

Blockchain and Smart Contracts – From Theory to Practice

Bruno Rodrigues¹, Eder Scheid¹, Roman Blum²,
Thomas Bocek², Burkhard Stiller¹

¹*Communication Systems Group CSG, University of Zürich UZH, Switzerland*

²*Hochschule für Technik Rapperswil HSR, Switzerland*

[rodrigues!scheid!stiller]@ifi.uzh.ch

[roman.blum!thomas.bocek]@hsr.ch



**University of
Zurich**^{UZH}



**HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL**

Schedule

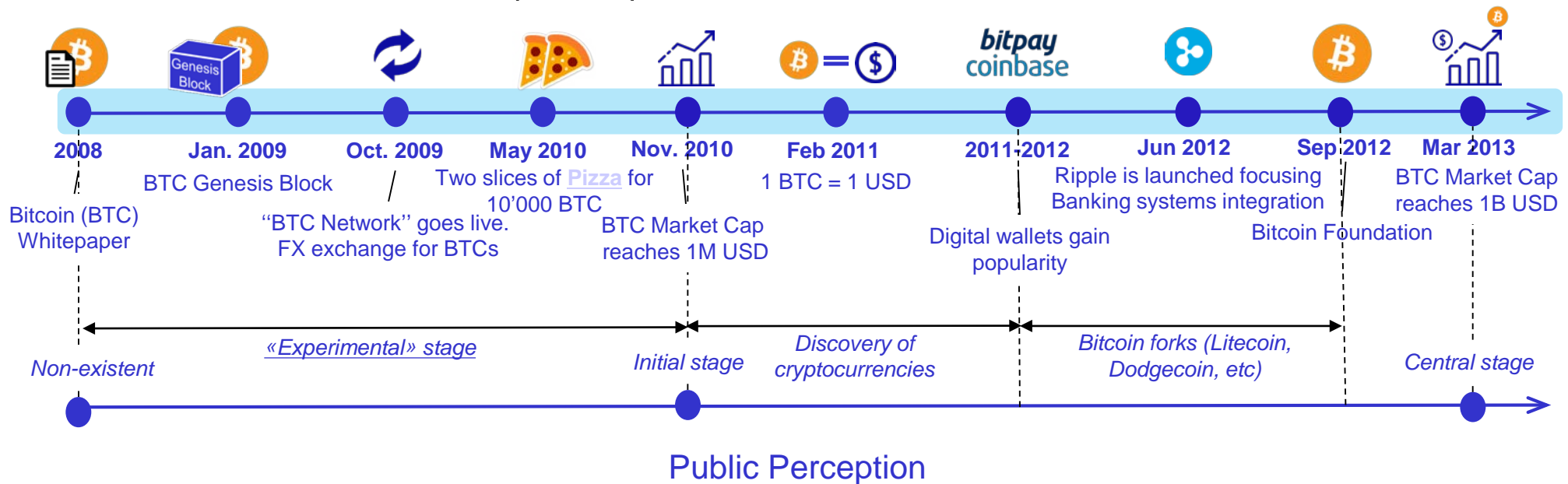
- ❑ Part I – Blockchain Motivation
 - Timeline and Basics
 - Consensus Mechanisms
 - Management Aspects
- ❑ Part II – Smart Contracts
 - Smart Contract
 - Best Practices
 - Security
- ❑ Part III – Discussion & Considerations
 - Challenges and Risks
 - Considerations

Part I – Blockchain Motivation

Blockchain 1.0

□ Digital Currency

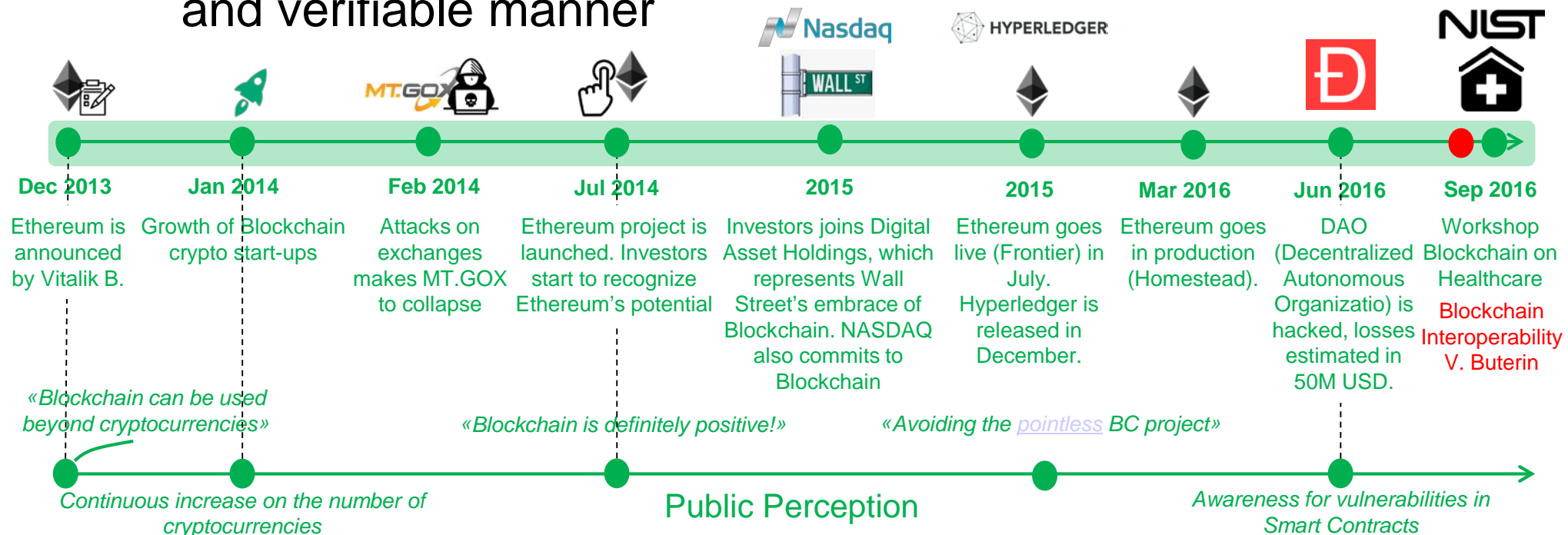
- Decentralized payment system
- Bitcoin as the father of digital currencies
 - Still, not much awareness of (other) Blockchain capabilities
- Proof-of-Work (PoW)



Blockchain 2.0

□ Smart Contracts

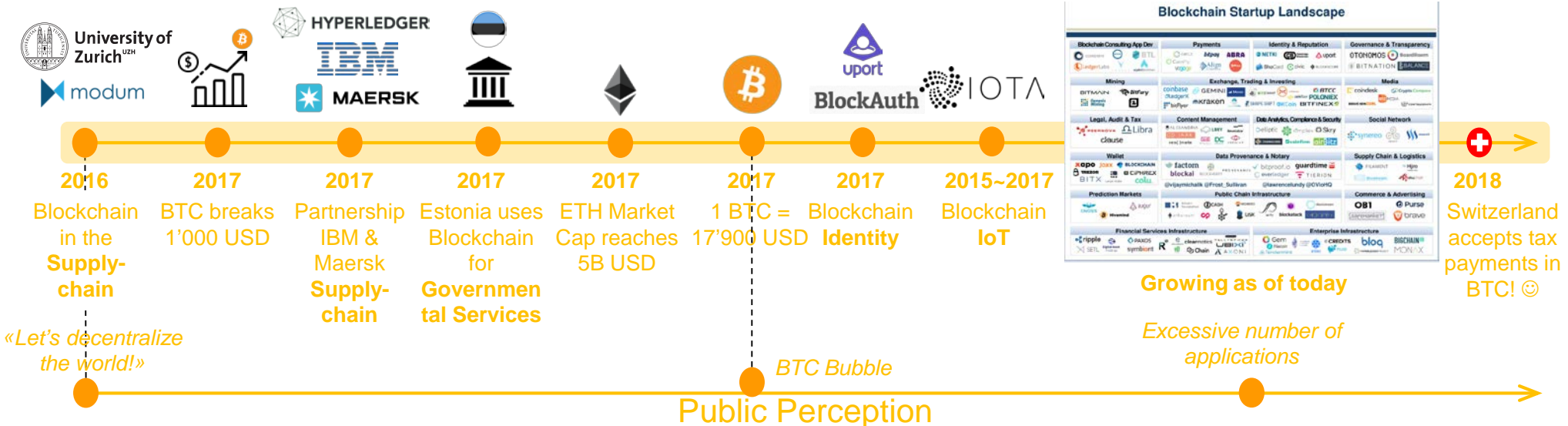
- Ethereum unlocks the blockchain potential beyond cryptocurrencies
- Blockchain is able to run computer programs in a transparent and verifiable manner



Blockchain 3.0

❑ Decentralized Applications (DApps)

- Production stage:
 - Large number of applications
- Scalability/Performance issues:
 - Need for performance → new consensus protocols
 - Need for storage → off-chain storage tools



Blockchain 4.0

❑ Ecosystem and Industry Integration

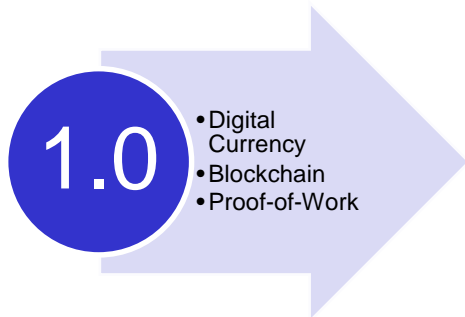
- Making blockchain effective in industry
- Decentralized and disconnected blockchain networks
 - Vendor-specific blockchain technology, interoperability
- Need for standardization

As of today



Blockchain Eras and Evolution

- ❑ 4 different BC eras are running in parallel today



- 1.0 – December 08/January 09: Bitcoins

- More than 2100 cryptocurrencies available today

Cryptocurrencies: 2095 • Markets: 15834



- 2.0 – 2012-14: Ethereum, Smart Contracts, Solidity, ...

- 3.0 – April 2012: Decentralized Apps (dApps) – “Satoshi Dice”

<https://hackernoon.com/dapp-and-things-you-need-to-know-4f50853a4cb7>

- Running on peer-to-peer network, all data transparent and tamper-proof

- 4.0 – App. 2015: BC ecosystems and industrial integration

- Countless Blockchain projects in many fields
 - FinTech, supply-chain, governmental, identity, ...

Driving Questions

- How and under which conditions to use blockchain?
 - Creator (e.g., researcher) or investor point-of-view
- Is there a right or wrong? A roadmap for blockchain usage, possibly.
 - There is no simple answer ...



Application

"What are application requirements?"

Blockchain

"Which different types of blockchain one can offer?"

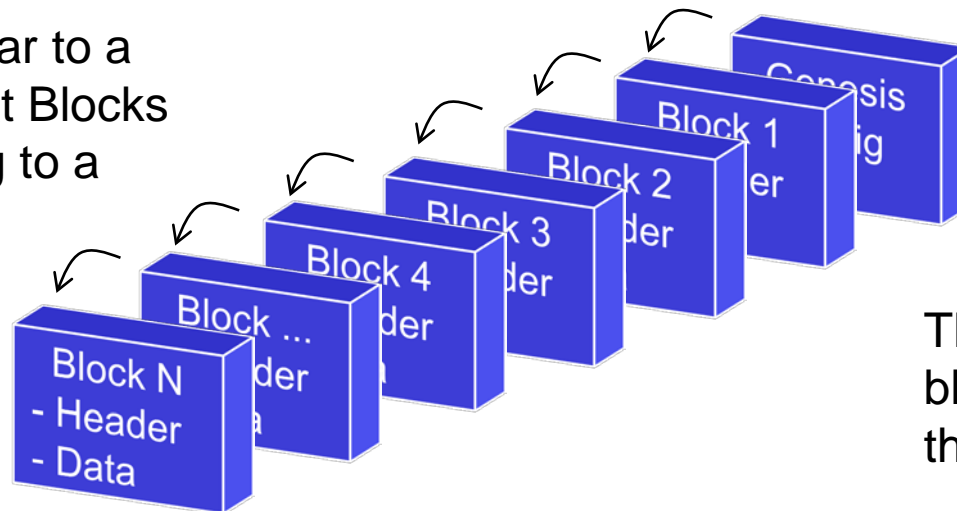


Blockchain (BC) Basics

Definition

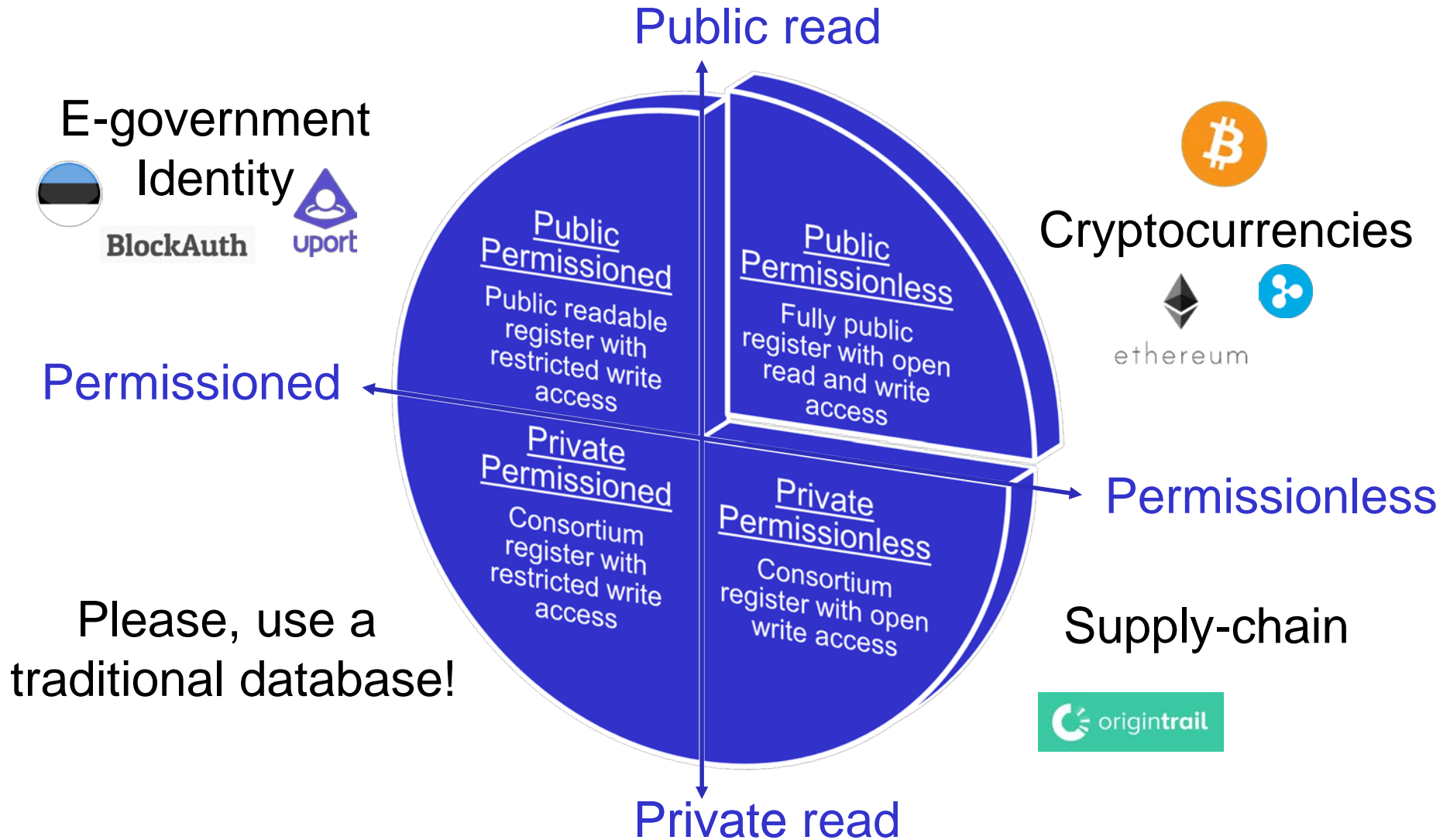
- ❑ A **Blockchain** (BC) or **Distributed Ledger** (DL) is a **decentralized** digital ledger that **transparently** and **permanently** record blocks of transactions across computers based on a consensus algorithm without modifying the subsequent blocks.

A blockchain is similar to a **linked list** except that Blocks are added according to a consensus protocol



The genesis or the first block define the settings of the blockchain

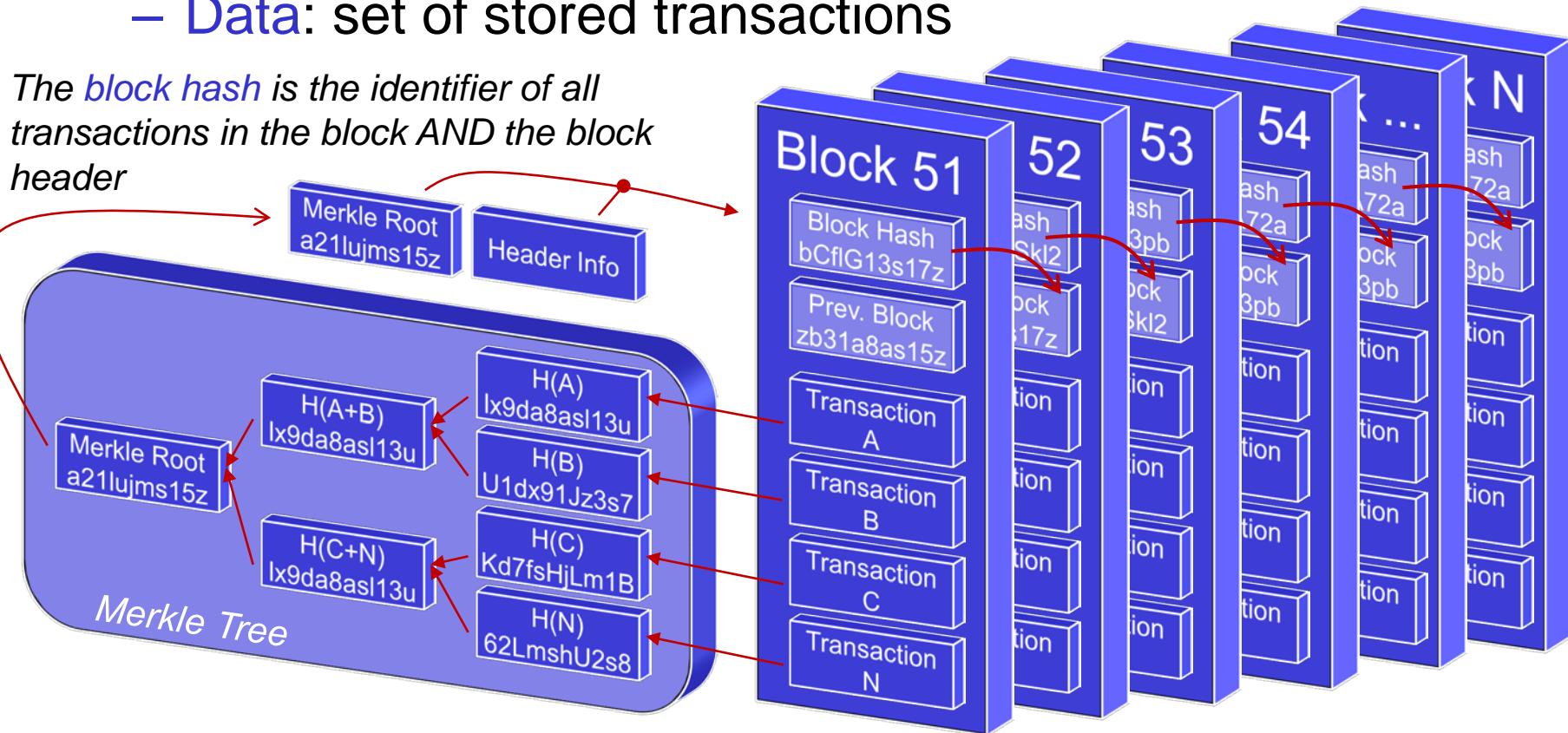
Permissions and Transparency



Block

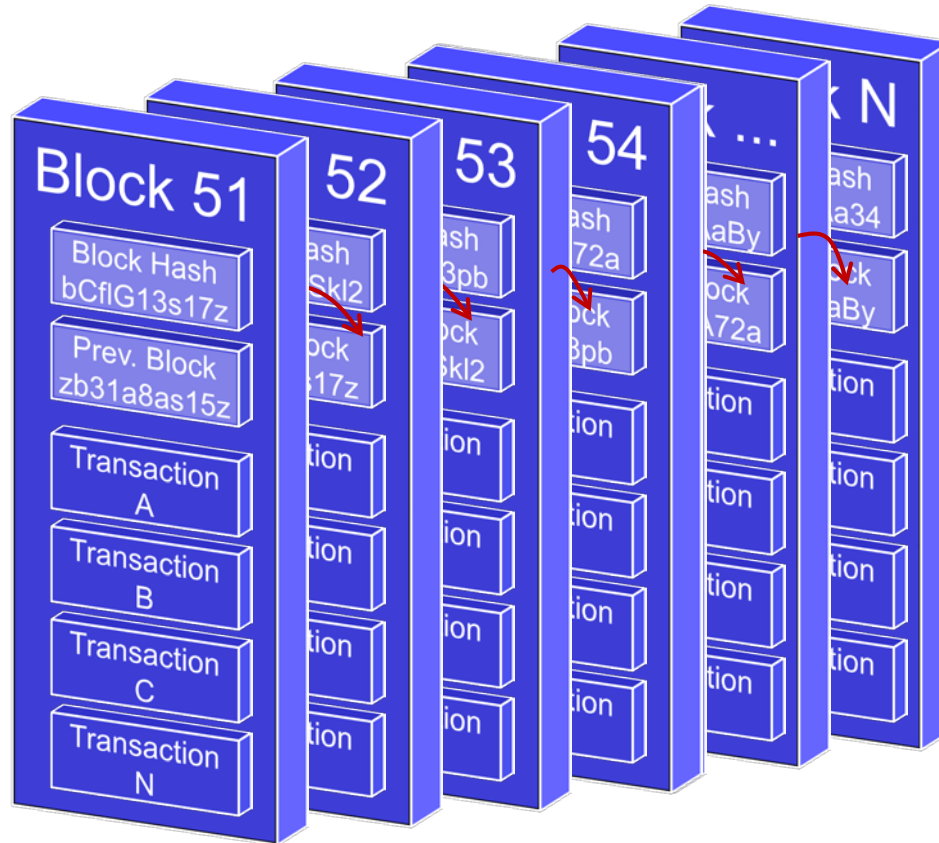
- A block is a structure to store data (transactions)
 - **Header**: information to identify the block.
 - **Data**: set of stored transactions

*The **block hash** is the identifier of all transactions in the block AND the block header*

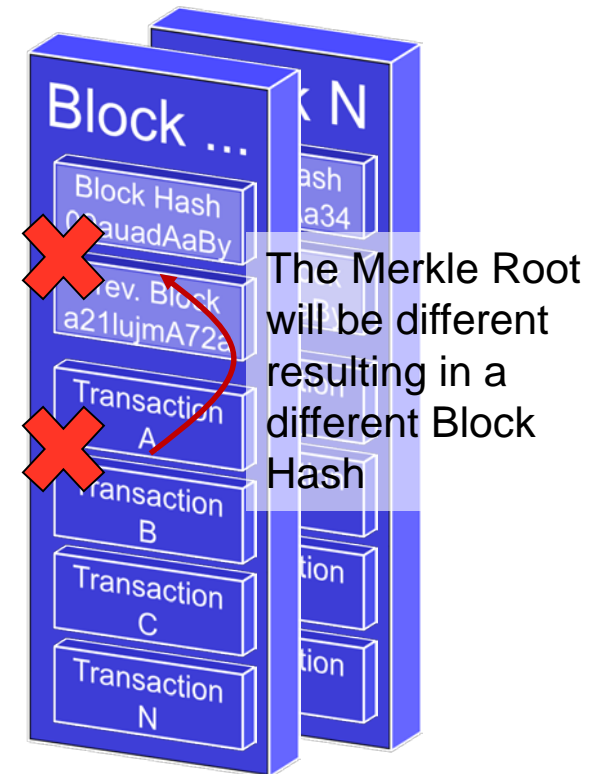


Integrity, Merkle Tree

- In practice, the Merkle Tree guarantees **immutability**



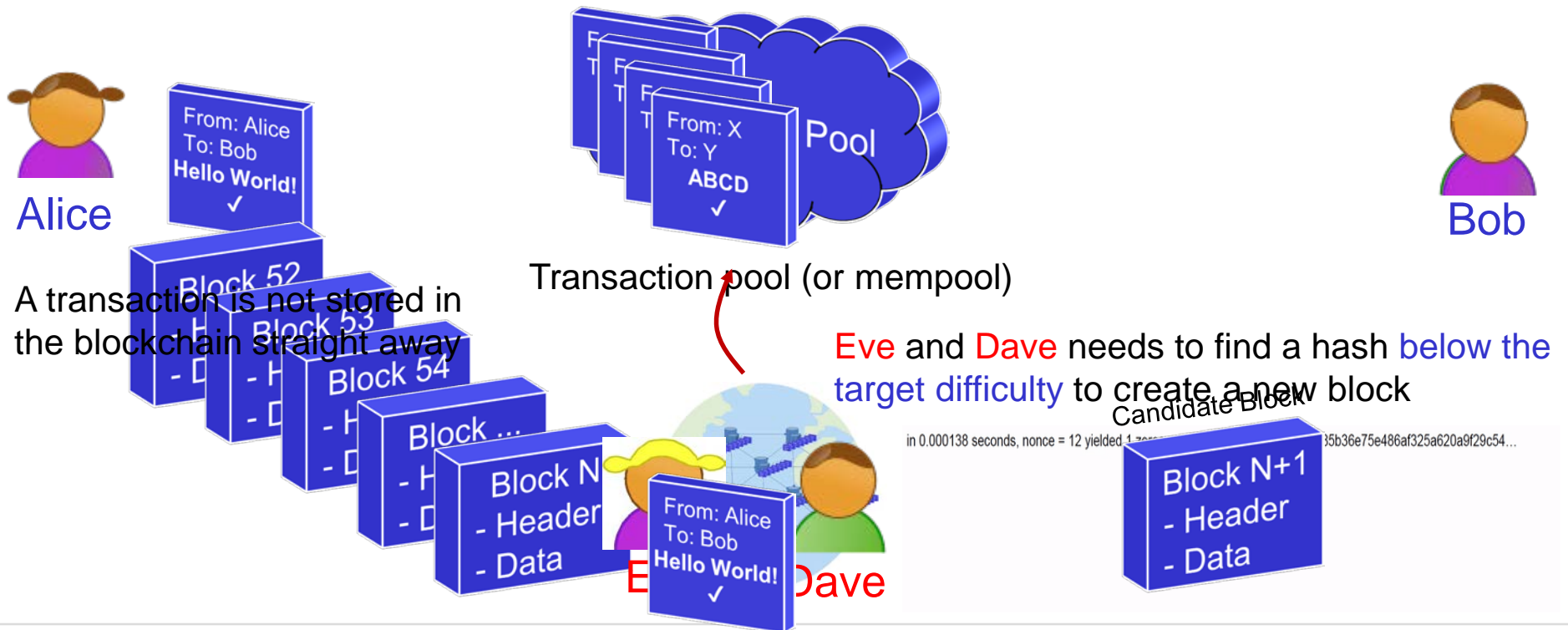
Then, a parallel (forked) chain is created



Imagine if one wants to remove/change a transaction

Transactions

- ❑ How are transactions stored in a block?
 - *Transaction pool or mempool*
 - *Temporary storage structure (RAM) available on each full node (Ethereum)*



Blockchain Consensus

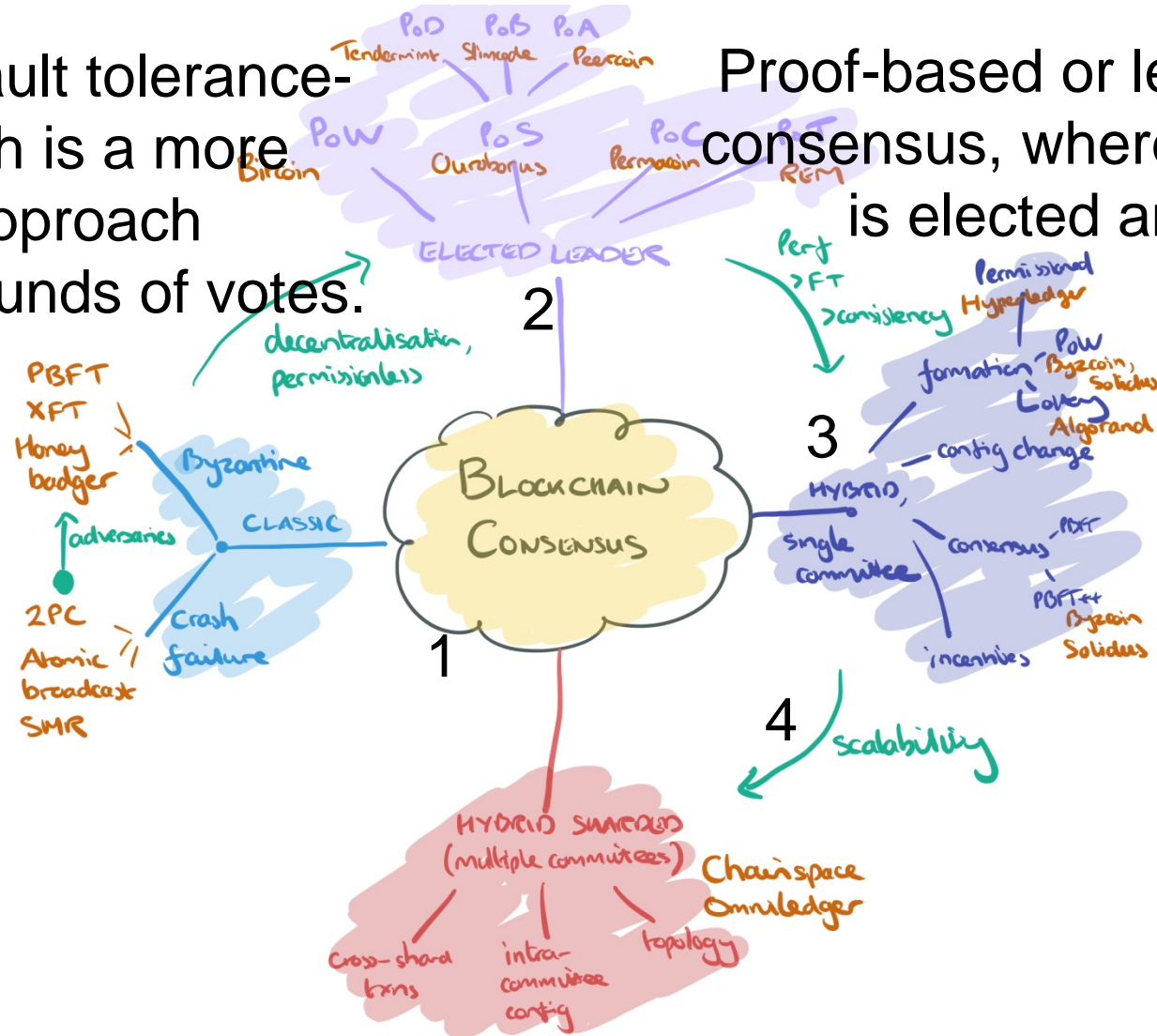
Mechanisms for Distributed Agreement

- ❑ Also called “Distributed Consensus” algorithms
- ❑ The 4 key characteristics <https://pradeeplogathan.com/blockchain/consensus/>
 - Uniform agreement: No two nodes decide differently
 - Integrity: No node decides twice
 - Validity: If a node decides on value v , then v was proposed by some node
 - Termination: Every node that does not crash eventually decides on some value
- ❑ Given a cluster of N nodes and a set of proposals P_1 to P_m , every non-failing node will eventually decide on a single proposal P_x without the possibility to revoke that decision. All non-failing nodes will decide on the same P_x .

Overview

Byzantine fault tolerance-based, which is a more traditional approach based on rounds of votes.

Proof-based or leader-based consensus, whereby a leader is elected and proposes a final value



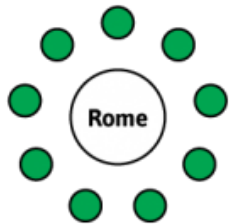
Figure

Classical Consensus Mechanisms (1)

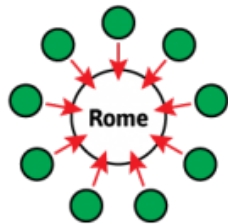
❑ Classical Consensus Models

- Crash failure models → honest nodes failing
- Byzantine Failure Tolerance (BFT)
 - State machine replication
- BFT General's Problem

HyperLedger (SOLO, Kafka mechanisms), Stellar



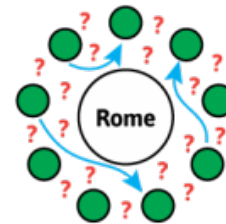
Imagine Rome being besieged by **nine armies**, each commanded by a (Byzantine) general.



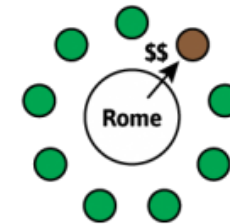
In order to launch a successful **attack** or retreat, all armies have to **do the same**, otherwise they will be decimated by Rome's armies.



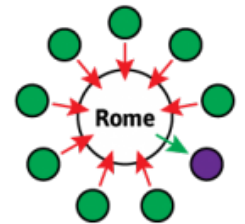
The decision to either **attack** or retreat is put up to a **vote**. Whichever option receives **more than 50%** of the votes, that's what the Generals will do (**retreat** in the example above).



Problem 1
The generals communicate by using **couriers**, who have to cross unknown areas controlled by the Romans, risking capture or their message becoming corrupt.



Problem 2
Each of the generals could be bribed by the Romans: **Traitorous Generals**.



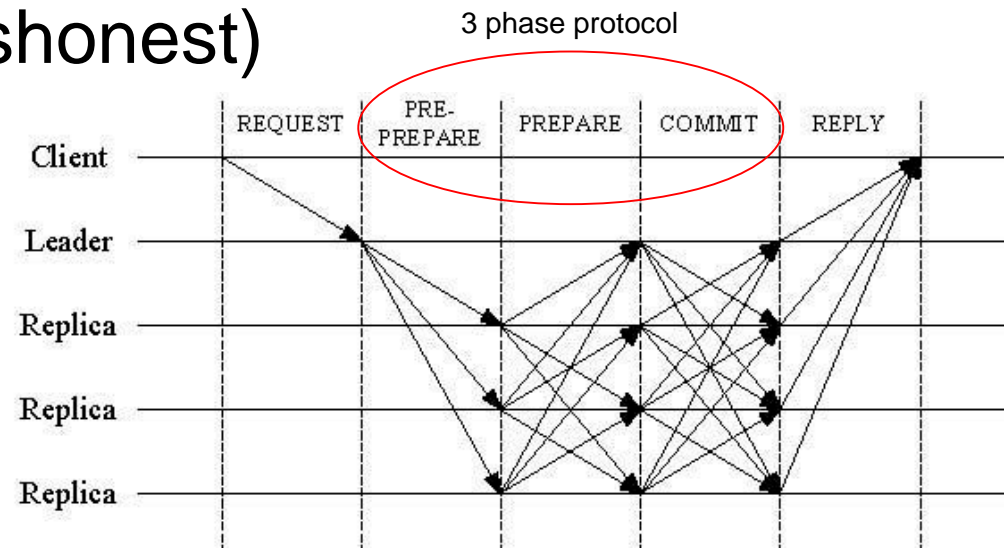
Problem 3
Any of the Generals can make the wrong decision, regardless of the vote: **Improperly Functioning Generals**.

Byzantine Fault Tolerance (BFT)

- ❑ Described as the capacity of a system to handle or survive unreliable situations and (all kinds of) failures
- ❑ **Practical BFT (PBFT)**: assume a small fraction of nodes as Byzantines (dishonest)

1. A **client** sends a request to invoke a service
2. The primary **leader** multicasts the requests to the replicas
3. Replicas execute the request and send a reply to the client
4. The client waits for $F+1$ replies from different replicas with the same result

- ❑ Other examples: XFT, HoneyBadger



n = Total # of nodes in network

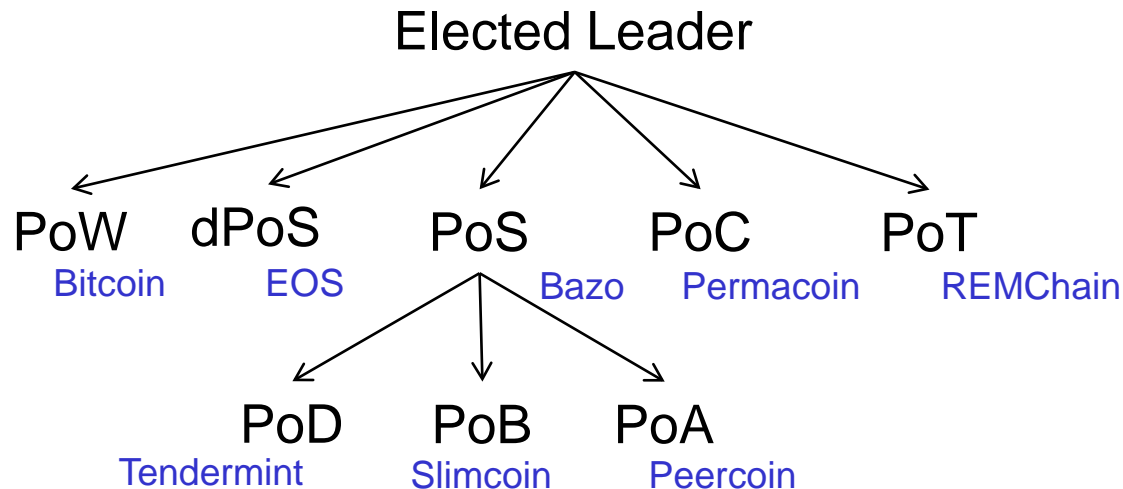
$$f = \frac{n-1}{3} \text{ (Max \# of faulty nodes)}$$

PBFT property

Classical Consensus Mechanisms (2)

□ Elected Leader Models

- Probabilistic elected leader in a
 - Lottery-like
 - Competition, or
 - Probabilistic algorithm



PoX: Proof-of-X, where X=

A: Authority

B: Burn

C: Capacity (storage)

D: Deposit

S: Stake

T: Trust

W: Work

d: delegated

Proof-of-Work (PoW)

- ❑ Set of transactions becomes available, a block is created by utilizing the following data:
 - Transaction(s), hash of previous block
 - Nonce (arbitrary number, can only be used once)
 - Other information (depending on BC)
- ❑ Hash of new block is calculated
- ❑ Checking performed once hash was computed
 - Hash is above the target value → Another miner may have found a suitable hash, block attached to local BC, but miner lost the lottery, otherwise nonce will be incremented, retry
 - Hash is below the target value → This miner won the lottery and the new block's hash determines the PoW result

Hash-based PoW (1)

- ❑ Key: One cannot compute an input from an output
 - To find a hash with N zeros at input start, requires 2^N computations, which proves computational work performed
 - Hashing an incrementing “nonce” as hash input, leads to zeros

in 3e-05 seconds, nonce = 0 yielded 0 zeros. value = 4c8f1205f49e70248939df9c7b704ace62c2245aba9e81641edf...

in 0.000138 seconds, nonce = 12 yielded 1 zeros. value = **0**5017256be77ad2985b36e75e486af325a620a9f29c54...

in 0.000482 seconds, nonce = 112 yielded 2 zeros. value = **00**ae7e0956382f55567d0ed9311cfd41dd2cf5f0a7137...

in 0.014505 seconds, nonce = 3728 yielded 3 zeros. value = **000**b5a6cfc0f076cd81ed3a60682063887cf055e47b...

in 0.595024 seconds, nonce = 181747 yielded 4 zeros. value = **0000**af058b74703b55e27437b89b1ebcc46f45ce55d6....

in 3.491151 seconds, nonce = 1037701 yielded 5 zeros. value = **00000**e55bd0d2027f3024c378e0cc511548c94fbed0e....

in 32.006105 seconds, nonce = 9913520 yielded 6 zeros. value = **000000**77a77854ee39dc0dc996dea72dad8852afbde6....

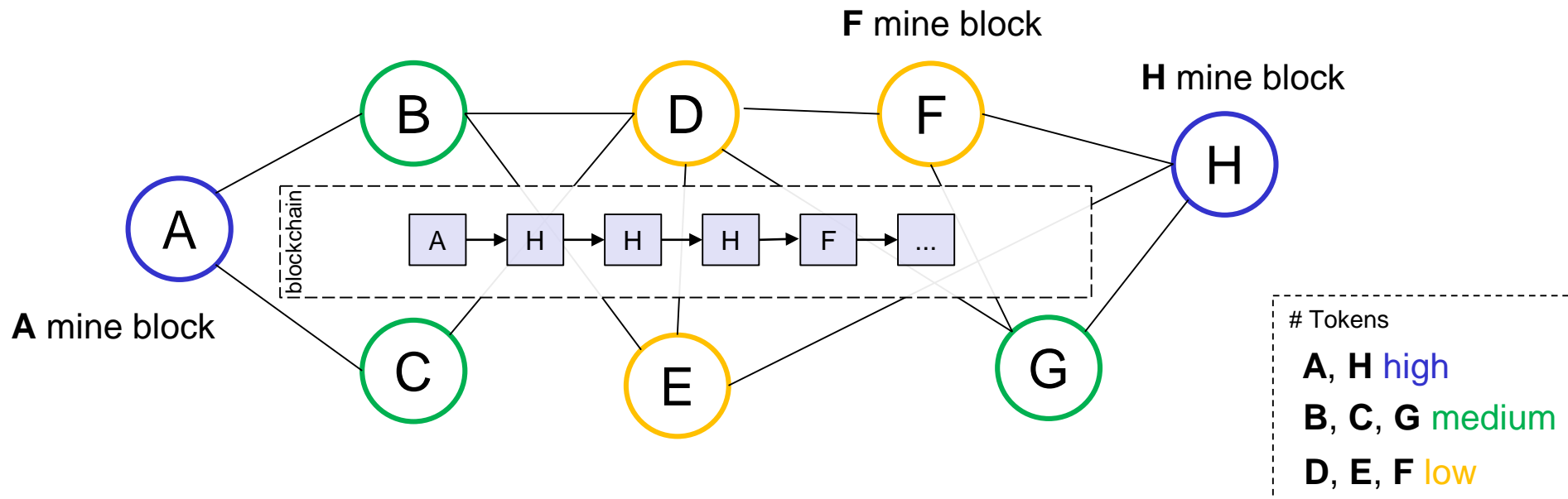
in 590.89462 seconds, nonce = 186867248 yielded 7 zeros. value = **0000000**225060b16117b23dbea9ce6be86ac439d....

in 4686.171007 seconds, nonce = 1424462909 yielded 8 zeros. value = **00000000**2dd743724609a9f57260e2492908d....

- ❑ Distributed game sets the difficulty N of the game
- ❑ Players accumulate points by creating blocks
 - Hashing the previous block, finding a hash of the new block with enough zeros, and transmitting this block to everyone

Proof-of-Stake – PoS (1)

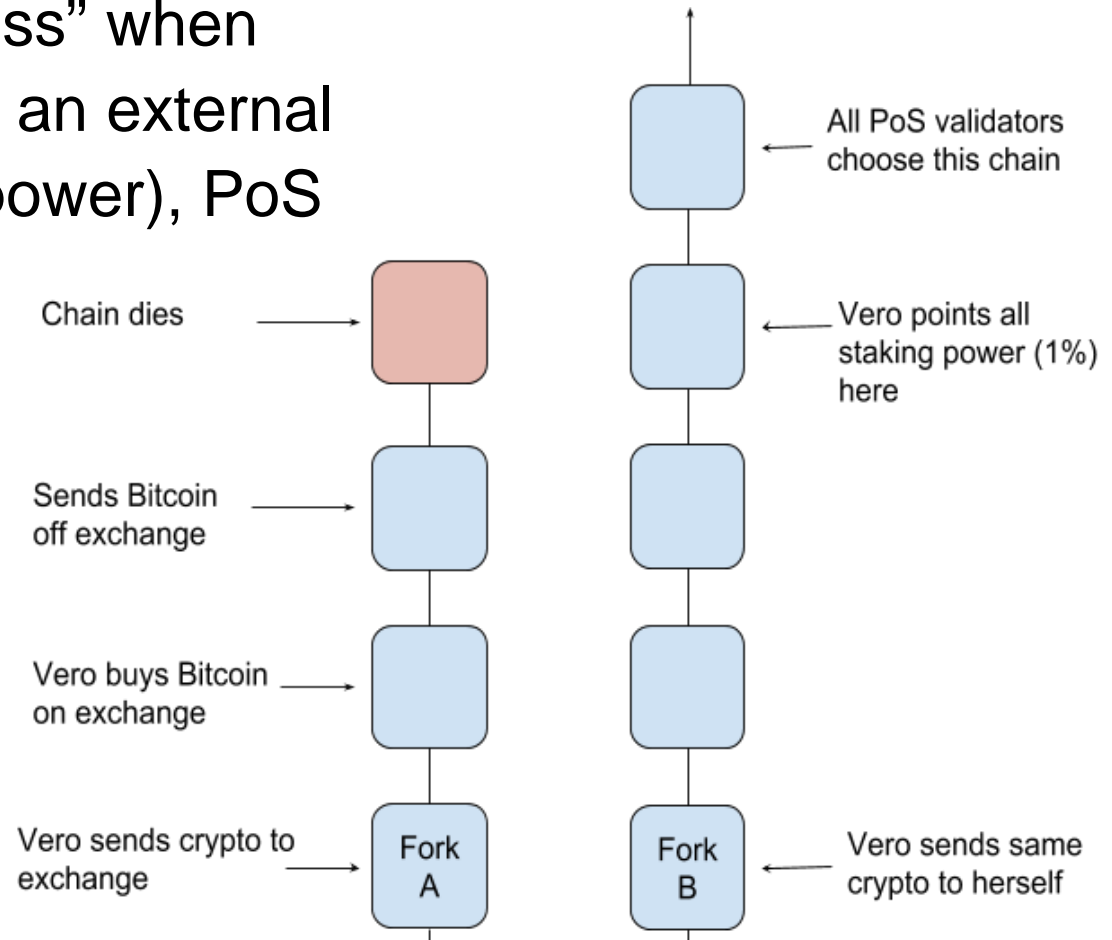
- ❑ Blocks are “mined” according to the amount of “tokens” he or she holds:
 - The higher is the number of tokens (coins) at stake, the higher is the “mining power”
 - Nodes gets the block reward as incentive



Proof-of-Stake – PoS (2)

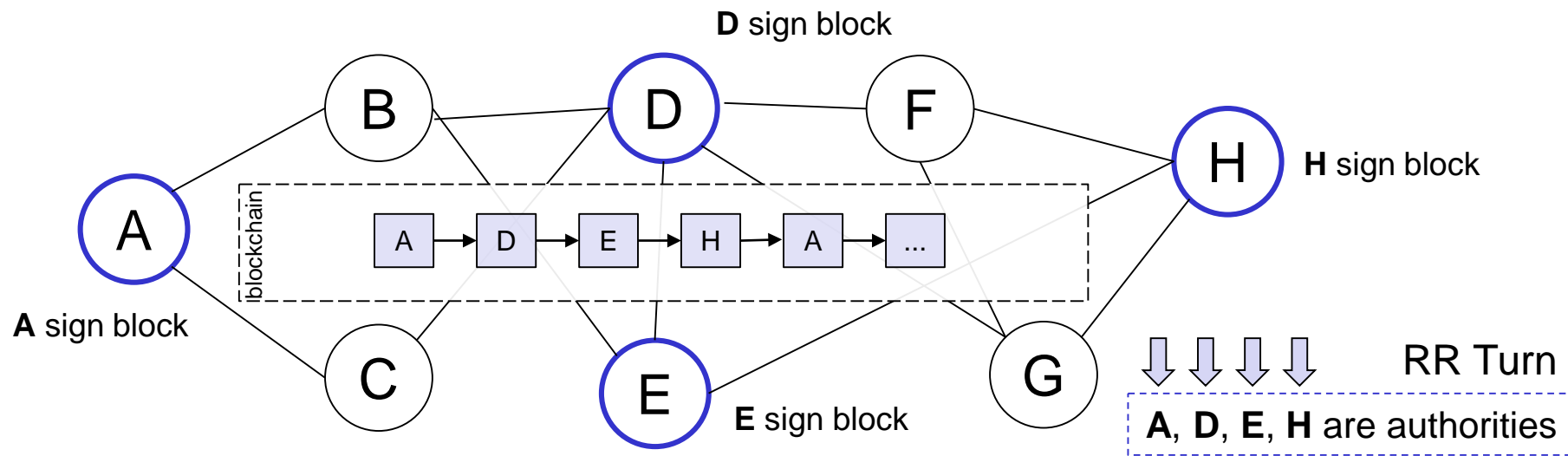
❑ Nothing at stake issue:

- Creating forks is “costless” when someone is not burning an external resource (e.g., mining power), PoS alone is “unworkable”



Proof-of-Authority (PoA)

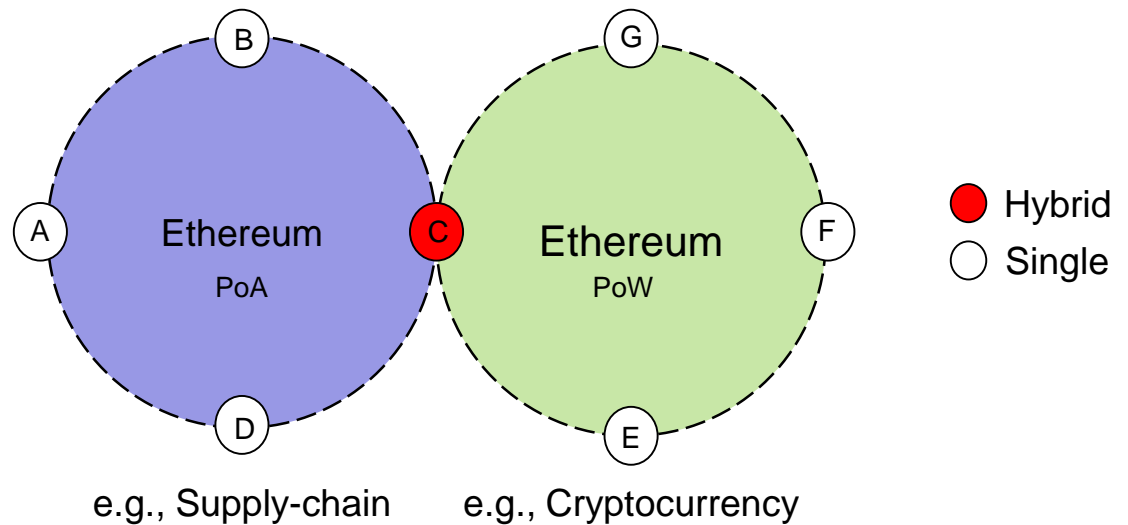
- ❑ PoA is a modified form of PoS where instead of stake a validator's **identity** performs the role of **stake**
- ❑ Authorities (nodes) are allowed to create new blocks
 - Clique (practical implementation) of PoA
 - Requires $N/2+1$ (more than 50%) of signers to be honest
 - Authorities sign new blocks in a Round-robin (RR) fashion



Hybrid Consensus

□ Hybrid Consensus Models

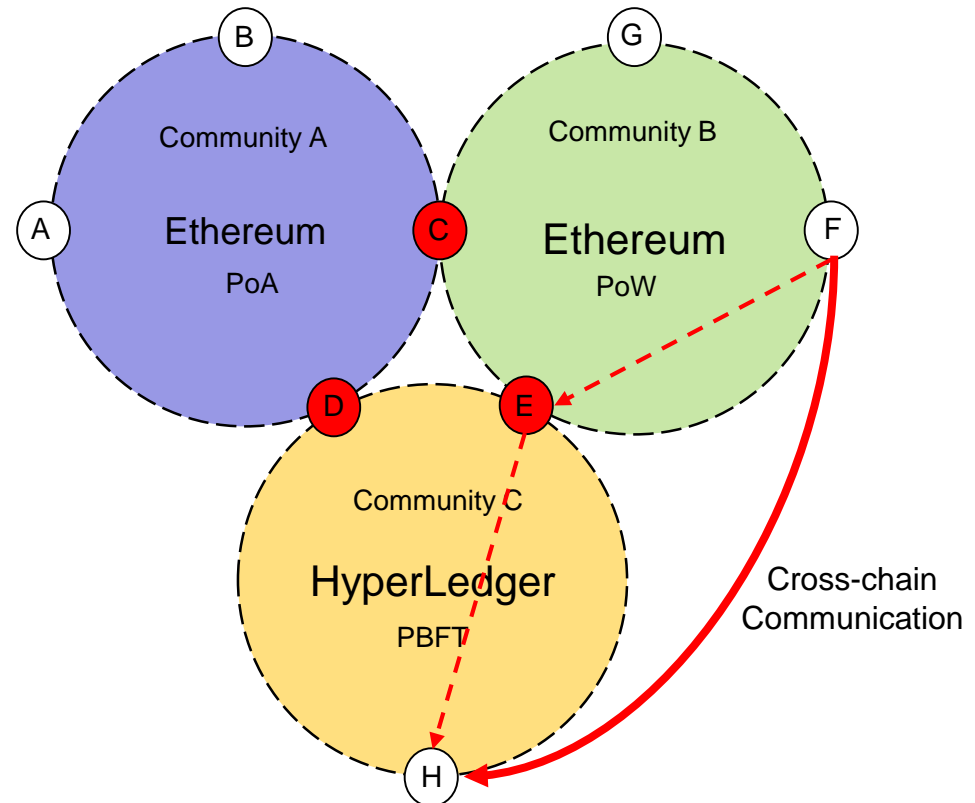
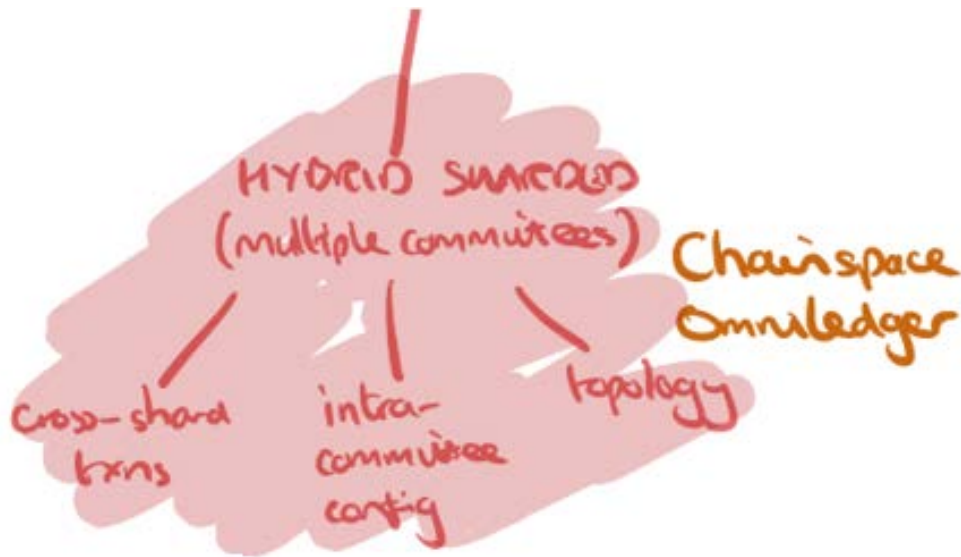
- Using a single consensus has many limitations
 - Combine different consensus mechanisms



Hybrid Sharding

□ Hybrid Sharding

- System can be organized into shards (communities)
- Cross-chain communications



Comparison of Consensus Mechanisms

Incomplete!

Mecha-nism	Security Level	Depending on	Scalability	Remarks
BFT	“Reasonable” Leader pre-elected 51% failure	-	Medium	Trust in pre-election
dBFT	“Reasonable” Set of leaders pre-elected	-	Medium	Trust in set of leaders
PoW	High 51% attack	Hashes	Controversial	Energy consumption high, needed to ensure high security level (by design)
PoS	Unknown “Nothing-at-Stake”	PoW-based “stake”	Under discussion	“Costless” forking, thus, measurable assets needed
PoA	Identity-based	PoS, PoW	Under discussion	Authorities required
Shards	Unknown	Any PoX	Unknown	Communities, interoperability

Blockchain Adoption

Choosing a Blockchain

Posted November 22, 2015 by [Gideon Greenspan](#) in [Private blockchains](#).

Avoiding the pointless blockchain project

How to determine if you've found a real blockchain use case

Do you need a Blockchain?

The Use of Blockchains: Application-Driven Analysis of Applicability

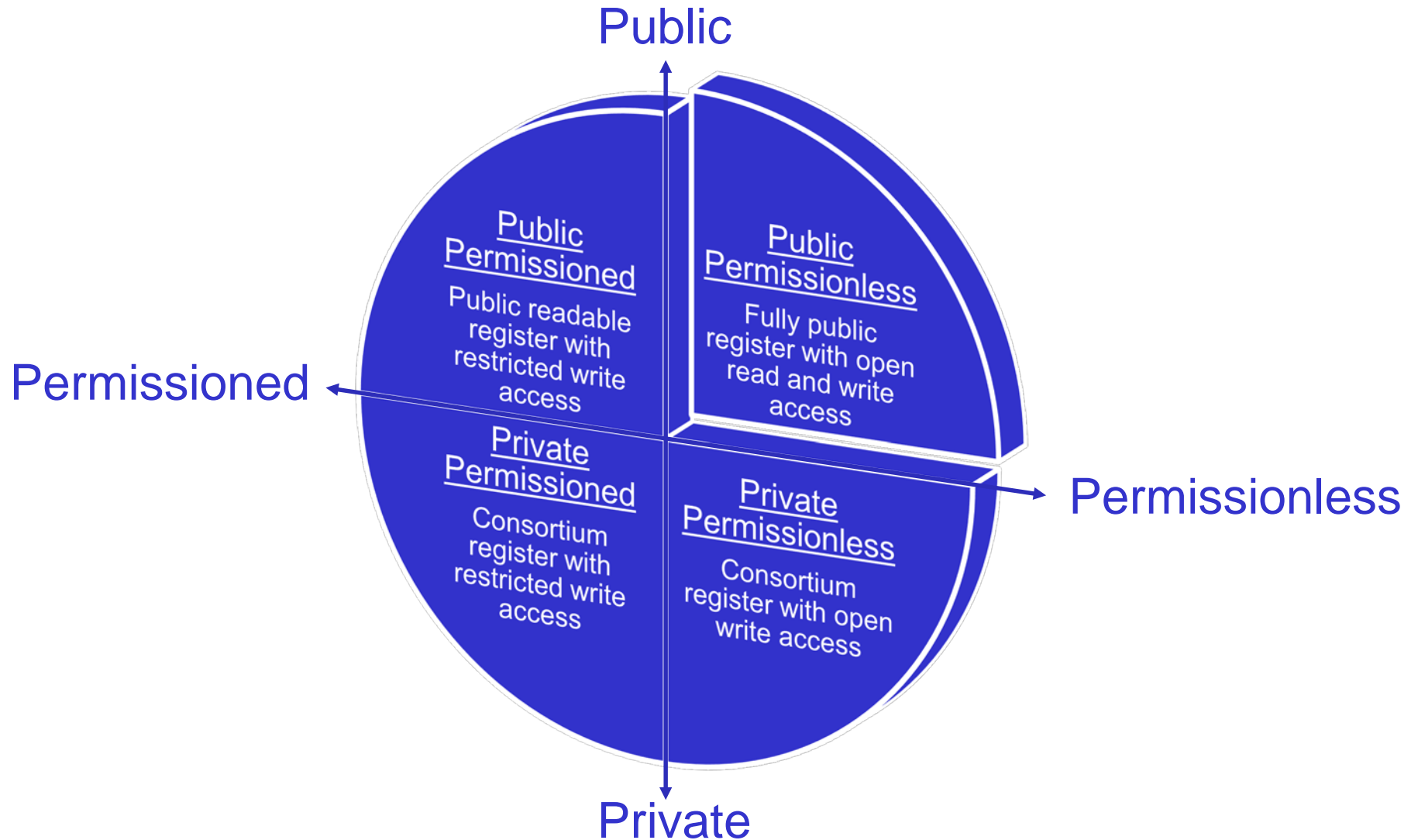
Bruno Rodrigues, Thomas Bocek, Burkhard Stiller

Communication Systems Group (CSG), Department of Informatics (IfI), Universität Zürich (UZH),
Zürich, Switzerland

Karl Wüst
Department of Computer Science
ETH Zurich
karl.wuest@inf.ethz.ch

Arthur Gervais
Department of Computing
Imperial College London
a.gervais@imperial.ac.uk

Deployment Models

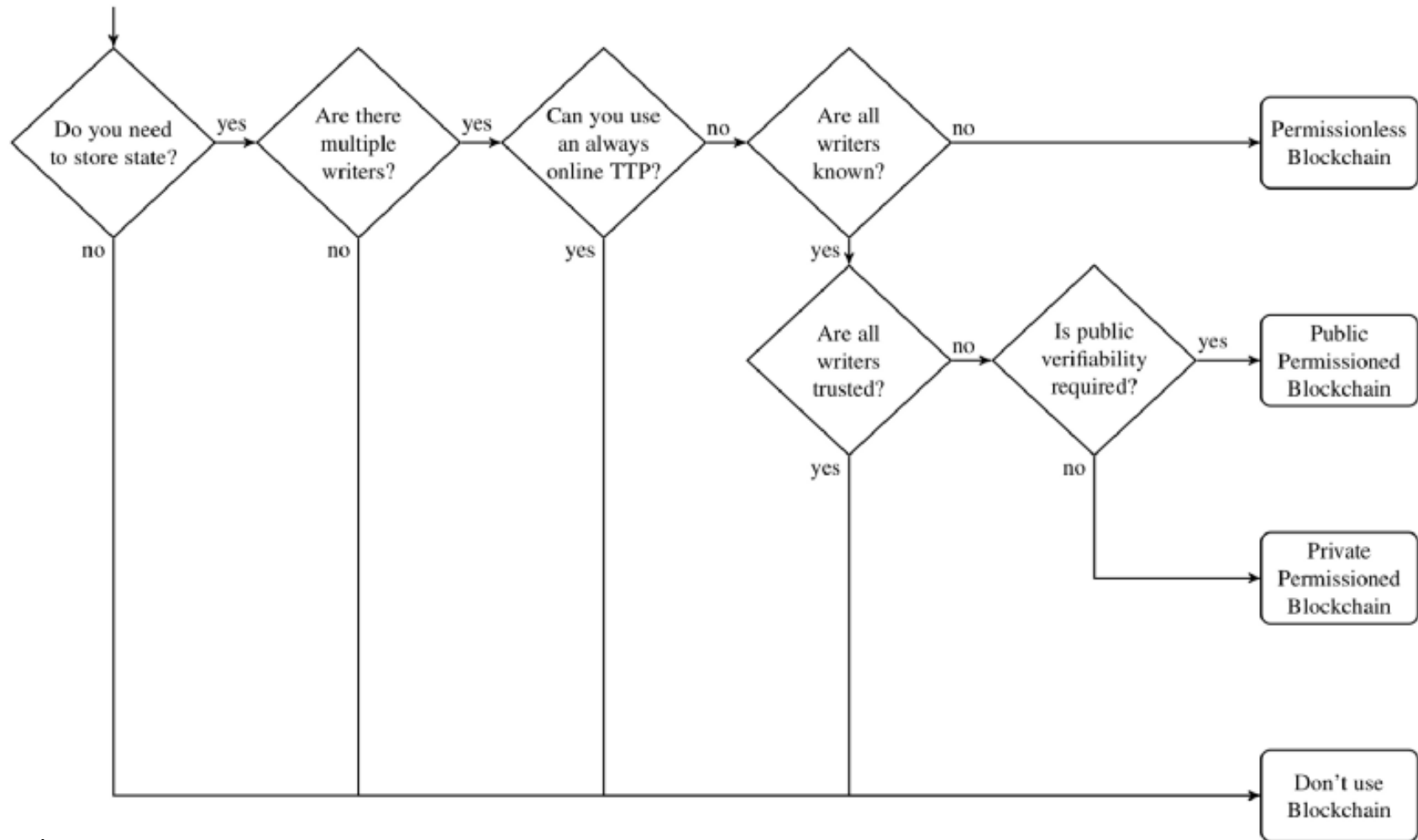


G. Greenspan (2015)

Key Points	When to use BC	Traditional DBs
Database	Shared	Centralized, Shared
Multiple writers	Multiple writers	Single or multiple
Absence of trust	Database with multiple non-trusting writers	Trust
Disintermediation	No trusted intermediaries	Trusted intermediary
Transaction interaction	There is a dependency between transactions	Trust the intermediary to mediate interactions
Set the rules	Clear rules applied to all writers	Different rules based on roles/groups of writers
*Pick your validators	Trust in the validation scheme (single entity or democratic)	
*Back your assets	Translation of digital assets into the real world	

***Recommendations**

K. Wüst, A. Gervais (2018)



Based on
K. Wüst, A. Gervais

K. Wüst, A. Gervais (2018) – Cont.

- Performance and scalability requirements impacts of alternative BC solutions and data bases in comparison

	Permissionless Blockchain	Permissioned Blockchain	Central Database
Throughput	Low	High	Very High
Latency	Slow	Medium	Fast
Number of readers	High	High	High
Number of writers	High	Low	High
Number of untrusted writers	High	Low	0
Consensus mechanism	Mainly PoW, some PoS	BFT protocols (e.g. PBFT [6])	None
Centrally managed	No	Yes	Yes

BFT: Byzantine Fault Tolerance
PBFT: Practical Byzantine Fault Tolerance

Application Trade-offs




(B. Rodrigues, T. Bocek, B. Stiller, 2018)

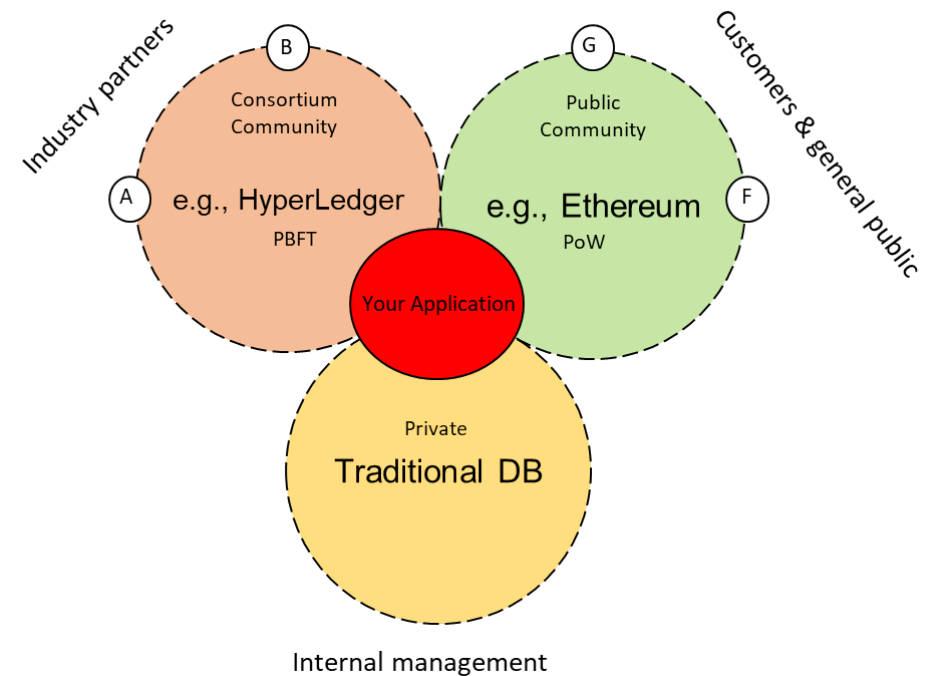
- ❑ Based on Blockchain characteristics:
 - Performance vs Reliability
 - BC offers slow throughput but more robustness than traditional DBs
 - Confidentiality vs Transparency
 - More transparency (trust) and less confidentiality
 - Distributed vs centralized control
 - No central authority (PoW) or trusted nodes (PBFT)
- ❑ Limited storage
- ❑ Unknown regulations
 - Different countries, different regulations
- ❑ Lack of standards
 - Blockchain 4.0 target

Distributed vs. Centralized Control

- ❑ Distributed control based on **elected** leader (e.g., PoW)
- ❑ Partially based on **selected** leaders (e.g., PoA, PBFT)
- ❑ Centralized Control based on **trust** (e.g., traditional databases)

- ❑ Multiple possibilities
 - At the same time...

Company	Model	Control
		Distributed
		Partial
		Centralized



Mapping Tradeoffs to Blockchain Types

	Public Permissionless	Public Permissioned	Private Permissionless	Private Permissioned
Transparency	World visibility	World visibility	Community visibility	Role-based visibility
Control	Distributed due to the election process	Distributed but validators are defined in a selection process	Distributed but validators are defined in a selection process	Centralized based on trusted nodes
Reliability	Full replication (light nodes always rely on full nodes)	Full or partial replication (possible to define super nodes)	Full or partial replication (possible to define super nodes)	Full or partial replication (master- slave)
Performance	Slow due the consensus and replication models	Intermediate depending on consensus and replication models	Intermediate depending on consensus and replication models	Fast because its mostly centrally managed

Part II - Smart Contracts

Smart Contracts

- ❑ A **Smart Contract** (SC) may reside inside transactions
 - Executed & validated on every node upon persisting that block
 - *E.g.*, for **Bitcoins** (blockchain-based cryptocurrency) SCs specify how to withdraw, escrow, refund, or transfer BTC from A to B
- ❑ SCs first mentioned in 1996:

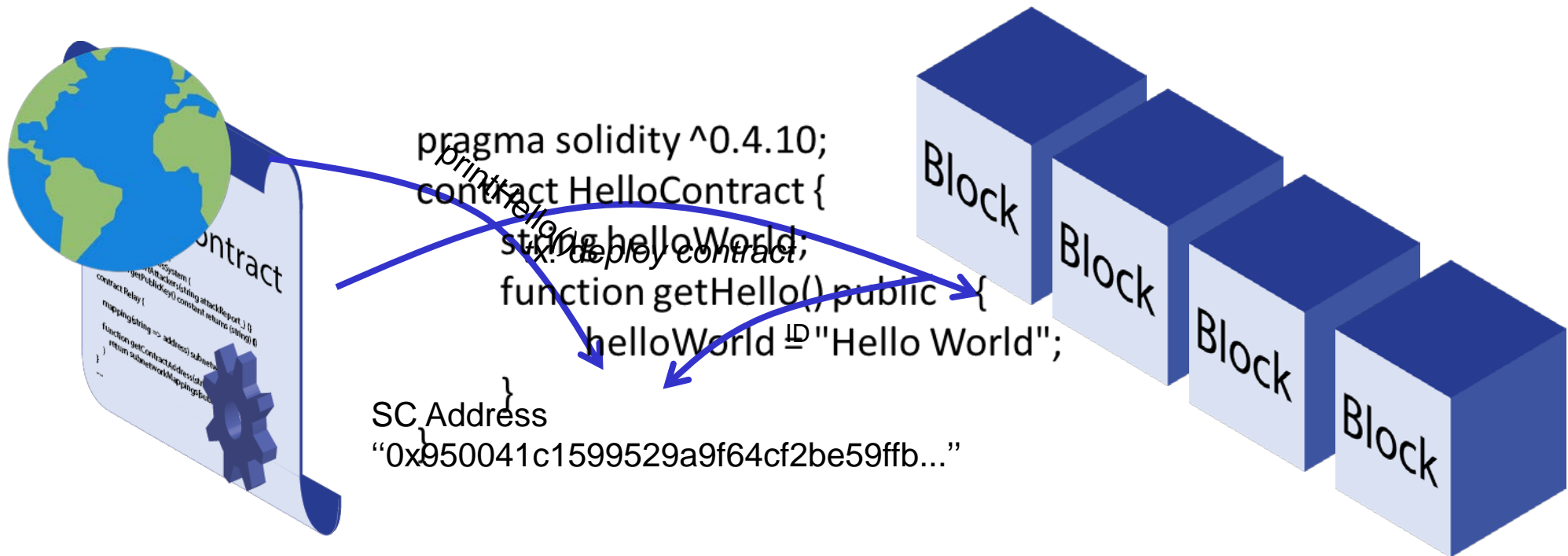
“Active” database!

A smart contract is a **computerized transaction protocol** that executes the terms of a contract. The general objectives of [a] smart contract[’s] design are to satisfy common contractual conditions (such as payment terms, liens, confidentiality, and even enforcement), minimize exceptions both malicious and accidental, and **minimize the need for trusted intermediaries**. Related economic goals include lowering fraud loss, arbitrations and enforcement costs, and other transaction costs.

- ❑ Smart contracts **alone** are not “smart”
 - They need an **infrastructure** (“technology”)
 - A **blockchain** forms **the ideal, distributed basis** for SCs
- ❑ The **legal relevance** of “coded”, more general contracts?

N. Szabo

Ethereum/Solidity



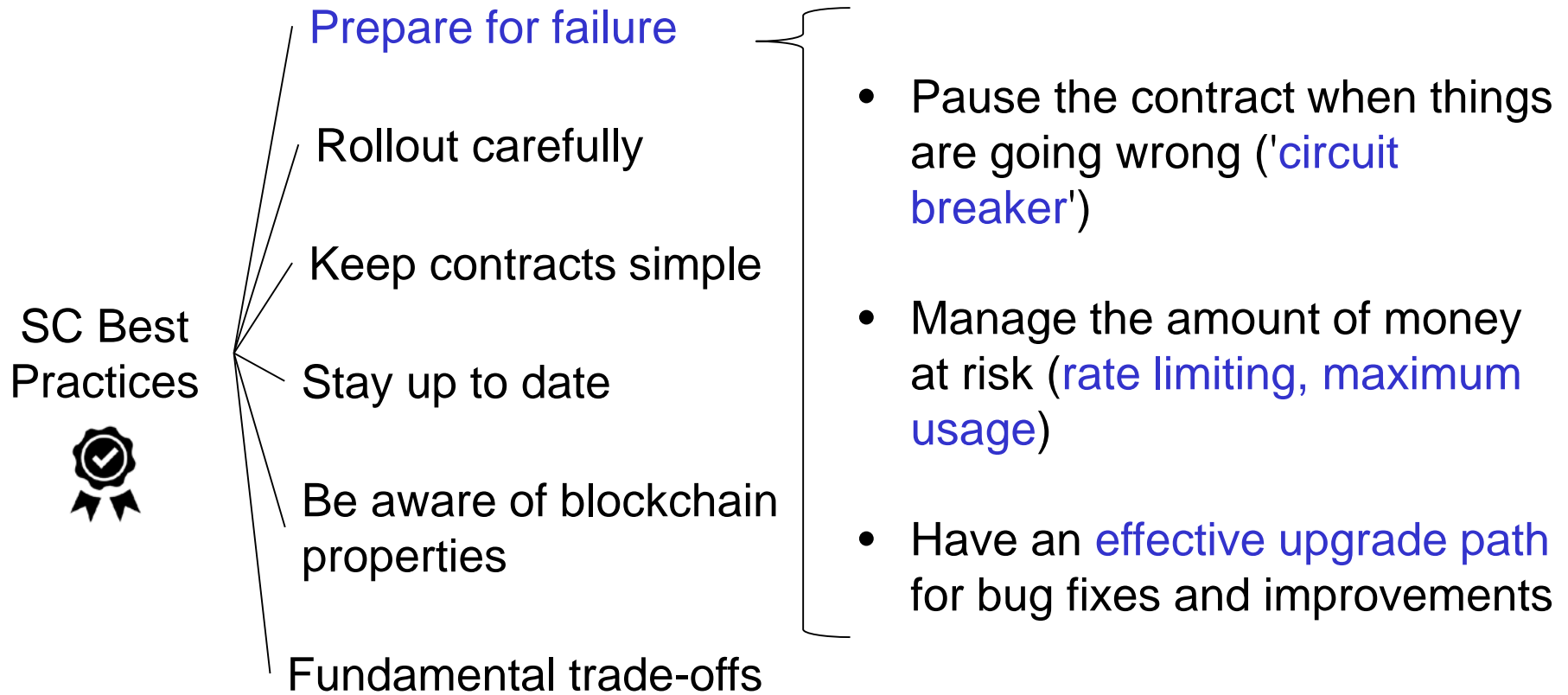
Smart Contract Best Practices

- ❑ Blockchain is a **relatively new** and most implementations are **experimental**
- ❑ SC programming requires a different mindset:
 - **Changes are not possible** once SC is deployed
 - Cost of **failures** can be high (e.g., DAO)
 - Still.... not immune to **vulnerabilities**
- ❑ Best practices are essential!!
 - Code security, efficiency, readability, ...

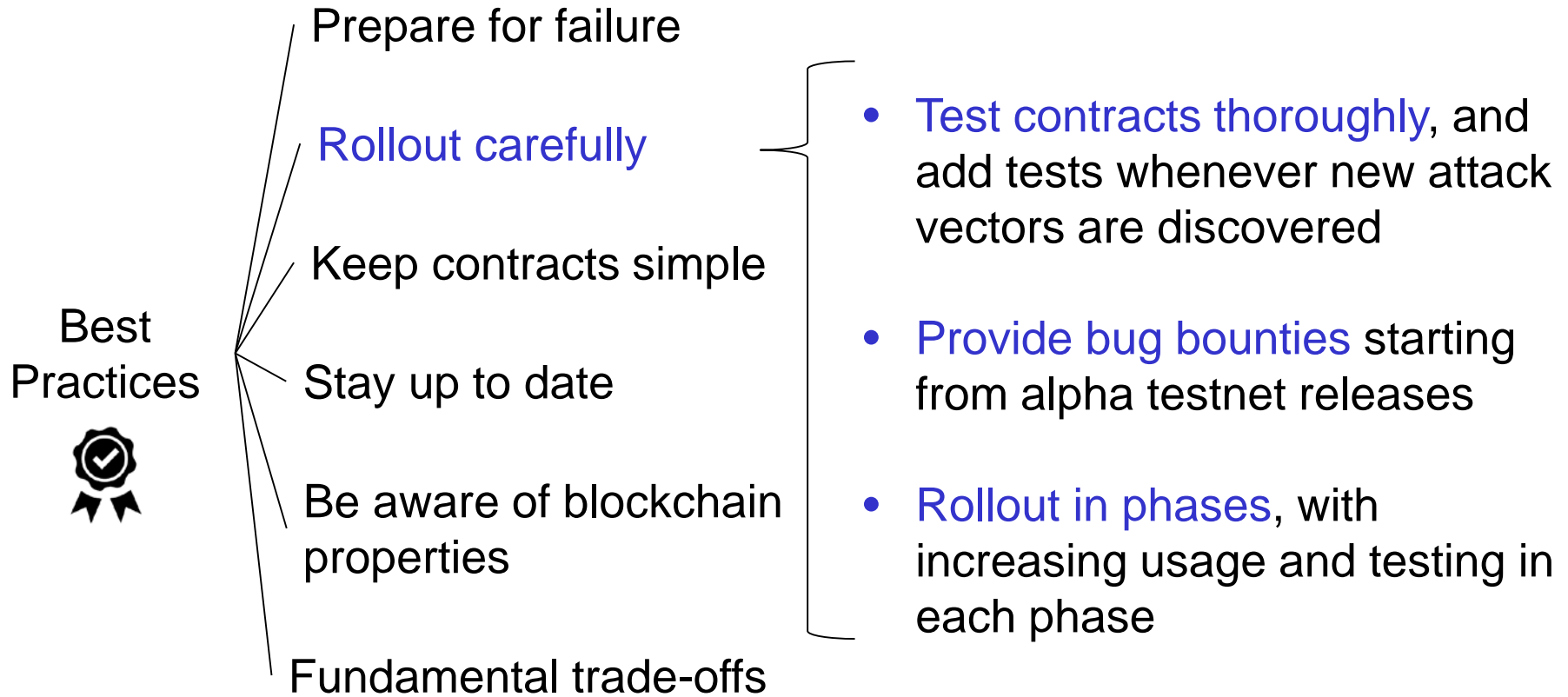


https://consensys.github.io/smart-contract-best-practices/general_philosophy/

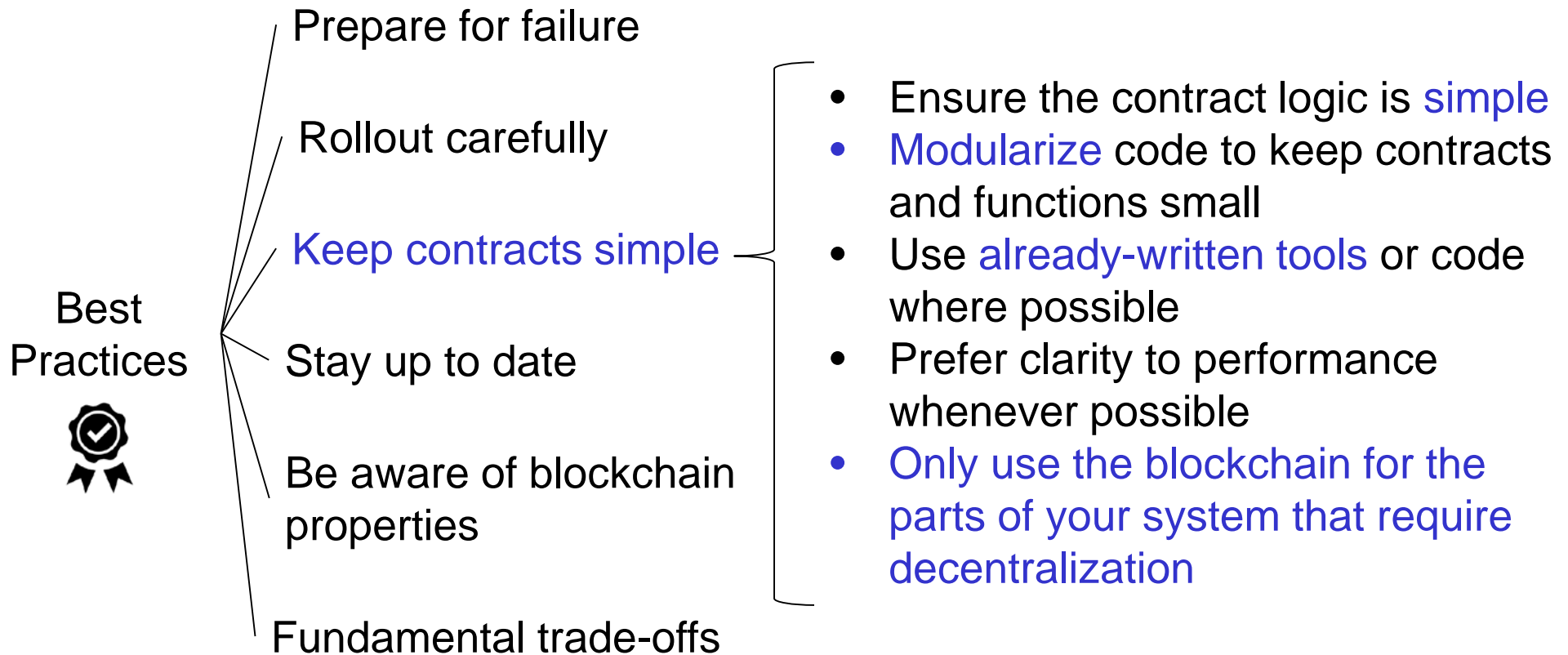
Prepare for failure



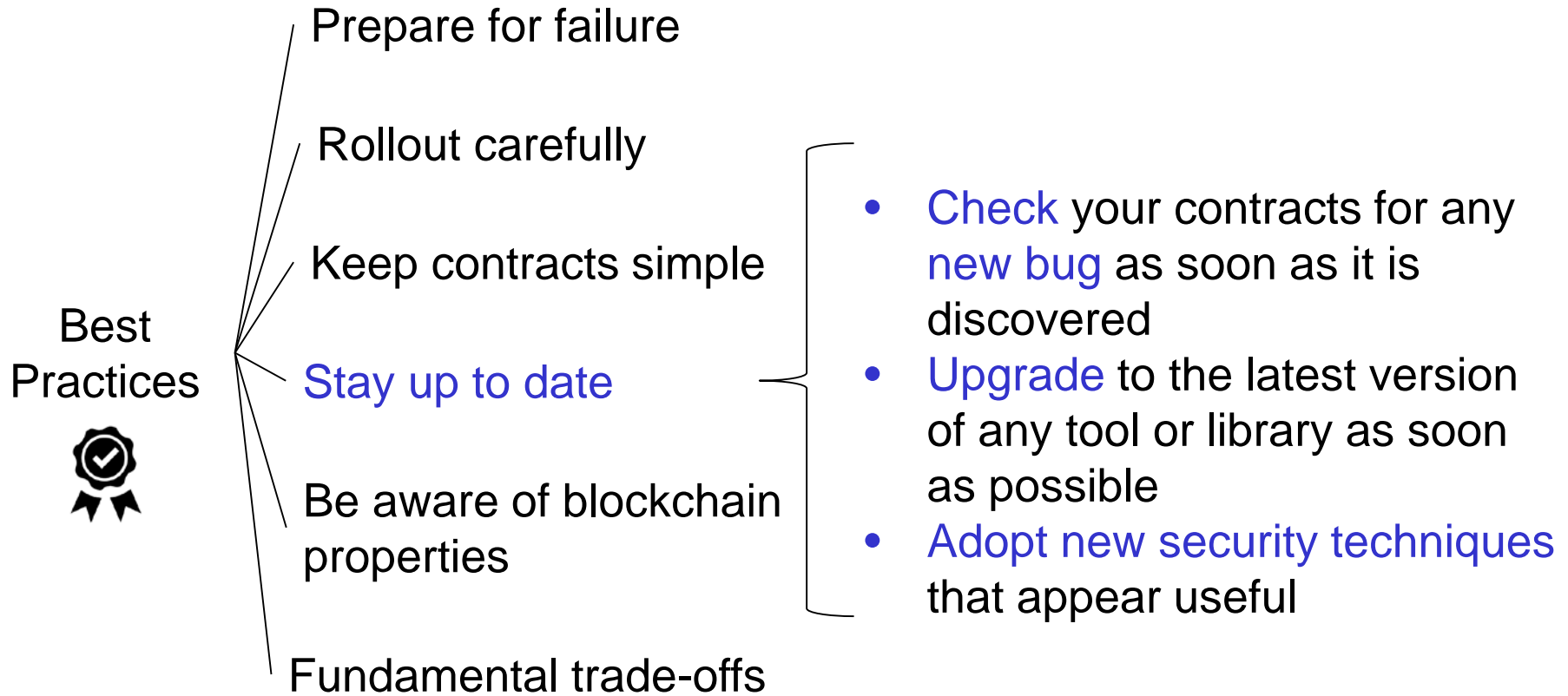
Rollout Carefully



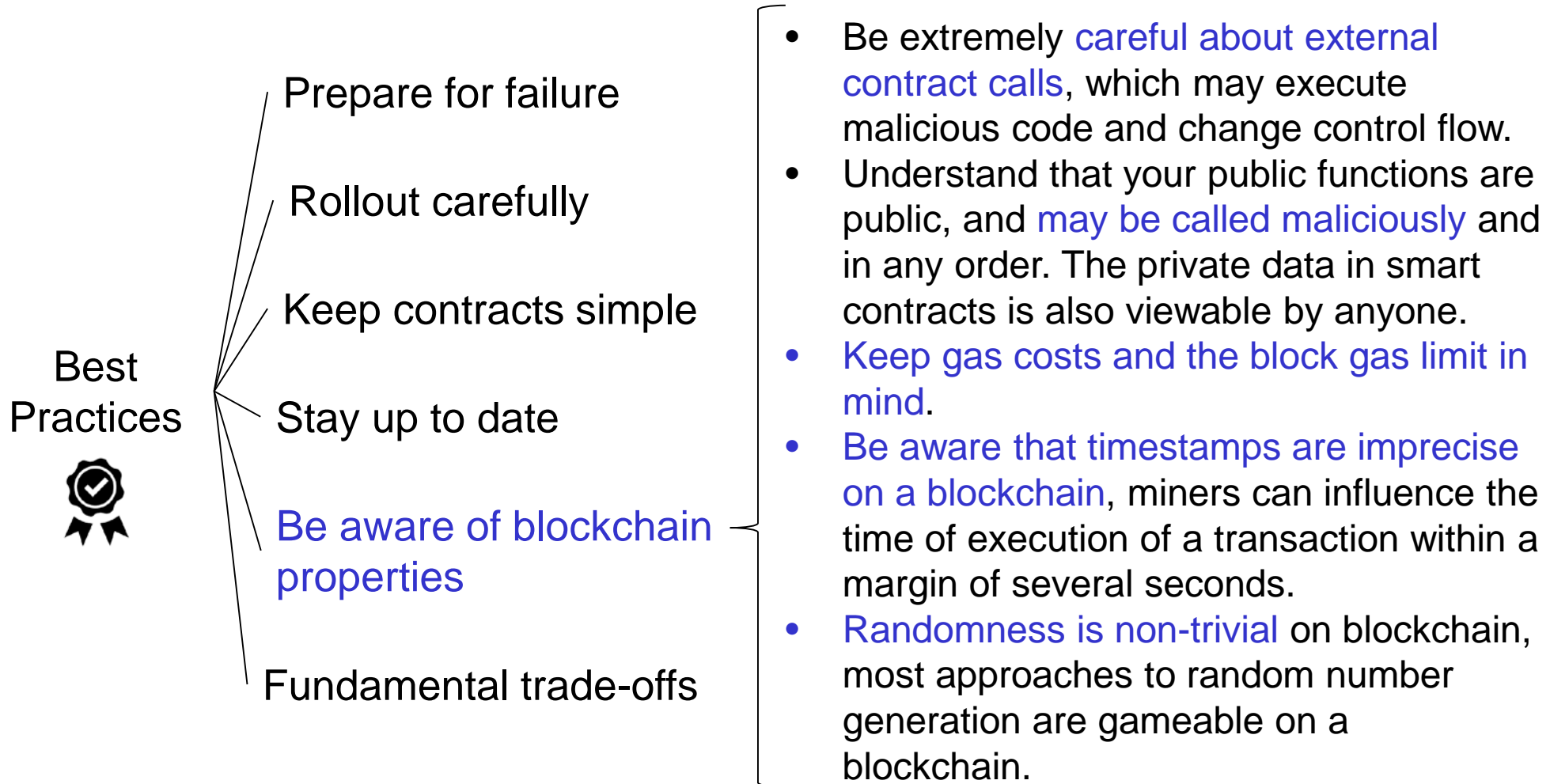
Keep Contracts Simple



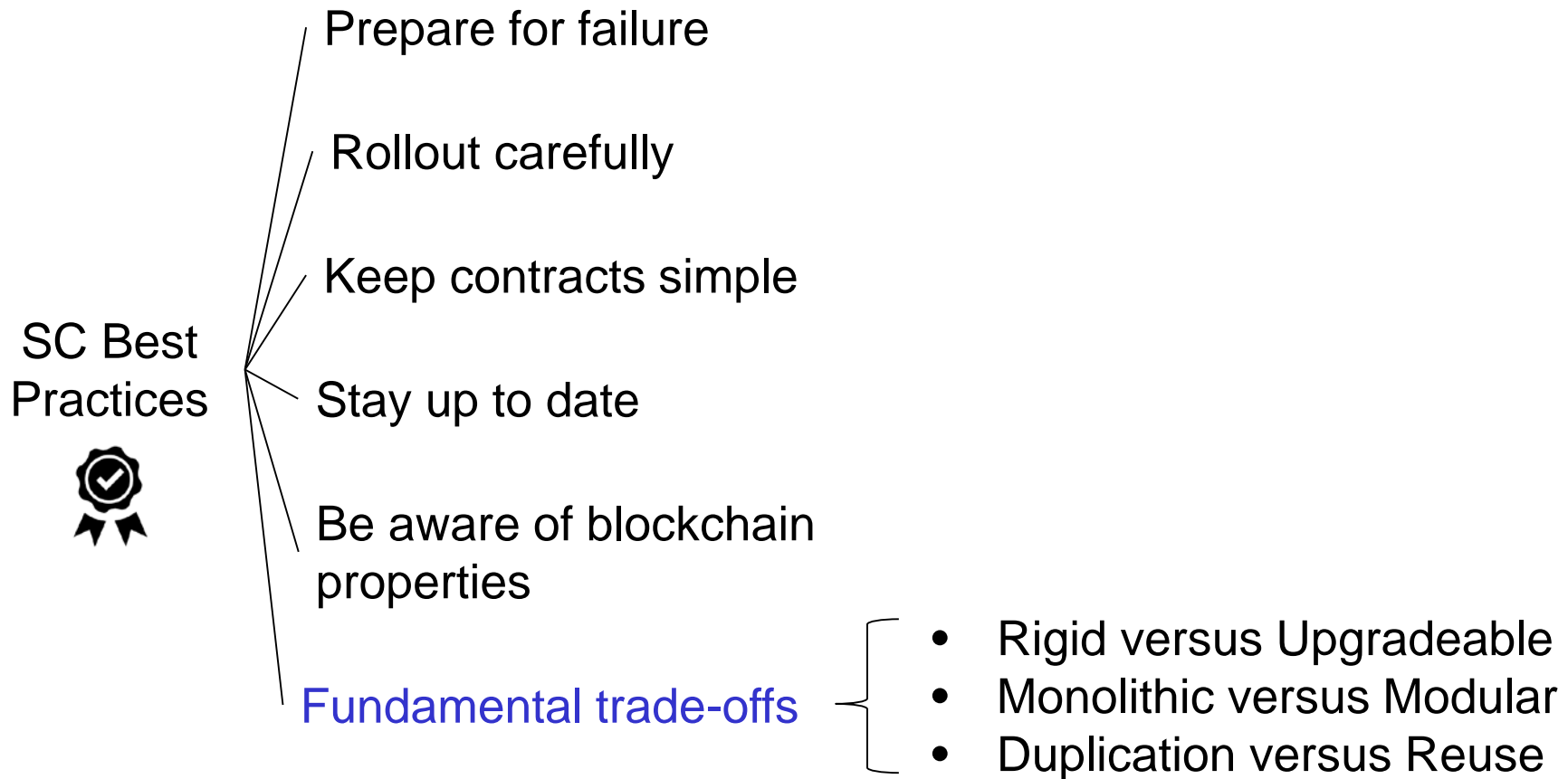
Stay Up to Date



Awareness of BC Properties



Fundamental Trade-offs



Tools for Security Visualization

❑ Surya:

- Visual outputs and information about the contracts' structure. Also supports querying the function call graph.

❑ Solgraph:

- Generates a DOT graph that visualizes function control flow and highlights potential security vulnerabilities.

❑ EVM Lab

- Rich tool package to interact with the EVM. Includes a VM, Etherchain API, and a trace-viewer.

❑ ethereum-graph-debugger

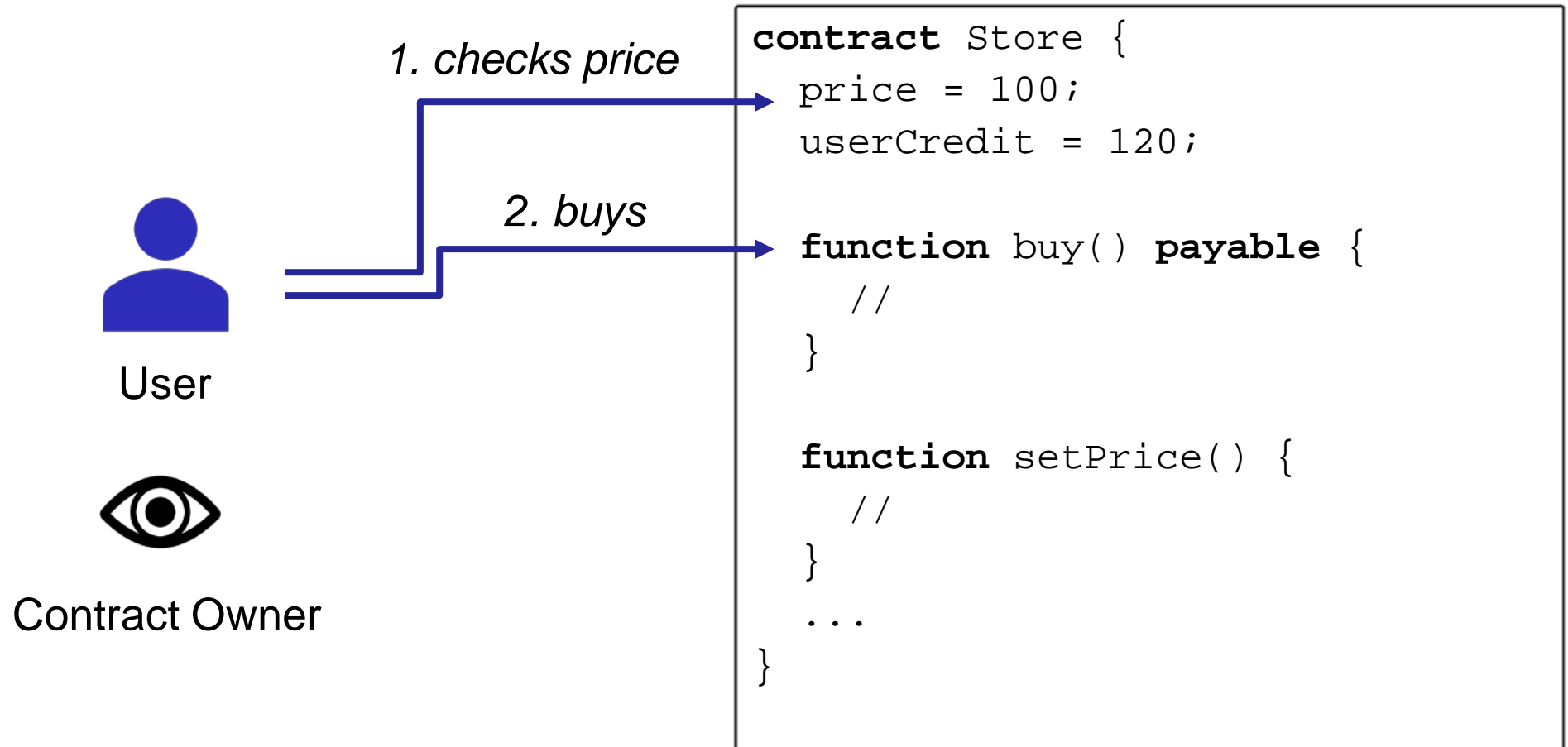
- A graphical EVM debugger. Displays the entire program control flow graph.

Smart Contract Security Examples

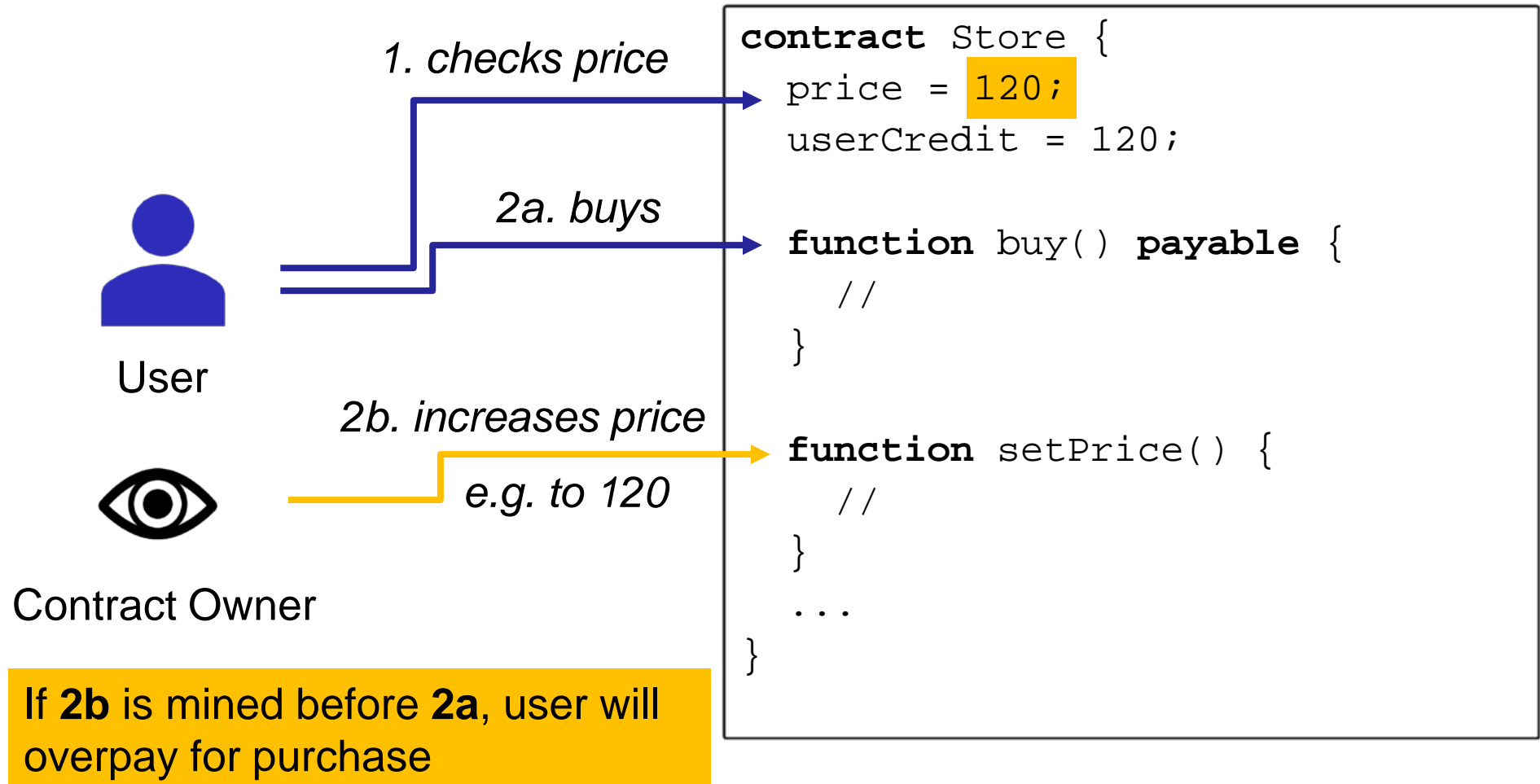
- ❑ **Transaction Ordering**
 - > Blockchain Shop
- ❑ **Reentrancy Attacks**
 - > Good ATM | Bad ATM

Source: James Chiang

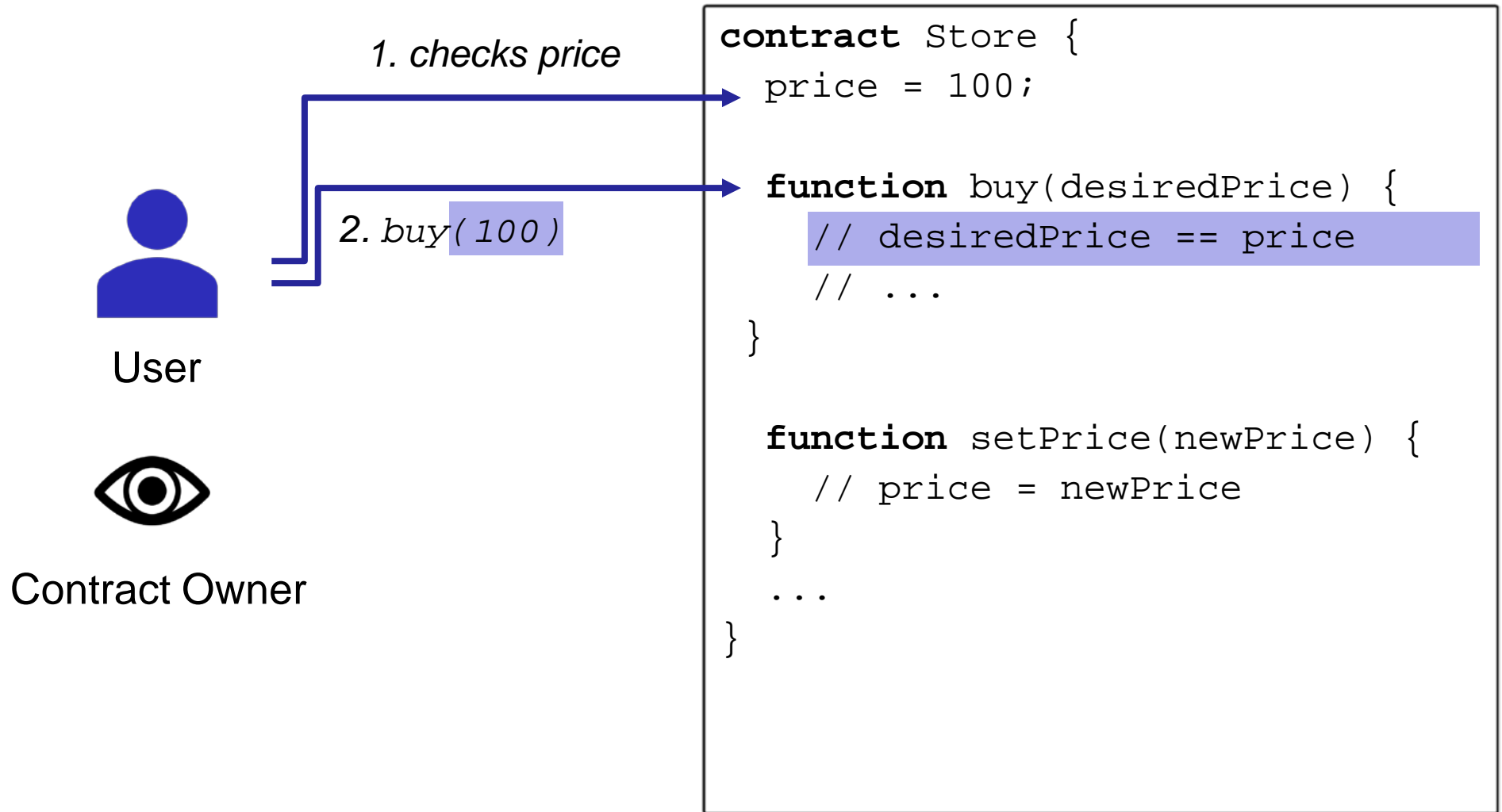
Transaction Ordering



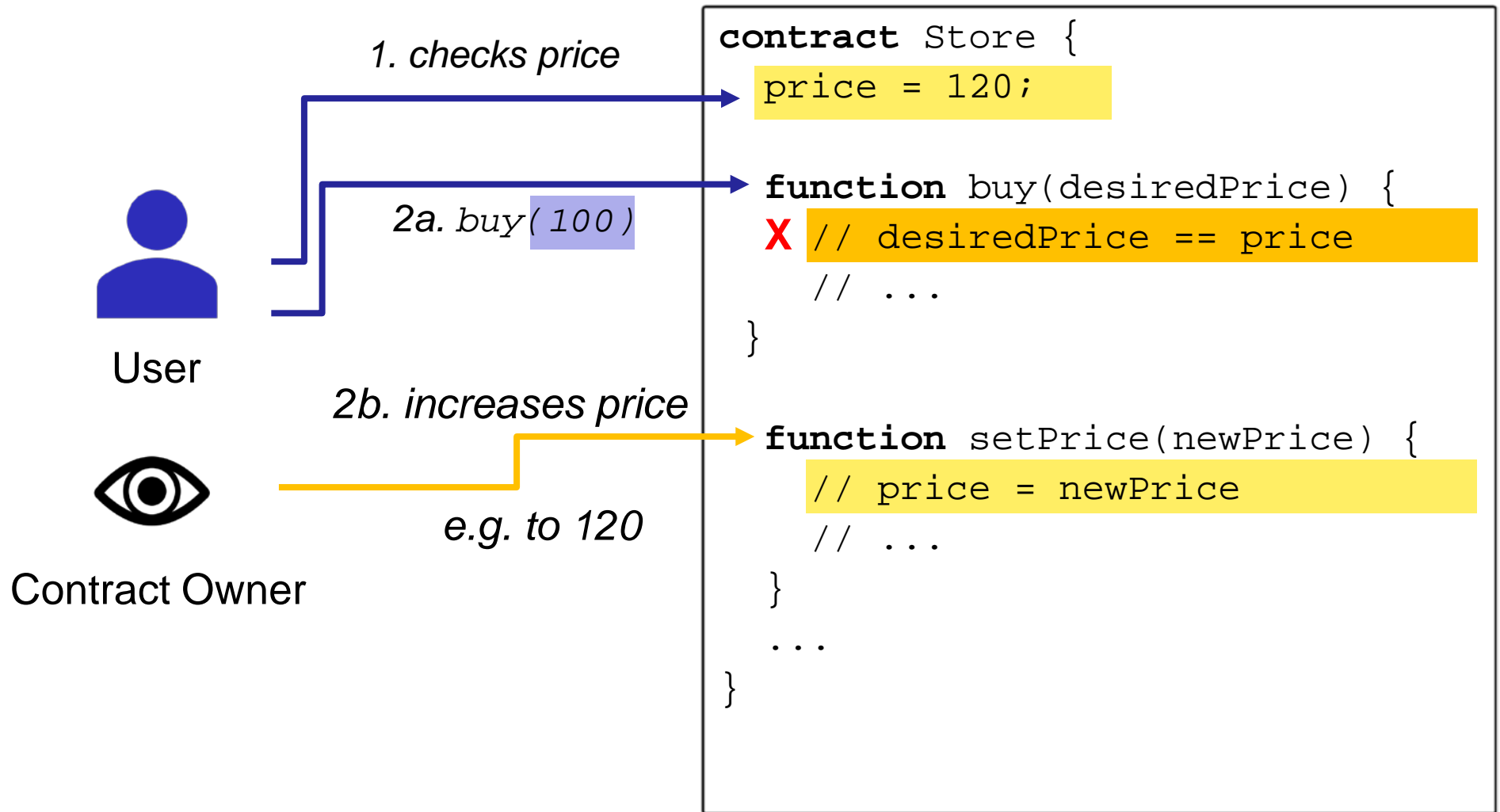
Transaction Ordering



Transaction Order Guard



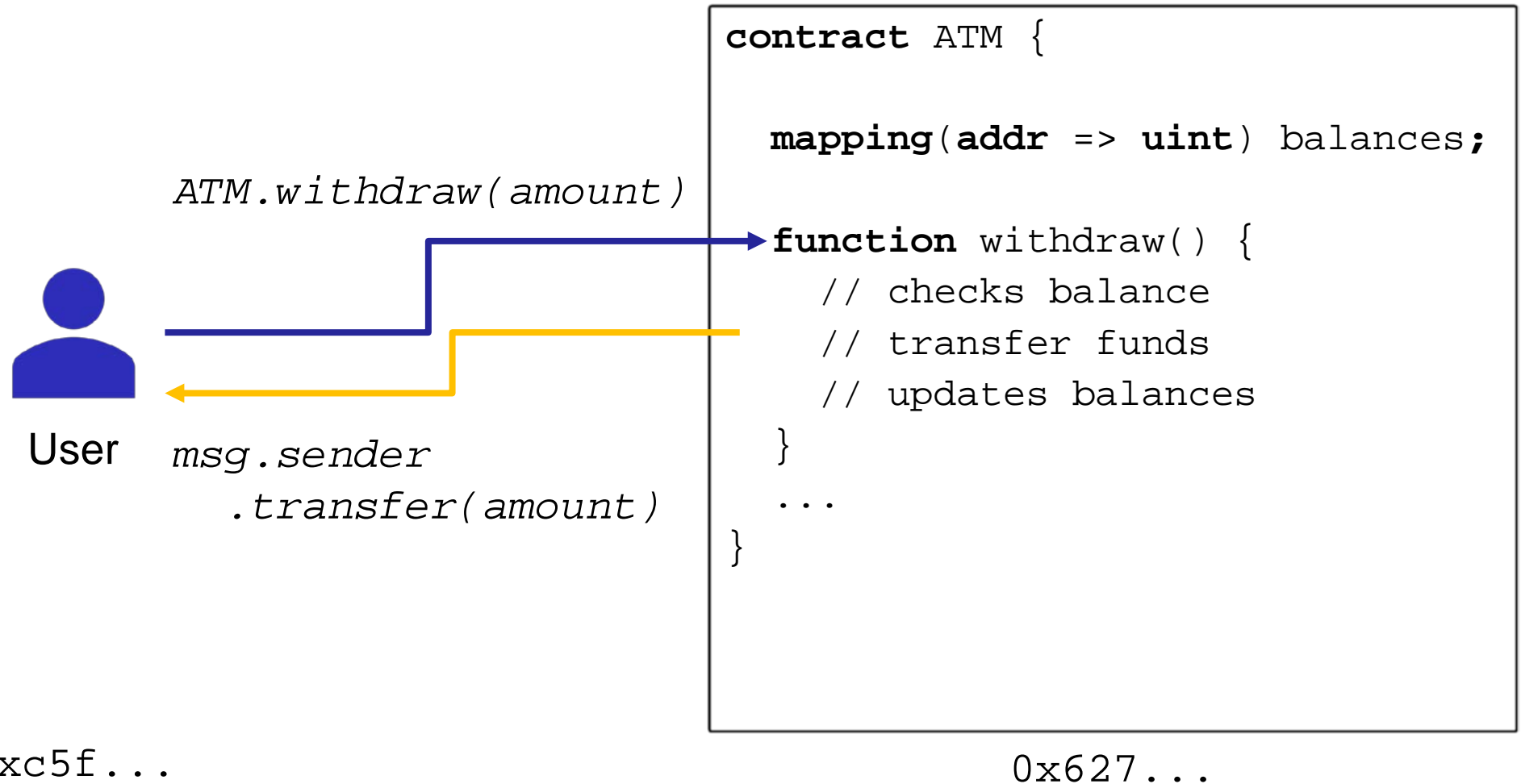
Transaction Order Guard



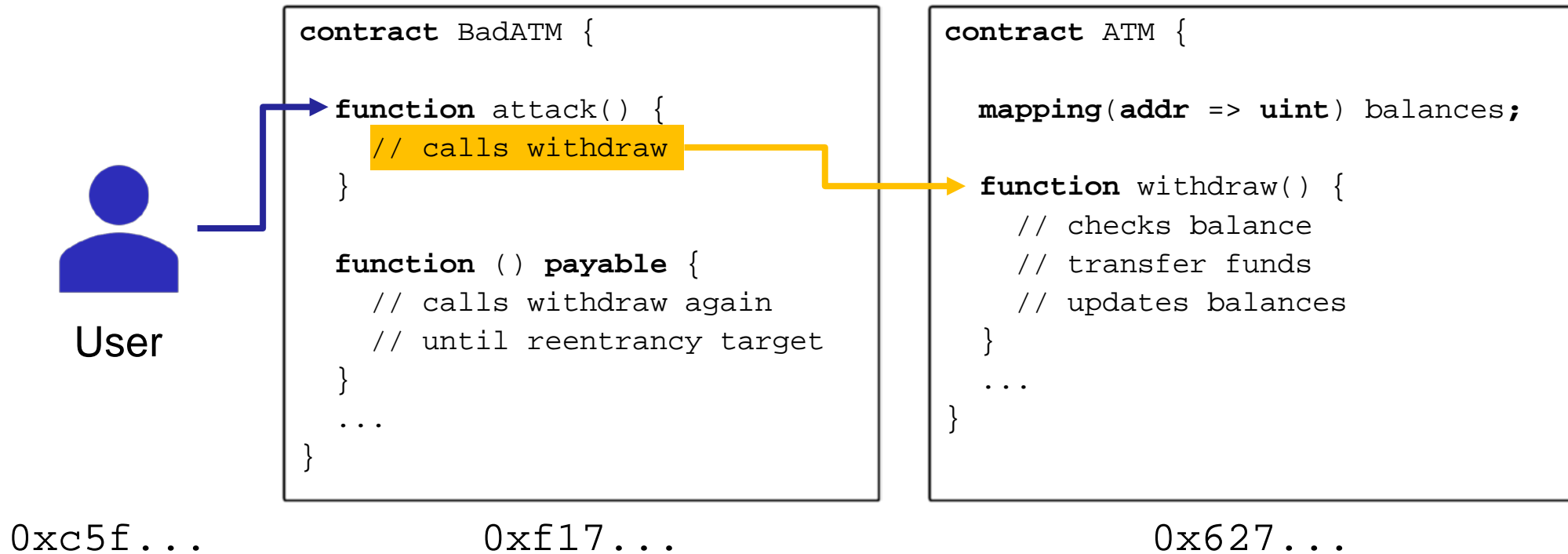
Smart Contract Security

- ❑ **Transaction Ordering**
 - > Blockchain Shop
- ❑ **Reentrancy Attacks**
 - > Good ATM | Bad ATM

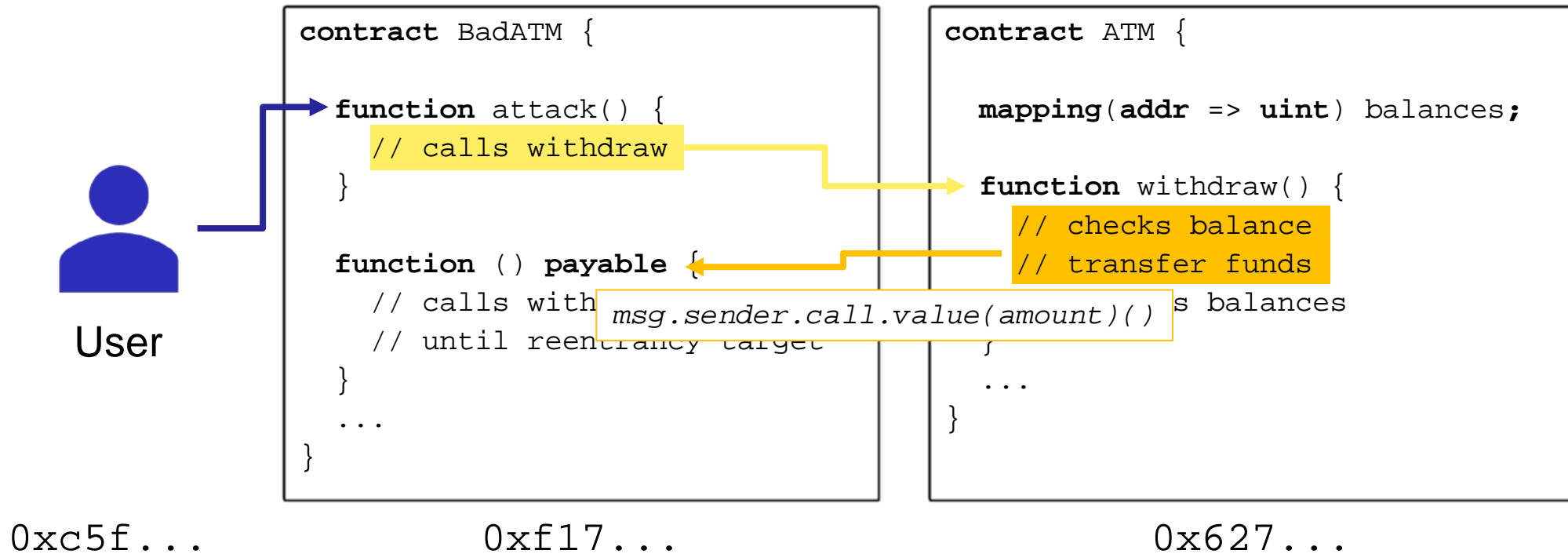
ATM Contract



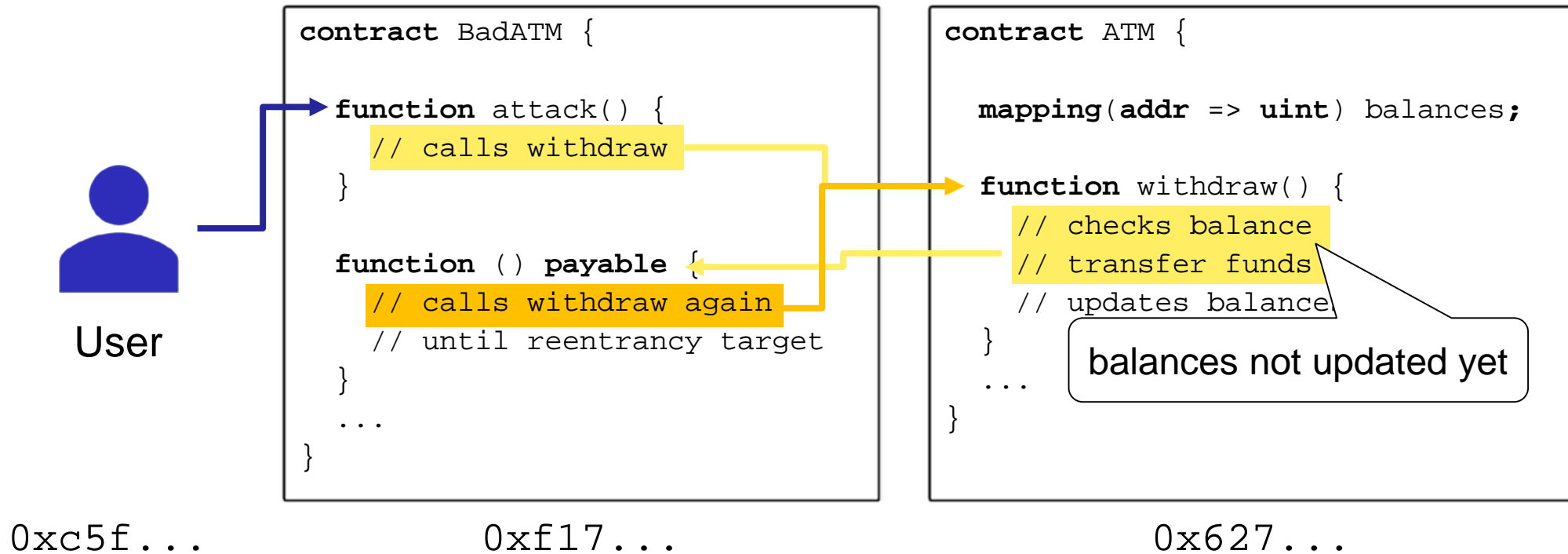
Reentrancy Attack



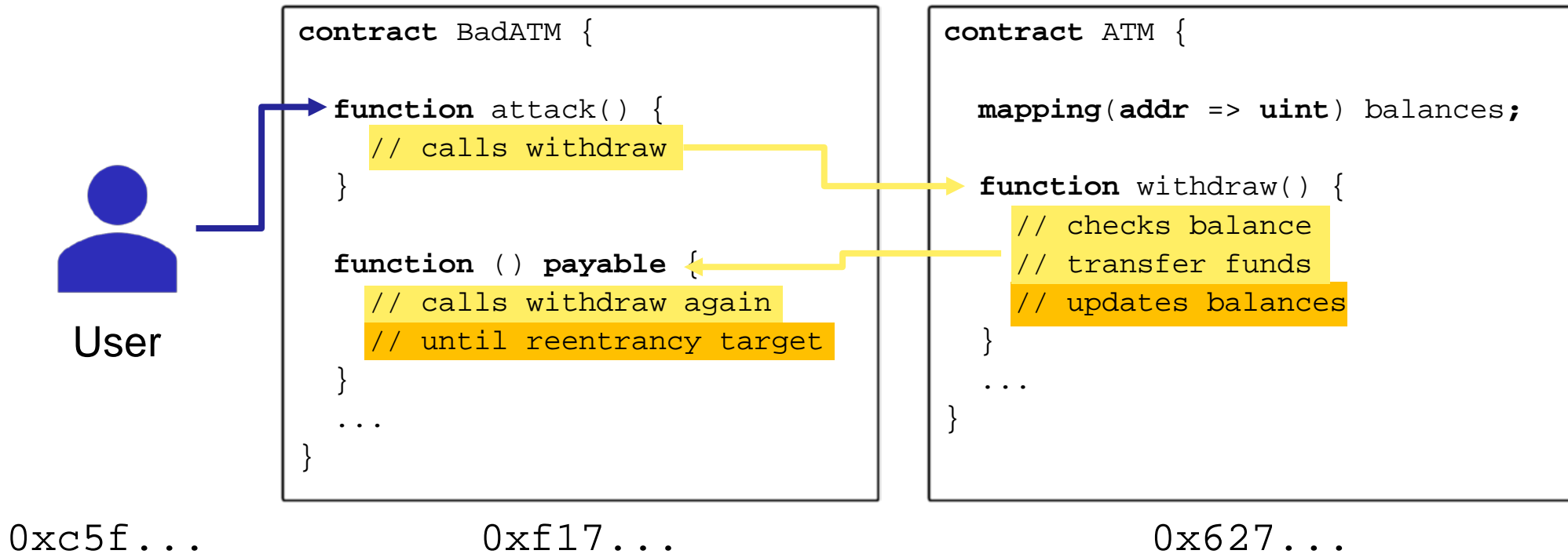
Reentrancy Attack



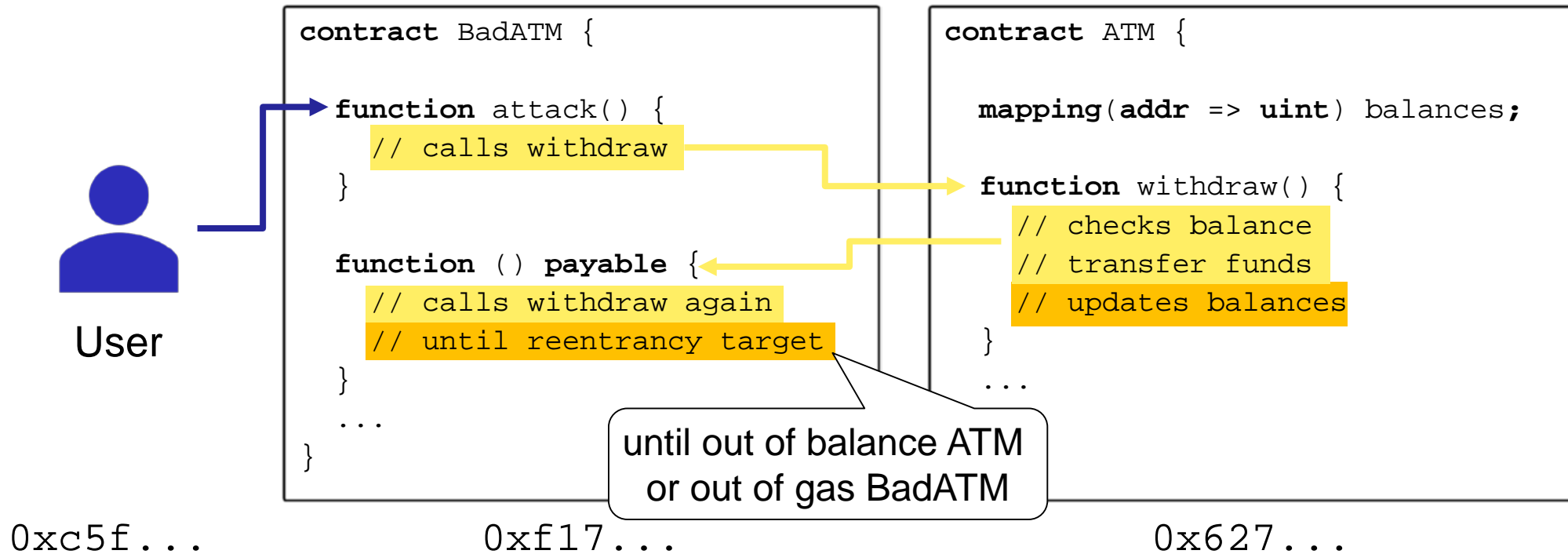
Reentrancy Attack



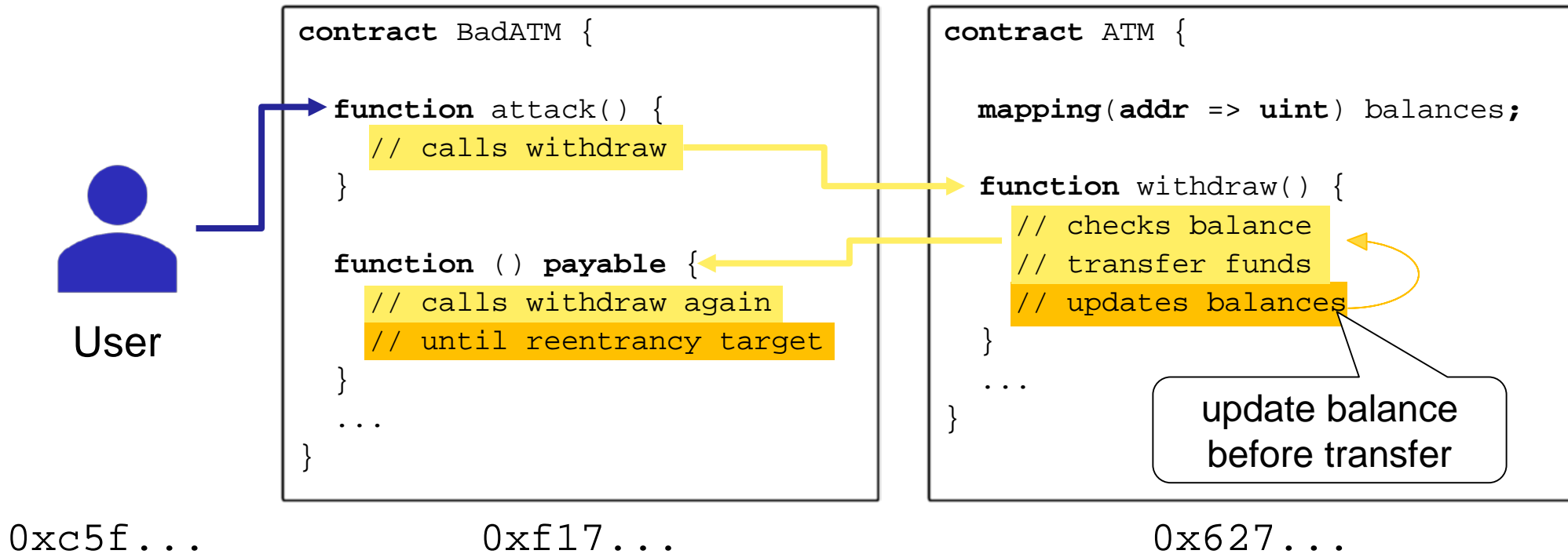
Reentrancy Attack



Reentrancy Attack



Reentrancy Attack



Contract Locks



User

```
contract BadATM {  
  
    function attack() {  
        // calls withdraw  
    }  
  
    function () payable {  
        // calls withdraw again  
        // until reentrancy target  
    }  
    ...  
}
```

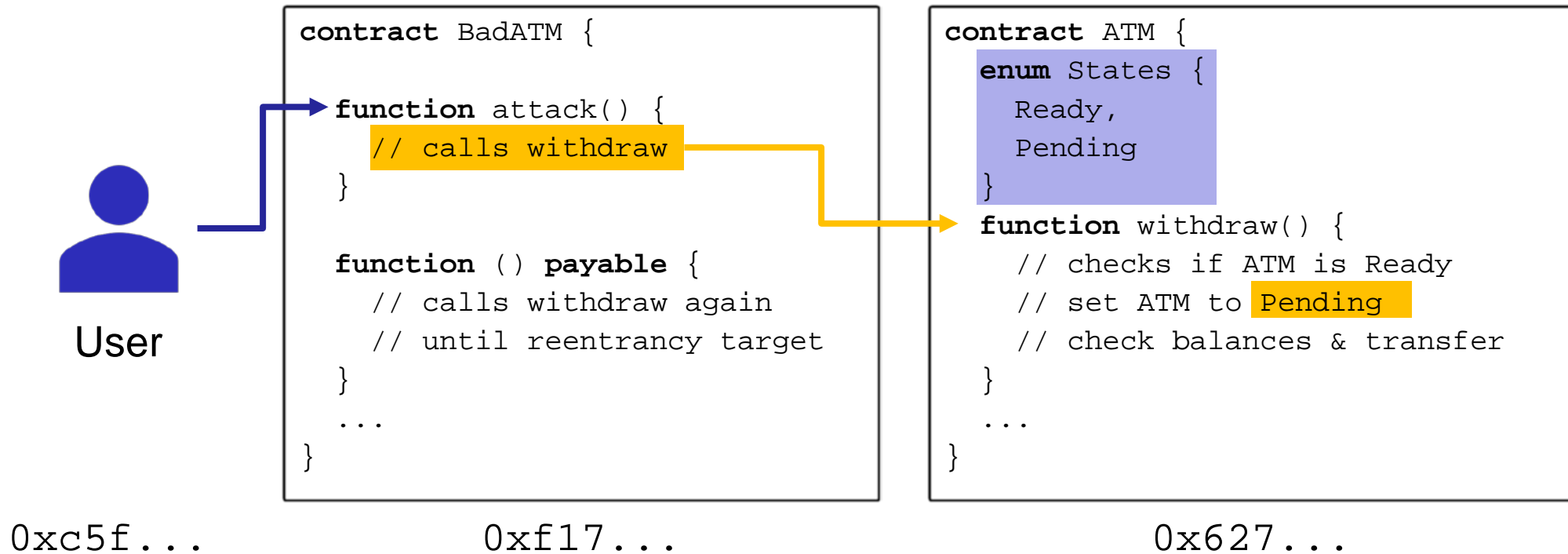
```
contract ATM {  
    enum States {  
        Ready,  
        Pending  
    }  
    function withdraw() {  
        // checks if ATM is Ready  
        // set ATM to Pending  
        // check balances & transfer  
    }  
    ...  
}
```

0xc5f...

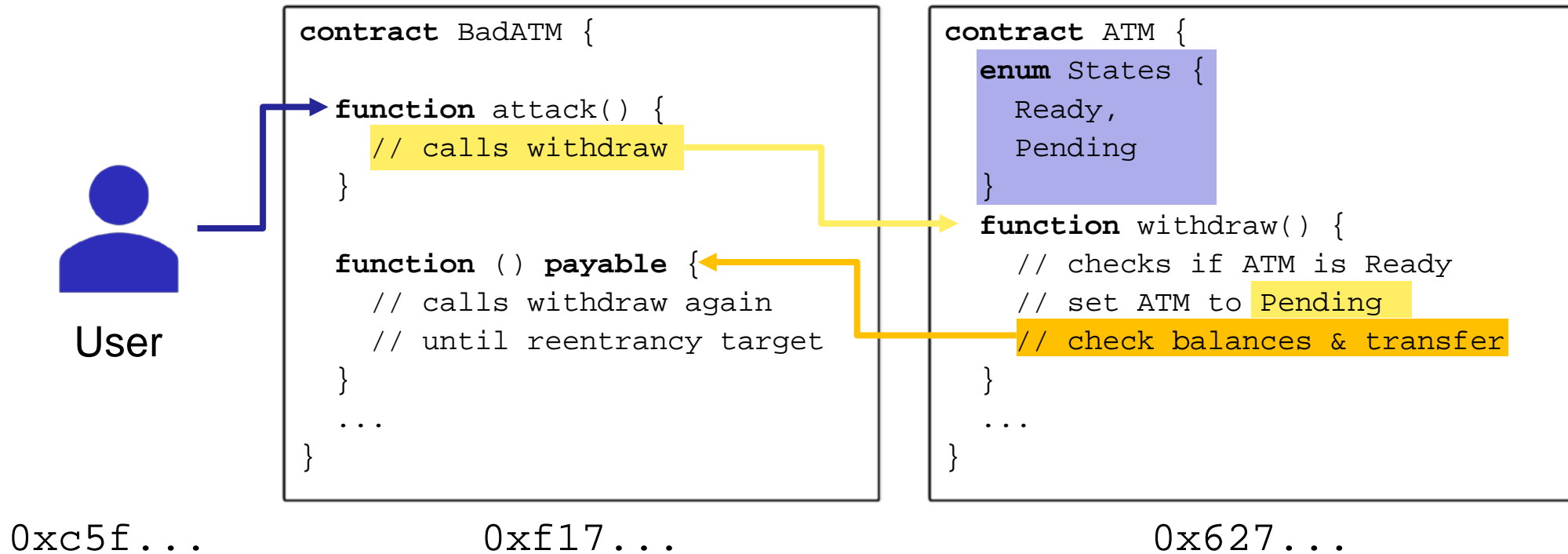
0xf17...

0x627...

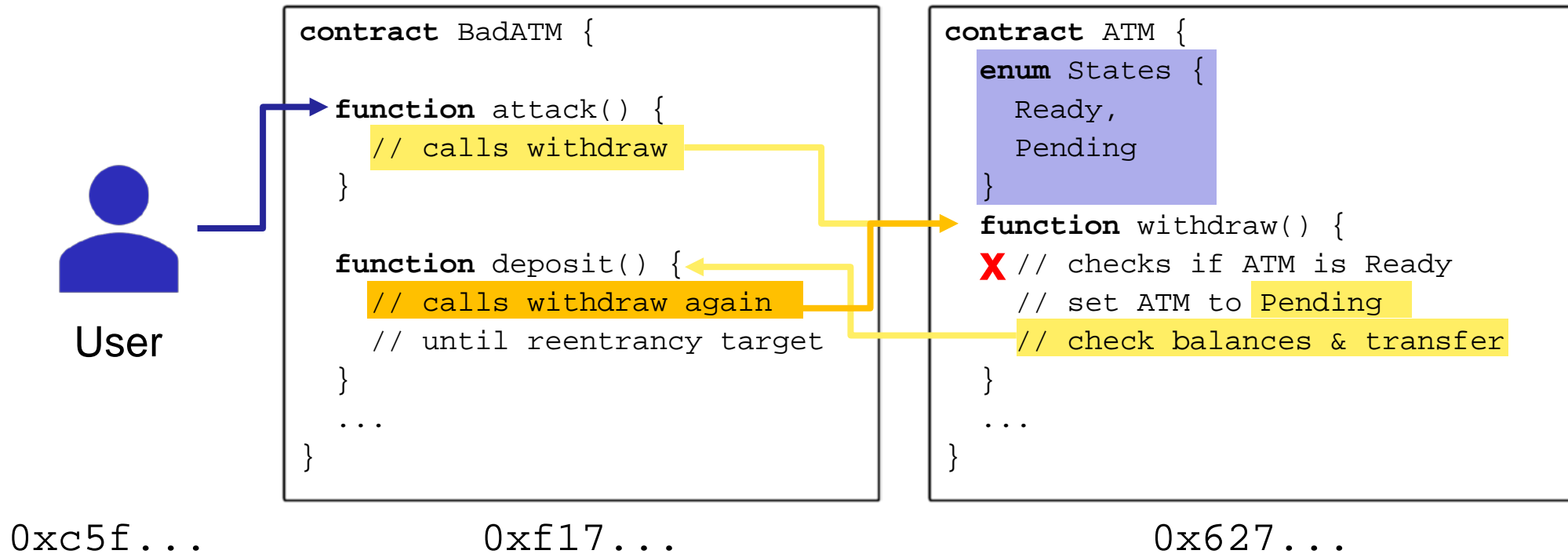
Contract Locks



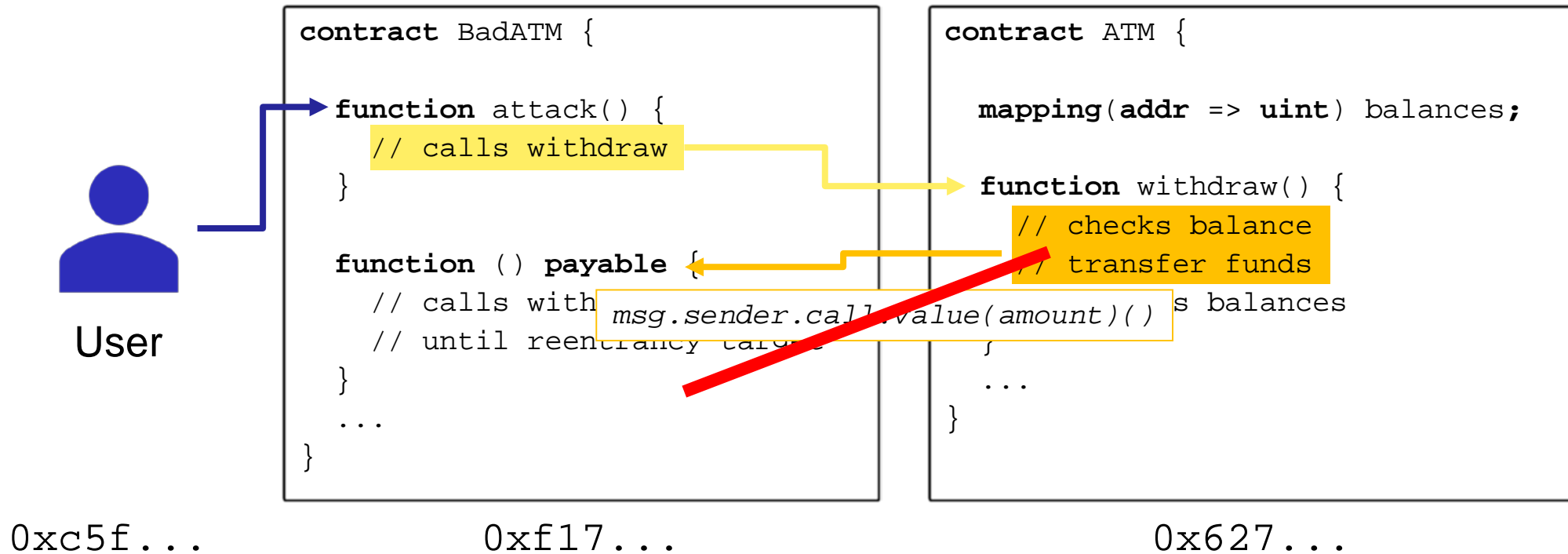
Contract Locks



Contract Locks

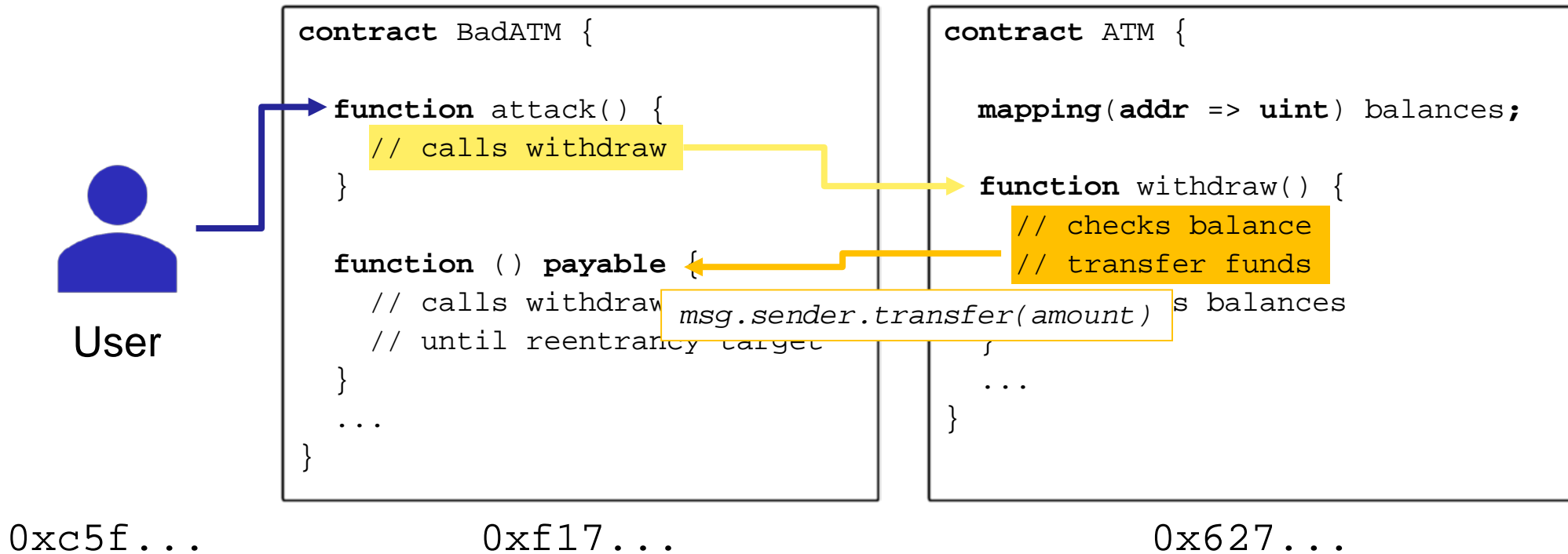


Reentrancy Attack



The transfer-Function

- ❑ Only a small amount of gas is sent along (21,000 gas).
- ❑ The receiver can only emit one single event at max, safe by “accident”.



In Conclusion - Best Practices

- ❑ Prepare for failure
- ❑ Rollout carefully
- ❑ Keep contracts simple
- ❑ Stay up to date
- ❑ Be aware of blockchain properties

Others

- ❑ Safe Math (Overflow)
- ❑ Error Handling (Revert / Require / Throw)
- ❑ Best Practices e.g. [Recommendation for Smart Contract Security in Solidity](#)

Part III - Discussion and Considerations

Overview of Blockchain Challenges

- How to handle reliably **tangible (non-digital) assets** in BC?
 - A Bitcoin is represented as bits vs. property, real estate as physical items
- **Sustainability: Energy efficiency** of consensus mechanisms?
 - Energy consumption for Bitcoin BC alone in 2017 \approx Iceland's production
- **Scalability**: BC throughput as a number of **transactions per second**, **volume of data** persisted in Mega (?) bytes, **costs**?
 - *E.g.*, BC sizes grow faster than the density of HDDs/SSDs
 - BC (always) better than a (distributed) data base? Exorbitant costs?
- **Identity management** (users, objects) and **anonymity**
- **Standardized APIs** for switching BCs for BC-based dapps
 - *E.g.*, in contrast, databases from different vendors offer “similar” APIs
- Many **economic effects** of BC-based cryptocurrencies unknown
 - Role of **national “e”-currency**, **interrelationships** of about 2000+ cryptoc.
- **Legal/regulative** compliance, **societal/governmental** acceptance

Mapping Challenges (1)

	Public Permissionless	Public Permissioned	Private Permissionless	Private Permissioned
Scalability	Public usage → size growth hard to be controlled	Only selected nodes create blocks → more control over size	Blockchain designed for specific use case → controlled size	Blockchain designed for specific use case → controlled size
Data Storage	Not designed as DB → High fees, size is limited	Know writers → No fees, no size limit	Know participants → Low fees, partial size limit	Know writers → No fees, no size limit
Sustainability	PoW → computational power has no “social benefit”	PoA → Sustainable, no significant computations	PoS → Sustainable, no significant computations	PoA → Sustainable, no significant computations
Identity Management	Pseudo-anonymity, data visible → Hard to link to physical user, data encryption	Data is supposed to be visible → Ensuring integrity	Know participants → Trusted environment	Know participants → Trusted environment

Mapping Challenges (2)

	Public Permissionless	Public Permissioned	Private Permissionless	Private Permissioned
Standardization	No standard → Complex Interoperability	No standard → Complex Interoperability	No standard → Complex Interoperability	No standard → Complex Interoperability
Trust	Data in the BC → Trusted Input data → Untrusted	Know writers → Trusted	Know participants → Trusted Environment	Know participants → Trusted Environment
Economics and Regulations	No clear regulations → Gray area	No clear regulations → Gray area	Regulated by participants → Defined rules	Regulated by participants → Defined rules

Public Blockchain Risks

- ❑ BCs' "true semantics" depends on the input received!
- ❑ BCs' security, privacy, and reliability
 - Unknown attack vectors (& 51% attack), Programming errors in SCs
 - Alternative consensus mechanisms beyond PoW? Security at stake?
 - The breaking of currently used security algorithms
 - Long-term storage? Quantum Computing impacts?
 - Privacy: persisted data at stake? GDPR?
 - The right to forget vs. immutability
 - Transparency (public knowledge of BC) vs. privacy (private data)
- ❑ Networking infrastructure's reliability (critical infrastructures)
 - Lacking Internet connectivity for a "longer" period of time?
- ❑ Economic/legal risks (cryptocurrency/tokens/coins, BC)
 - Fraudulent profitability projections, volatility, dispute resolutions

Conclusions

1. Blockchains **do** show a logical evolution of linked lists, however, they “exaggerate” processing demands
 - Especially Proof-of-Work (PoW), but this ensures immutability
2. The technical future of blockchains is based on **security ingredients** of today’s technology, however, long-term storage and security management is not known by now
 - *E.g.*, unknown impact of Quantum computing (on all IT!)
3. Blockchains are **no** revolution, but a typical Computer Science (Abstract Data Type) evolution of linked lists
 - The “distribution” of consensus does not always make sense
 - Any system as of the past has **not** been replaced fully by a BC

Thank you for your attention.



Questions?