BLOCKCHAIN

A confluence of Cryptography, Game Theory and Distributed Computing

by

Kannan Srinathan

IIIT-Hyderabad

Blockchain solves a very hard problem

In this lecture:

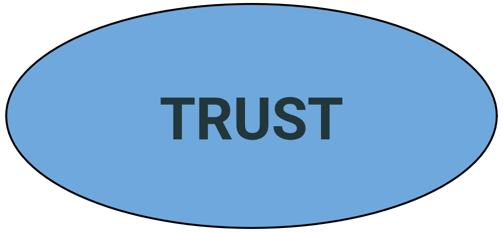
- What is the hard problem?
- What is the pleasing solution?

The problem
(High level Overview)

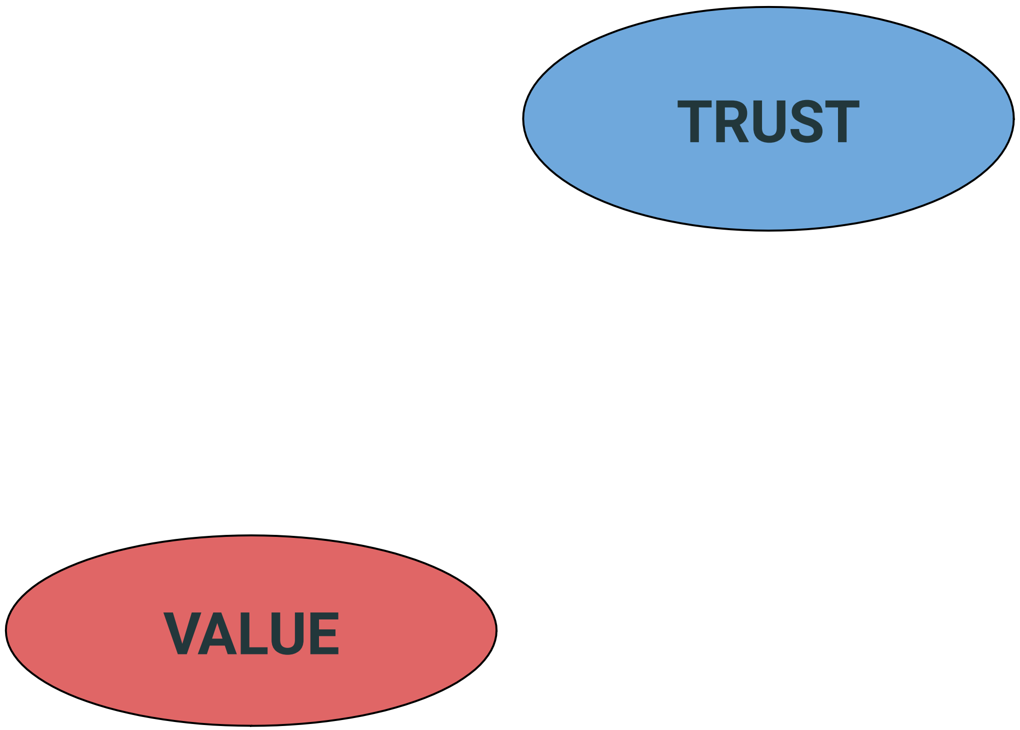
The Big Picture

The Big Picture

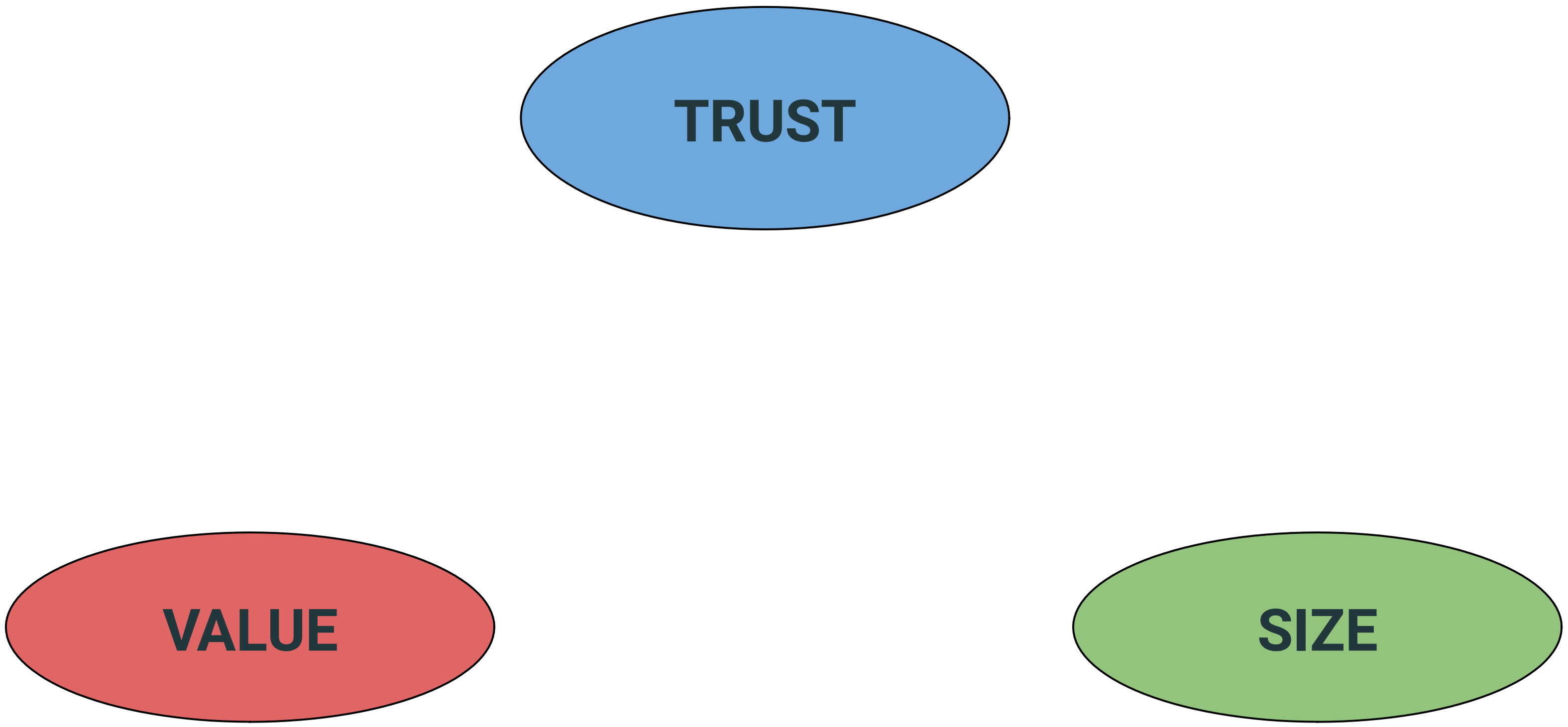
The Big Picture



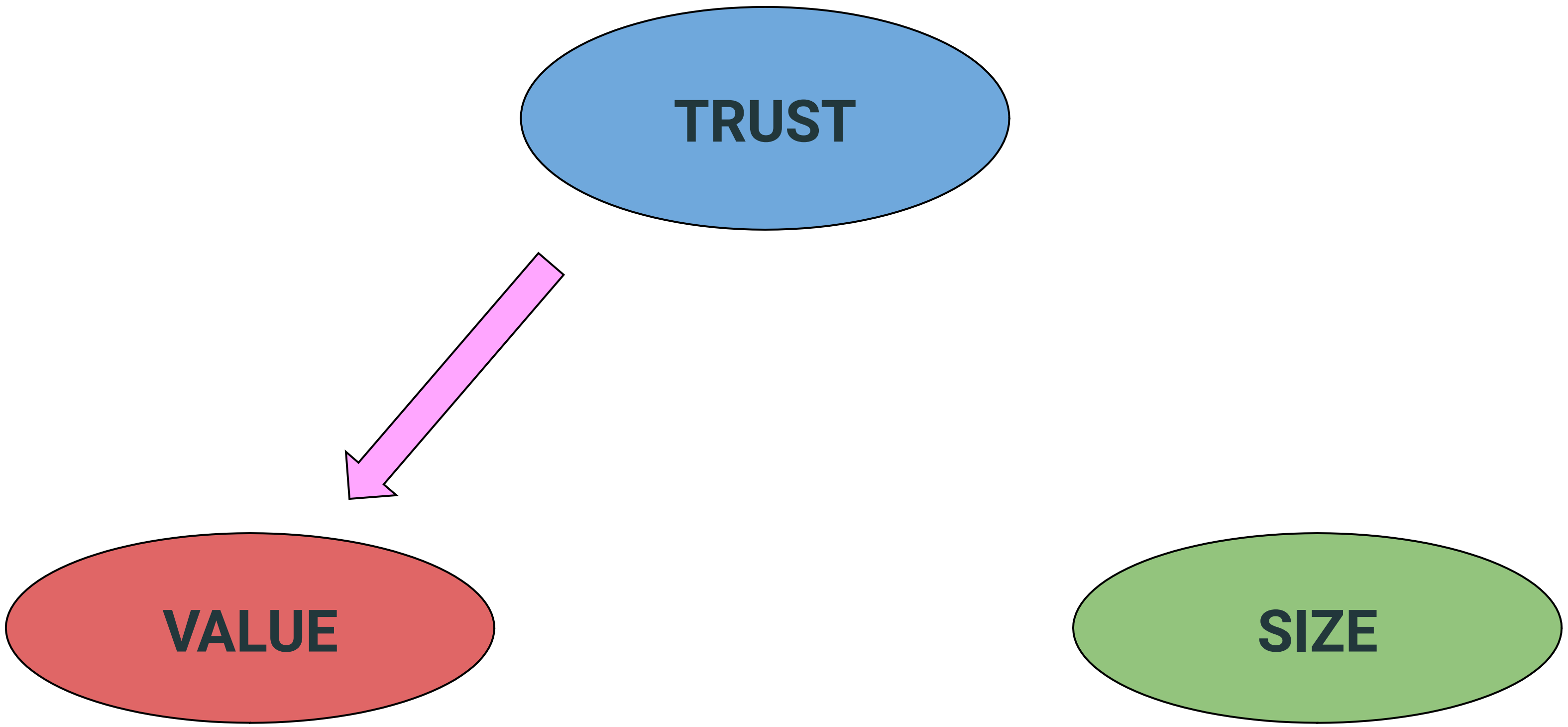
The Big Picture



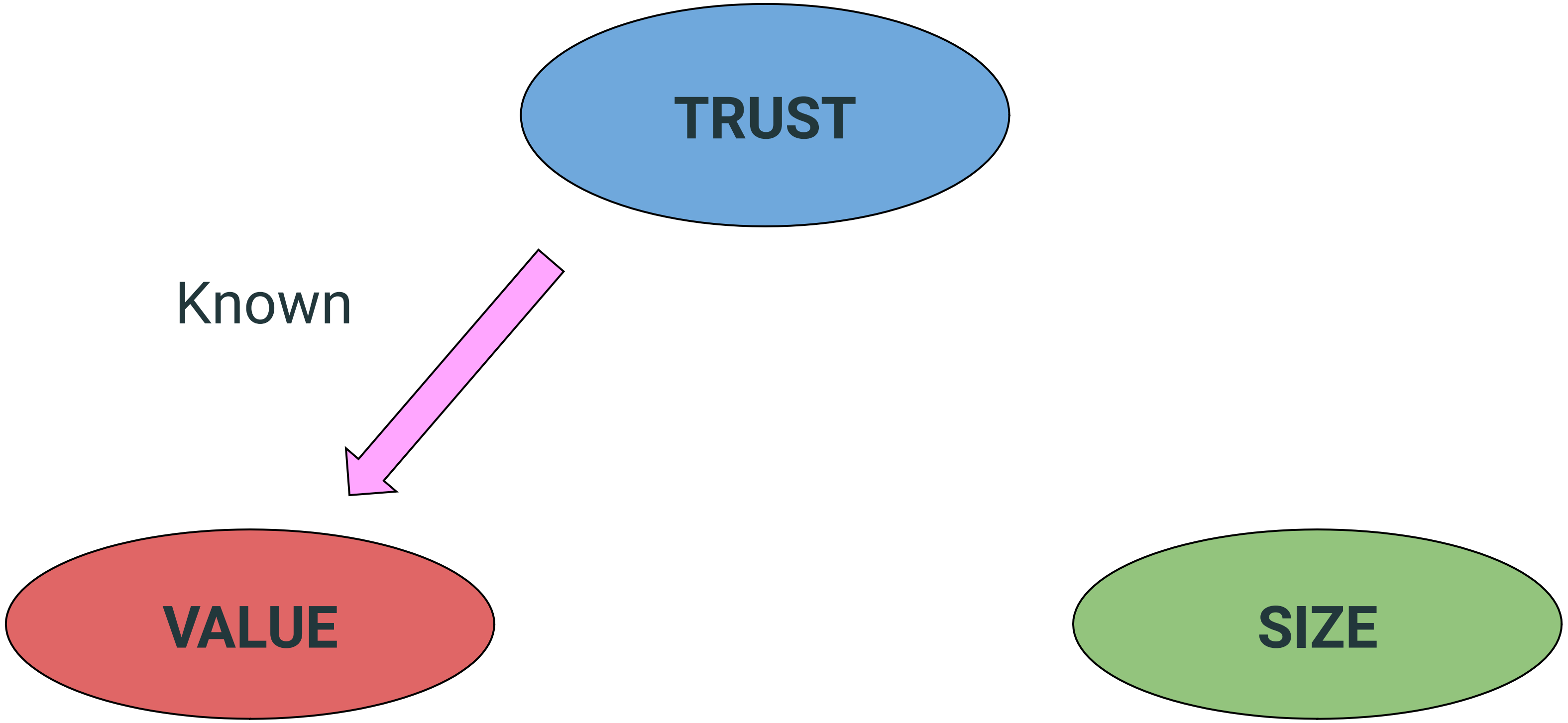
The Big Picture



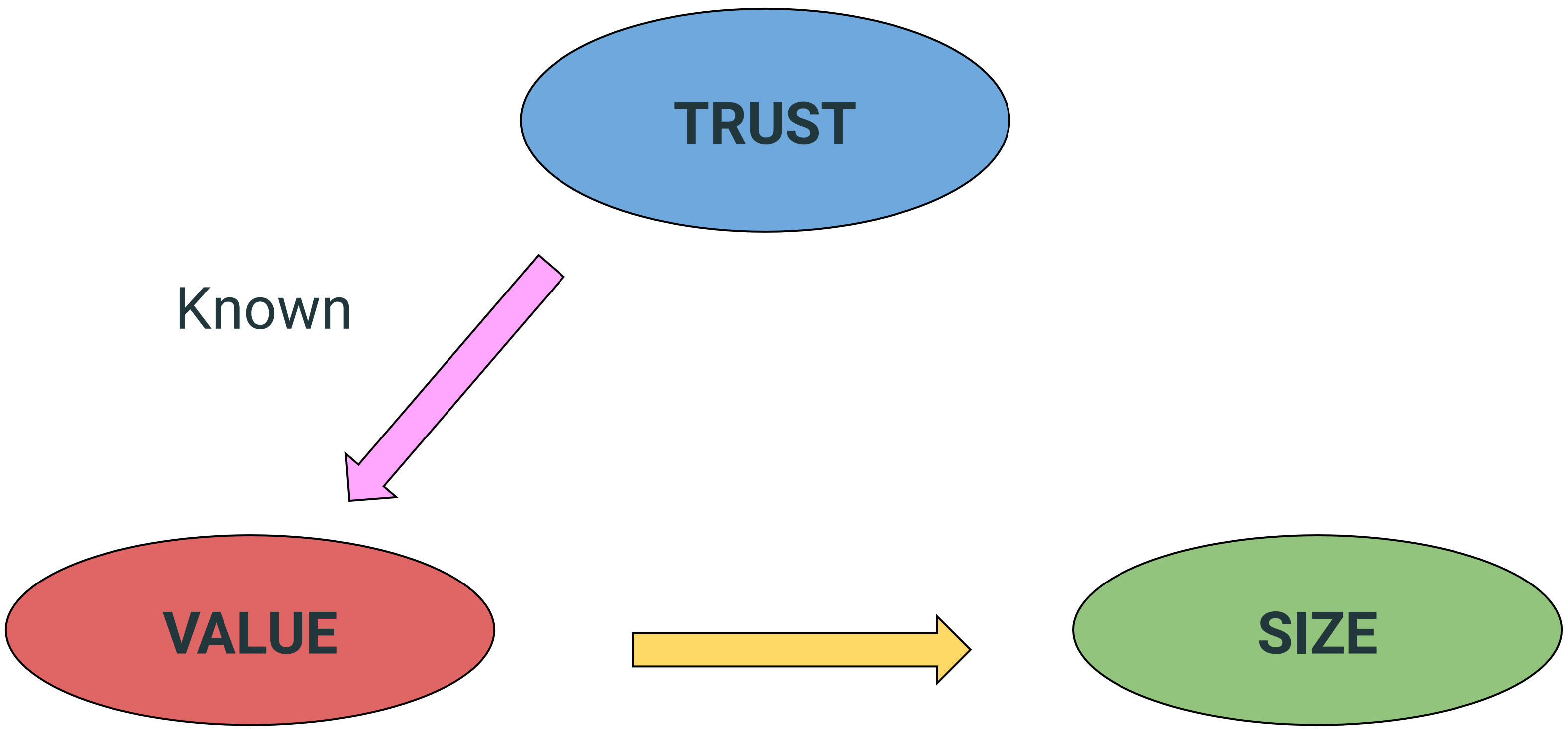
The Big Picture



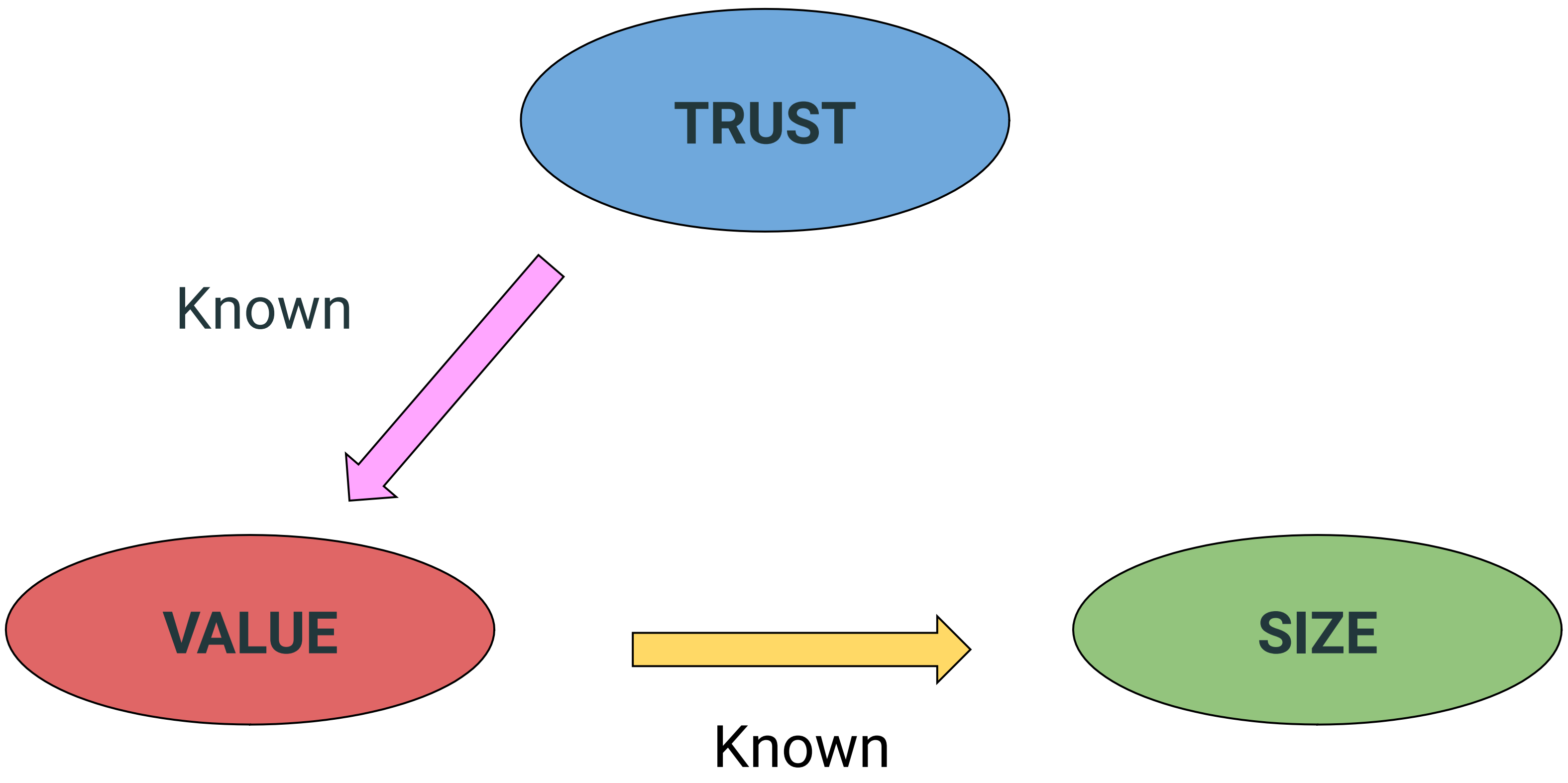
The Big Picture



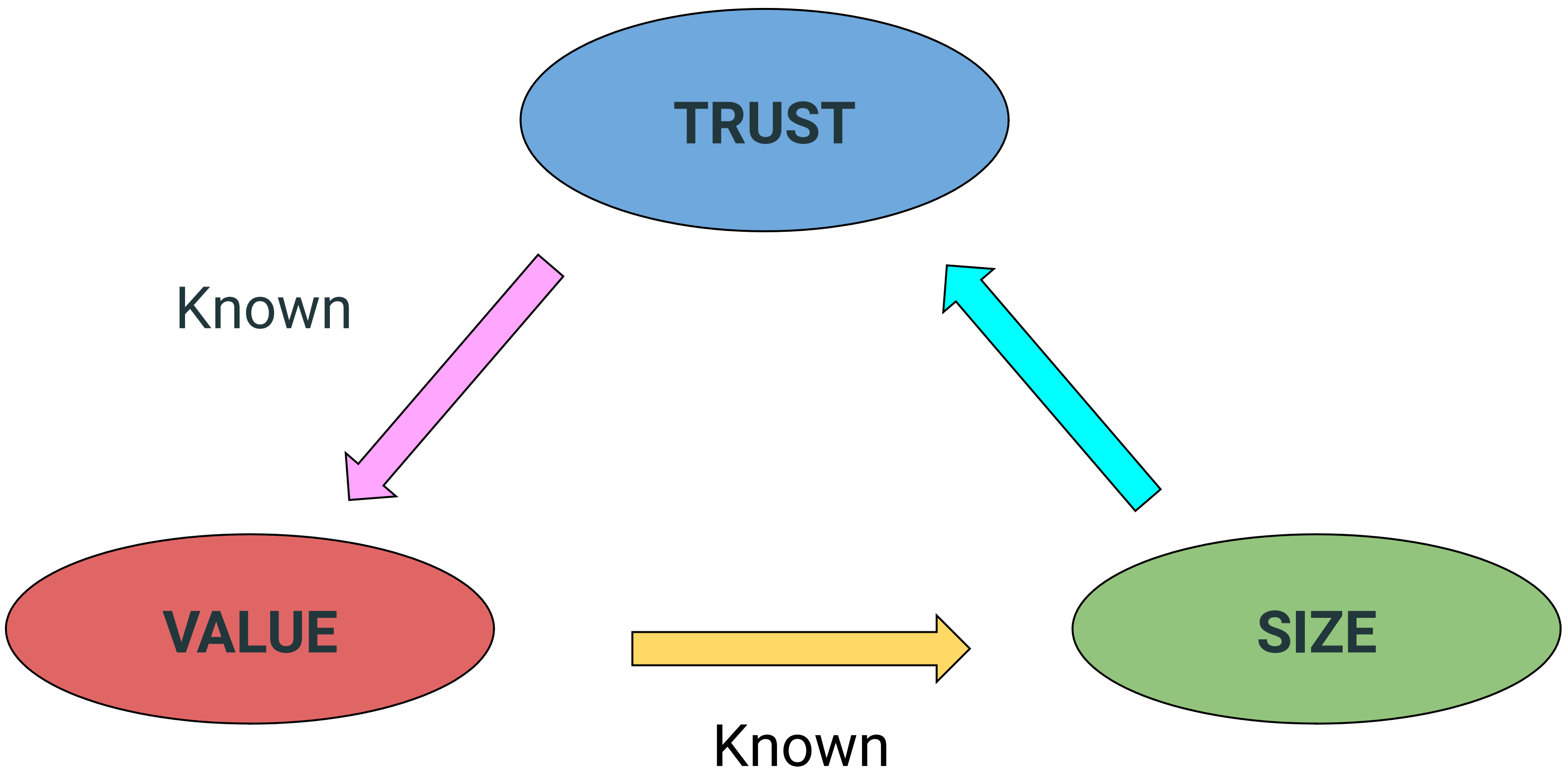
The Big Picture



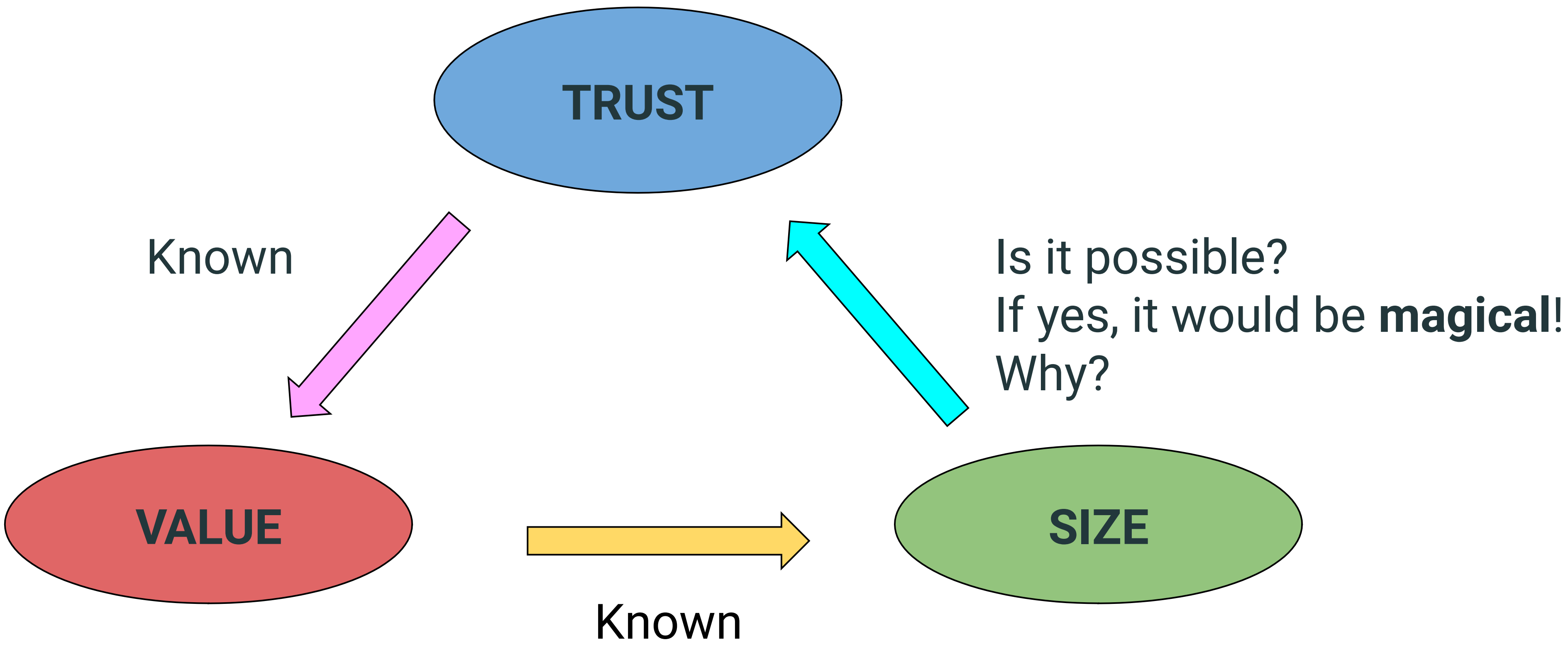
The Big Picture



The Big Picture



The Big Picture



What’s The Challenge?

What's The Challenge?

Distributing Trust

Can a collection of untrustworthy nodes simulate a trustworthy one?

What's The Challenge?

Distributing Trust

Can a collection of untrustworthy nodes simulate a trustworthy one?

Recall: Too many (good) cooks spoil the broth!

(Consensus is not easy)

What to say if some of them are outright malicious?

Byzantine Agreement

Byzantine Agreement

A Fundamental Problem

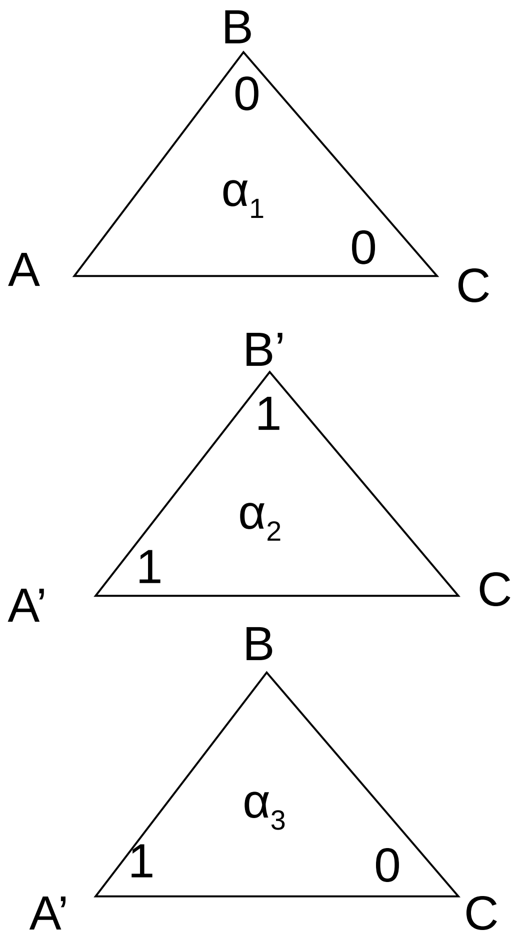
Simulating Broadcast in P2P Networks

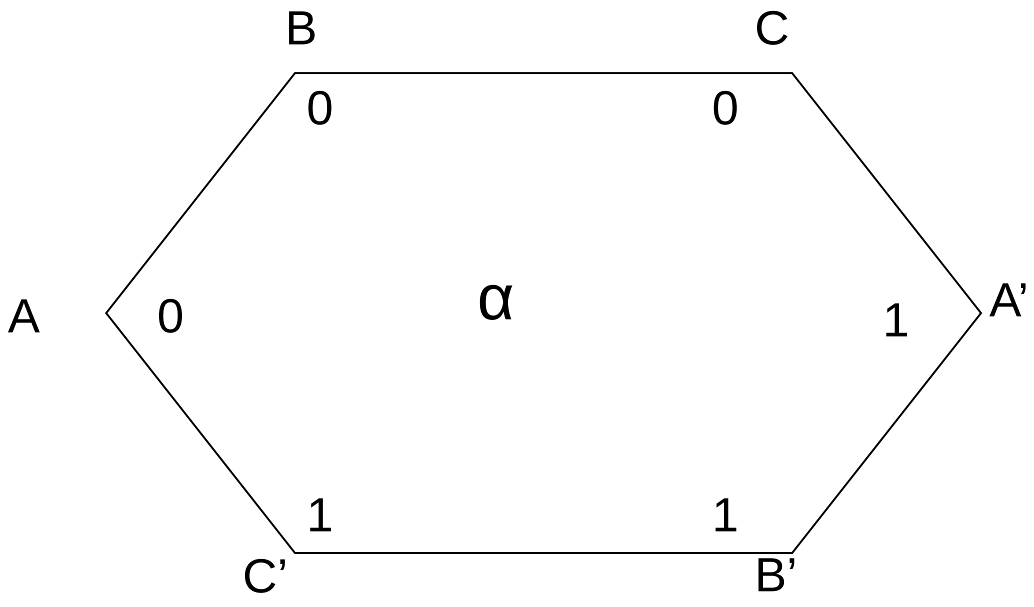
Simulating Broadcast in P2P Networks

What's the issue?

Atomicity!

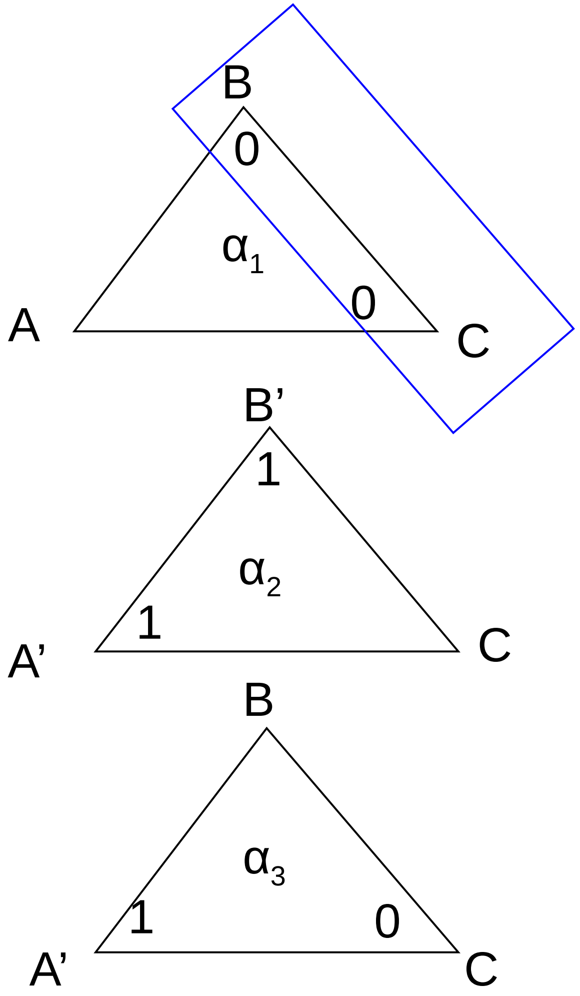
Impossibility for 1 out of 3 [2]

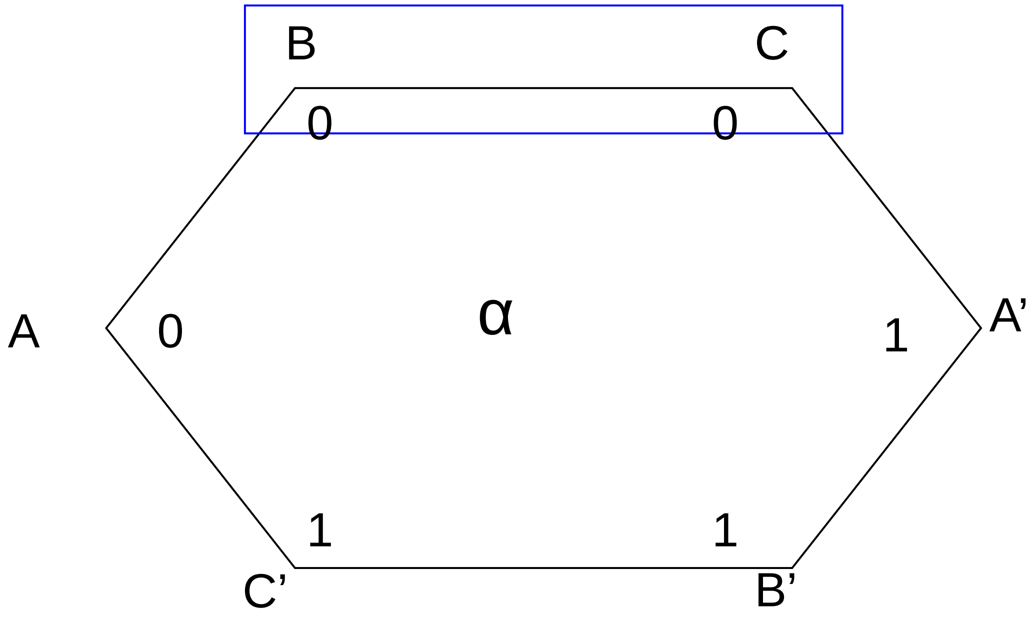




[2] Michael J. Fischer, Nancy A. Lynch, Michael Merritt. **Easy impossibility proofs for distributed consensus problems** , PODC '85, 59-70.

Impossibility for 1 out of 3 [2]

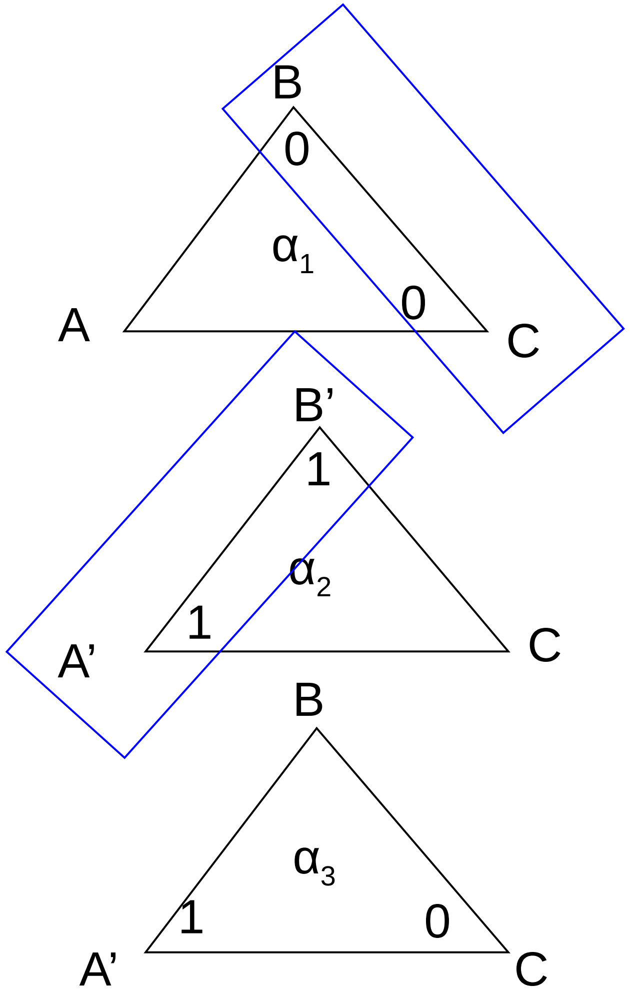


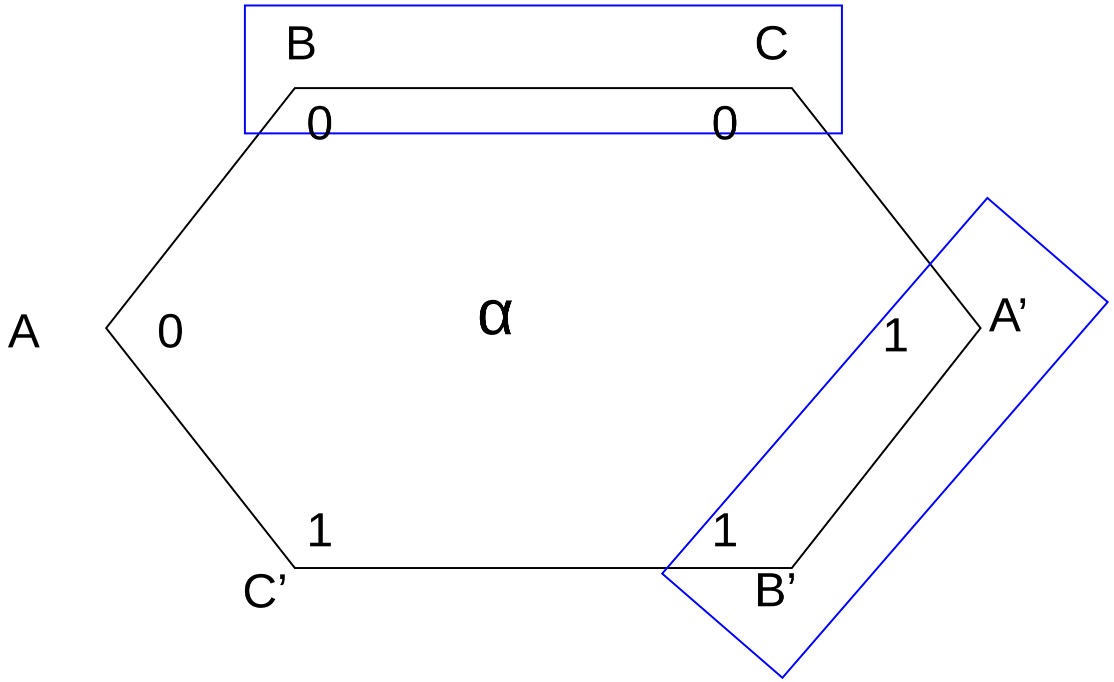


Executions α_1 and α are indistinguishable to processes B and C

[2] *Michael J. Fischer, Nancy A. Lynch, Michael Merritt*. **Easy impossibility proofs for distributed consensus problems** , PODC '85, 59-70.

Impossibility for 1 out of 3 [2]

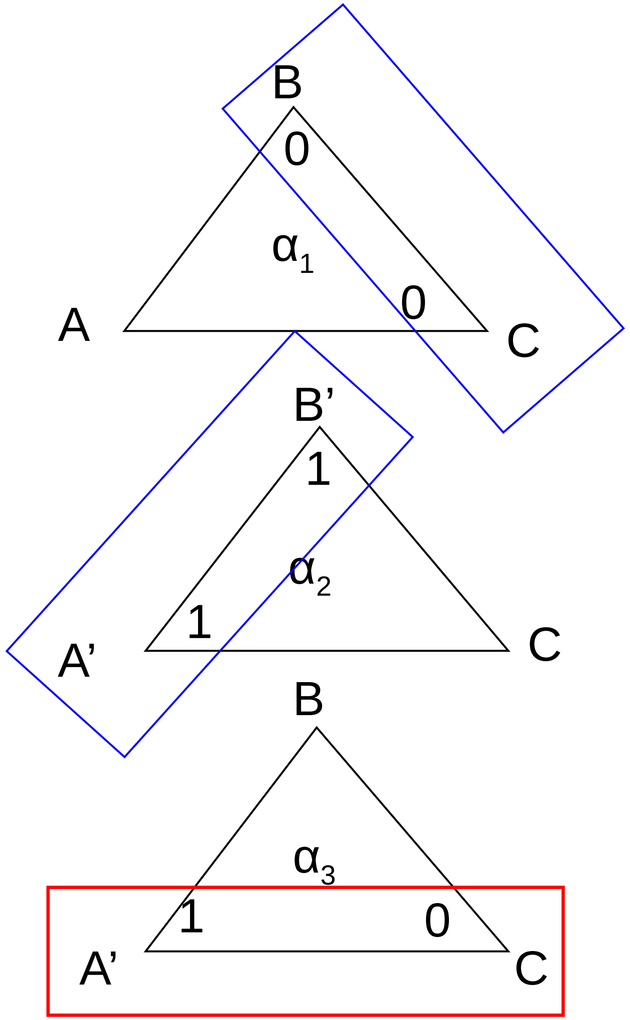


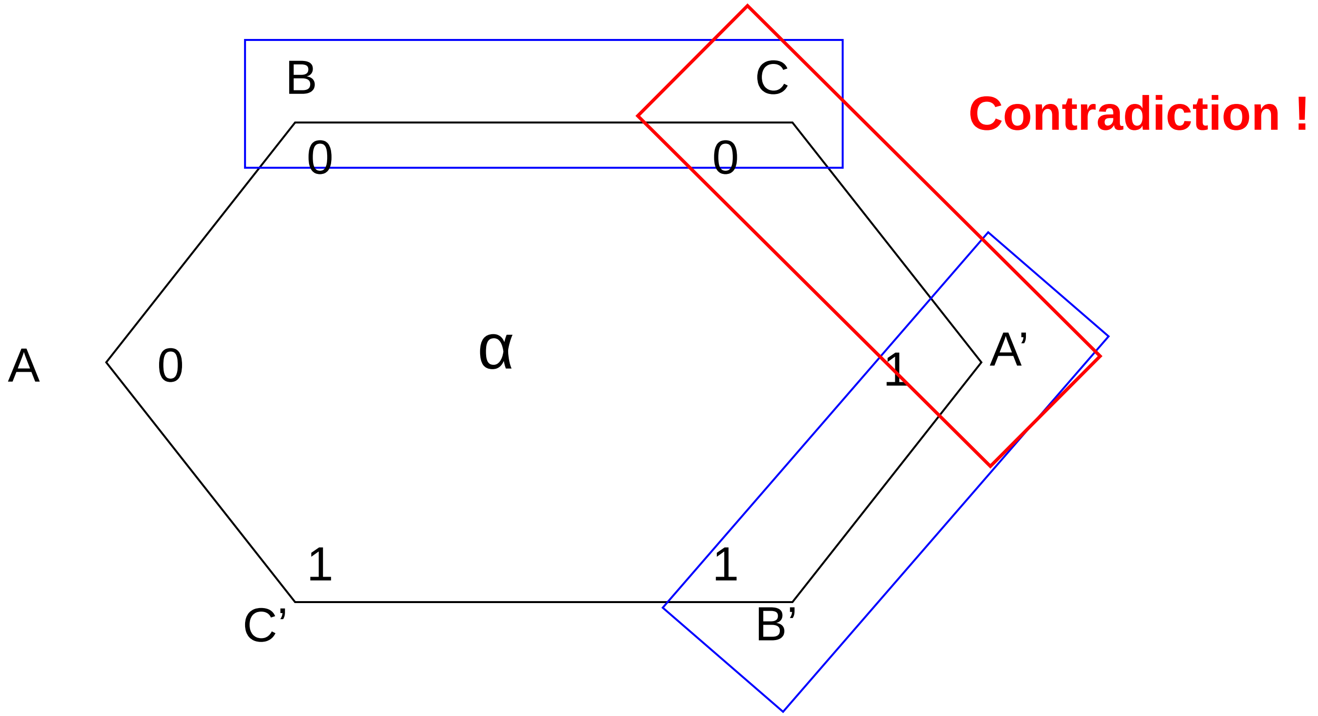


Executions α_2 and α are indistinguishable to processes A’ and B’

[2] Michael J. Fischer, Nancy A. Lynch, Michael Merritt. **Easy impossibility proofs for distributed consensus problems** , PODC '85, 59-70.

Impossibility for 1 out of 3 [2]





Executions α_3 and α are indistinguishable to processes A' and C

[2] *Michael J. Fischer, Nancy A. Lynch, Michael Merritt*. **Easy impossibility proofs for distributed consensus problems** , PODC '85, 59-70.

(S)ample (Un)related Challenges

(S)ample (Un)related Challenges

Cluster computing: Distributing workload

- Can a collection of slow nodes simulate a fast, if not an omnipotent one?

(S)ample (Un)related Challenges

Cluster computing: **Distributing workload**

- Can a collection of slow nodes simulate a fast, if not an omnipotent one?

Network cloud memory: **Distributed storage**

- Can a collection of low memory modules simulate a huge if not omniscient memory?

(S)ample (Un)related Challenges

Cluster computing: Distributing workload

- Can a collection of slow nodes simulate a fast, if not an omnipotent one?

Network cloud memory: Distributed storage

- Can a collection of low memory modules simulate a huge if not omniscient memory?

Ubiquitous/pervasive computing: Distributing I/O

- Can a collection of locally accessible nodes simulate a wide, if not an omnipresent access?

(S)ample (Un)related Challenges

Cluster computing: **Distributing workload**

- Can a collection of slow nodes simulate a fast, if not an omnipotent one?

Network cloud memory: **Distributed storage**

- Can a collection of low memory modules simulate a huge if not omniscient memory?

Ubiquitous/pervasive computing: **Distributing I/O**

- Can a collection of locally accessible nodes simulate a wide, if not an omnipresent access?

(Future) **Conscious Computing**: Distributing consciousness/intelligence.

Gearing-Up Our Ammunition

Gearing-Up Our Ammunition

The “Utopian” Ammunitions:

Gearing-Up Our Ammunition

The “Utopian” Ammunitions:

Omnipresence

- The ability to be anywhere, instantaneously.

Gearing-Up Our Ammunition

The “Utopian” Ammunitions:

Omnipresence

- The ability to be anywhere, instantaneously.

Omnipotence

- The ability to execute any program, instantaneously.

Gearing-Up Our Ammunition

The "Utopian" Ammunitions:

Omnipresence

- The ability to be anywhere, instantaneously.

Omnipotence

- The ability to execute any program, instantaneously.

Omniscience

- The ability to access/know any data, instantaneously.

The Three Fields involved

The Three Fields involved

Non-omniness creates/solves problems:

The Three Fields involved

Non-omniness creates/solves problems:

Non-Omnipresence

- Distributed Computing

The Three Fields involved

Non-omniness creates/solves problems:

Non-Omnipresence

- Distributed Computing

Non-Omnipotence

- Cryptography

The Three Fields involved

Non-omniness creates/solves problems:

Non-Omnipresence

- Distributed Computing

Non-Omnipotence

- Cryptography

Non-Omniscience

- Game theory

The Three Fields *Can Be* Mutually “Complementary”

The Three Fields *Can Be* Mutually "Complementary"

Is there a problem, so hard, that it is (best) solved by the combined efforts of all the three fields?

How To Solve It ?

(By Non-Omnipotence of machines?)

How To Solve It ?

(By Non-Omnipotence of machines?)

Welcome to **Crypto**!

Crypto is a *Fantastic* Story Because …

Crypto is a *Fantastic* Story Because …

… no other field of science has ever had to so brazenly circumvent logical *no-go theorems* …

Sample No-Goes

Sample No-Goes

- Illustrating Logical No-Go (Russell's Paradox): Let S be the set of all sets that do not contain itself? Does S belong to S?
  Ans: Yes *and* No!

Sample No-Goes

- Illustrating Logical No-Go (Russell’s Paradox): Let S be the set of all sets that do not contain itself? Does S belong to S?
  Ans: Yes *and* No!
- Should the machine know your *password*?
  Ans: Yes (for checking) *and* No (for secrecy)

Sample No-Goes

- Illustrating Logical No-Go (Russell's Paradox): Let S be the set of all sets that do not contain itself? Does S belong to S?
  Ans: Yes *and* No!
- Should the machine know your *password*?
  Ans: Yes (for checking) *and* No (for secrecy)
- Can you spend your digital cash?
  Ans: Yes (the original) *and* No (the copies)

Sample No-Goes

- Illustrating Logical No-Go (Russell's Paradox): Let S be the set of all sets that do not contain itself? Does S belong to S?
  Ans: Yes *and* No!
- Should the machine know your *password*?
  Ans: Yes (for checking) *and* No (for secrecy)
- Can you spend your digital cash?
  Ans: Yes (the original) *and* No (the copies)
- Should there be CCTV cameras?
  Ans: Yes (for policing) *and* No (for privacy)

(S)ample "Successes" against Logical Impossibilities

(S)ample “Successes” against Logical Impossibilities

- Compression *without* Collision!

(S)ample “Successes” against Logical Impossibilities

- Compression *without* Collision!
- Authenticity *with* Anonymity!

(S)ample “Successes” against Logical Impossibilities

- Compression *without* Collision!
- Authenticity *with* Anonymity!
- Blinding *but* Binding!

(S)ample "Successes" against Logical Impossibilities

- Compression *without* Collision!
- Authenticity *with* Anonymity!
- Blinding *but* Binding!
- Answering correctly *without* knowing the Query!

(S)ample "Successes" against Logical Impossibilities

- Compression *without* Collision!
- Authenticity *with* Anonymity!
- Blinding *but* Binding!
- Answering correctly *without* knowing the Query!
- Alice proves (some true statement) to Bob but Bob *cannot* prove it to Charlie!

(S)ample “Successes” against Logical Impossibilities

- Compression *without* Collision!
- Authenticity *with* Anonymity!
- Blinding *but* Binding!
- Answering correctly *without* knowing the Query!
- Alice proves (some true statement) to Bob but Bob *cannot* prove it to Charlie!
- Privacy Preserving Personalization!

It is naturally *Fundamental* because …

It is naturally *Fundamental* because …

… cryptography has *famously* extended its success story by revolutionarily circumventing logical no-go theorems in *other areas* too!

Founding Members of the Association on “Beneficiaries of Cryptography”

[Ironically, they are also the prominent members of the Club that Cryptography Benefits From!]

Founding Members of the Association on “Beneficiaries of Cryptography”

[Ironically, they are also the prominent members of the Club that Cryptography Benefits From!]

- Coding Theory: Detecting 100% adversarial noise is feasible!

Founding Members of the Association on "Beneficiaries of Cryptography"

[Ironically, they are also the prominent members of the Club that Cryptography Benefits From!]

- Coding Theory: Detecting 100% adversarial noise is feasible!
- Distributed Computing: Byzantine Agreement

Founding Members of the Association on "Beneficiaries of Cryptography"

[Ironically, they are also the prominent members of the Club that Cryptography Benefits From!]

- Coding Theory: Detecting 100% adversarial noise is feasible!
- Distributed Computing: Byzantine Agreement
- Algorithms: Derandomization

Founding Members of the Association on "Beneficiaries of Cryptography"

[Ironically, they are also the prominent members of the Club that Cryptography Benefits From!]

- Coding Theory: Detecting 100% adversarial noise is feasible!
- Distributed Computing: Byzantine Agreement
- Algorithms: Derandomization
- Mathematics: IP = PSPACE = ZKP = QIP!

Our first exemplary problem

Is Secure Communication a Logical No-Go?

Is Secure Communication a Logical No-Go?

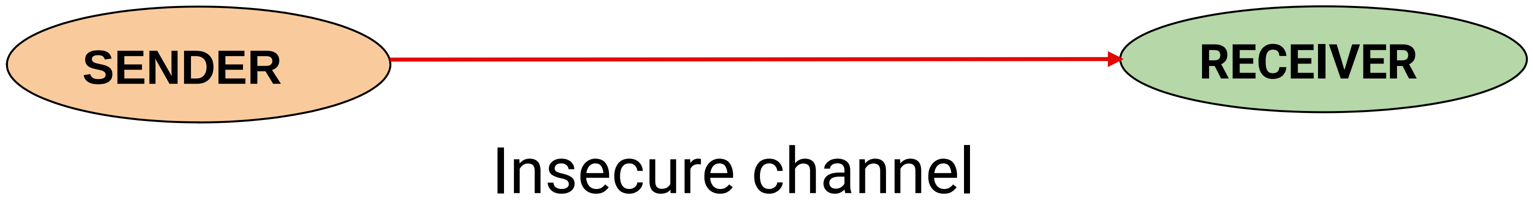
Yes!

Why?

Secure Communication is Impossible!

Secure Communication is Impossible!

Secure Communication is Impossible!



Secure Communication is Impossible!

Adversary

SENDER

RECEIVER

Insecure channel

Secure Communication is Impossible!

At time t_0:

Information@Receiver = Information@Adversary

Recall: Kerckhoff’s Principle

Secure Communication is Impossible!

At time t_0:
Information@Receiver = Information@Adversary
Recall: Kerckhoff's Principle

At every subsequent instant of time:
Information gained by receiver = Information gained by adversary

How to Circumvent the Impossibility?

How to Circumvent the Impossibility?

Ans: Non-Omnipotence of the eavesdropper!

How to Circumvent the Impossibility?

Ans: Non-Omnipotence of the eavesdropper!

Representation matters, indeed.

Natural Numbers, Efficiency of Operations and Modern Cryptography

Ease of Computation Depends on the Representation

Ease of Computation Depends on the Representation

It also depends on the operation!

Ease/Speed of Operation Depends on the Representation

- viii * xvi = cxxviii
- 8 * 16 = 128
- $2^3 * 2^4 = 2^7$

- viii + xvi = xxiv
- 8 + 16 = 24
- $2^3 + 2^4 = 2^3.3$

- viii < ix is true
- 8 < 9 is true
- $2^3 < 3^2$ is true

Is There a Representation Where all Common Operations are FAST?

- Addition (+)
- Comparison (<)
- Multiplication (*)

Is There a Representation Where all Common Operations are FAST?

Not Easy!

- Addition (+)
- Comparison (<)
- Multiplication (*)

Why is the Decimal System Popular?

Why is the Decimal System Popular?

	Addition	Multiplication	Comparison
ROMAN	SLOW	SLOW	SLOW
DECIMAL	**FAST**	MEDIUM	**FAST**
PRIME PRODUCT	SLOW	**FAST**	SLOW
RESIDUE SYSTEM	**FAST**	**FAST**	MEDIUM

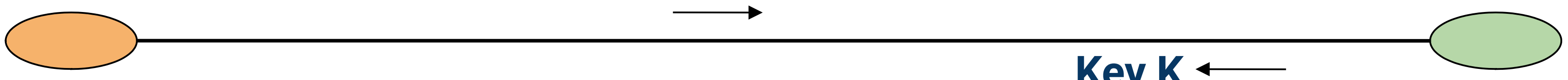
Slowness is advantageous too!

Slowness is <u>advantageous</u> too!

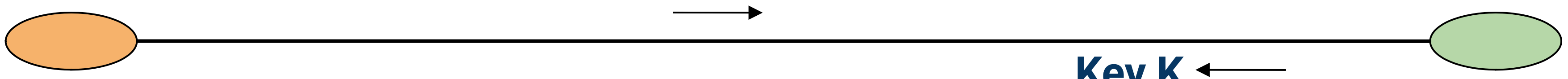
Public Key Cryptography

Secure Communication

Secure Communication



Secure Communication



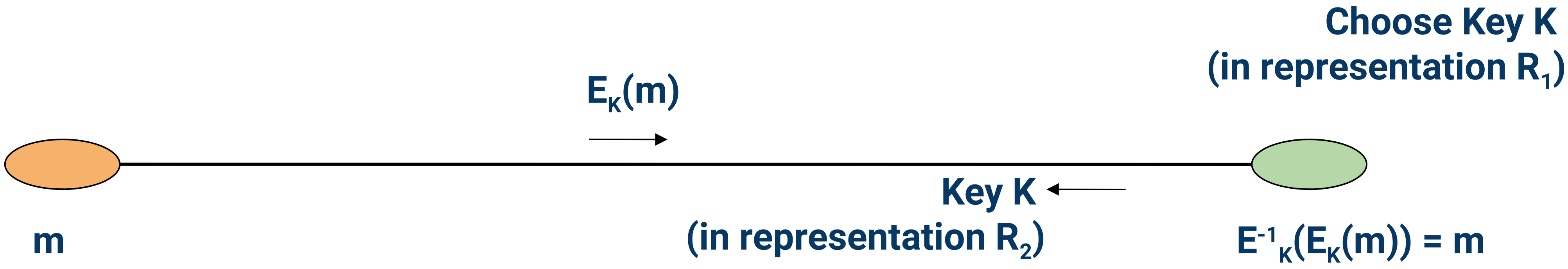
<u>In Representation R_2</u>

- Operation E_K is FAST
- Operation E_K^{-1} is <u>VERY SLOW</u>

<u>In Representation R_1</u>

- Operation E_K^{-1} is FAST

Secure Communication



<u>In Representation R_2</u>

- Operation E_K is FAST
- Operation E_K^{-1} is <u>VERY SLOW</u>

<u>In Representation R_1</u>

- Operation E_K^{-1} is FAST

<u>EXAMPLE RSA Cryptosystem</u>

R_1 : Product of Primes

R_2 : Decimal

E_K : Modular Exponentiation
$m^e \bmod K$

Our second exemplary problem

Is Collision-Resistant Hashing a Logical No-Go?

Is Collision-Resistant Hashing a Logical No-Go?

Yes!

Why?

Compression Leads To Collisions!

Compression Leads To Collisions!

Ans: Hash functions take arbitrary length strings and compress them into shorter strings.

Compression Leads To Collisions!

Ans: Hash functions take arbitrary length strings and compress them into shorter strings.

Compression **implies** Collision!

How to Circumvent the Impossibility?

How to Circumvent the Impossibility?

Ans: Non-Omnipotence of the collision finder!

How to Circumvent the Impossibility?

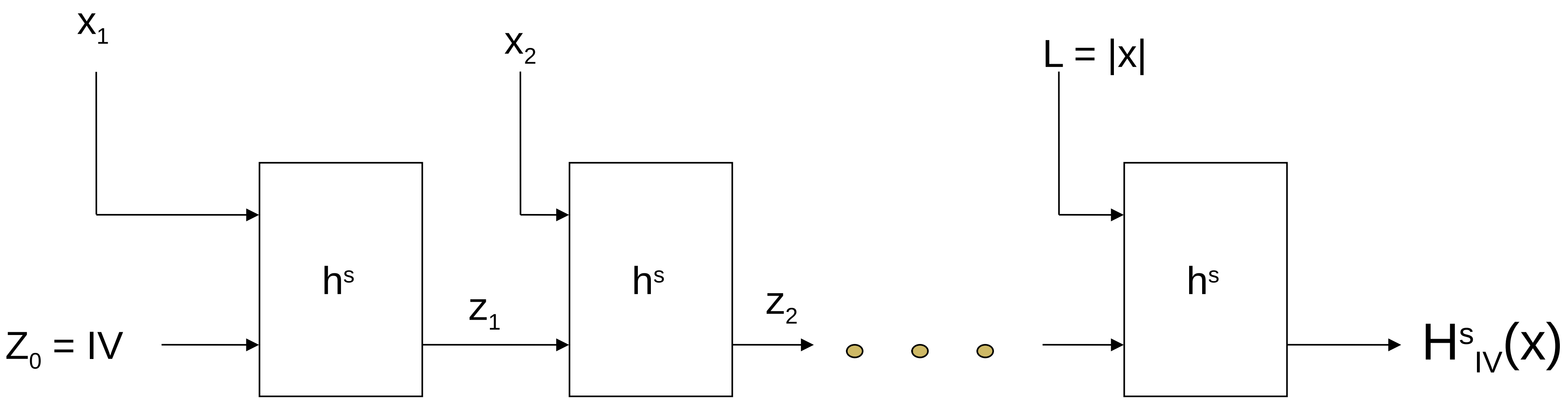
Ans: Non-Omnipotence of the collision finder!

Amplifying compression with collision-resistance:

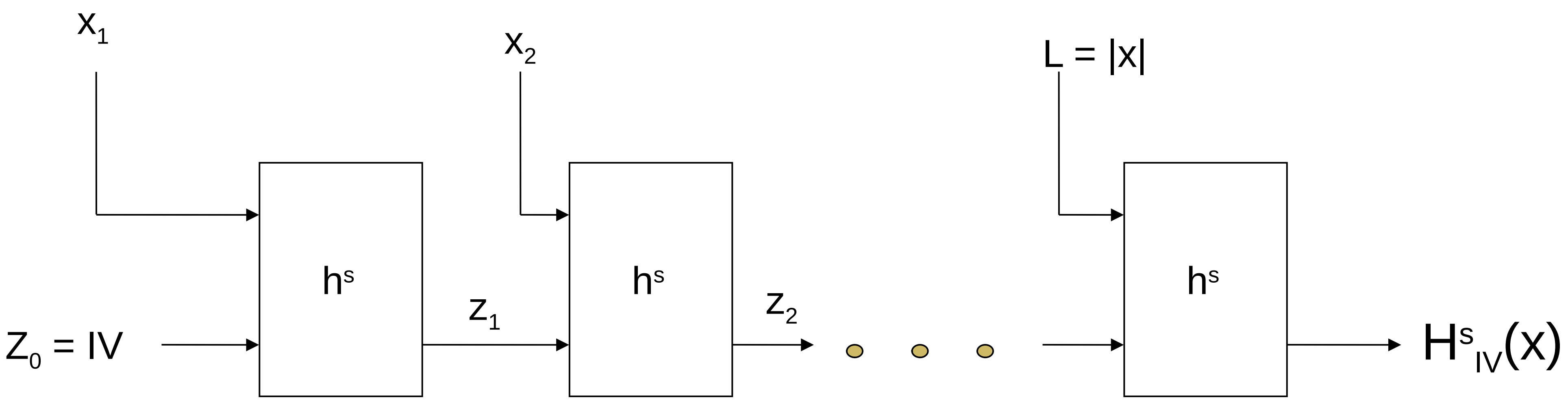
The Merkle-Damgard Transform

Merkle Damgard Transform

Merkle Damgard Transform



Merkle Damgard Transform



Theorem: If (Gen,h) is a fixed length collision resistant hash function, then (Gen, H) is a collision resistant hash function.

Our third exemplary problem

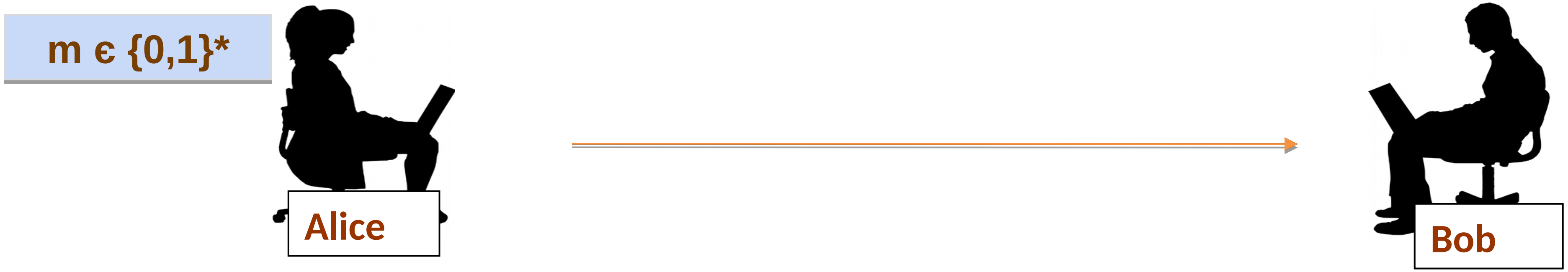
Digital Signatures

Digital Signatures

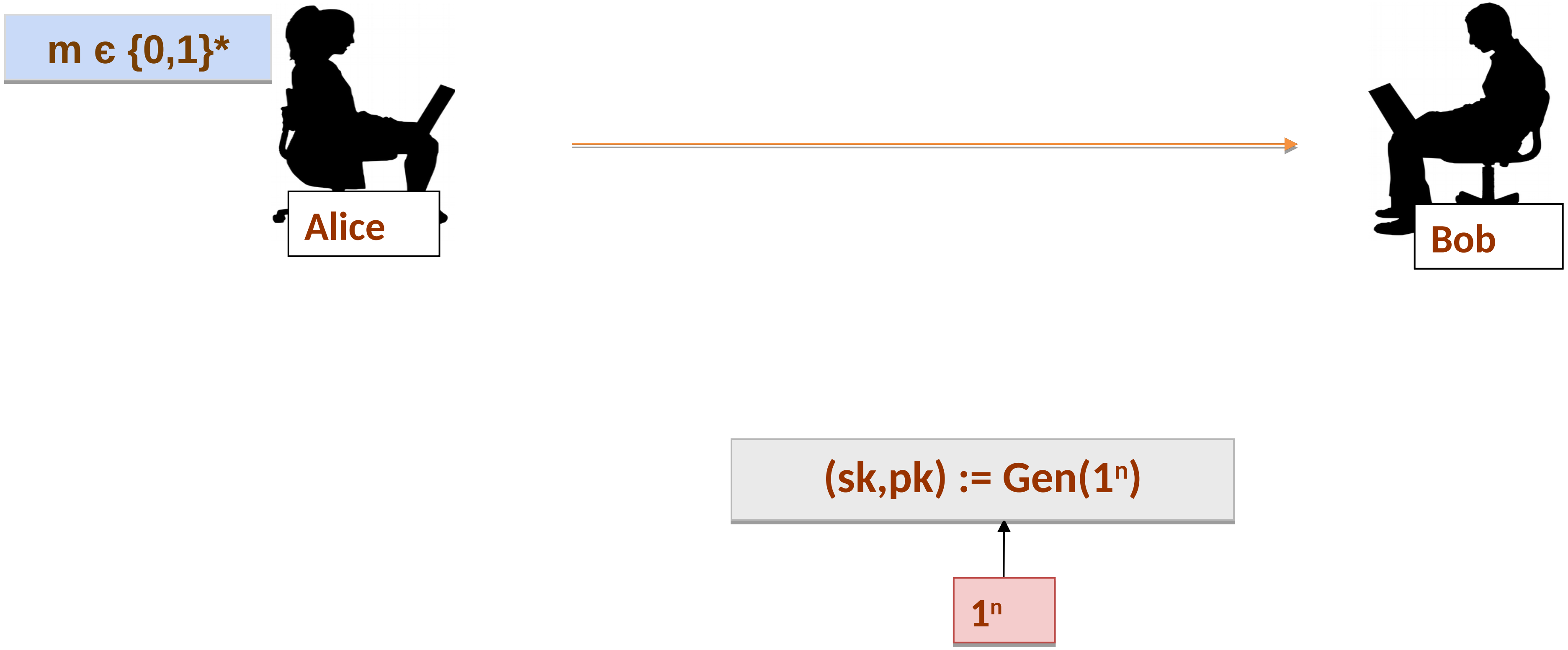
Perfect Signatures are Impossible!

Why?

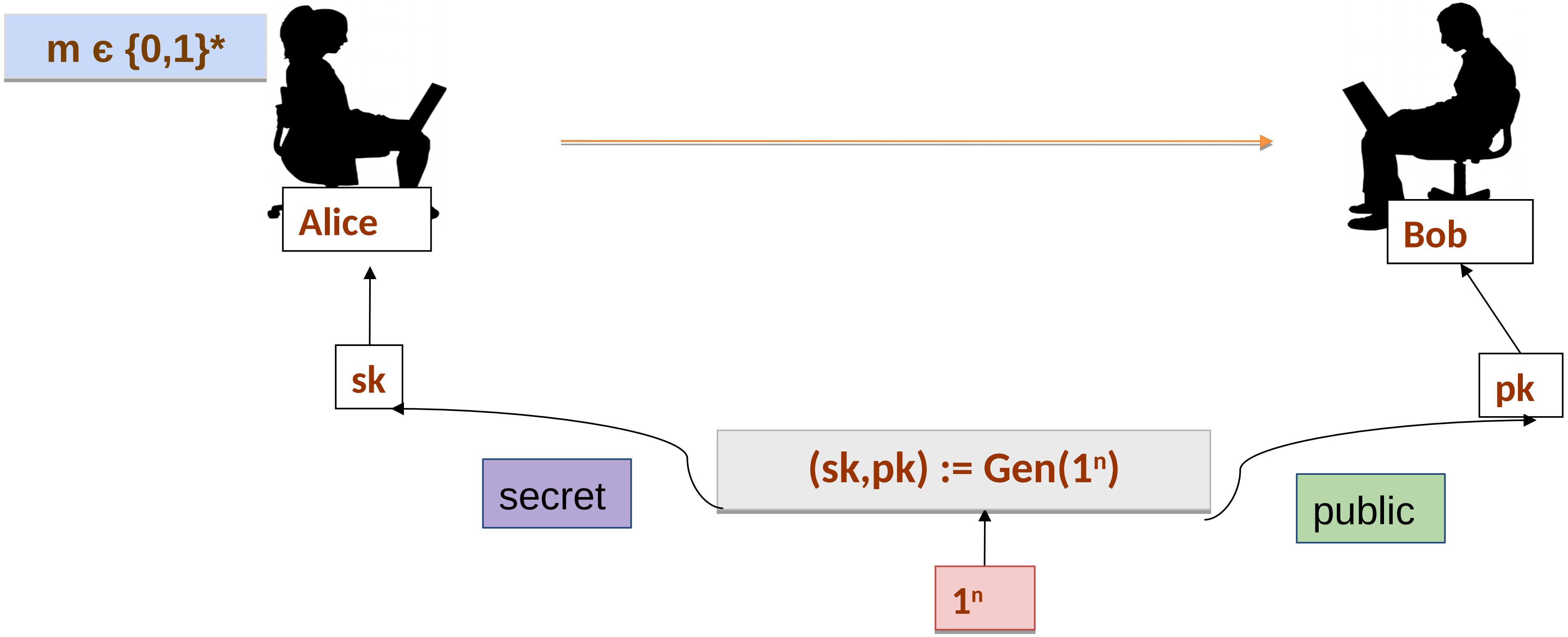
Digital Signature Scheme



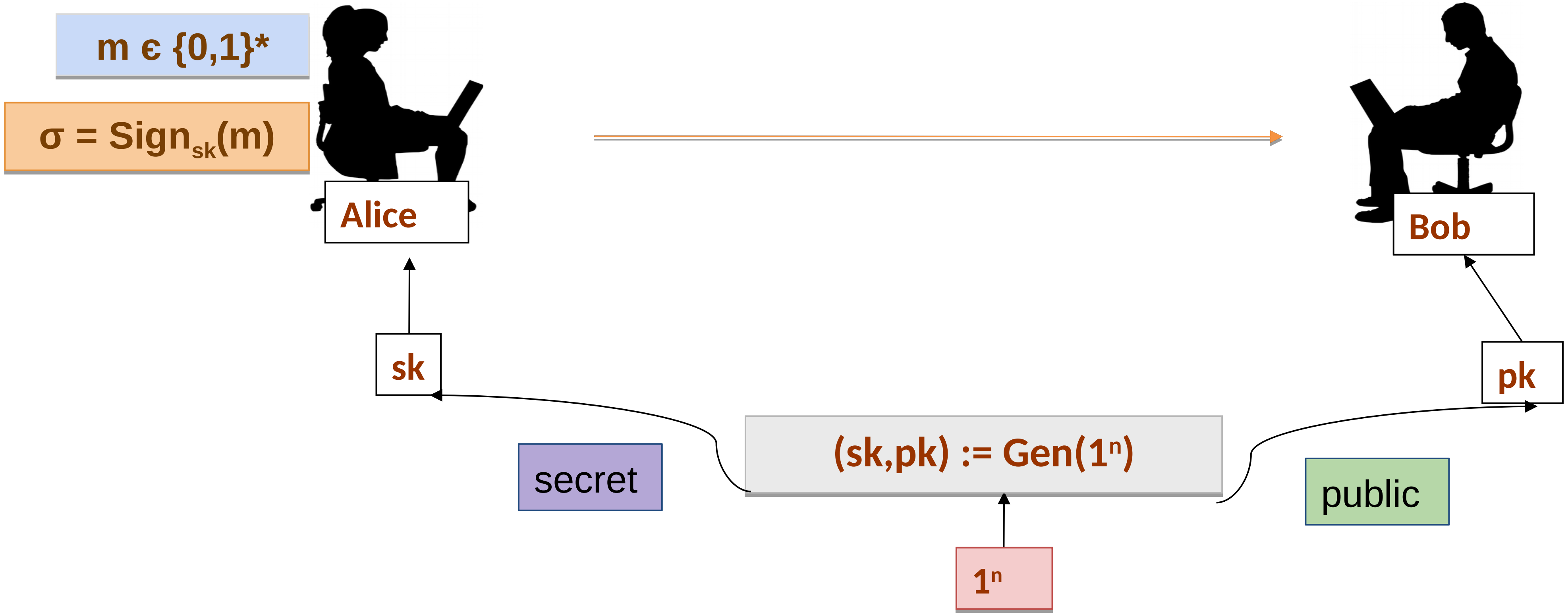
Digital Signature Scheme



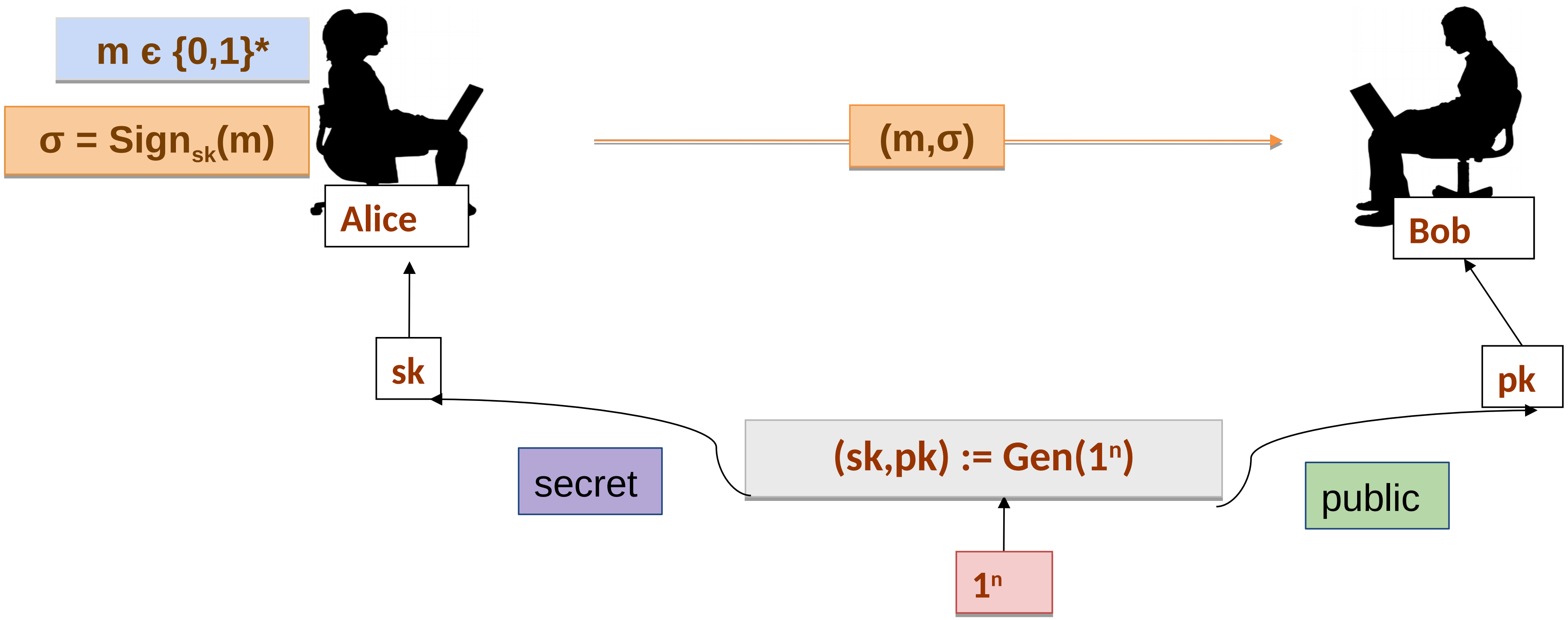
Digital Signature Scheme



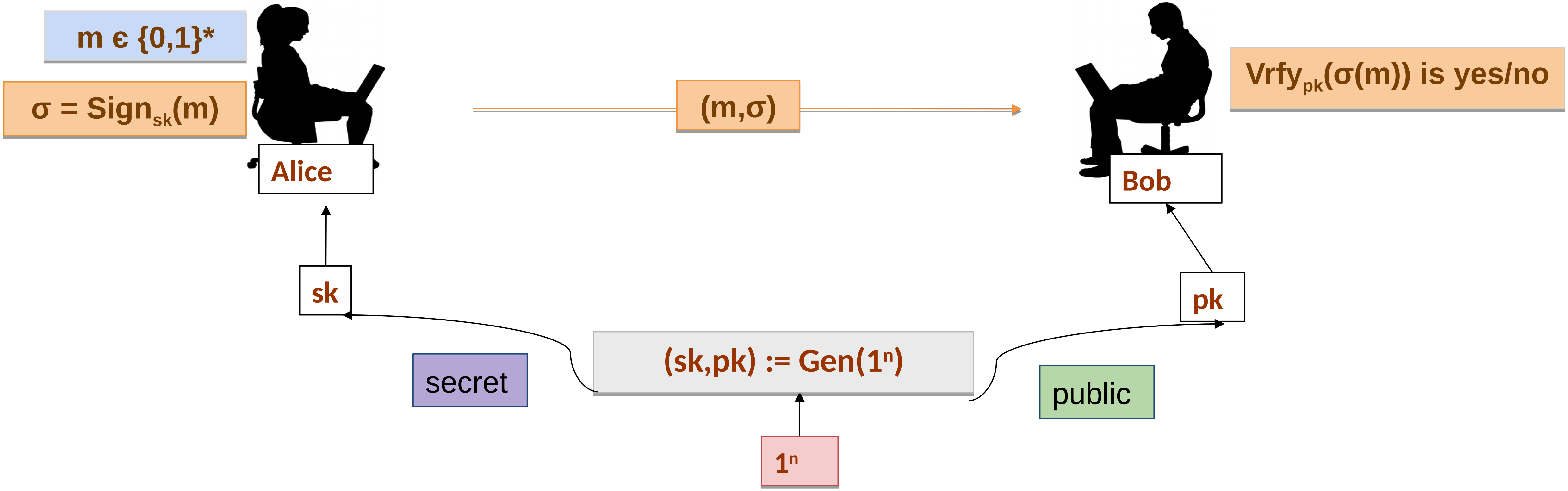
Digital Signature Scheme



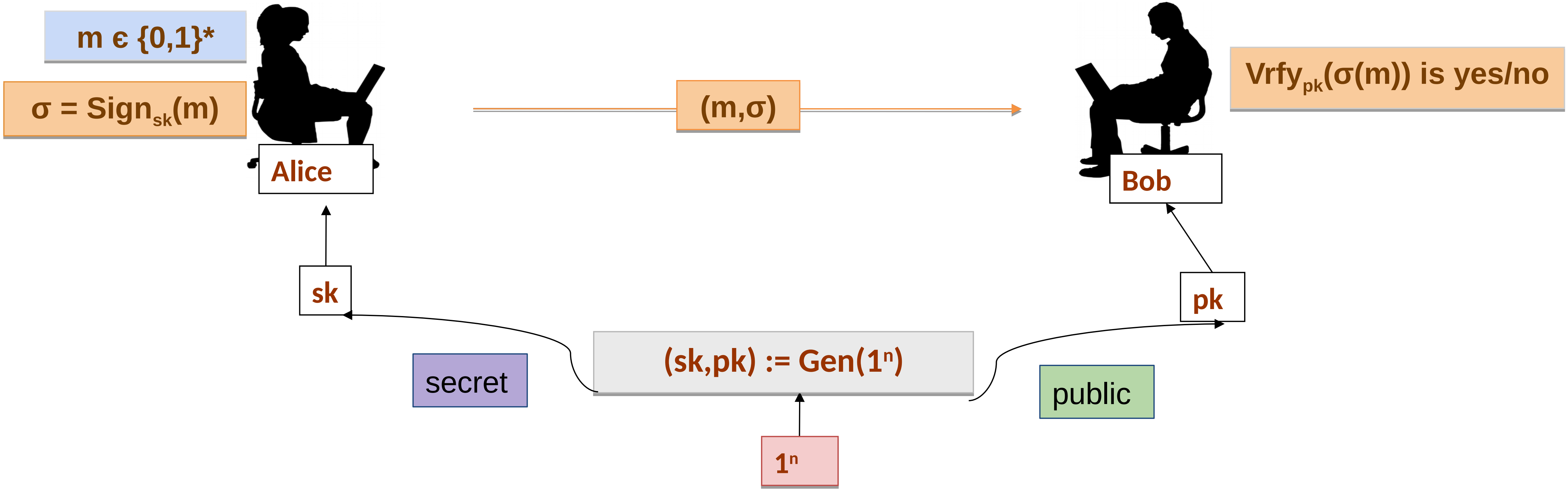
Digital Signature Scheme



Digital Signature Scheme



Digital Signature Scheme



Assuming non-omnipotence:
Forging Alice’s signature without knowing sk is computationally hard

Cryptography requisites

Cryptography requisites

- Hashing
  - SHA256, RIPEMD
- Signing
  - ECDSA, DSS
- Public-key Schemes
  - RSA, ElGamal

MyCoin: A Use-Case for Public Key Cryptography,Collision-Resistant Hashing and Digital Signatures

MyCoin

MyCoin

Let’s say I want to design my own coin (MyCoin), what are the basic points to keep in mind:

MyCoin

Let's say I want to design my own coin (MyCoin), what are the basic points to keep in mind:

1. a) I should be able to generate MyCoin.
   b) Anyone should be able to verify that MyCoin belongs to me.

MyCoin

Let’s say I want to design my own coin (MyCoin), what are the basic points to keep in mind:

1. a) I should be able to generate MyCoin.
   b) Anyone should be able to verify that MyCoin belongs to me.

2. I, as a owner of the MyCoin, should be able to spend it (and transfer ownership).

MyCoin

The Structure of MyCoin:

CoinID : CreateCoin()

My Publicly verifiable Unique Signature, $Sign_{me}$: Sign($PrivK_{me}$)

MyCoin

Transaction in MyCoin:

$Sign_{me}$: Sign($PrivK_{me}$)

Pay to $PubK_{you}$: H( )

CoinID : CreateCoin()

$Sign_{me}$: Sign($PrivK_{me}$)

MyCoin

Multiple Transaction in MyCoin:

MyCoin

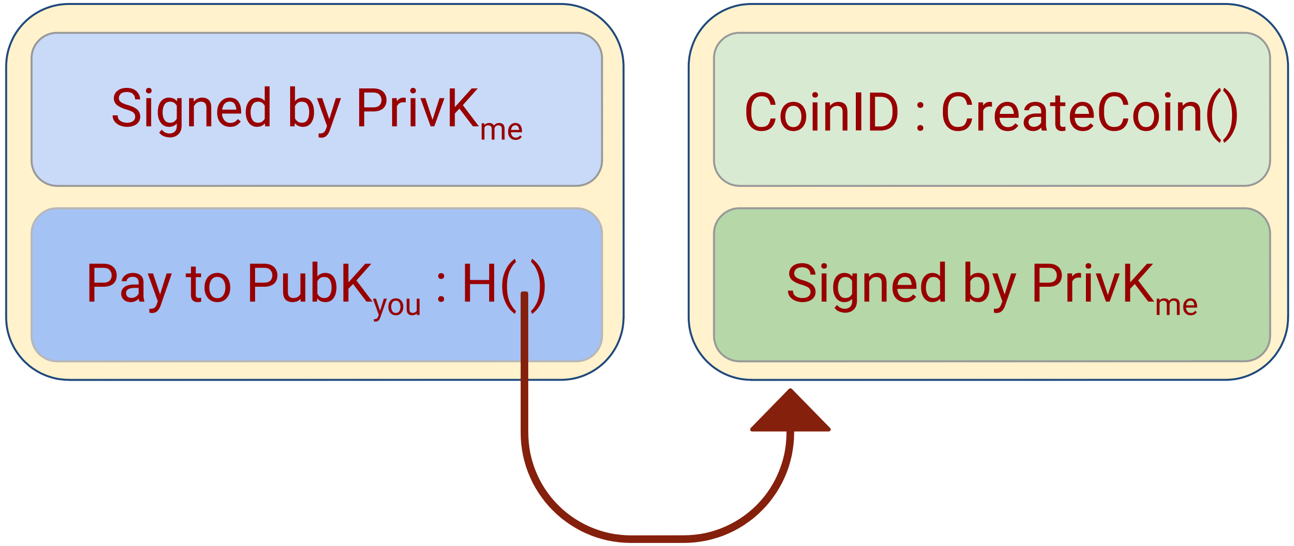
Multiple Transaction in MyCoin:

CoinID : CreateCoin()

Signed by PrivK_{me}

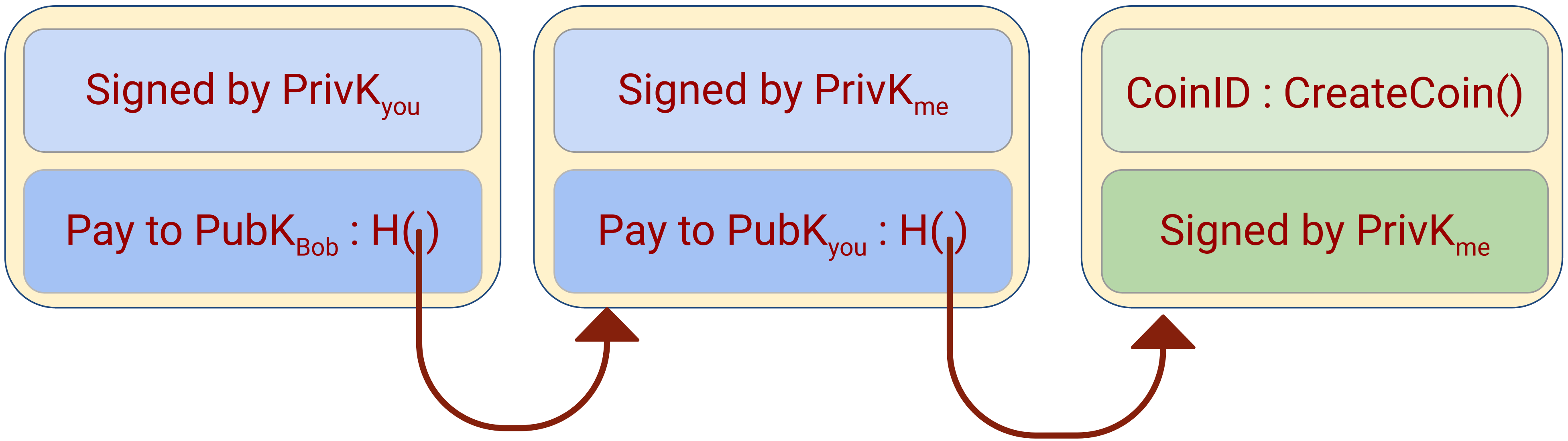
MyCoin

Multiple Transaction in MyCoin:



MyCoin

Multiple Transaction in MyCoin:



What can go wrong with MyCoin ?

What can go wrong with MyCoin ?

Consider the following transaction:

What can go wrong with MyCoin ?

Consider the following transaction:

What can go wrong with MyCoin ?

Consider the following transaction:

Signed by PrivK_{me}

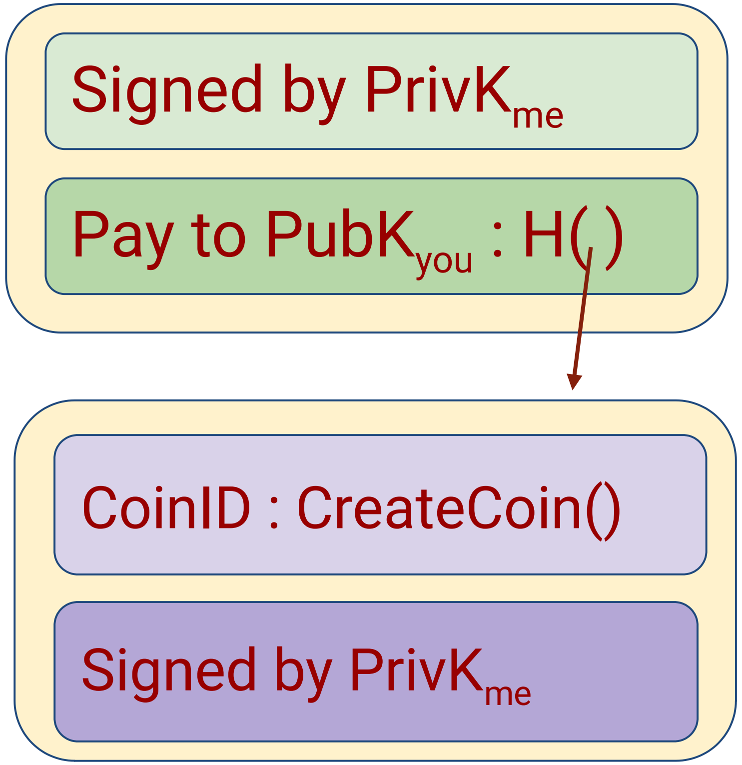
Pay to PubK_{you} : H( )

CoinID : CreateCoin()

Signed by PrivK_{me}

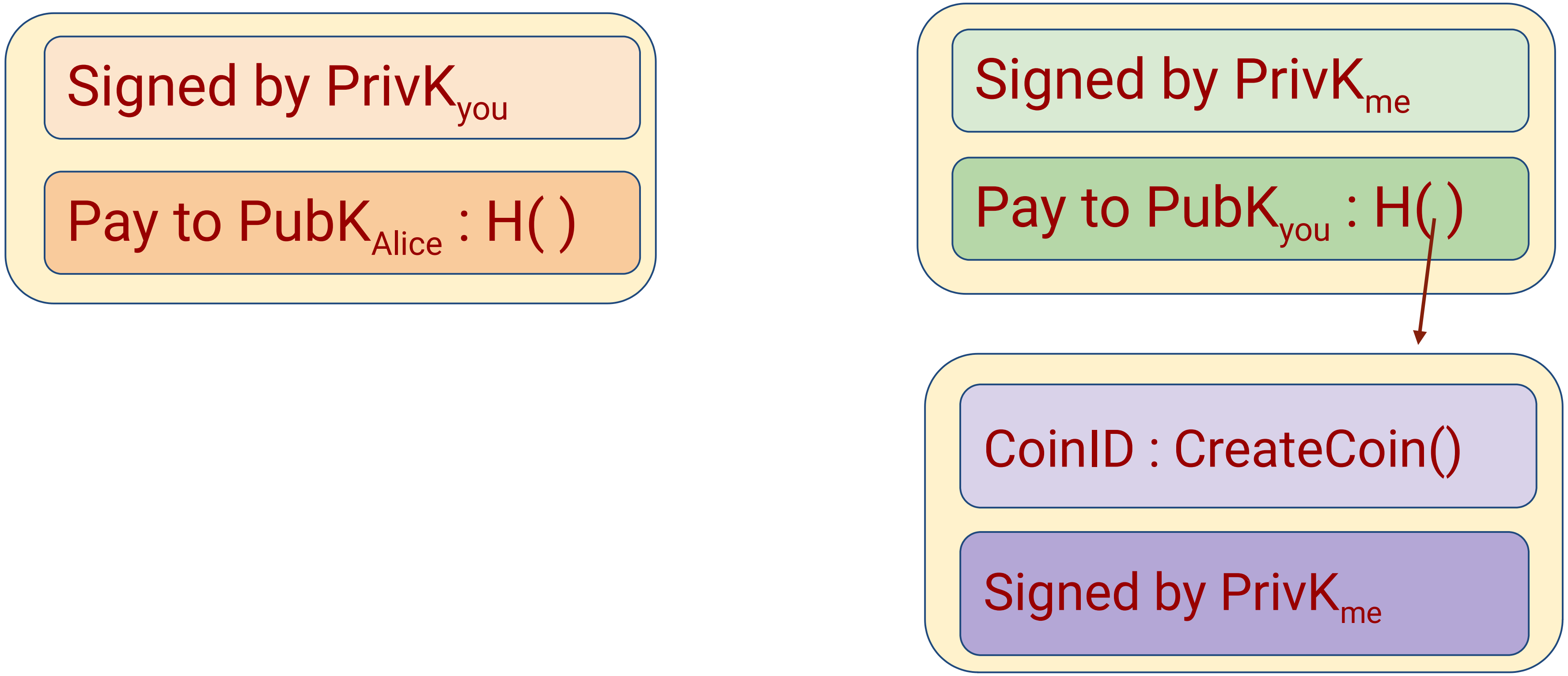
What can go wrong with MyCoin ?

Consider the following transaction:



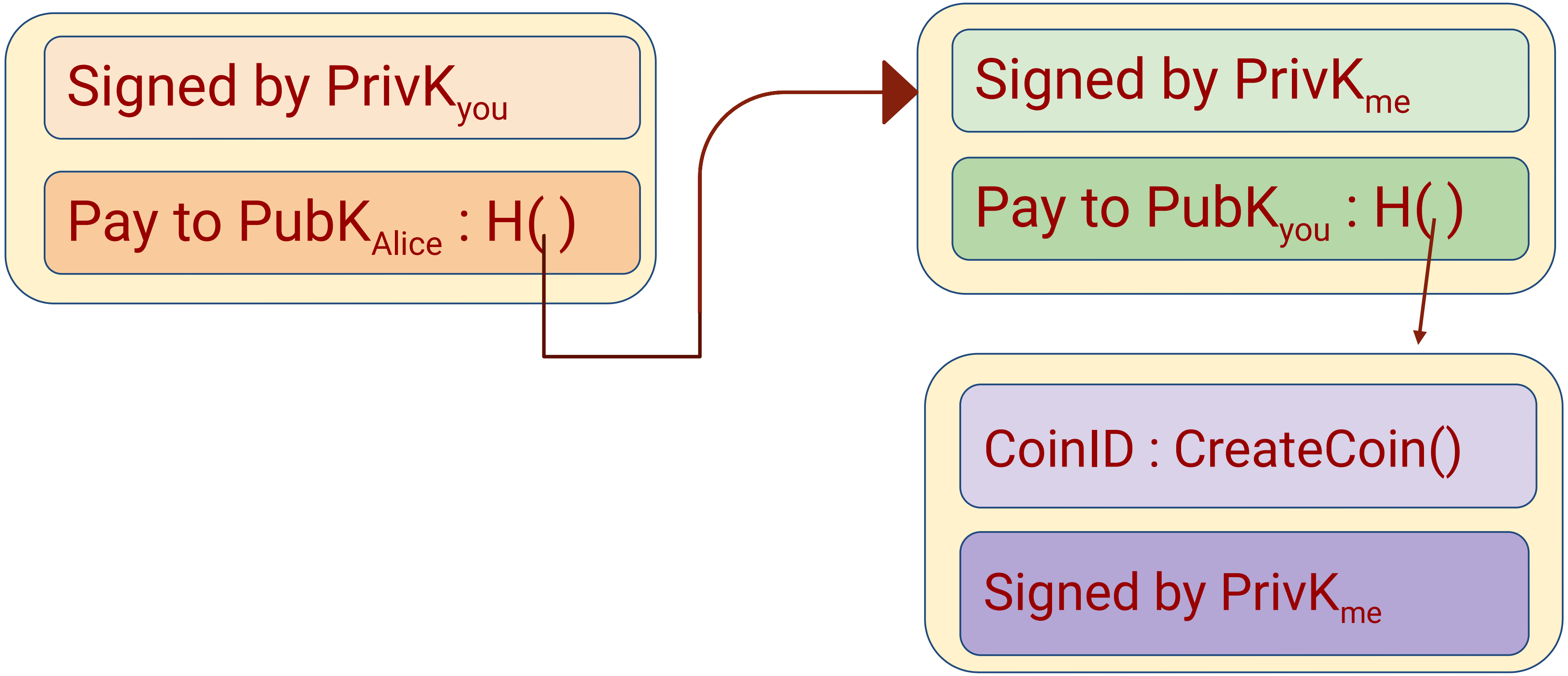
What can go wrong with MyCoin ?

Consider the following transaction:



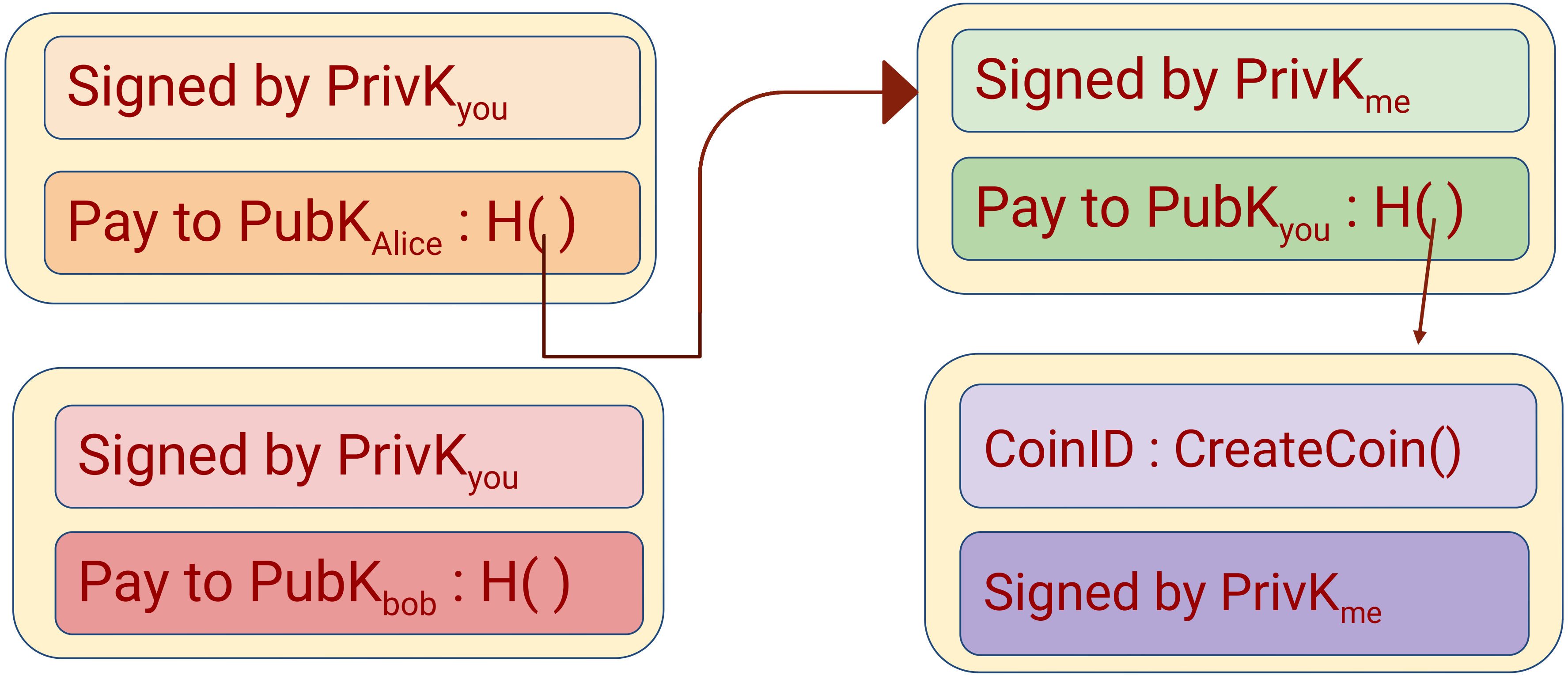
What can go wrong with MyCoin ?

Consider the following transaction:



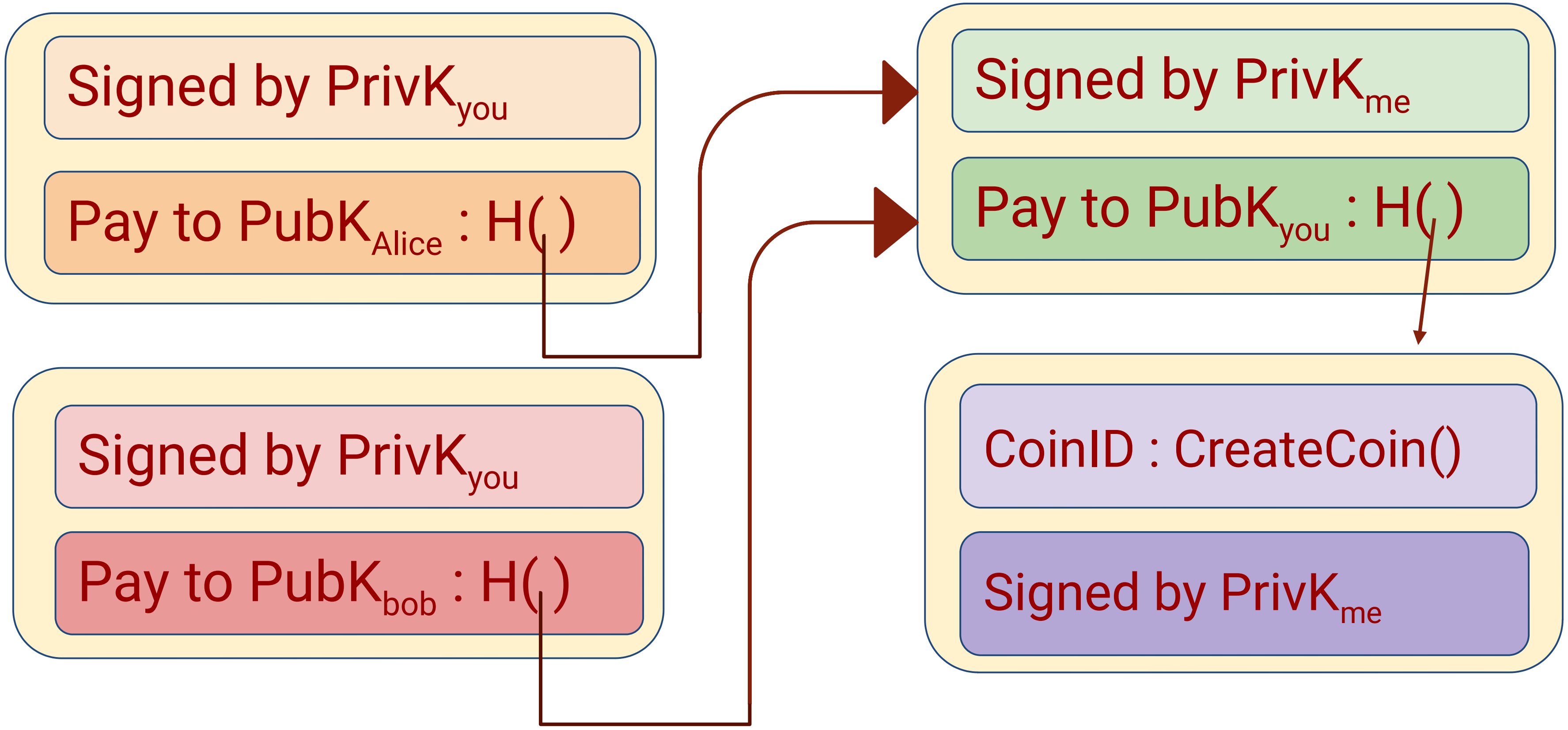
What can go wrong with MyCoin ?

Consider the following transaction:



What can go wrong with MyCoin ?

Consider the following transaction:



What can go wrong with MyCoin?

What can go wrong with MyCoin?

The Problem of **Double Spending**

What can go wrong with MyCoin?

The Problem of **Double Spending**

A tough one!

How to resolve Double Spending Problem?

How to resolve Double Spending Problem?

Use Distributed Computing (Level I)
(Introducing **TrustMeCoin**)

How to resolve Double Spending Problem?

Use Distributed Computing (Level I)
(Introducing **TrustMeCoin**)

Using Secure DC (Level II)
(Introducing **SimulatedTrustCoin**)

How does **TrustMeCoin** solve Double Spending?

How does **TrustMeCoin** solve Double Spending?

Publish The Transaction History

How does TrustMeCoin solve Double Spending?

Publish The Transaction History

H ( )

How does **TrustMeCoin** solve Double Spending?

Publish The Transaction History

Hash that is calculated and signed by me

H ( )

How does TrustMeCoin solve Double Spending?

Publish The Transaction History

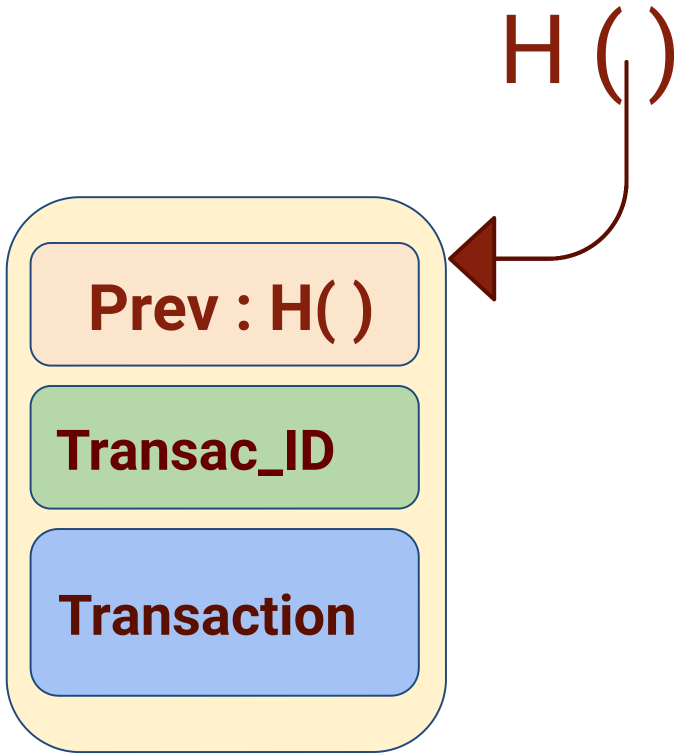
Hash that is calculated and signed by me

H ( )

How does TrustMeCoin solve Double Spending?

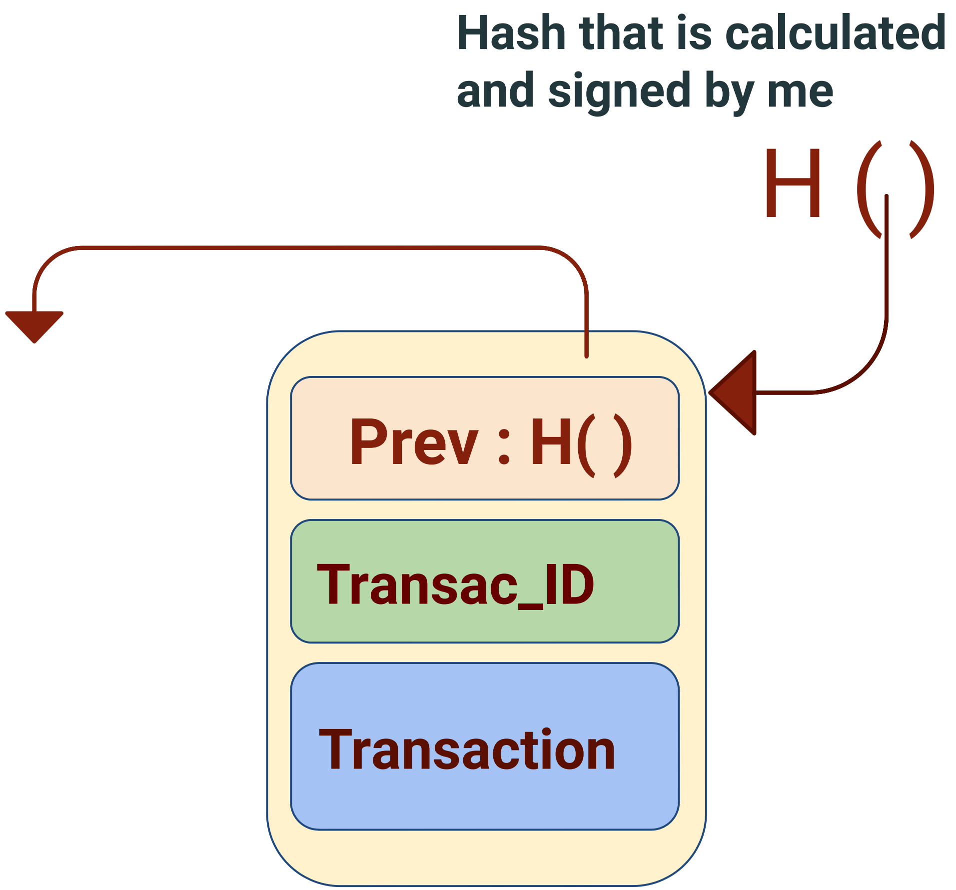
Publish The Transaction History

Hash that is calculated and signed by me



How does TrustMeCoin solve Double Spending?

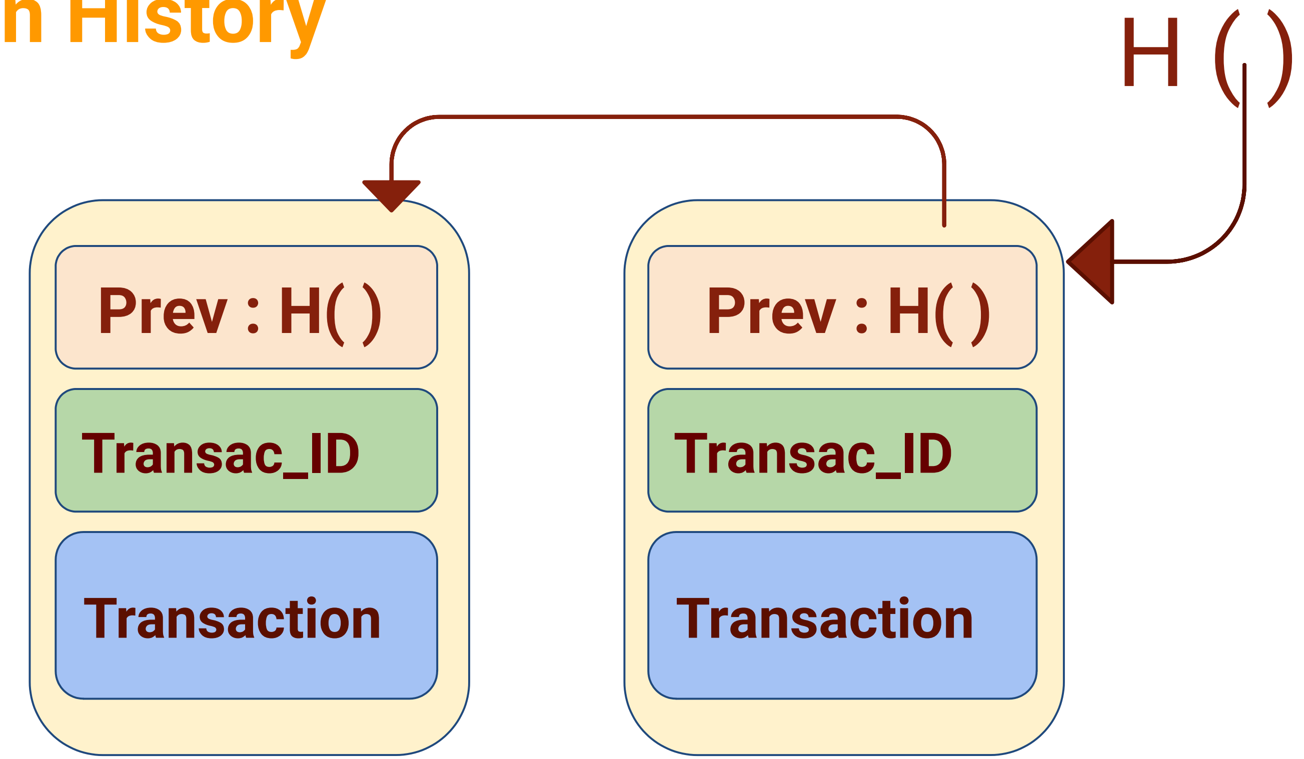
Publish The Transaction History



How does TrustMeCoin solve Double Spending?

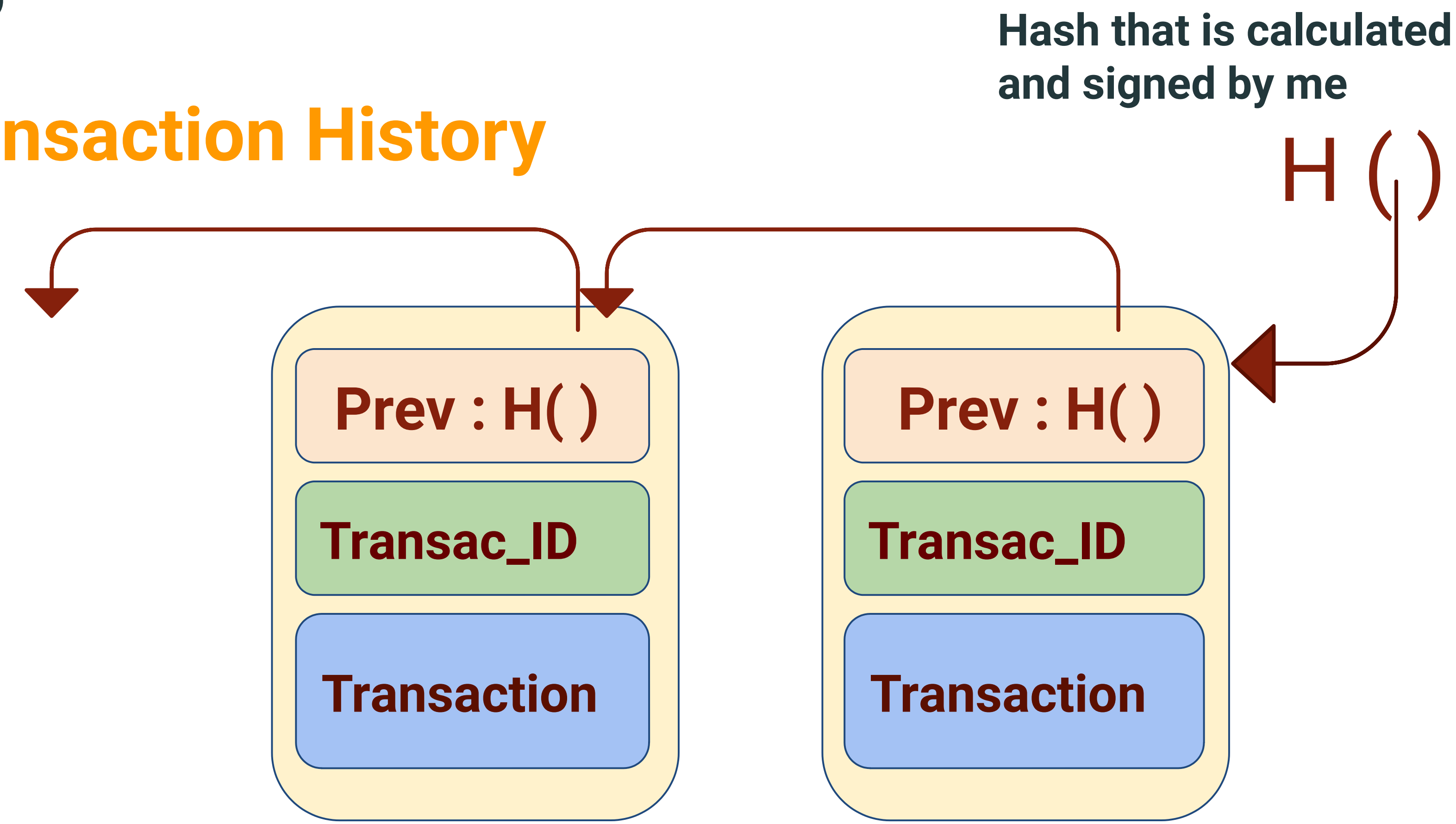
Publish The Transaction History

Hash that is calculated and signed by me



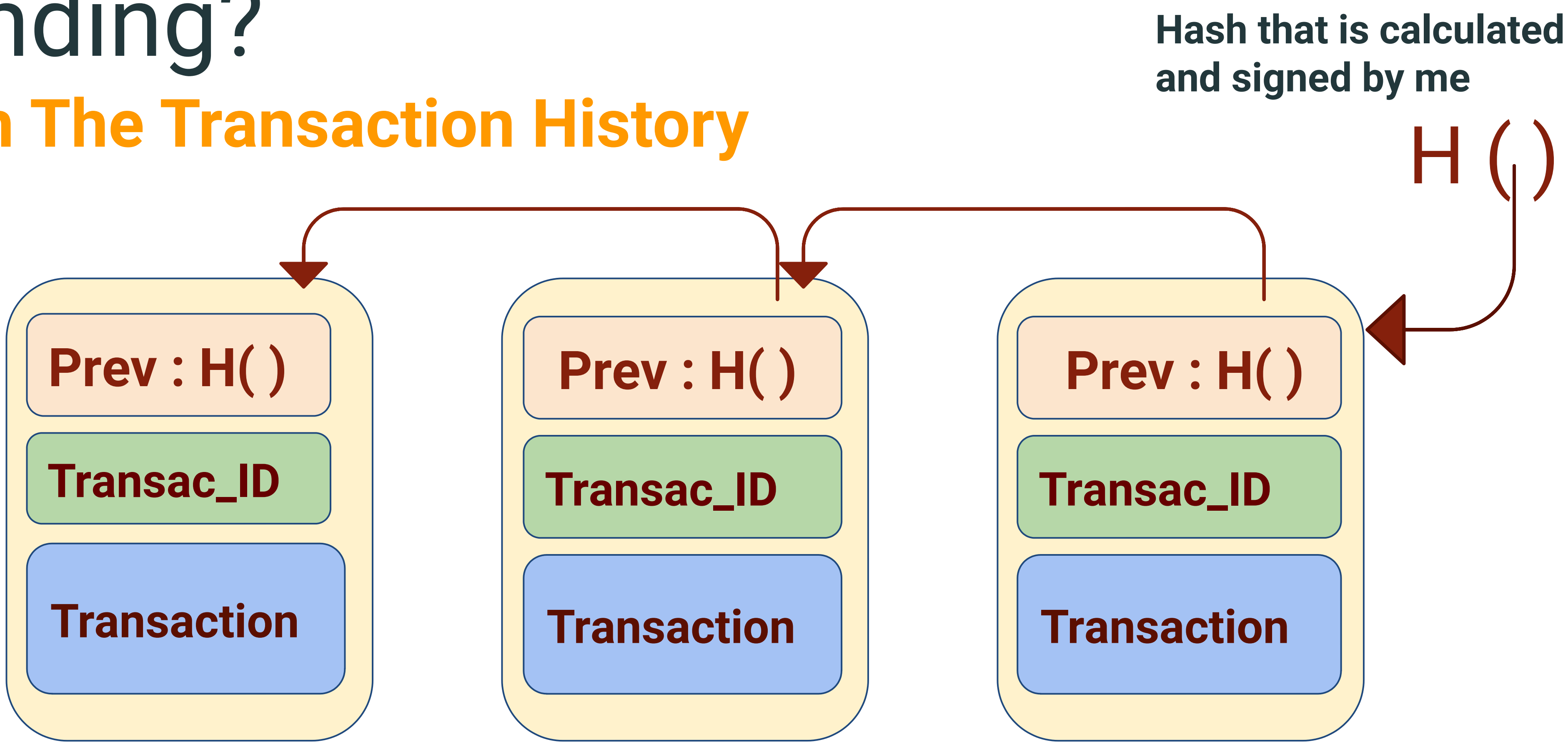
How does TrustMeCoin solve Double Spending?

Publish The Transaction History



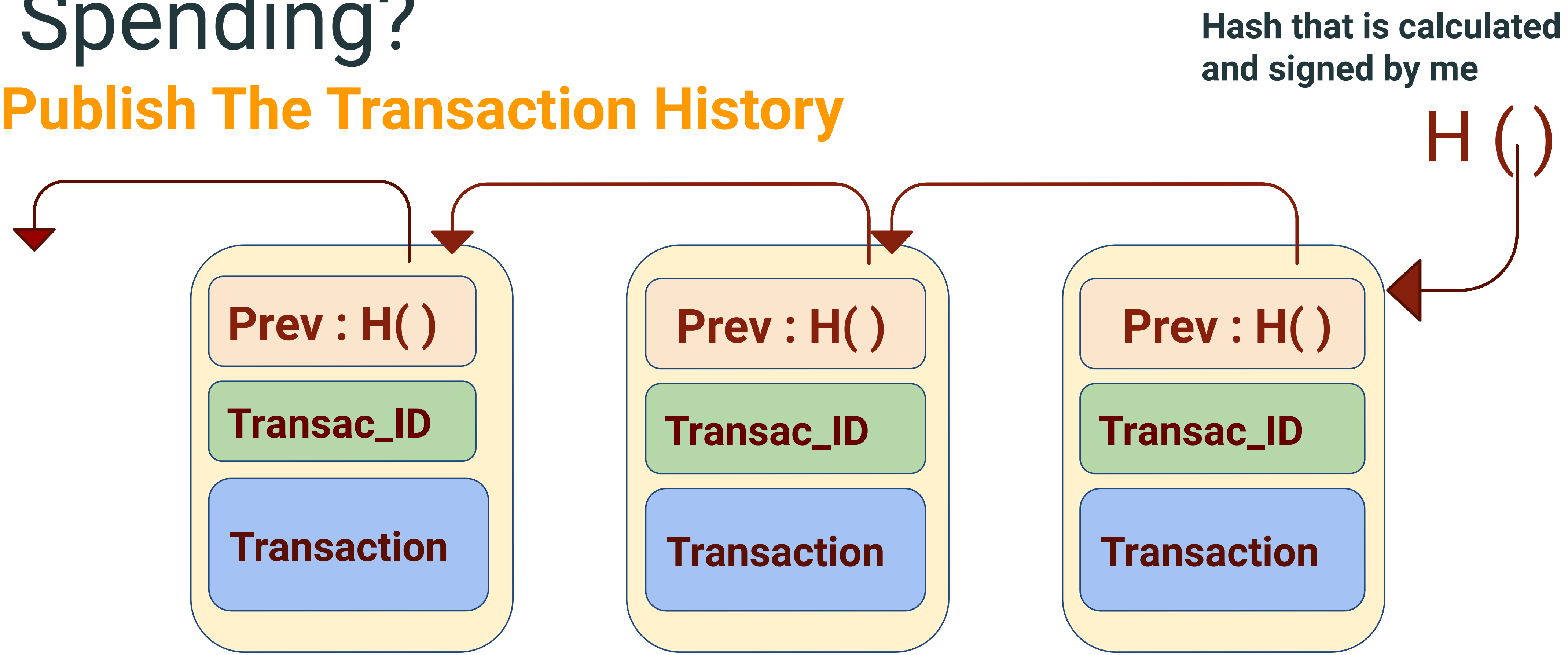
How does TrustMeCoin solve Double Spending?

Publish The Transaction History



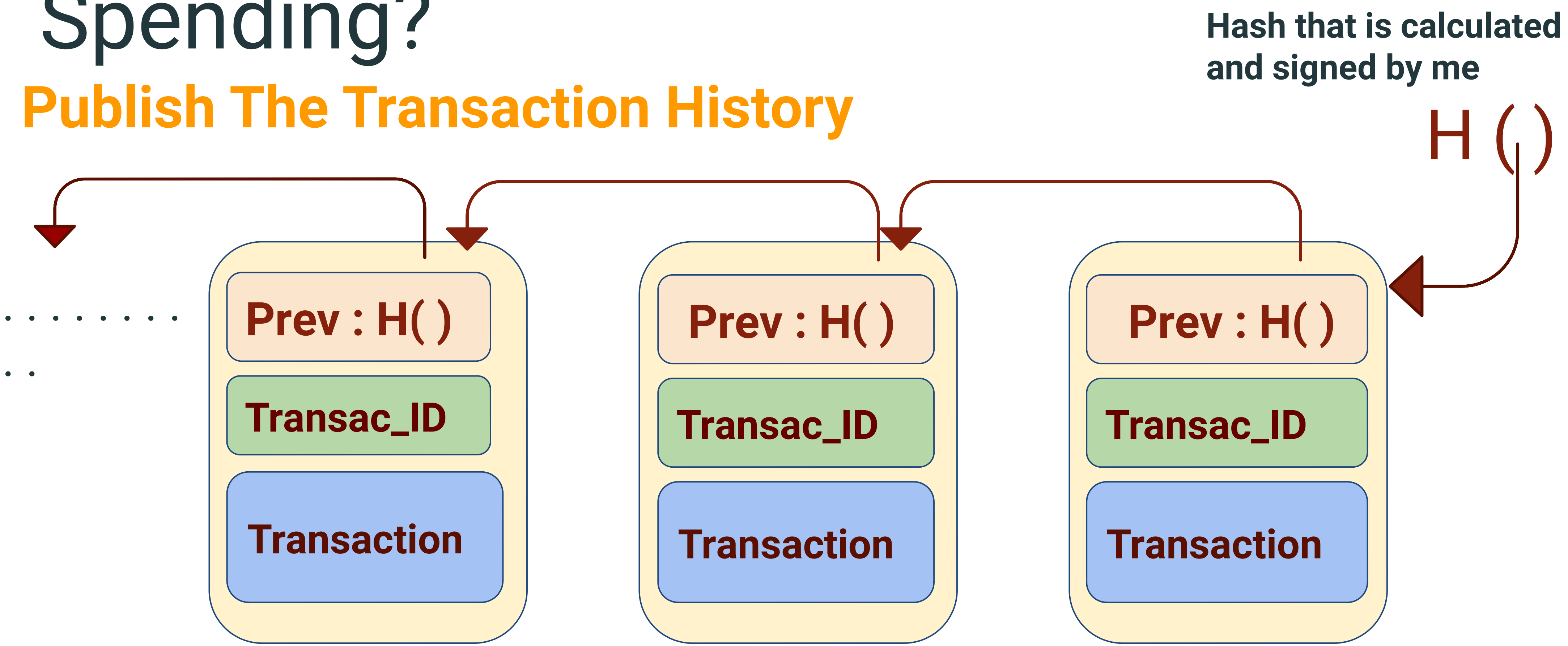
How does TrustMeCoin solve Double Spending?

Publish The Transaction History



How does TrustMeCoin solve Double Spending?

Publish The Transaction History



TrustMeCoin : Two types of Transactions

TrustMeCoin : Two types of Transactions

Transaction Type 1 : CreateCoin, for generating new coins

TrustMeCoin : Two types of Transactions

Transaction Type 1 : CreateCoin, for generating new coins

TransId : 53 Type : CreateCoin

Coin Created

Num	Value	Recipient
1	4.7	0x……...
2	3.7	0x……...
3	2.3	0x……...

Coin Id : 53 (0)

Coin Id : 53 (1)

Coin Id : 53 (2)

TrustMeCoin : Two types of Transactions

Transaction Type 1 : CreateCoin, for generating new coins

TransId : 53 Type : CreateCoin

Coin Created

Num	Value	Recipient
1	4.7	0x……...
2	3.7	0x……...
3	2.3	0x……...

Every created coin is valid , because I have the rule that all coins generated by me are valid.

Coin Id : 53 (0)

Coin Id : 53 (1)

Coin Id : 53 (2)

TrustMeCoin : Two types of Transactions

Some reference taken from free draft of Bitcoin and Cryptocurrency Technologies

TrustMeCoin : Two types of Transactions

Transaction Type 2 : PayCoin, for payment using TrustMeCoin

TrustMeCoin : Two types of Transactions

Transaction Type 2 : PayCoin, for payment using TrustMeCoin

TransId : 67 Type : PayCoin

Input Coin Id(s): 53(0), 53(1), 53(2)

Coin Created

Num	Value	Recipient
1	3.7	0x……...
2	2.3	0x……...
3	4.7	0x……...

Coin Id : 67 (0)

Coin Id : 67 (1)

Coin Id : 67 (2)

Some reference taken from free draft of Bitcoin and Cryptocurrency Technologies

TrustMeCoin : Two types of Transactions

Transaction Type 2 : PayCoin, for payment using TrustMeCoin

TransId : 67 Type : PayCoin

Input Coin Id(s): 53(0), 53(1), 53(2)

Coin Created

Num	Value	Recipient
1	3.7	0x……..
2	2.3	0x……..
3	4.7	0x……..

Coin Id : 67 (0)

Coin Id : 67 (1)

Coin Id : 67 (2)

Transaction is Valid if:
- Consumed coins are valid.
- Not Already Consumed.
- Total input = Total Output.
- All consumed coin owners have signed it.

Some reference taken from free draft of Bitcoin and Cryptocurrency Technologies

The problem with **TrustMeCoin**?

The problem with **TrustMeCoin**?

Trust is the Problem.

TrustMeCoin is robust if only you ***trust me***!

Decentralizing the Trust-on-Me

Decentralizing the Trust-on-Me

A Not-So-Simple Distributed Cryptographic Approach

(accentuates the simplicity of blockchains)

How To Solve It ?

(By Non-Omnipotence of machines?)

How To Solve It ?

(By Non-Omnipotence of machines?)

Welcome to **Distributed (Crypto) Computing**!

Our first exemplary problem (non-omnipresence)

Secret Sharing

t-Secret Sharing

Shamir's Protocol:

- Choose a t-degree polynomial p() over a (large enough) finite field
- Let Secret s = p(0)
- Shares are p(1), p(2), …p(n) for some n

*Ref: **Adi Shamir. How to Share a Secret. CACM, 1979***

Our second exemplary problem

Recall: Secure Communication

Secure Communication: Distributed Algorithmic Solution

- Essentially the distribution helps in improving security. Even perfect security is achievable in some cases.
- Key Ingredient: Secret Sharing

Perfectly Secure Communication

Dolev *et al.* Protocol:

1. Sender shares the message using secret sharing and sends the shares to Receiver along different paths.

2. Receiver collects all the shares, and reconstructs the message.

Ref: ***Danny Dolev, Cynthia Dwork, Orli Waarts, Moti Yung Perfectly secure message transmission, J. ACM 40, 1 (1993), 17-47.***

Recall: Byzantine Agreement

Authenticated Byzantine Agreement (ABA)

- Computationally Bounded Adversary Model
- Processes are supplemented with “magical powers” to a*uthenticate* their communication – *Digital Signatures*.
- Using authentication, fault tolerance can be increased to $t < n$.

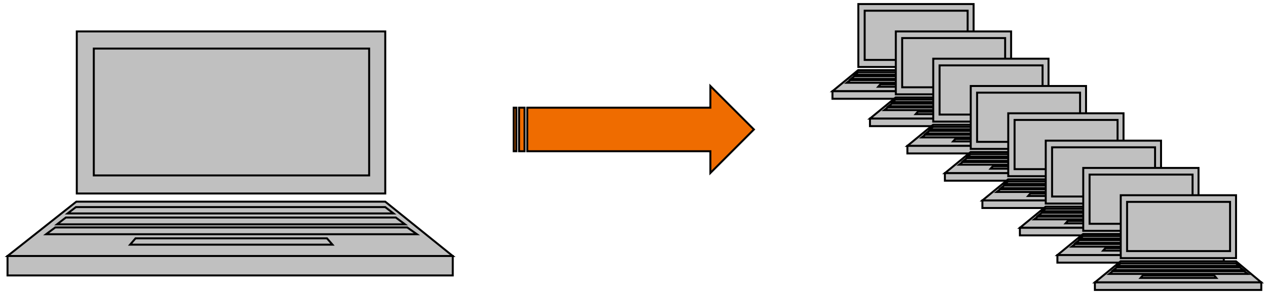
ABA protocol for 1 out of 3

- Player k maintains a set W_{ik}, $\forall\, i \in P$. Initially $W_{kk}=\{\sigma\}$ where σ is player k's input value.
- Repeat the following steps for 2 rounds:
  - Receive values from neighbors and for each received value do:
    - If the message is properly signed, he append its content to the set W_{ik}
    - Sends i, W_{ik} to his neighbors.
- He deletes W_{ik} if $|W_{ik}| \neq 1$.
- Since all remaining W_{ik}'s are singleton, he takes majority over all values. If a majority exists he decides on it, else decides on the default value.

Our last exemplary problem

Secure Multiparty Computation

Simulating Secure Nodes: Basic Idea



- The data in each memory register of the secure virtual server is secret shared and stored across several nodes in the network
- Each CPU instruction of the secure virtual server is simulated via a network protocol
- Our instruction set architecture: XOR, AND, SEND and RECEIVE

Secure Addition Protocol

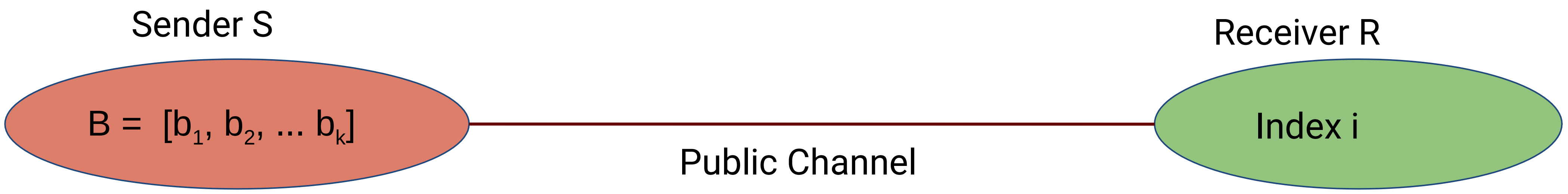
Simple: Local addition entails global addition!

Secure Multiplication Protocol

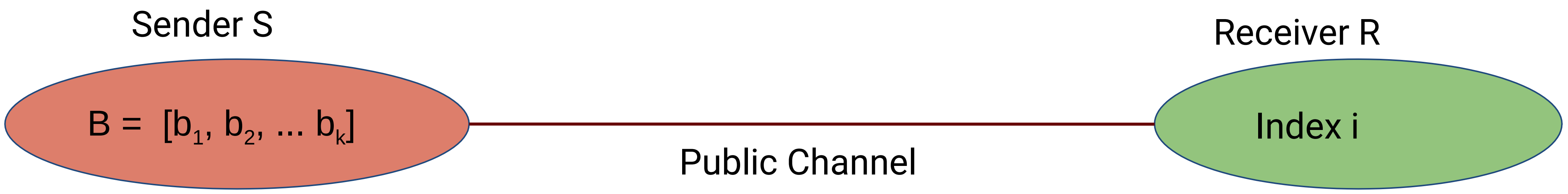
Cryptographic Solution:
Uses Oblivious Transfer

Oblivious Transfer

Oblivious Transfer



Oblivious Transfer



Objective :

- R should be able know b_i without revealing i to S.
- S should be able to send b_i to R without revealing $b_{j \neq i}$

Oblivious Transfer (Contd.)

Oblivious Transfer (Contd.)

- Step 1: R selects uniformly at random k strings $x_1,x_2,\ldots,x_k$ and sets
  $y_i = Enc_{PubKey(S)}(x_i)$ and $y_j = x_j$ for every $j \neq i$; R sends $y_1,y_2,\ldots,y_k$ to S.

Oblivious Transfer (Contd.)

- Step 1: R selects uniformly at random k strings $x_1,x_2,\ldots,x_k$ and sets
  $y_i = Enc_{PubKey(S)}(x_i)$ and $y_j = x_j$ for every $j \neq i$; R sends $y_1,y_2,\ldots,y_k$ to S.
- Step 2: S decrypts y_j's and obtains z_j's. S sets $c_j=b_j$ xor z_j. S sends $c_1,c_2,\ldots,c_k$ to R.

Oblivious Transfer (Contd.)

- Step 1: R selects uniformly at random k strings $x_1,x_2,\ldots,x_k$ and sets
  $y_i = Enc_{PubKey(S)}(x_i)$ and $y_j = x_j$ for every $j \neq i$; R sends $y_1,y_2,\ldots,y_k$ to S.
- Step 2: S decrypts y_j's and obtains z_j's. S sets $c_j=b_j$ xor z_j. S sends $c_1,c_2,\ldots,c_k$ to R.
- Step 3: R outputs c_i xor x_i

Multiplication in the Shared Domain

- Two-party Passive Adversary Case (all computations are in GF(2)).
- Party P_1 has a_1 and b_1; Party P_2 has a_2 and b_2 ; here $a=a_1+a_2$ and $b=b_1+b_2$
  (that is, a and b are secret shared).
- Objective: P_1 must obtain c_1 and P_2 must obtain c_2 such that:

$$c_1+c_2 = (a_1+a_2).(b_1+b_2).$$

Multiplication in the Shared Domain (Contd.)

- Step 1: P_1 selects c_1 uniformly at random.
- Step 2: An Oblivious Transfer is performed with P_1 as sender and P_2 as receiver, k=4
- P_1's input: $\{c_1+a_1b_1, c_1+a_1b'_1, c_1+a'_1b_1, c_1+a'_1b'_1\}$
- P_2's input: $1+2a_2+b_2$ (from the set {1,2,3,4})

Ref: *Oded Goldreich. Foundations of Cryptography. Cambridge University Press. 2001*

SimulatedTrustCoin

SimulatedTrustCoin

Candidate Solution: Decentralizing the **Trust-on-Me** by simulating a trusted third party

Simplification: Introducing Blockchains

The Blockchain Approach

The Blockchain Approach

Simplify by Game Theory

The Blockchain Approach

Simplify by Game Theory

Incentivising Honesty

The Blockchain Approach

The Blockchain Approach

- Publish the transaction history

The Blockchain Approach

- Publish the transaction history
- Anyone (randomly picked) can append a new block (not only *me*, in TrustMeCoin)

The Blockchain Approach

- Publish the transaction history
- Anyone (randomly picked) can append a new block (not only *me*, in TrustMeCoin)
    - All double spends are *detected* (like in TrustMeCoin)
    - However no centralized authority exists to correct it!

The Blockchain Approach

- Publish the transaction history
- Anyone (randomly picked) can append a new block (not only *me*, in TrustMeCoin)
  - All double spends are *detected* (like in TrustMeCoin)
  - However no centralized authority exists to correct it!
- All the peers *agree* on the correct chain. How?

The Blockchain Approach

- Publish the transaction history
- Anyone (randomly picked) can append a new block (not only *me*, in TrustMeCoin)
    - All double spends are *detected* (like in TrustMeCoin)
    - However no centralized authority exists to correct it!
- All the peers *agree* on the correct chain. How?
    - The longest chain is deemed as *correct*.
    - Appending to the longest chain is rewarded [game theory]
    - Appending elsewhere is penalized [game theory]

Proof-Of-Work

Random Selection, Incentives and Penalties: **All-in-one-Go**!

Proof-Of-Work

Random Selection, Incentives and Penalties: **All-in-one-Go**!

- If it takes some minimum effort/cost to add a new block, there is penalty if your efforts are not a part of agreement.

Proof-Of-Work

Random Selection, Incentives and Penalties: **All-in-one-Go**!

- If it takes some minimum effort/cost to add a new block, there is penalty if your efforts are not a part of agreement.
- If you are compensated for the efforts, there is a reward, *if and only if* your transaction is in the longest chain.

Proof-Of-Work

Random Selection, Incentives and Penalties: **All-in-one-Go**!

- If it takes some minimum effort/cost to add a new block, there is penalty if your efforts are not a part of agreement.
- If you are compensated for the efforts, there is a reward, *if and only if* your transaction is in the longest chain.
- If the effort is a hash-puzzle, then none can keep winning, entailing random choice.

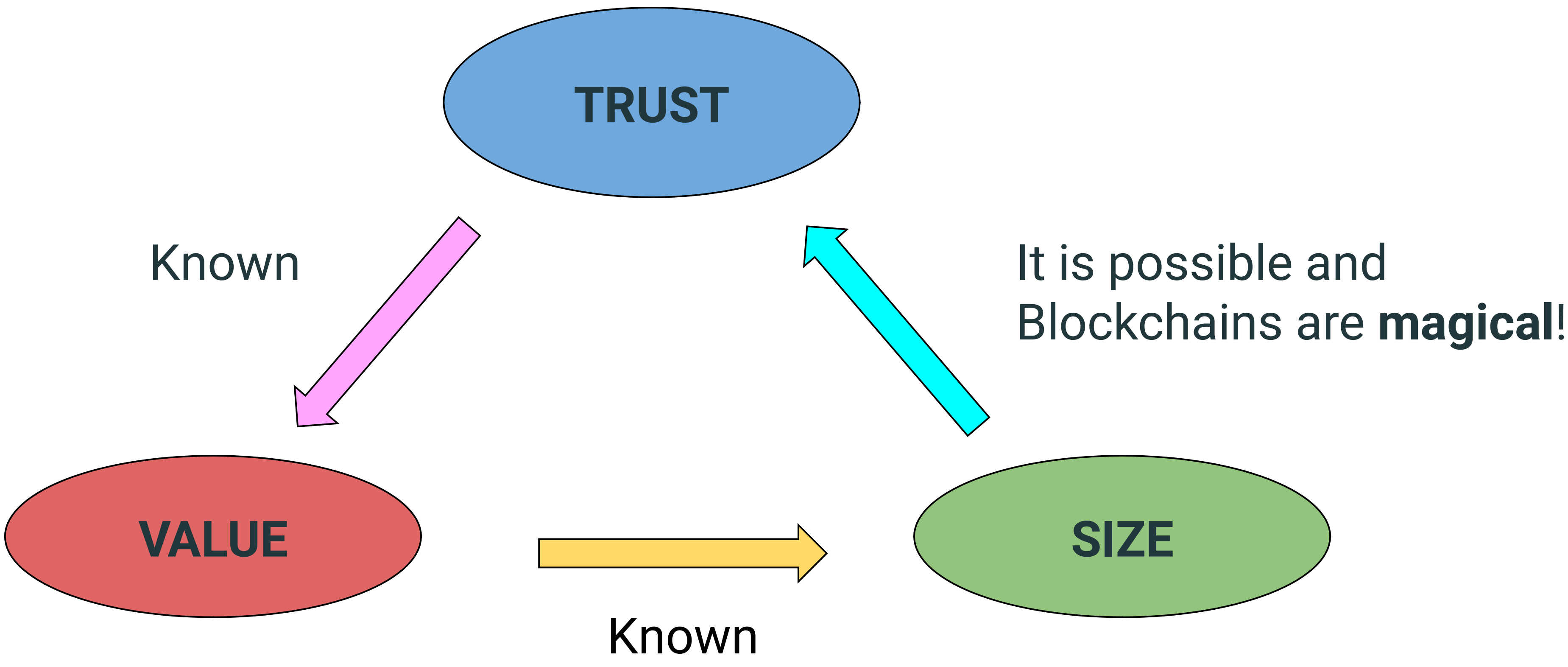
Summary

- Crypto takes us to MyCoin.

  Crypto-and-Distributed Computing takes us to SimulatedTrustCoin.

  Crypto-DC-Game-theory creates **BLOCKCHAIN**.

Conclusion


TRUST
Known
It is possible and
Blockchains are **magical**!
VALUE
SIZE
Known

Thank you.

Any questions?