inputs: [113][21](1)
Outputs: 10 → Eleonora,3.1→Flavia
Signed: Flavia

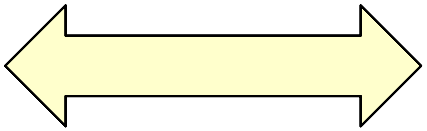
Technical Interlude 2: Identity without trust?

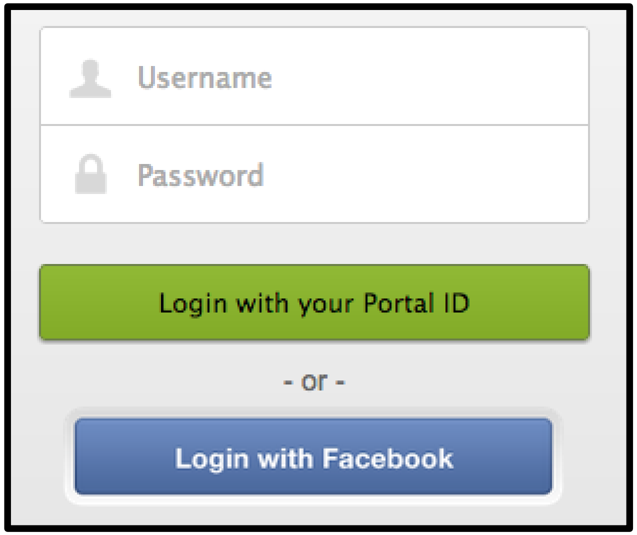
i.e.: how a person can perform transactions on the bitcoin ledger?

Identity providers

Linked in
BANK
g+
Agenzia per l'Italia Digitale
Presidenza del Consiglio dei Ministri
SPID
Sistema Pubblico
di Identità Digitale
ID







Username

Password

Login with your Portal ID

- or -

Login with Facebook

Some provide must know/authorize you!!

???

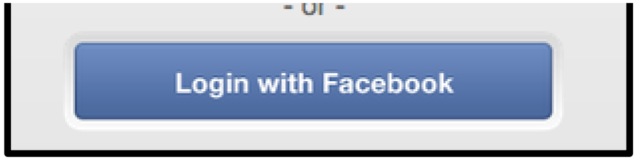
Identity providers

NOT REALLY…

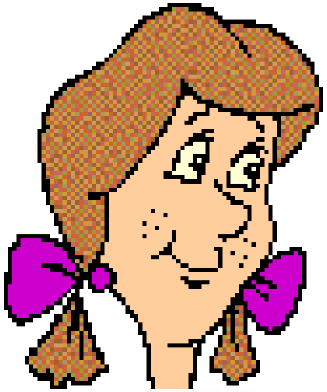
VERY OLD CRYPTO TRICK

Some provide must know/authorize you!!

???



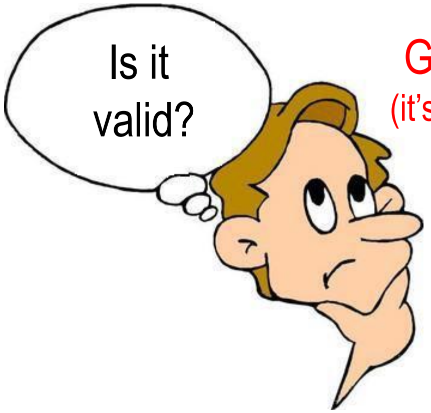
Background: digital signature



Public Key
PK

M = I'm transferring 1 BTC to Jon | H(M) Use SK

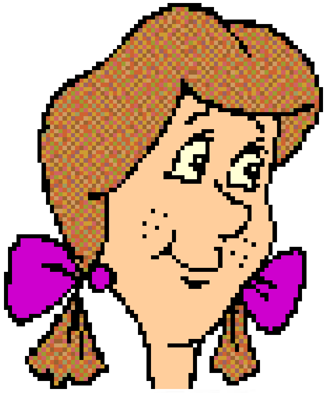
Private Key
SK



Get PK
(it's public!)

M = I'm transferring 1 BTC to Jon | H(M) Use SK

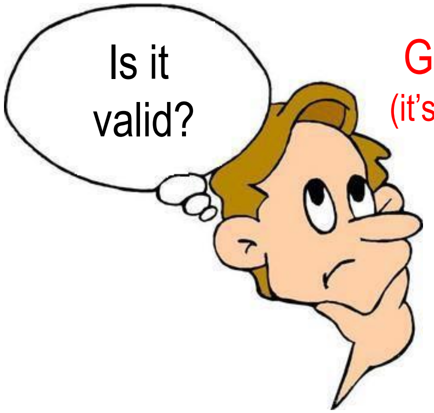
Background: digital signature



Private Key
SK

Public Key
PK

M = I’m transferring 1 BTC to Jon | H(M)
Use SK



Get PK
(it’s public!)

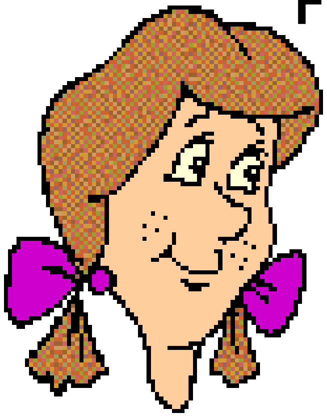
M = I’m transferring 1 BTC to Jon | H(M)
Use SK

Use PK to «invert» tag

Do they match?
If Y, transaction OK

Identity decentralization
(public keys as identities!)

Forge your own «identity» – can be many!



Generate a pair:
PK = Public key
SK = Private Key

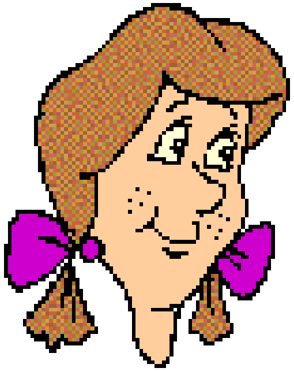
Use (hash of) PK as identity
hash[PK] is called «address» in bitcoin

Sign every transaction you perform with SK

Anyone which sees a transaction from you=PK can verify
that it’s really you, by simply checking the signature

But **you remain the ONLY one able to perform a transaction**
from YOUR (self-assigned) address H(PK)!!

Step by step…



ADDRESS = H[PK] = a1b2c3d41235ef

Account name: 256 bits, 64 hex string
in bitcoin (SHA-256)

ADDRESS = **867aff3432af**

A1b2c3d41235ef → **msg = «transfer 1 BTC to 867aff3432af» | signature(msg)**

TRANSACTION VERIFIER → retrieves PK for (anonymous) user a1b2c3d41235ef

→ checks that H[PK] = a1b2c3d41235ef

collision resistance protects from impersonation:
not possible to claim different PK for a given address

→ verify(PK, message, signature) = TRUE

only a1b2c3d41235ef knows private key SK!

No need for any intermediate/central authority to issue/manage «accounts»

Decent level of privacy (as long as multiple identities are used for multiple transactions)

2° blockchain dimension: Consensus

consensus

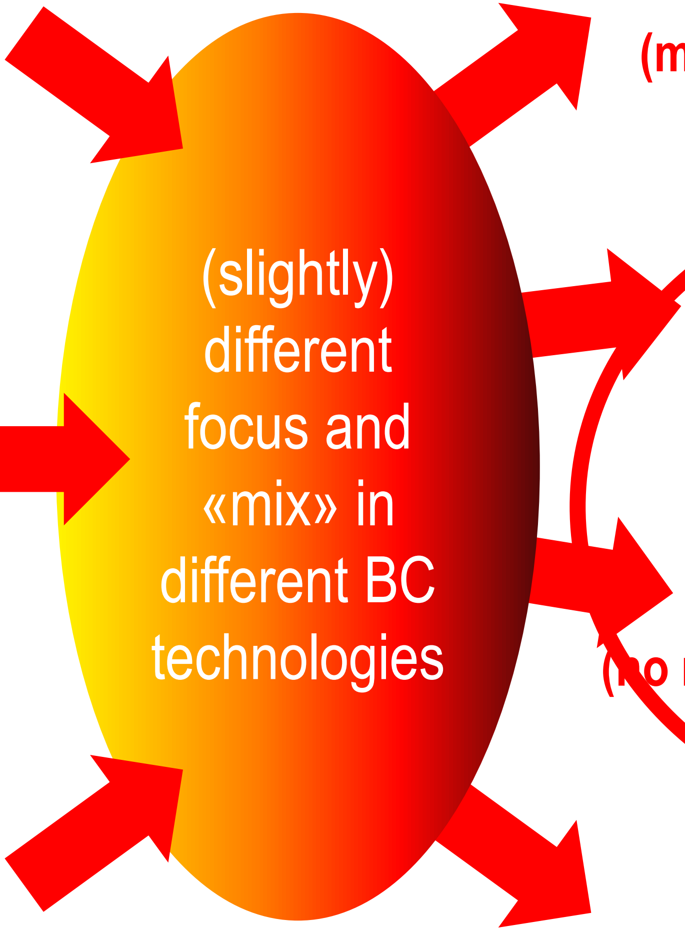
Technical asset

Outcome, impact

Append-only secure storage (hash-based ledger)

Trust without single trusted party (consensus protocols)

Transactions' validation and smart contracts (scripting languages)



(slightly) different focus and «mix» in different BC technologies

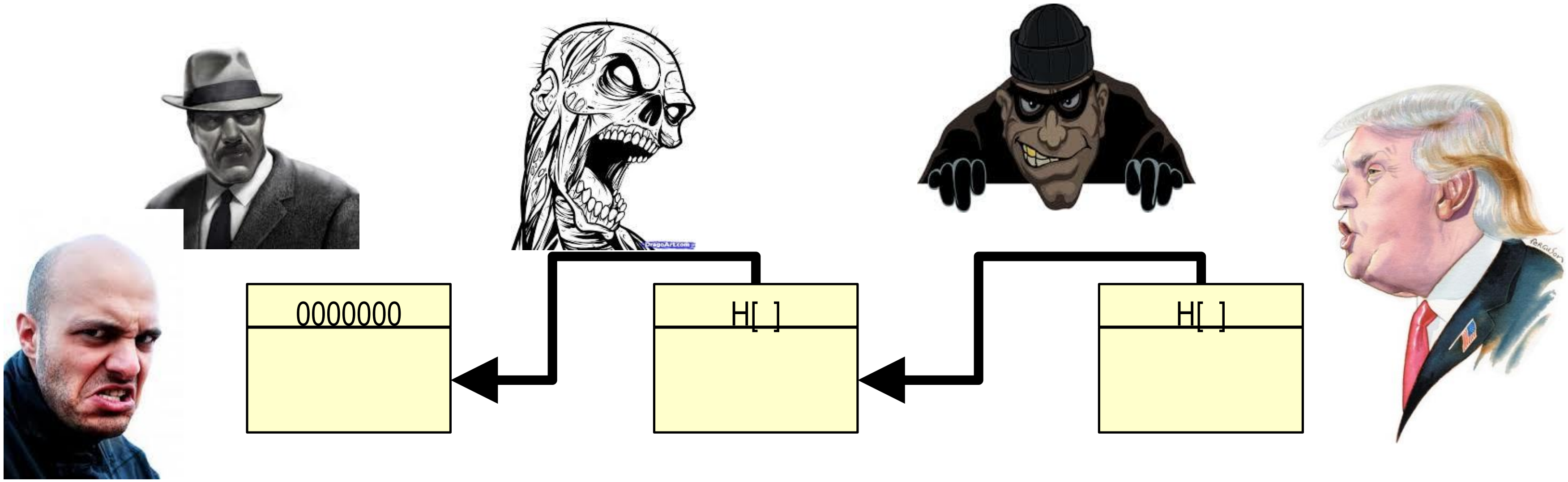
Transparency (many societal implications)

Indelibility (notary services)

Shareability across boundaries of trust (no need for single trust anchor)

(Very) sophisticated «ownership» Control (actually, more than this!)

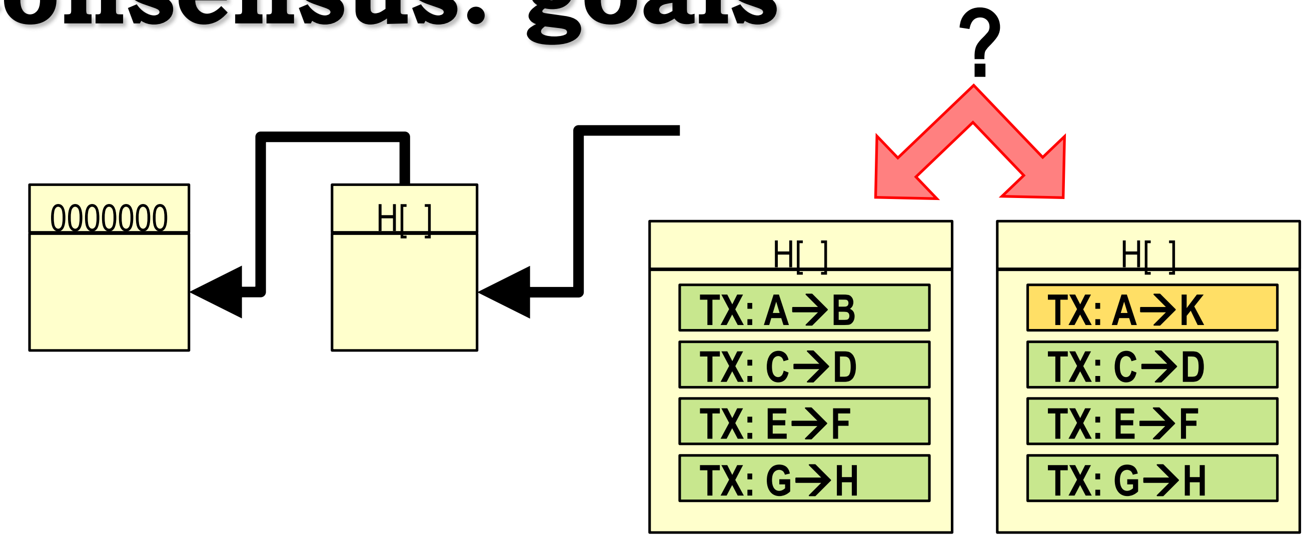
Concept: single (shared) storage!



➔ **Two very different scenarios**

- ⇨ **PERMISSIONED** *These do NOT drain (too much) energy* ☺
  - →known/controlled set of untrusted parties which «build» the chain
- ⇨ **PERMISSIONLESS**
  - →anyone can add a block: unknown/uncontrolled set of miners!
- ⇨ Well, sometimes third scenario: PRIVATE
  - →*Does it make sense?! Mah. Though interoperability is still an asset….*

Consensus: goals



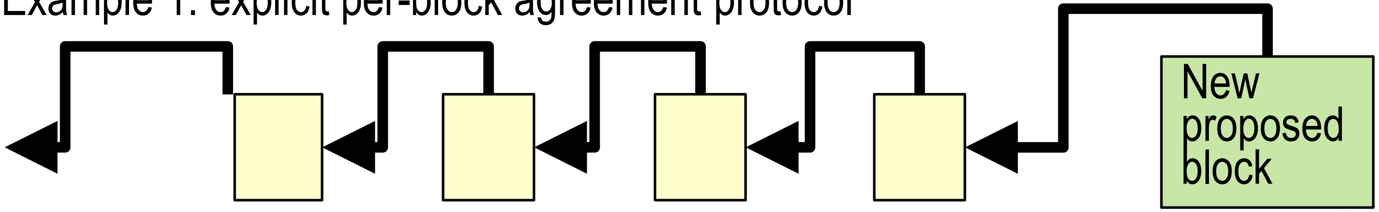
→**Two conceptually different forms of agreement**

- ⇨On the transactions contained in a block
- ⇨On the VALIDITY of such transactions!
  - →e.g., bitcoin: correct balance + correct signature

Permissioned Blockchains:
(many!) consensus protocols available

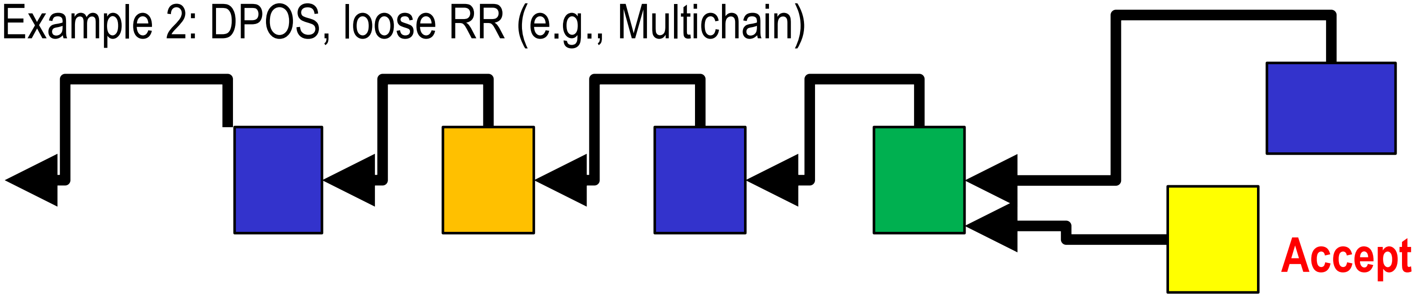
- ⇨ RAFT (Paxos), BTF-SMaRt, Byzantine Fault Tolerant variants (PBFT, XFT, CFT, …), Dynamic permissioned, loose (probabilistic) RR, DPOS, …
- ⇨ Consolidated literature since the 80ies
  - →Many subtleties… no time today…
- ⇨ You may **choose** consensus model in some platforms (e.g. Hyperledger)

Example 1: explicit per-block agreement protocol



Signature by (qualified) majority

Example 2: DPOS, loose RR (e.g., Multichain)



Rejected (already 2 in last 5)

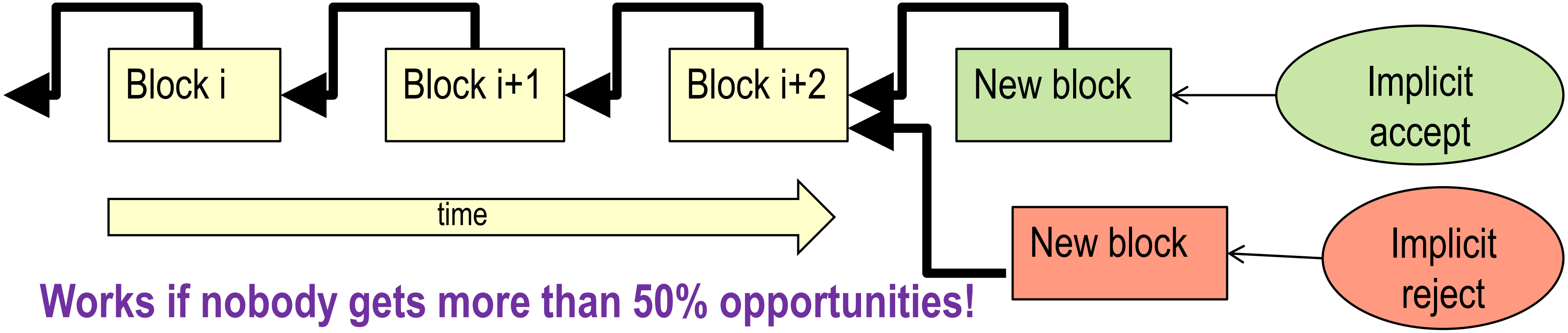
Accepted

Permissionless/wild Blockchains: much harder!

- ➔ **No support from theory!**
  - ⇨ Actually, negative results from theory
  - ⇨ Fischer-Lynch-Paterson's 1985 impossibility result: (asynchronous) consensus impossible even with a single (!) faulty node
- ➔ **So?**
- ➔ **Bitcoin' quite successful pragmatic approach!**
  - ⇨ Clever combination of incentive + Randomization via proof-of-work

If we could select at random…

- ➔ **No protocol! (leaders, masters, elections, messages, etc)**
- ➔ **Select random node at regular time (e.g., 10m)**
  - ⇨ How???!!! more later on this!
- ➔ **Selected node adds block to the chain**
  - ⇨ And gets an **incentive** for this *(e.g. bitcoins, fees)*
- ➔ **New block includes VALID transactions seen so far**
  - ⇨ delayed transactions not a problem, can be included in next block
- ➔ **Implicit acceptance – next selected node:**
  - ⇨ extends chain from there → implicitly accepts block
  - ⇨ Extends chain from previous block → implicitly rejects block

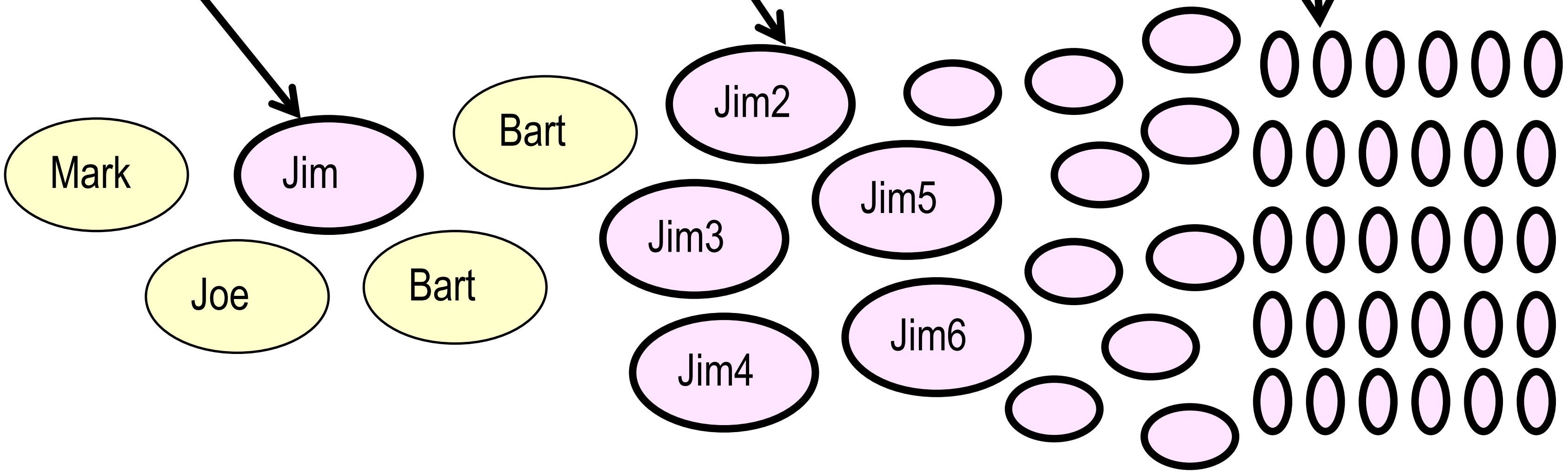


Works if nobody gets more than 50% opportunities!

How to select at random?

➔No trusted party available to «run» the selection!

➔Selection must resist SYBIL attacks!!



20% chance for Jim

60% chance to select one of Jim's controlled identities!

Critical issue when it is cheap to forge an identity!

Sybil-resistant random selection

➔Randomization NOT based on # identities
➔But based on some RESOURCE!!

➔E.g., Bitcoin's proof-of-work (PoW)

- ⇨ probability proportional to computational power owned

➔PoW is just «one possible» approach…

- ⇨Proof-of-stake: probability proportional to memory you have
- ⇨Proof-of-elapsed-time…
- ⇨Proof-of-****, where «****» prevents from sybil

KEEP IN MIND: permissioned BC do NOT have any of these problems!!

Scalability issues? Reasonable power consumption?
Not nearly a permissioned blockchain issue!!!

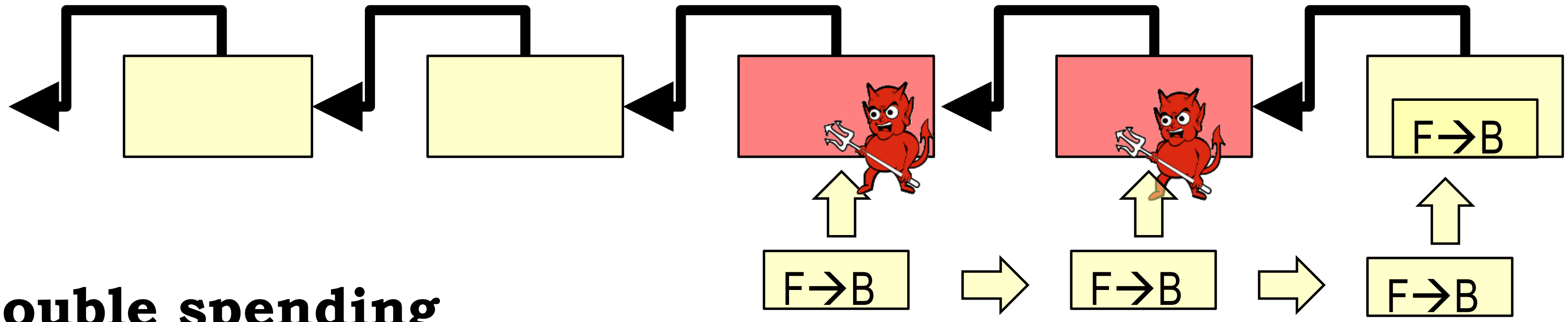
Possible attacks (to bitcoin chain)

➔ Steal your money/asset

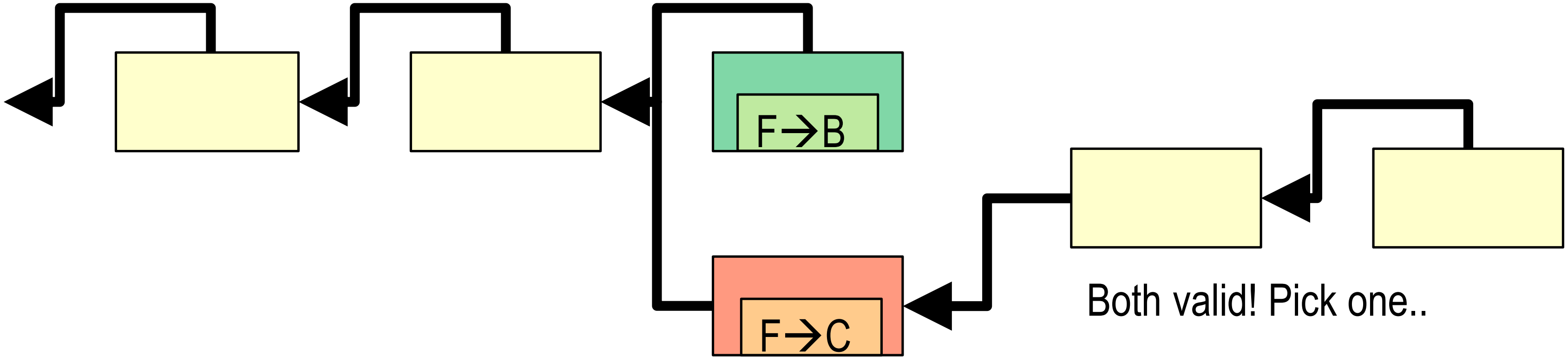
- ⇨ No way, attacker does not know your private key

➔ Keep you out of the blockchain

- ⇨ Not possible with explicit (signature based) consensus protocol;
- ⇨ With implicit consensus or randomization honest blocks will include you back ☺



➔ Double spending



Both valid! Pick one..

3° blockchain dimension: Scripting

scripting

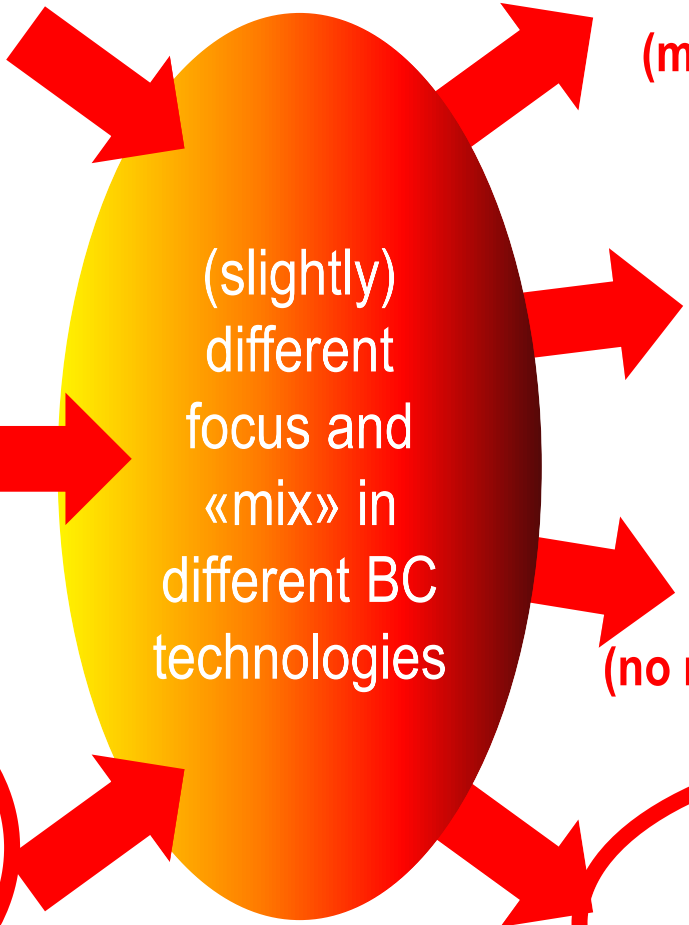
Technical asset

Outcome, impact

Append-only
secure storage
(hash-based ledger)

Trust without
single trusted party
(consensus protocols)

Transactions' validation
and smart contracts
(scripting languages)



(slightly)
different
focus and
«mix» in
different BC
technologies

Transparency
(many societal implications)

Indelibility
(notary services)

Shareability across
boundaries of trust
(no need for single trust anchor)

(Very) sophisticated
«ownership» Control
(actually, more than this!)

Bitcoin transactions → scripts

(slightly simplified)

metadata

input(s)

output(s)

```
{
      "hash":"5a42590fbe0a90ee8e8747244d6c84f0db1a3a24e8f1b95b10c9e050990b8b6b",
      "ver":1,
      "vin_sz":2,
      "vout_sz":1,
      "lock_time":0,
      "size":404,
      "in":[
        {
          "prev_out":{
            "hash":"3be4ac9728a0823cf5e2deb2e86fc0bd2aa503a91d307b42ba76117d79280260",
            "n":0
          },
        "scriptSig":"30440..."
        },
        {
          "prev_out":{
            "hash":"7508e6ab259b4df0fd5147bab0c949d81473db4518f81afc5c3f52f91ff6b34e",
            "n":0
          },
          "scriptSig":"3f3a4ce81...."
        }
      ],
      "out":[
        {
          "value":"10.12287097",
          "scriptPubKey":"OP_DUP OP_HASH160 69e02e18b5705a05dd6b28ed517716c894b3d42e
  OP_EQUALVERIFY OP_CHECKSIG"
        }
      ]
}
```

Identifier of this transaction (its hash)

Housekeeping + global advanced features (e.g. lock-time)

Pointer to previous transaction

Signature (proves ownership of prev transaction)

THIS (!) transaction – what you want to do now:
a SW program – script!

Source: princeton 2015 course (+ my additions)

Scripting: more than logging!!

- ➔ **Code associated to any (!) transaction**

- ➔ **Main role of a script:**
  - ⇨ Formalize verification conditions
    - →Transaction valid if script terminates OK → truthfulness formalized!
  - ⇨ May formalize a process involving players
    - → enable transition only if Mr. X has given permission

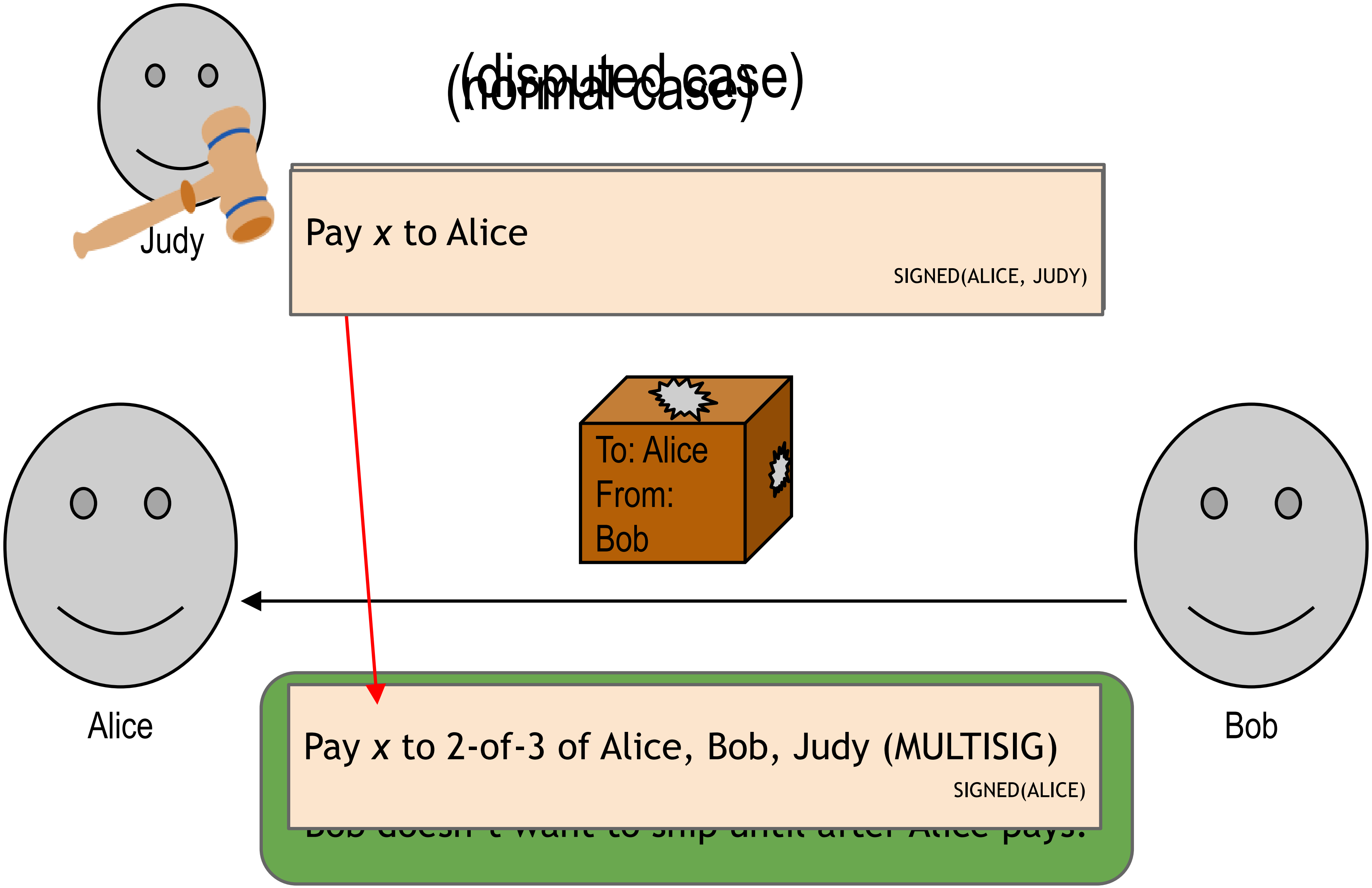
- ➔ **Smart contracts (not new – see Szabo 1996)**
  - ⇨ Broader view of scripting: not only validity, but also execution of actions
  - ⇨ Lots of promises, but also lots of concerns
    - →remember ETH DAO (2016) & Parity Wallets (2017)?!

- ➔ **Smarter scripting (e.g. Turing-complete)?**
  Or smarter crypto? (I'm for the latter)

Booster: multi signatures

(disputed case)

(normal case)



Pay *x* to Alice

SIGNED(ALICE, JUDY)

Pay *x* to 2-of-3 of Alice, Bob, Judy (MULTISIG)

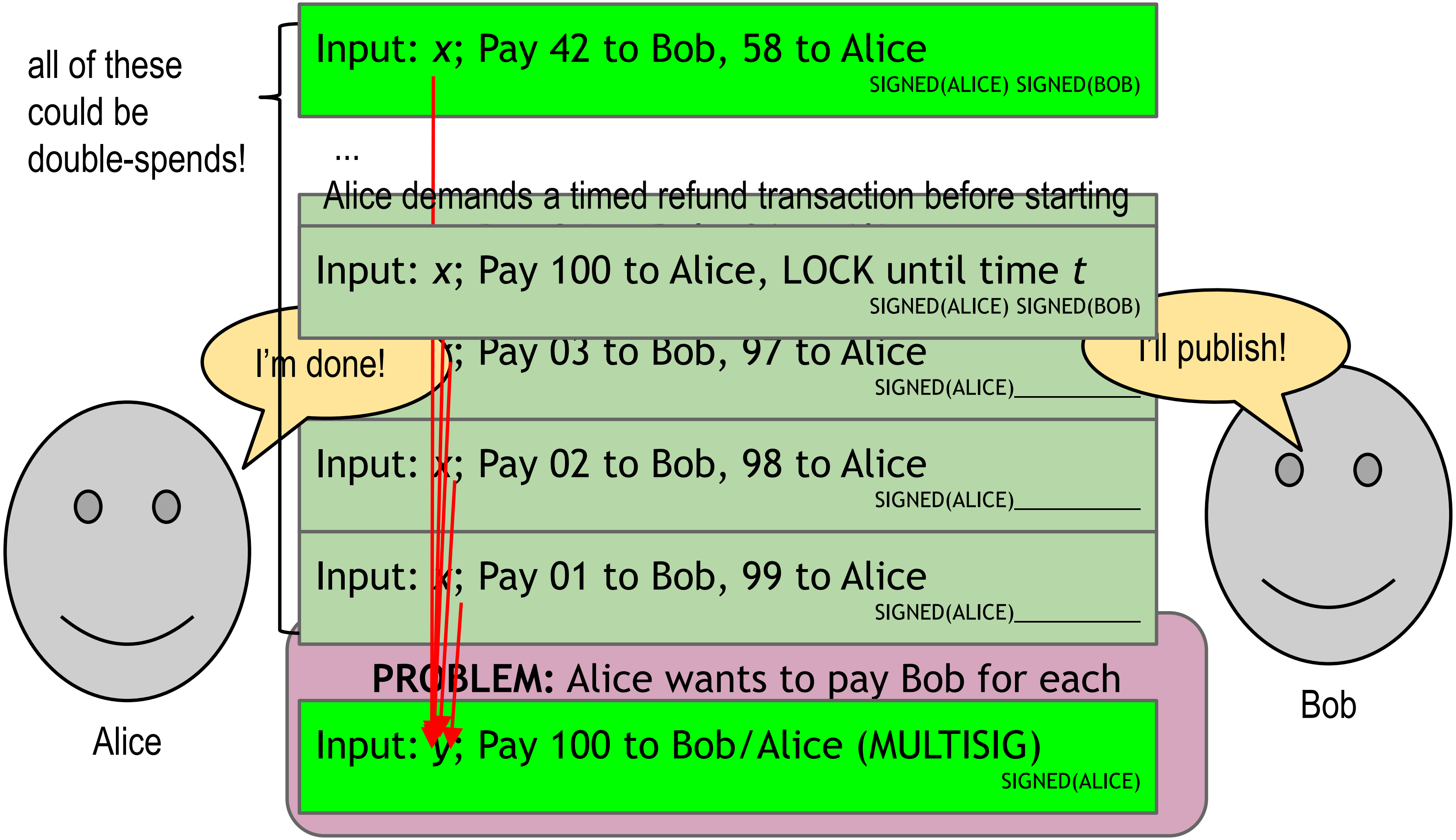
SIGNED(ALICE)

Bob doesn't want to ship until after Alice pays.

source: my adaptation of Princeton 2015 slide

Efficient micro-payments

What if Bob never signs??



all of these could be double-spends!

Input: *x*; Pay 42 to Bob, 58 to Alice
SIGNED(ALICE) SIGNED(BOB)

...

Alice demands a timed refund transaction before starting

Input: *x*; Pay 100 to Alice, LOCK until time *t*
SIGNED(ALICE) SIGNED(BOB)

; Pay 03 to Bob, 97 to Alice
SIGNED(ALICE)___________

Input: *x*; Pay 02 to Bob, 98 to Alice
SIGNED(ALICE)___________

Input: *x*; Pay 01 to Bob, 99 to Alice
SIGNED(ALICE)___________

PROBLEM: Alice wants to pay Bob for each

Input: *y*; Pay 100 to Bob/Alice (MULTISIG)
SIGNED(ALICE)

Slide taken from Princeton 2015 course

Applications?

- **Crypto currencies**
  - ⇨ of course! Though most scams / pump&dump
- **Asset transfering / transaction notarization**
  - ⇨ plenty of use cases
  - ⇨ More clever crypto conditions → more advanced apps
    - → e.g. involvement of notary attributes to restrict transactions' domain
    - → E.g. release of unblocking keys by transaction itself
- **Workflow management in complex scenarios**
  - ⇨ Blockchain = greater transparency and auditability
- **Identity management**
  - ⇨ Identity attributes come from multiple authorities…
  - ⇨ blockchain as shared interoperable database

Taking stocks…

- **Think twice before embarking into a blockchain deployment**
  - ⇨ An ordinary database may suffice (or even be superior!!)
- **Industrial applications focus on permissioned!**
  - ⇨ Very different story than public (e.g. bitcoin)
- **Less is (often) more!**
  - ⇨ Do you really need complex scripting and EVM?
  - ⇨ Think to your application requirements!
- **Very interesting Side effect: data/transactions are natively shareable/shared!**
  - ⇨ Interoperability not anymore an issue!

A few research topics

- ➔ **Consensus**
  - ⇨ Protocols for permissioned
  - ⇨ Alternative randomization (e.g. Algorand, IOTA's Tangle, etc)
  - ⇨ More scalable and sustainable Proof-of-*
- ➔ **Crypto/scripting for better contracts**
  - ⇨ Commitments, policy-based signatures, physical activation keys generation, …
  - ⇨ Optimizations (e.g. with Schnorr)
  - ⇨ Which scripting is best suited?
- ➔ **Alternative ledgers / architectures**
  - ⇨ E.g. AlgoRand, Tangle, R3/CORDA
- ➔ **bitcoin (& wild blockchain) evolution**
  - ⇨ Plenty of game theory involved!
    - →E.g. fees' management
    - →E.g. huge miners' pools likely not what Sakamoto had in mind
  - ⇨ Security, scalability, monitoring, …
- ➔ **And (mostly!!) meaningful applications & deployment…**