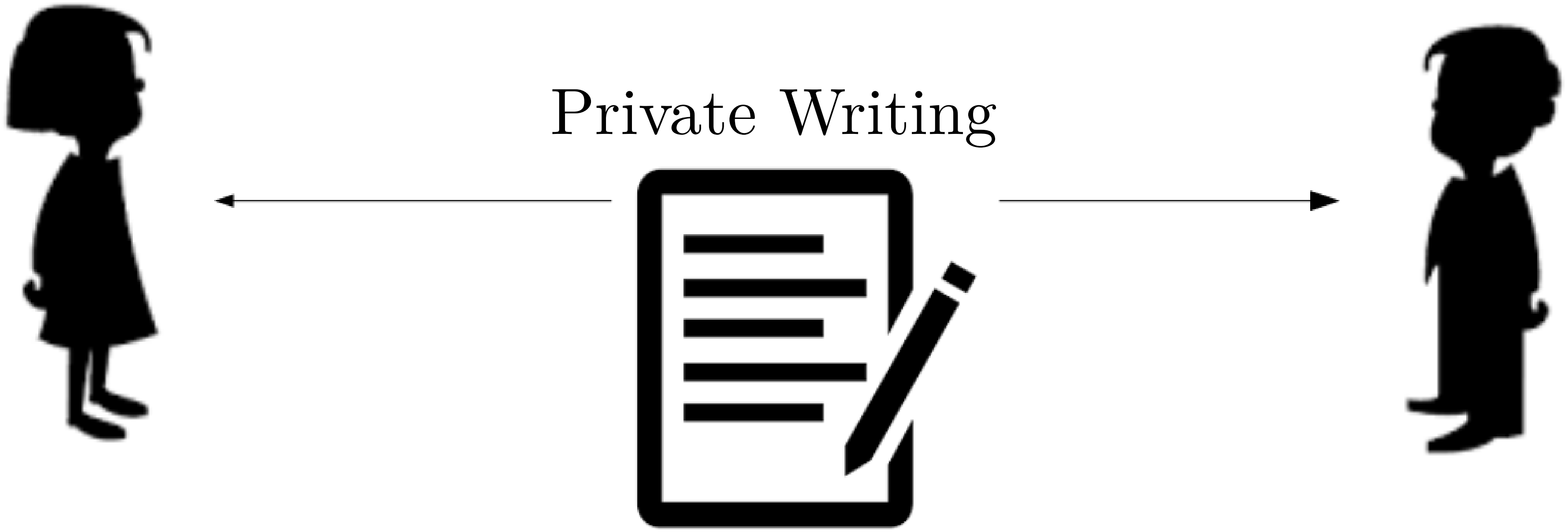
Smart Contracts and Blockchain[*]

Emanuele Natale

Minicorso su Blockchain
Dottorato in Ingegneria dell'Impresa
12 April 2019

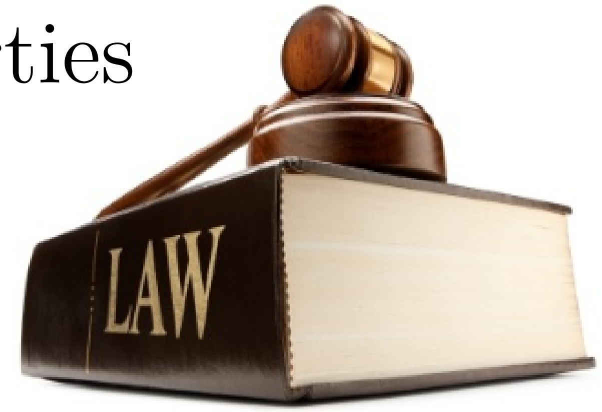
[*]Remaking of Giacomo Scornavacca's slides

Contracts



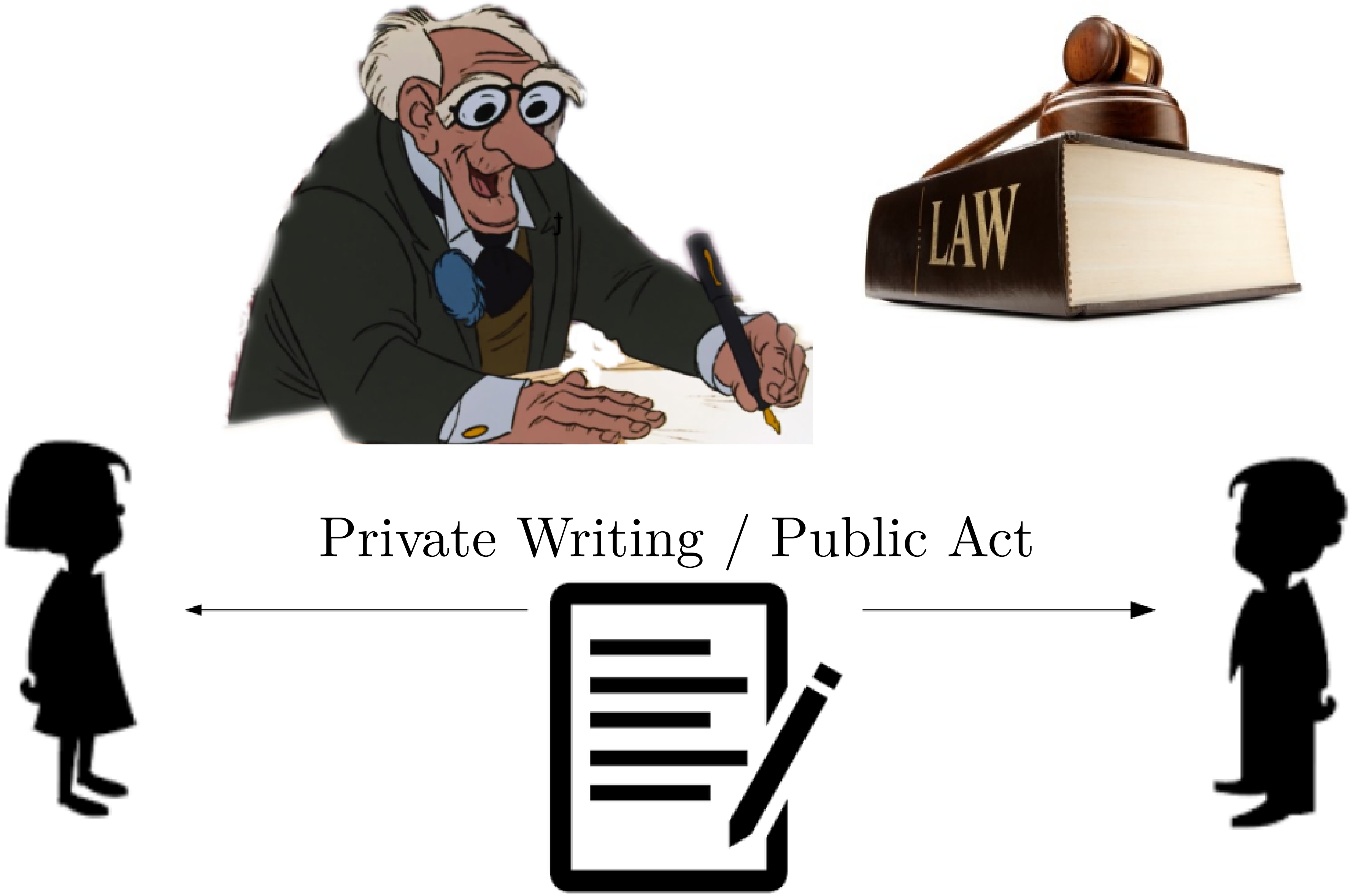
Contracts

Agreement between parties
Object
Cause

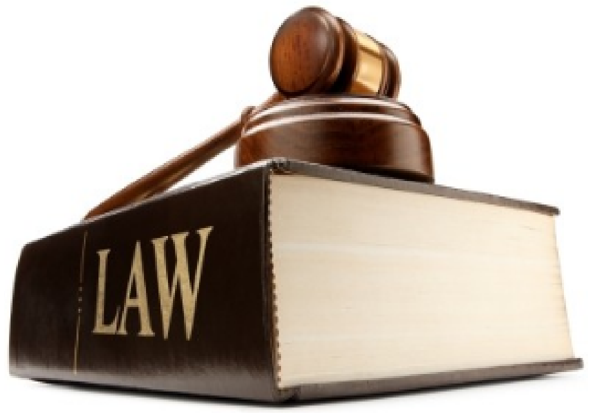


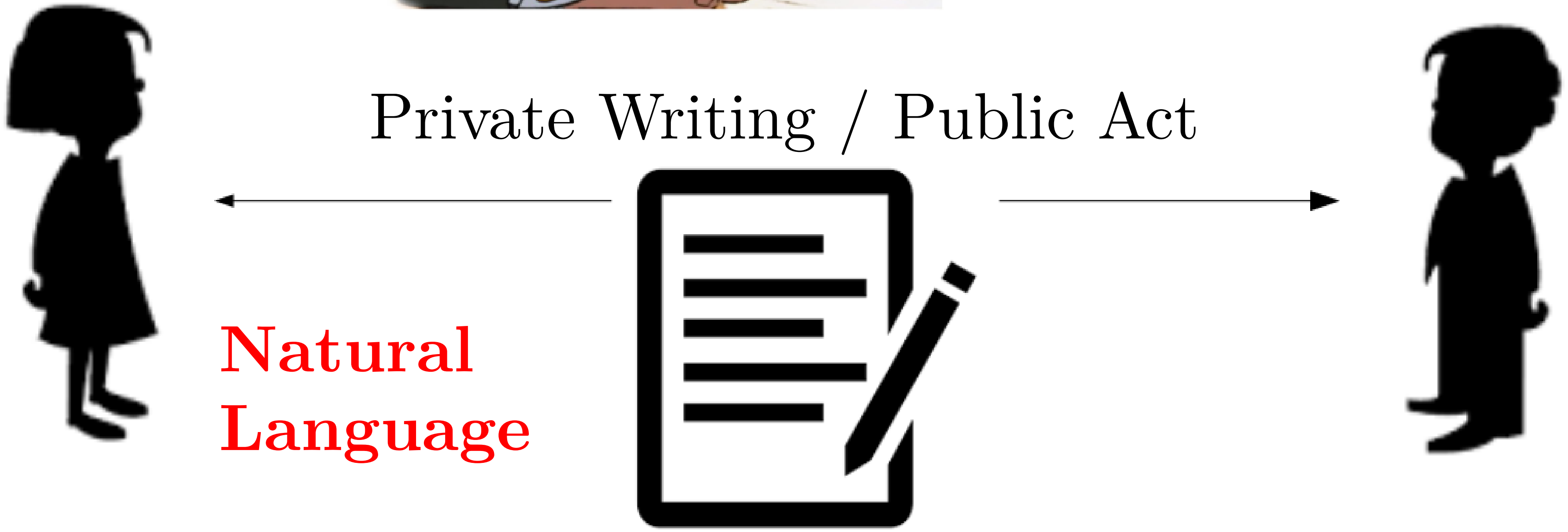
Private Writing

Contracts

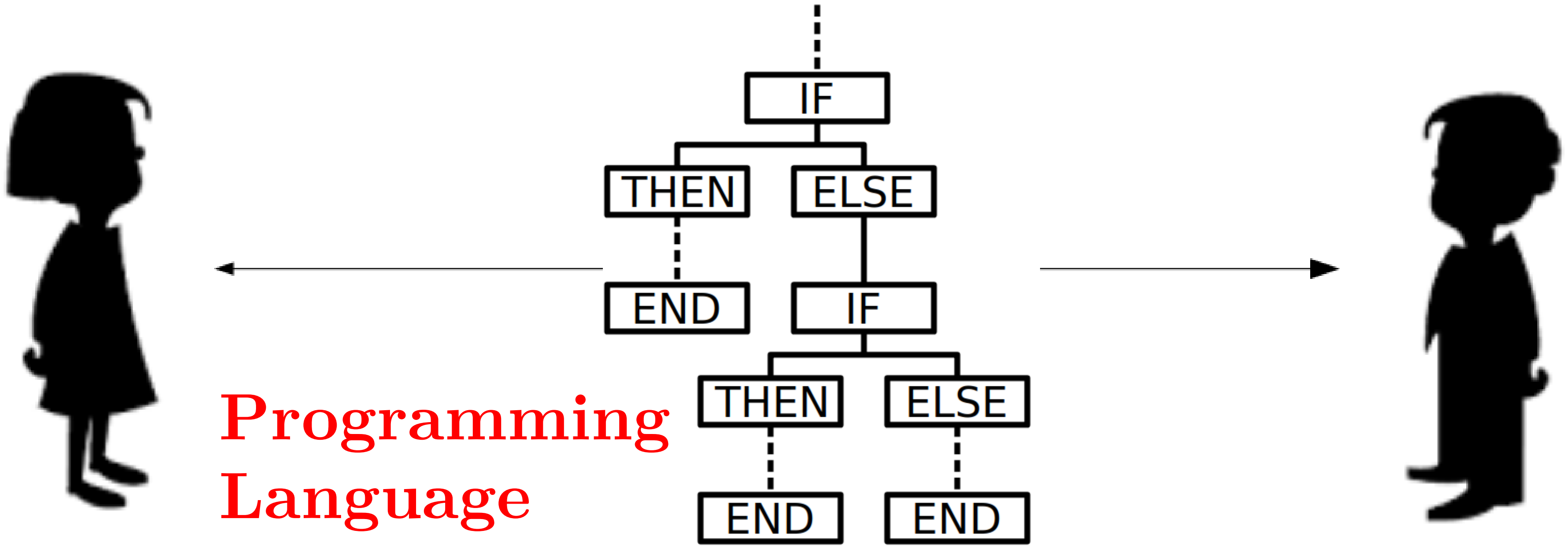


Contracts



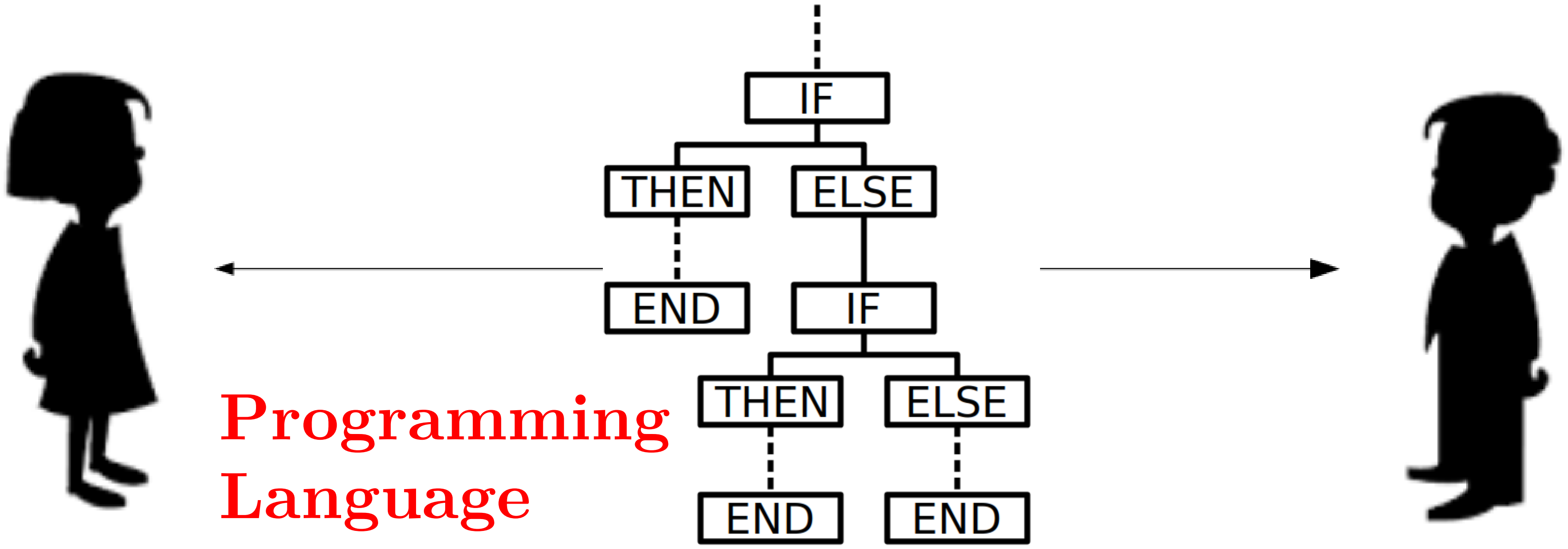


Smart Contracts

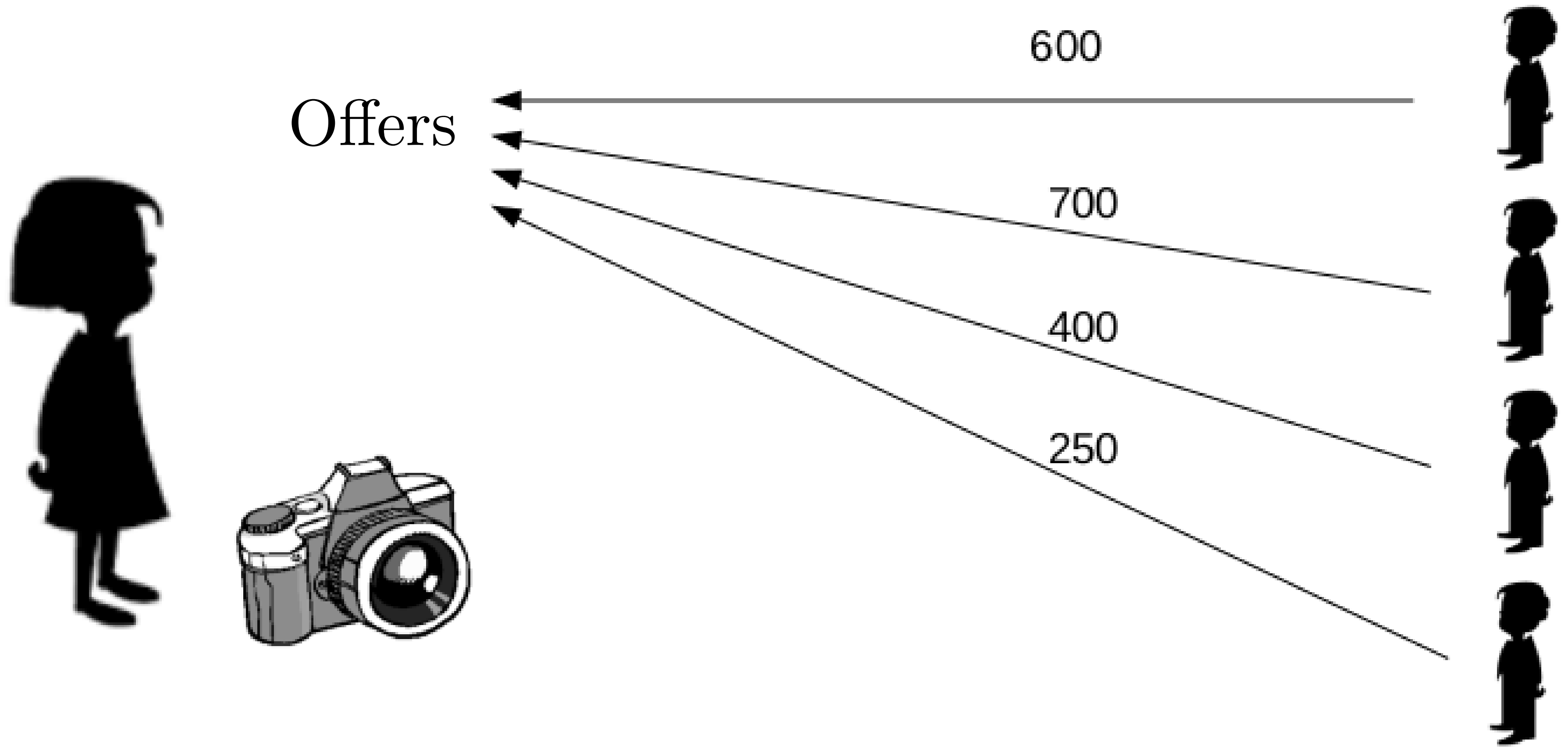


Smart Contracts

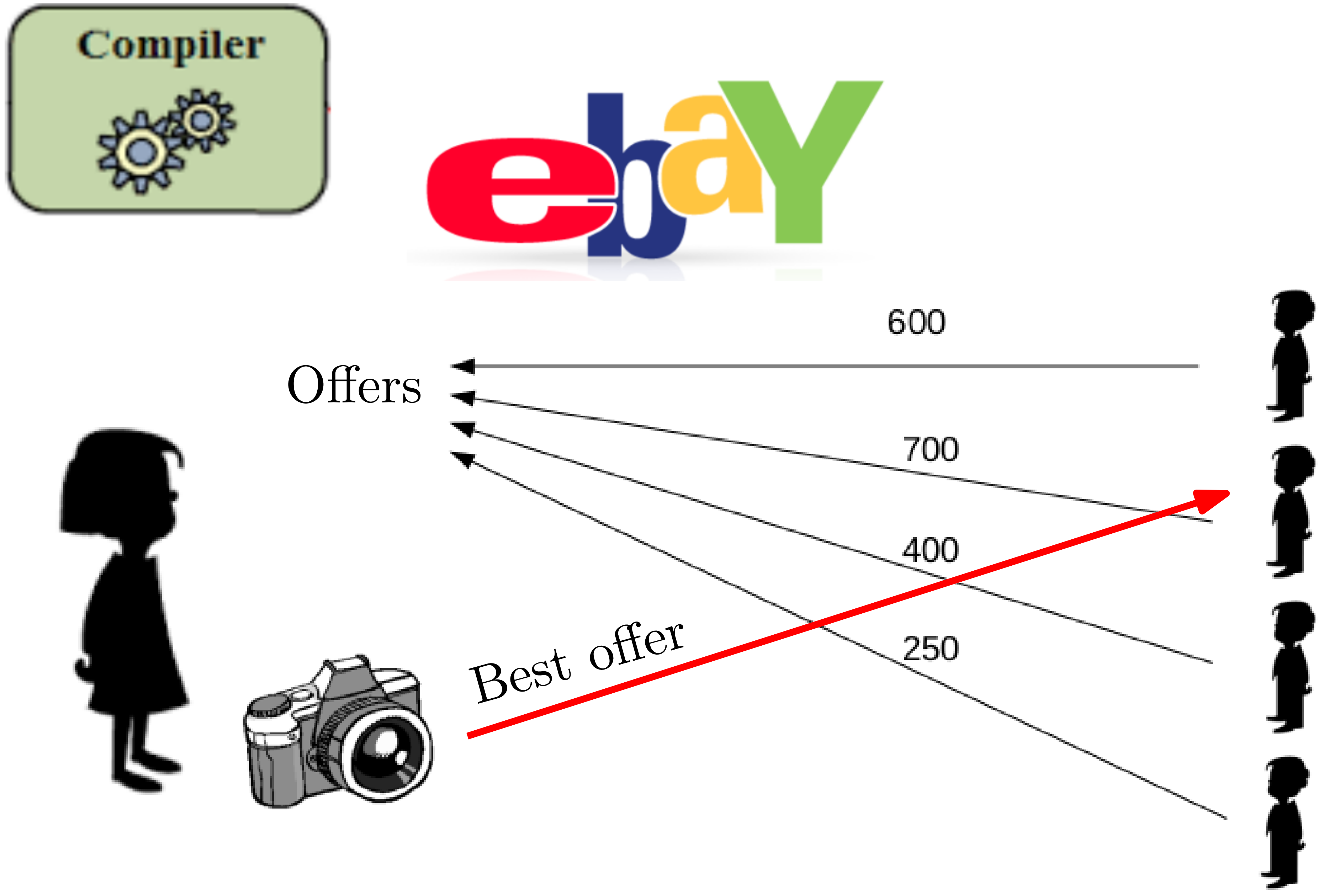
Smart contracs are computer protocols that assist, verify or enforce, the negotiation or the execution of a contract.



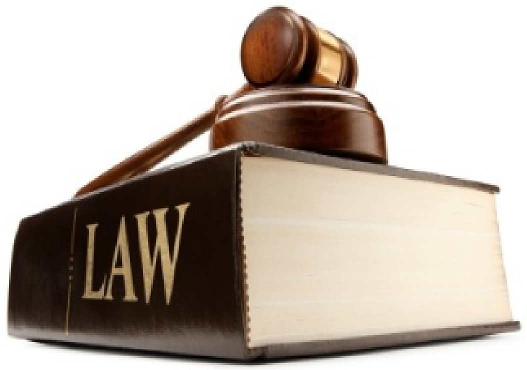
Online Auctions

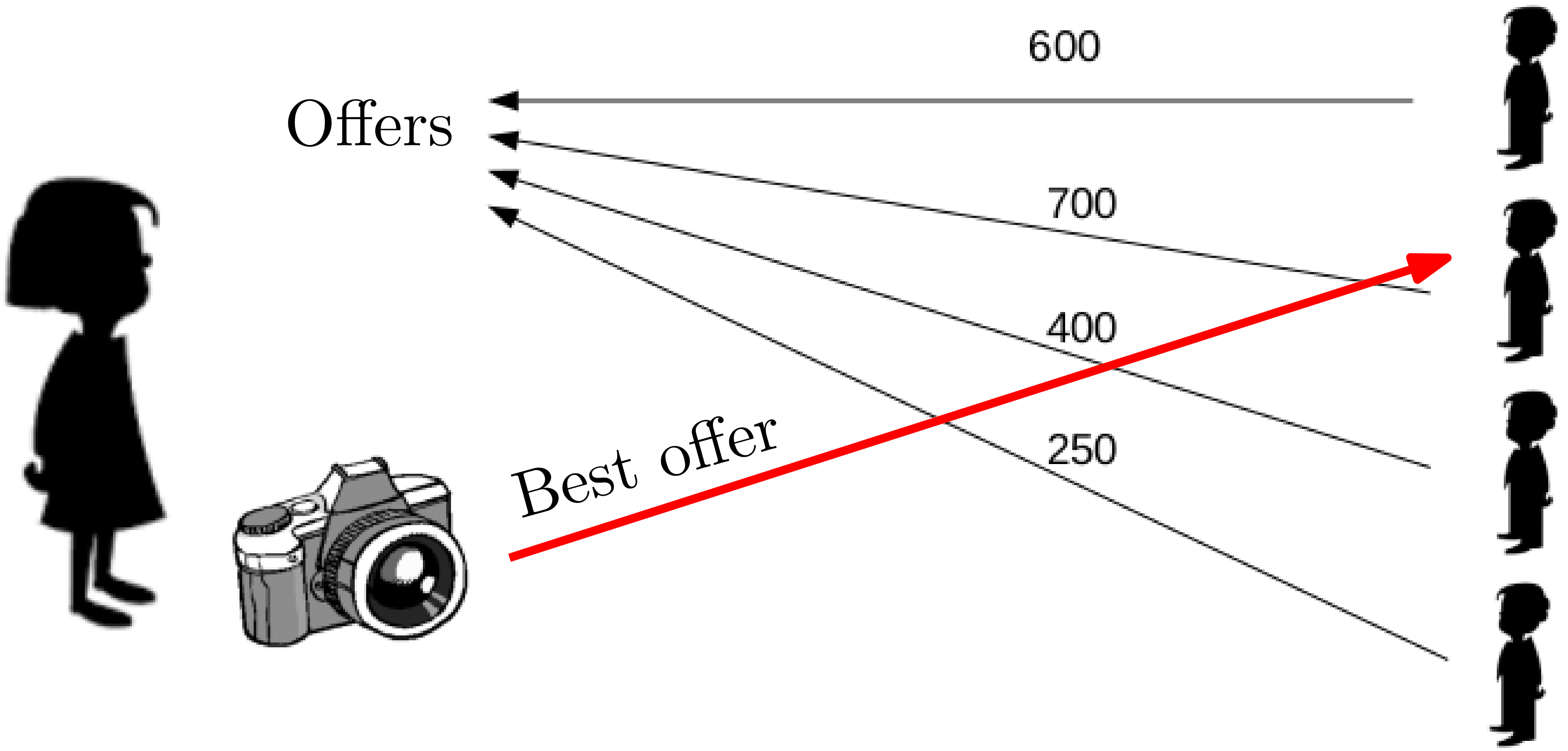


Online Auctions

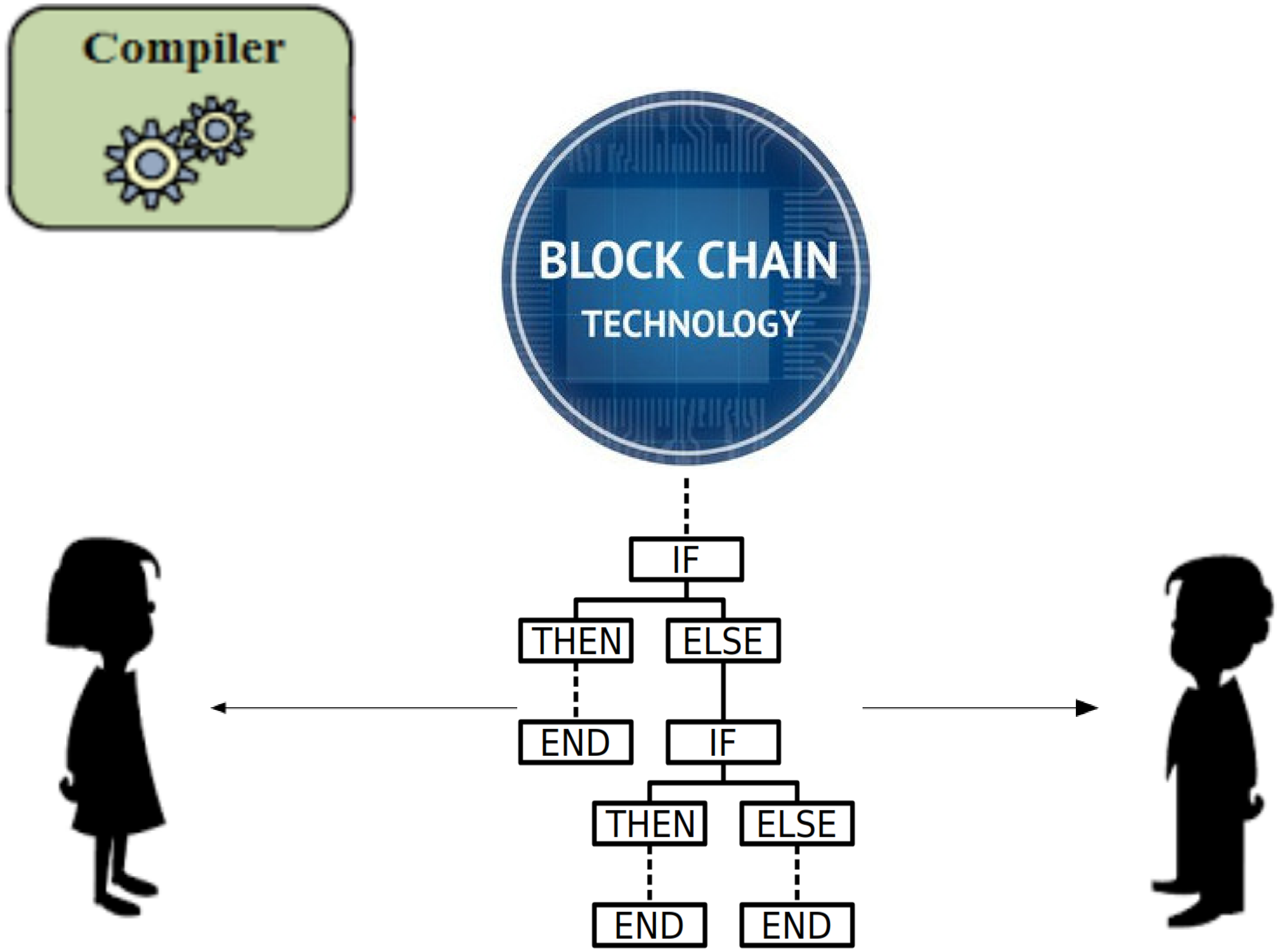


Online Auctions

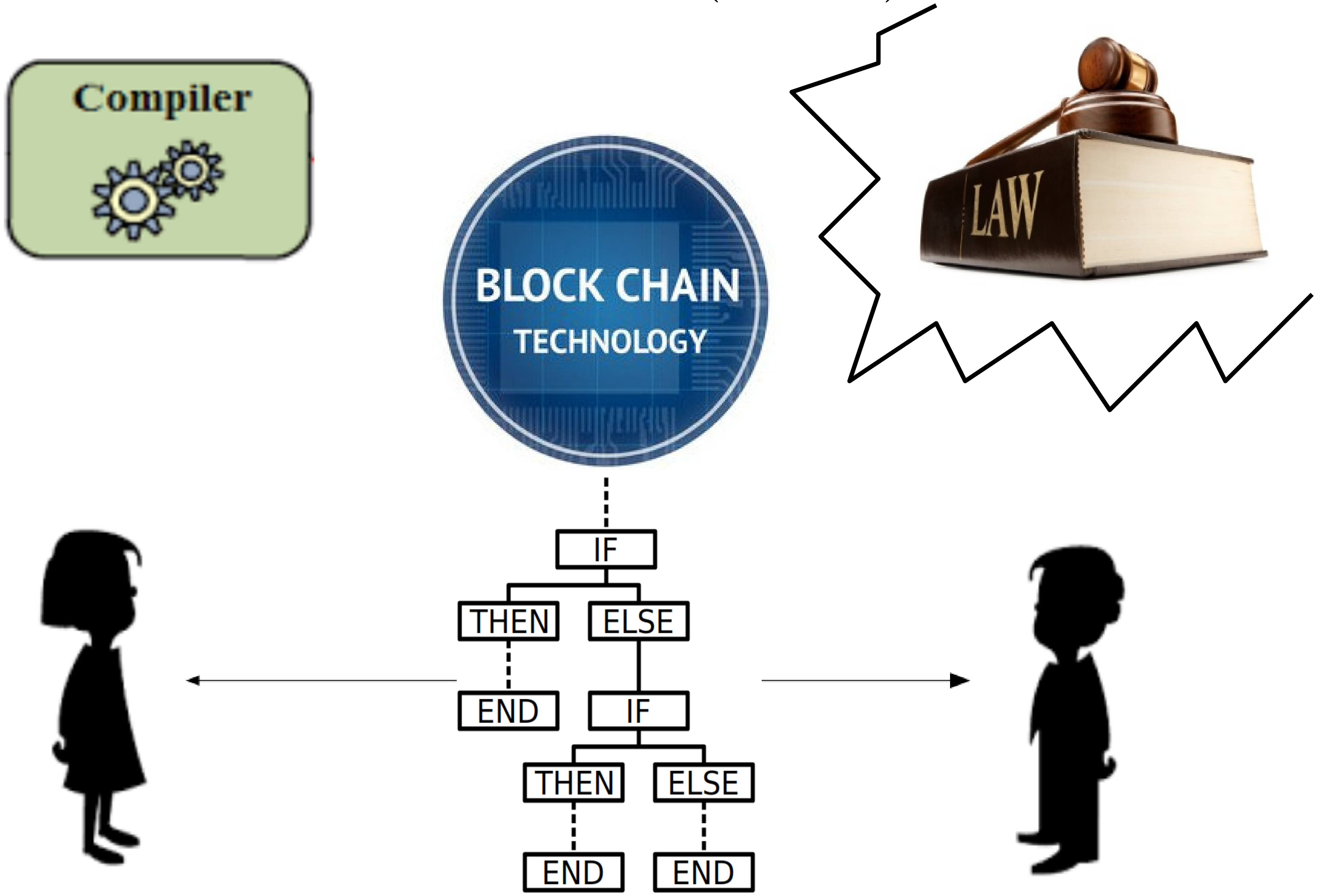




Smart Contract (Today)



Smart Contract (Today)



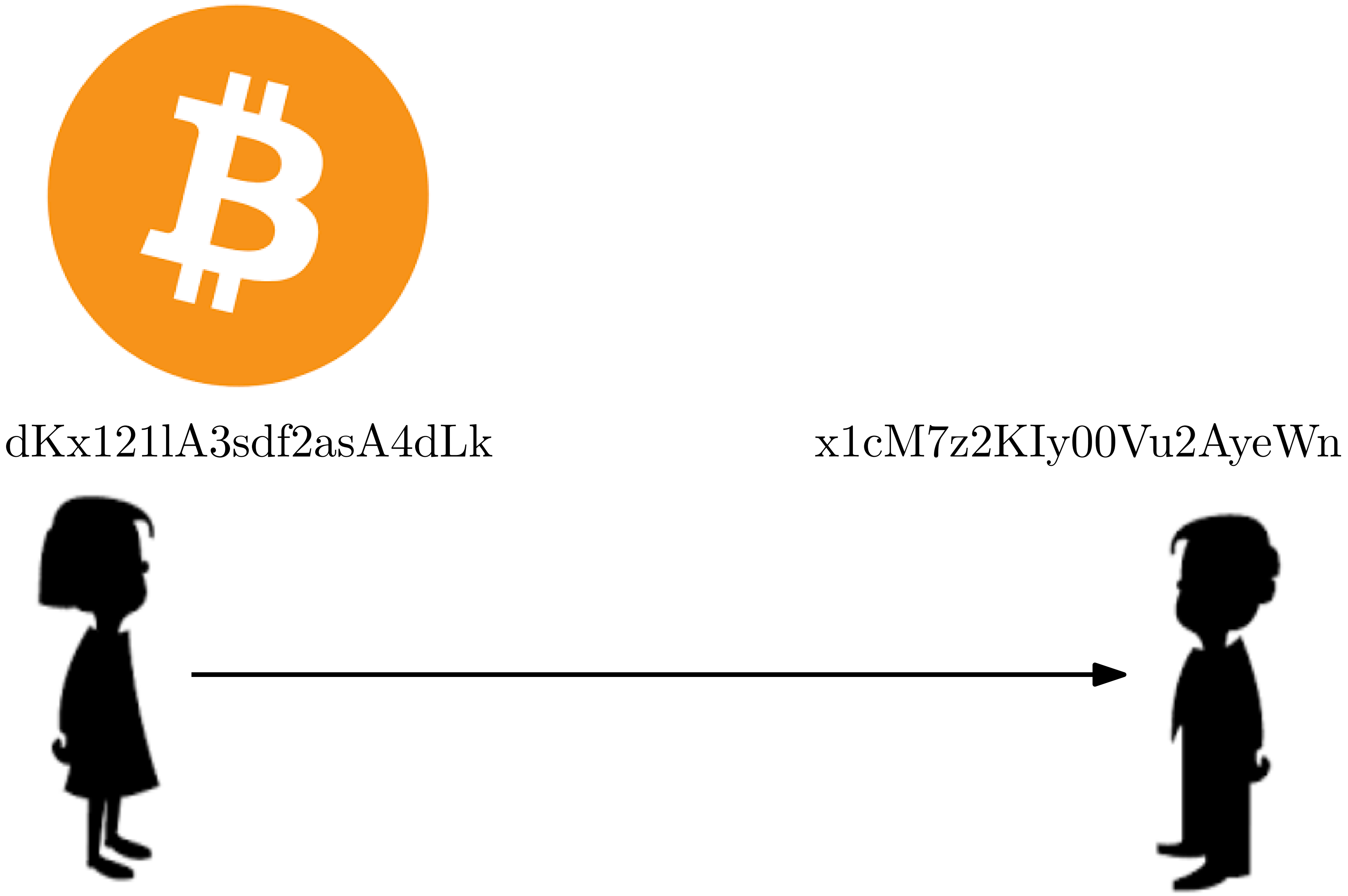
Different Approaches

Scripting
Language

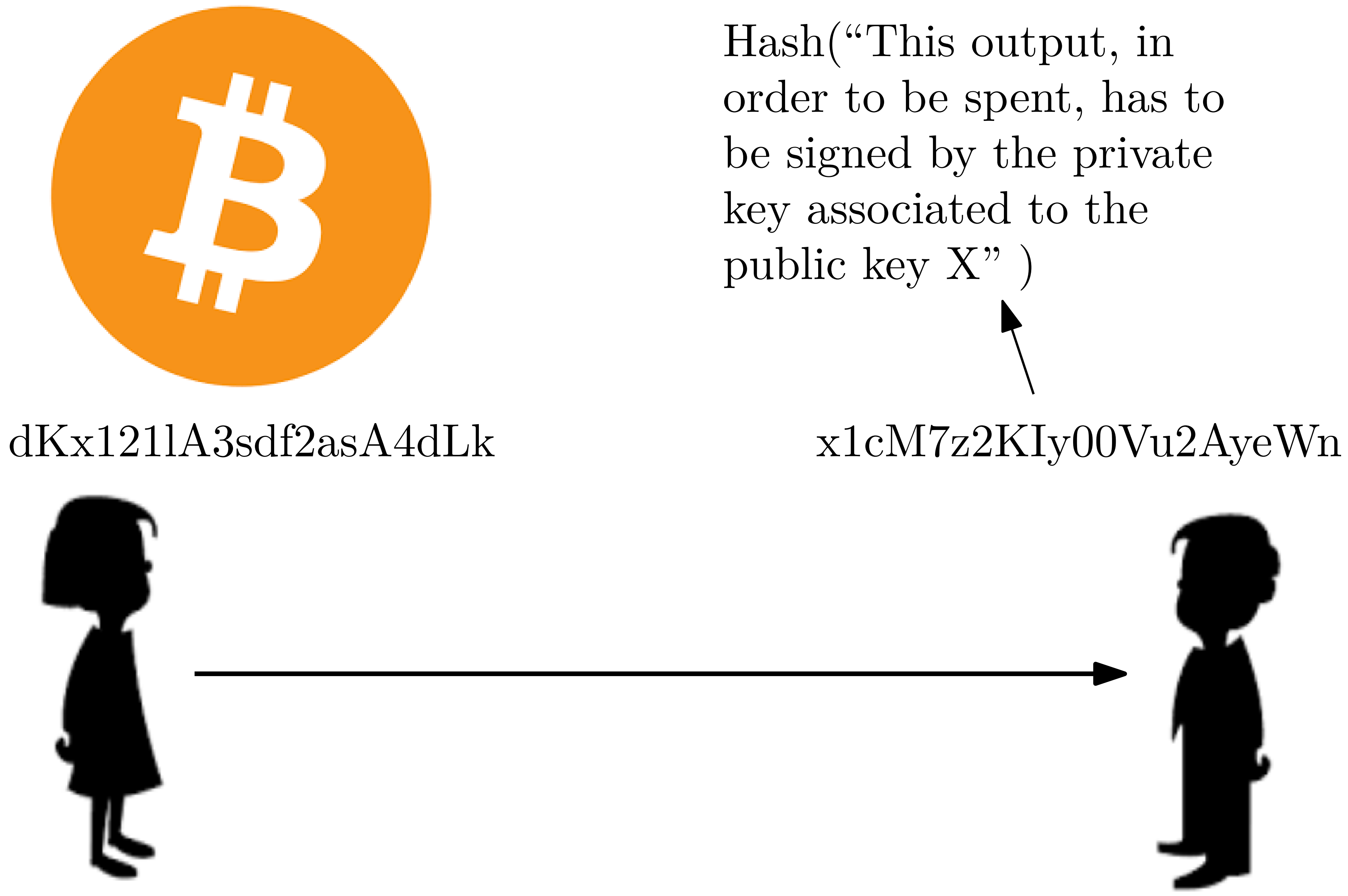
ethereum

Turing-Complete
Language

Normal Transactions



Normal Transactions



Hash("This output, in order to be spent, has to be signed by the private key associated to the public key X")

Normal Transactions

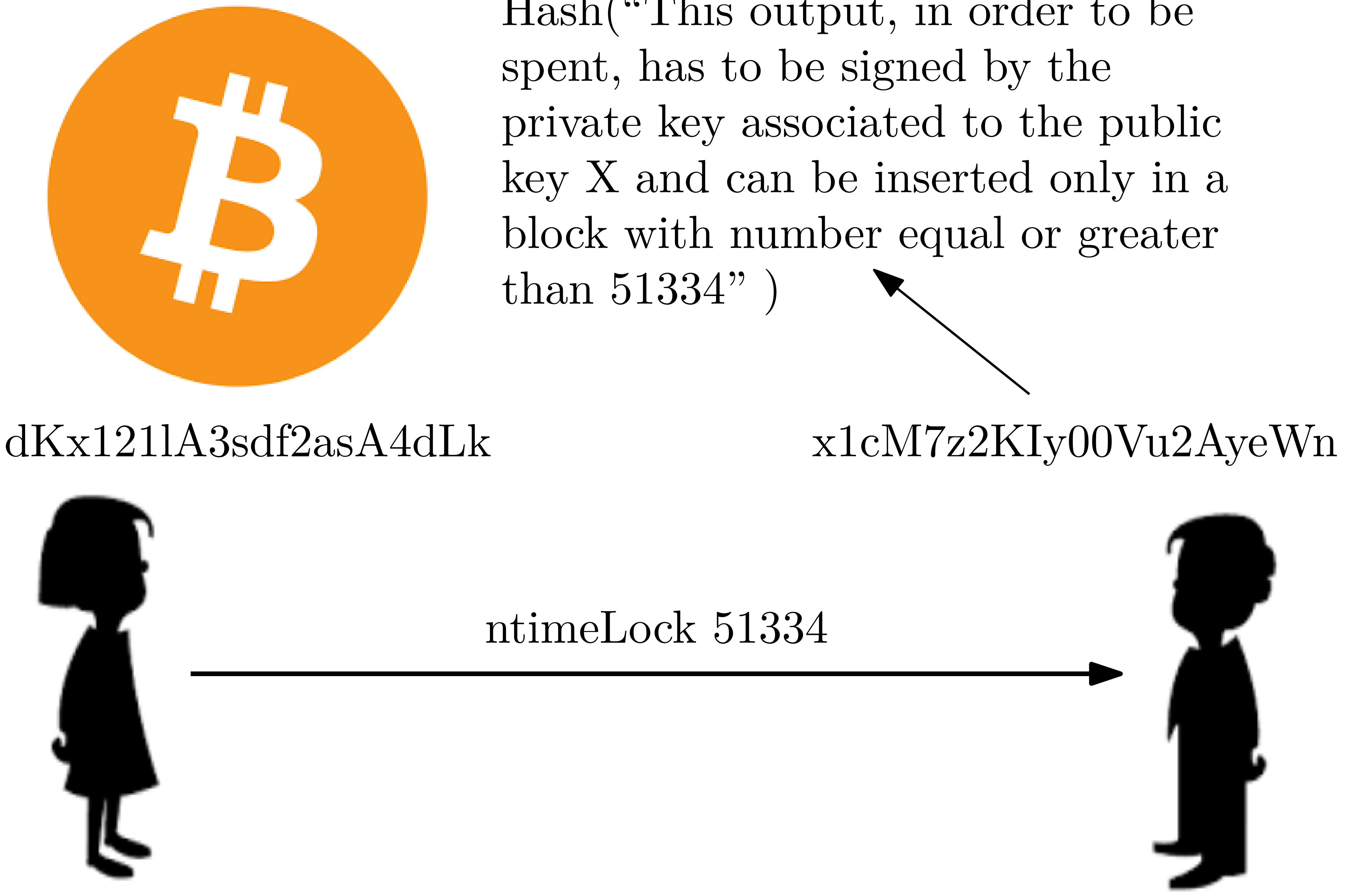
Hash("This output, in order to be spent, has to be signed by the private key associated to the public key X and Y")

dKx121lA3sdf2asA4dLk

x1cM7z2KIy00Vu2AyeWn

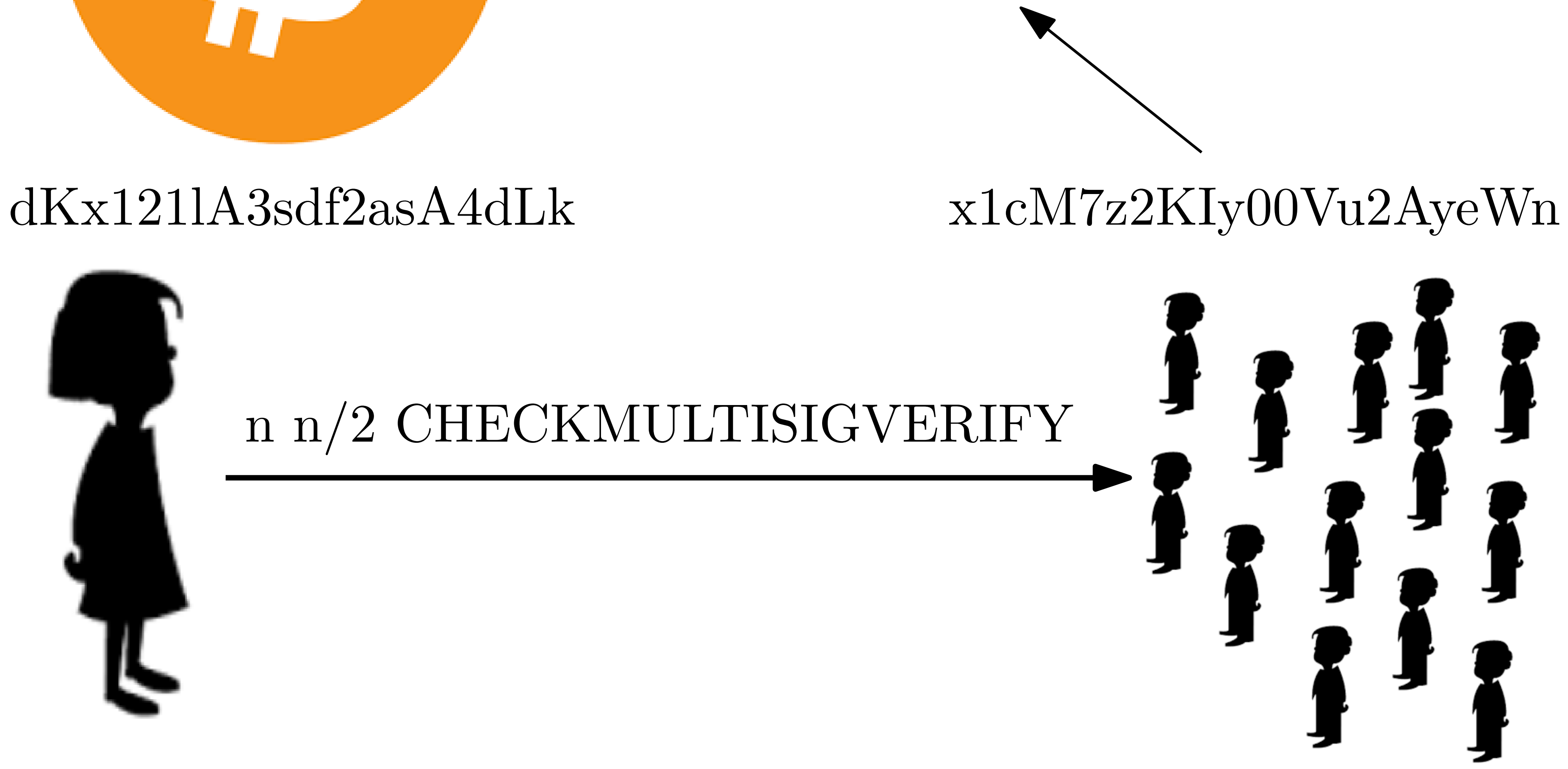
2 2 CHECKMULTISIGVERIFY

“Strange” Transaction I


Hash(“This output, in order to be spent, has to be signed by the private key associated to the public key X and can be inserted only in a block with number equal or greater than 51334”)

“Strange” Transactions

Hash(“This output, in order to be spent, has to be signed by the majority of private keys associated to the following public keys: $\{X, Y, ..., Z\}$”)



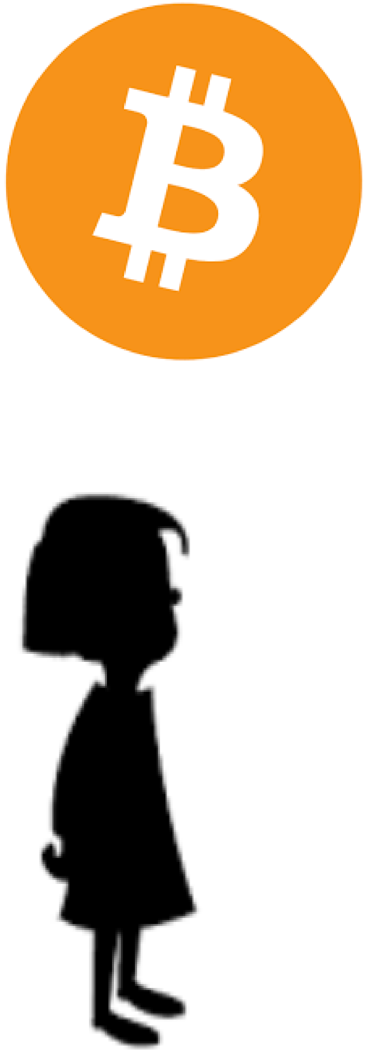
Different approaches

Scripting Language

A restricted programming language without loops is available

Turing-Complete Language

Providing a deposit



Bob is a server which provides a service free of charge

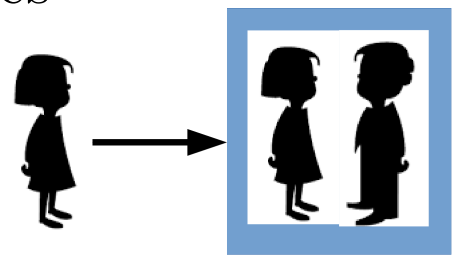
Providing a deposit

Bob is a server which provides a service free of charge

Alice must prove that she is not a spambot

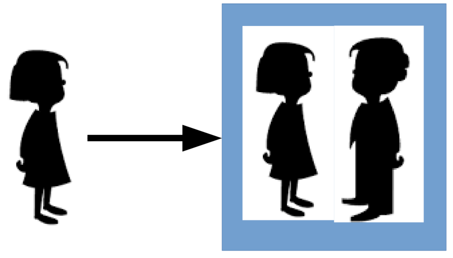
Providing a deposit

1) Alice prepares a transaction (signed) T1:

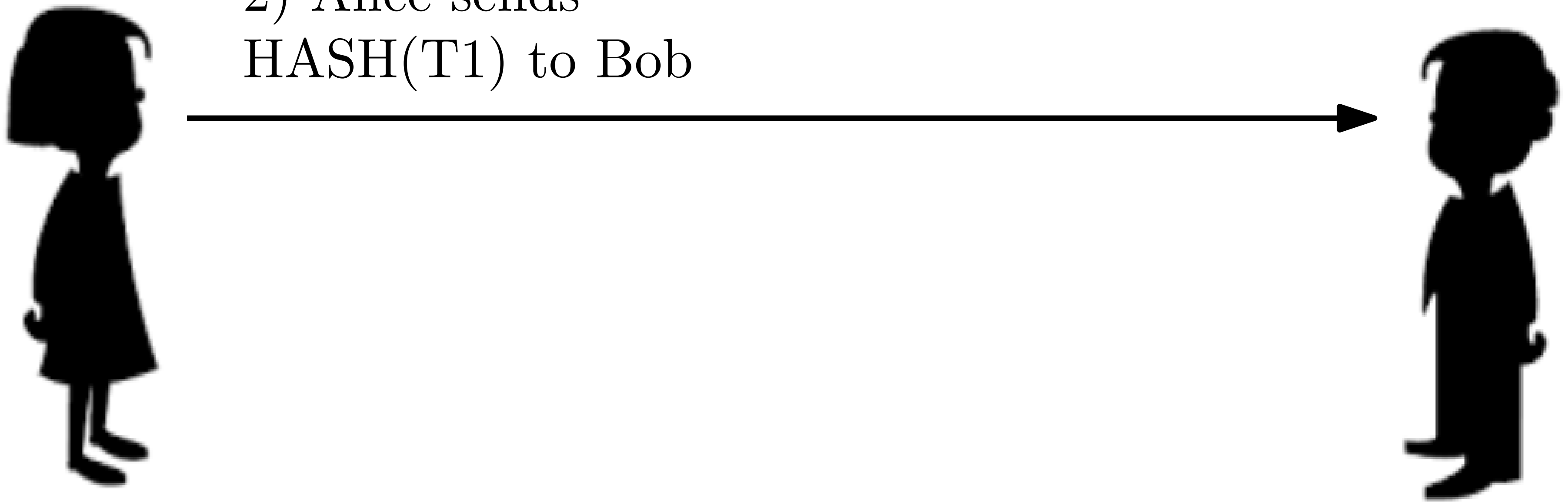


Providing a deposit

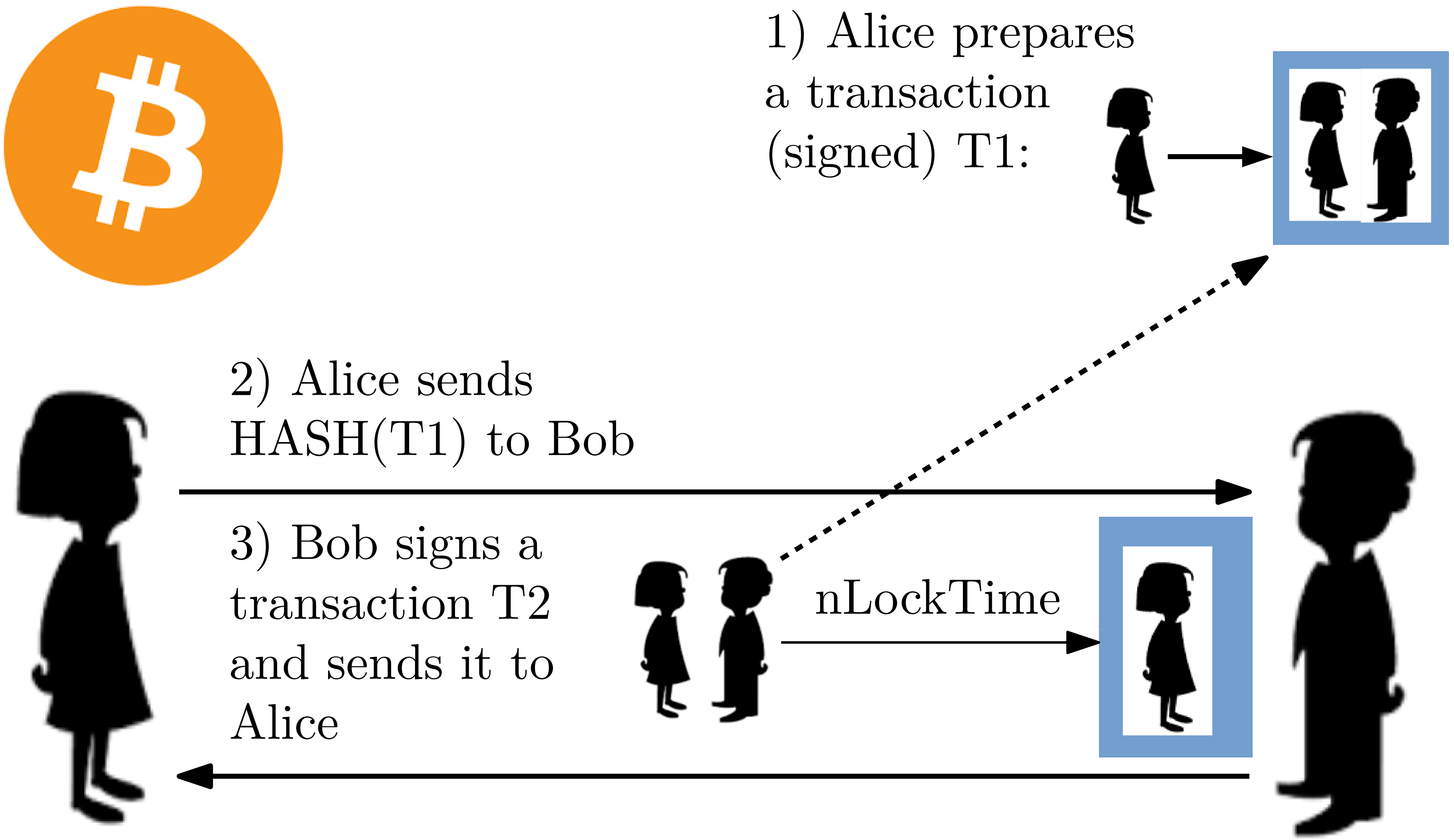
1) Alice prepares a transaction (signed) T1:



2) Alice sends HASH(T1) to Bob



Providing a deposit

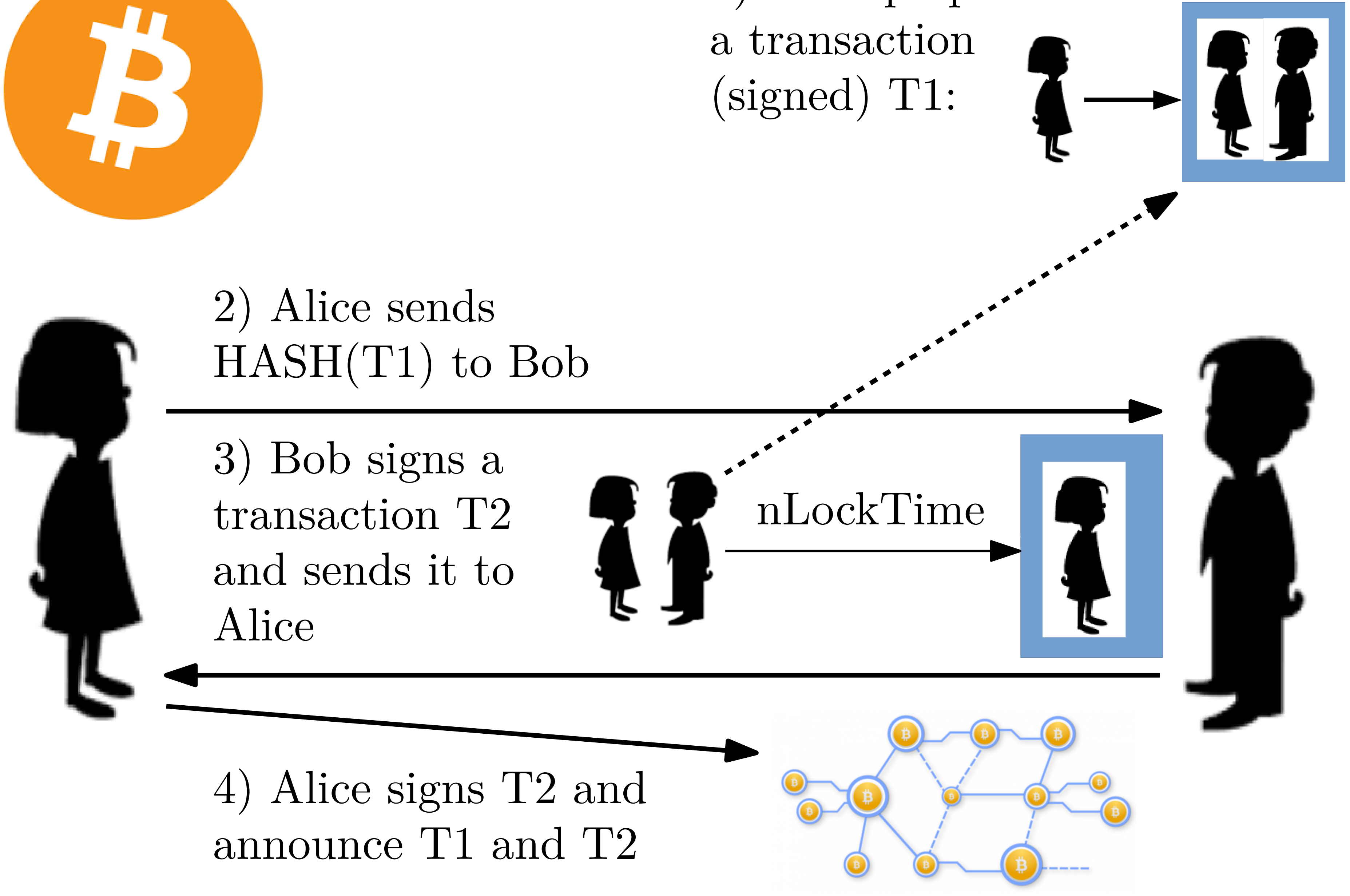


1) Alice prepares a transaction (signed) T1:

2) Alice sends HASH(T1) to Bob

3) Bob signs a transaction T2 and sends it to Alice

Providing a deposit



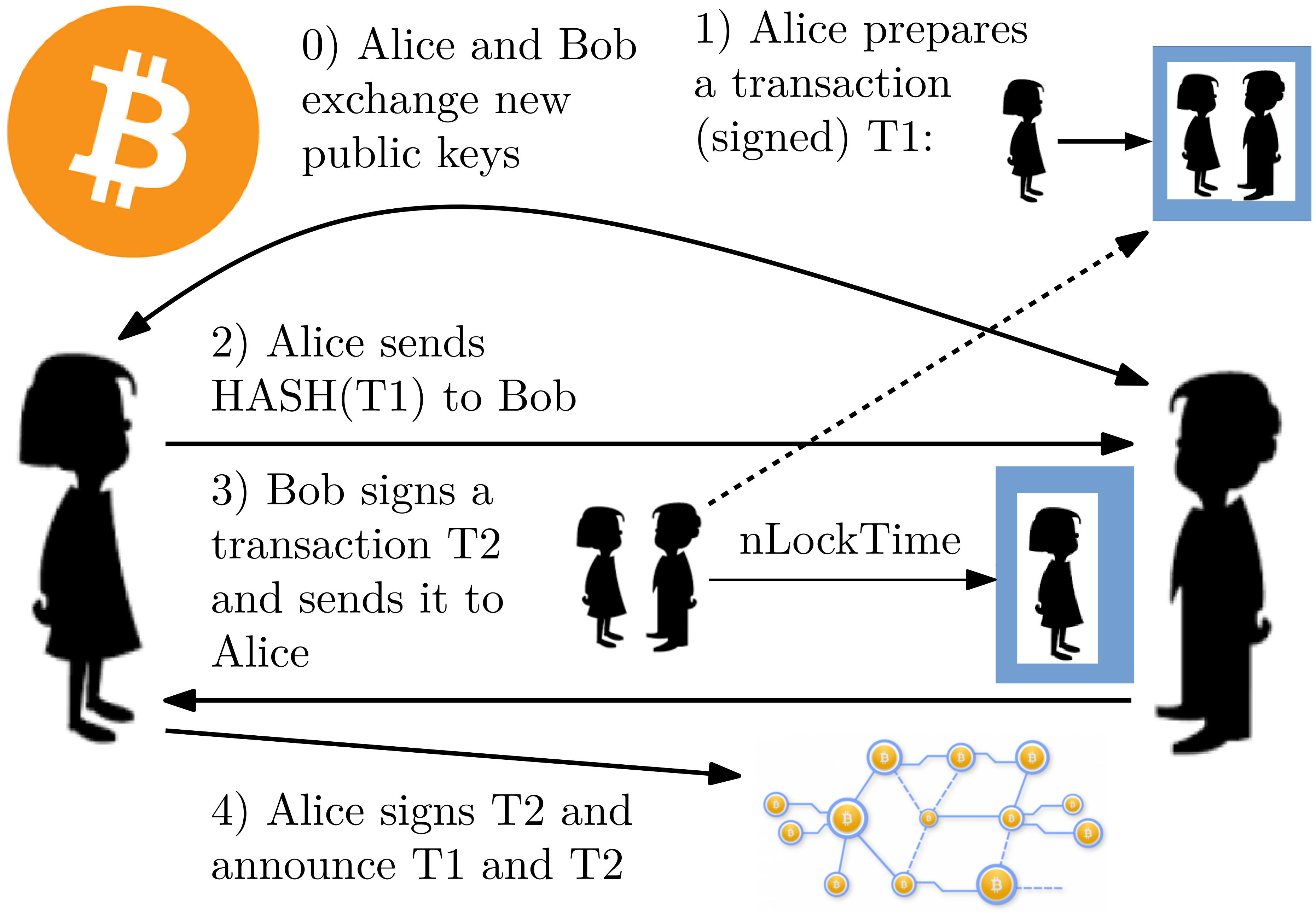
1) Alice prepares a transaction (signed) T1:

2) Alice sends HASH(T1) to Bob

3) Bob signs a transaction T2 and sends it to Alice

4) Alice signs T2 and announce T1 and T2

Providing a deposit



0) Alice and Bob exchange new public keys

1) Alice prepares a transaction (signed) T1:

2) Alice sends HASH(T1) to Bob

3) Bob signs a transaction T2 and sends it to Alice

4) Alice signs T2 and announce T1 and T2

Lottery I

Lottery I

Alice chooses a random number x

Bob chooses a random number y

Lottery I

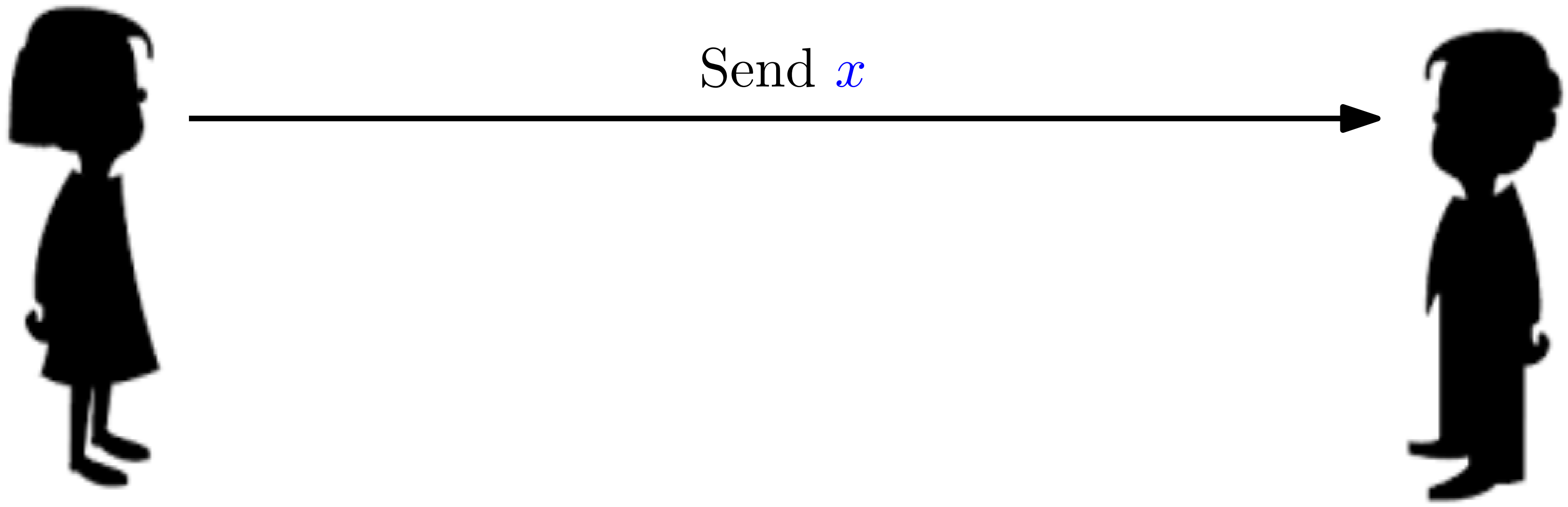
If $x + y$ is even Alice wins, otherwise Bob wins.

Alice chooses a random number x

Bob chooses a random number y

Lottery I

If $x + y$ is even Alice wins, otherwise Bob wins.

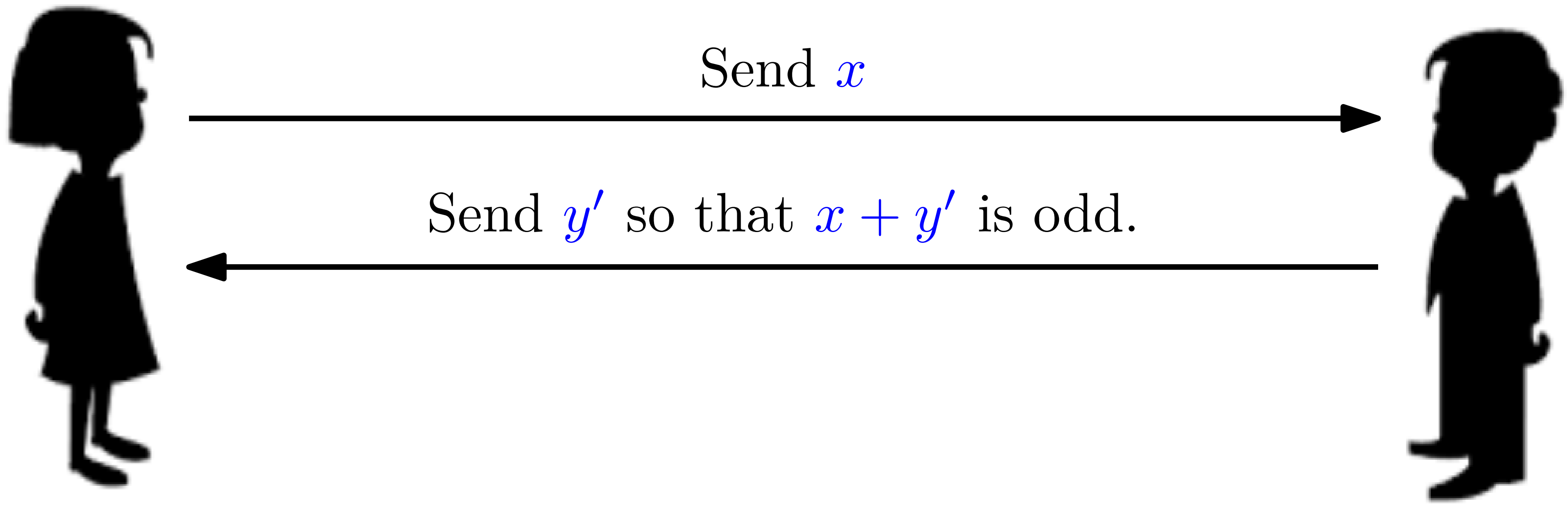


Alice chooses a random number x

Bob chooses a random number y

Lottery I

If $x + y$ is even Alice wins, otherwise Bob wins.

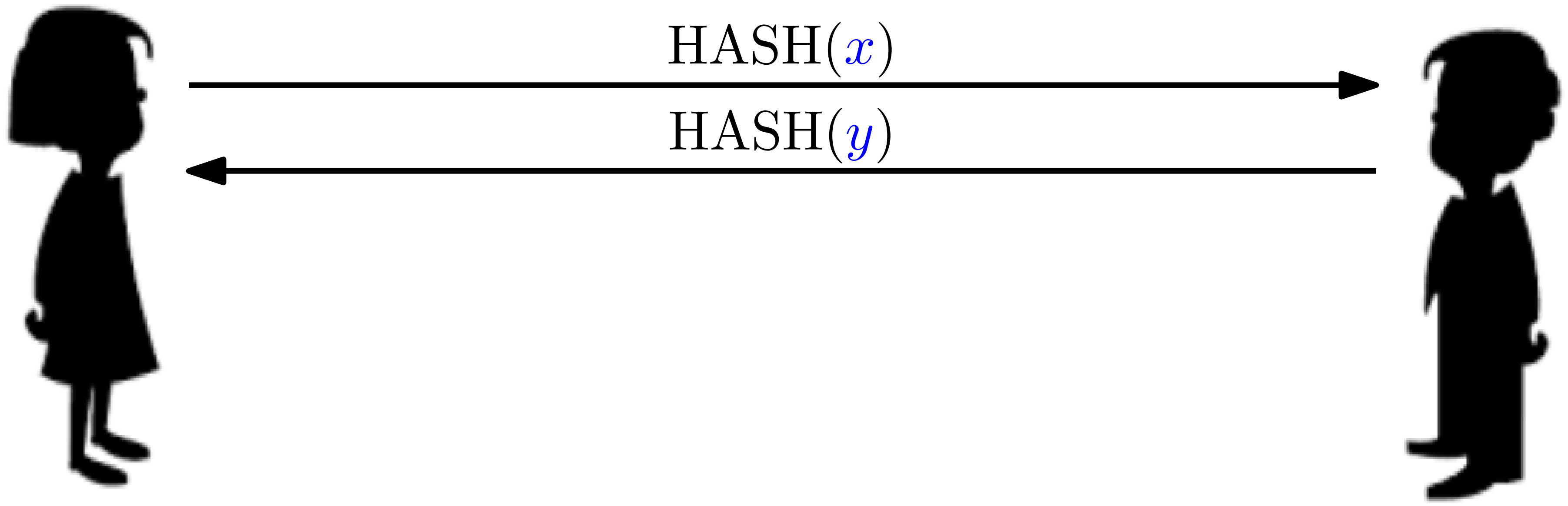


Alice chooses a random number x

Bob chooses a random number y

Lottery I

If $x + y$ is even Alice wins, otherwise Bob wins.

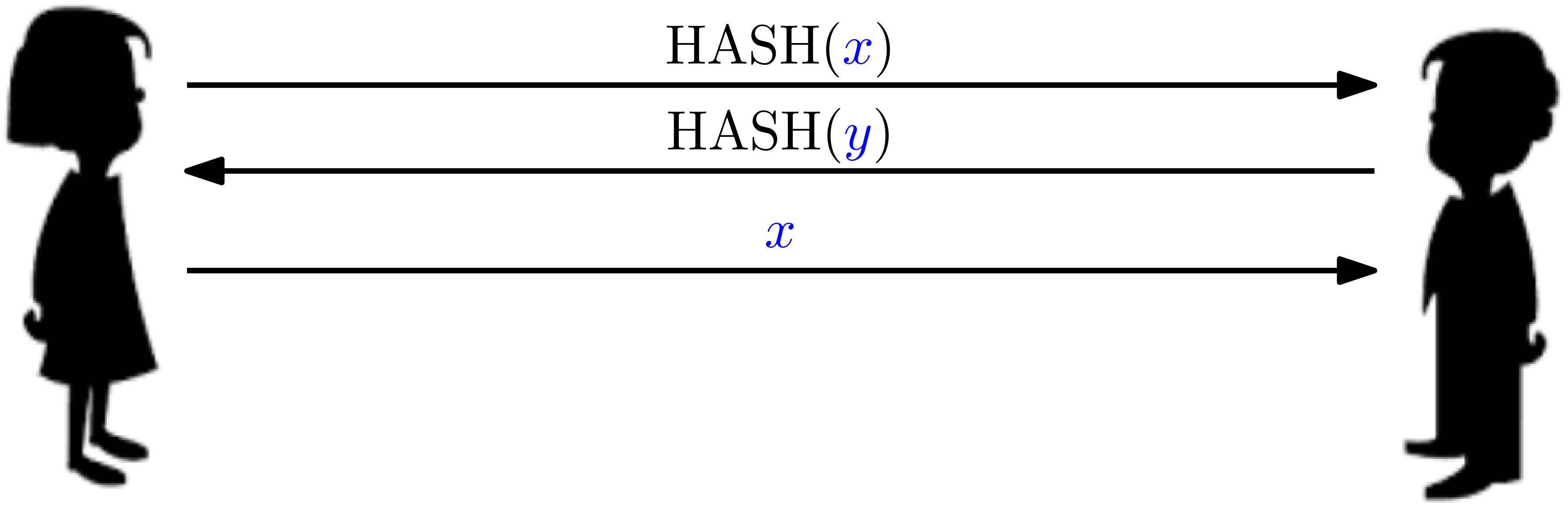


Alice chooses a random number x

Bob chooses a random number y

Lottery I

If $x + y$ is even Alice wins, otherwise Bob wins.

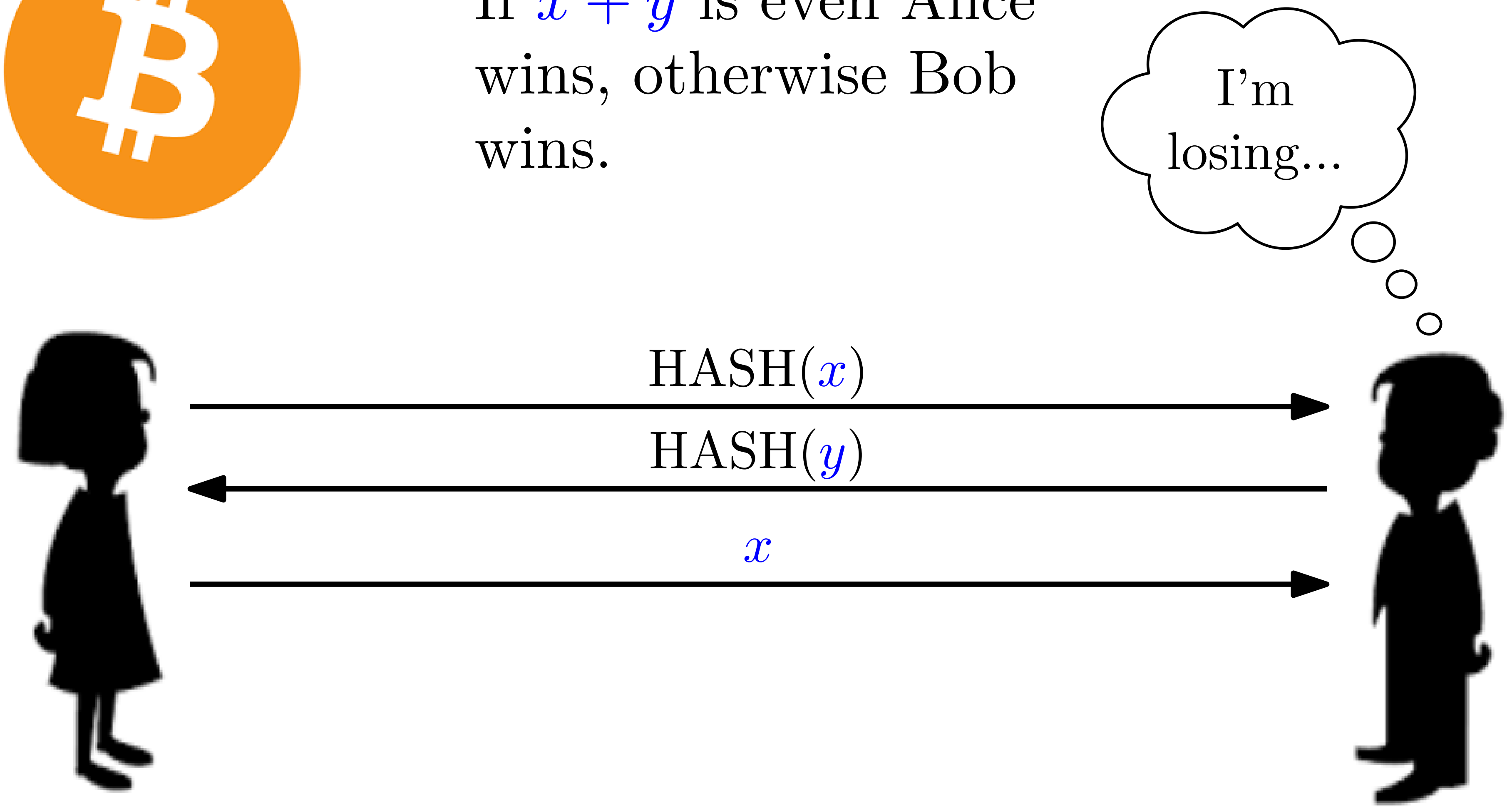


Alice chooses a random number x

Bob chooses a random number y

Lottery I

If $x + y$ is even Alice wins, otherwise Bob wins.

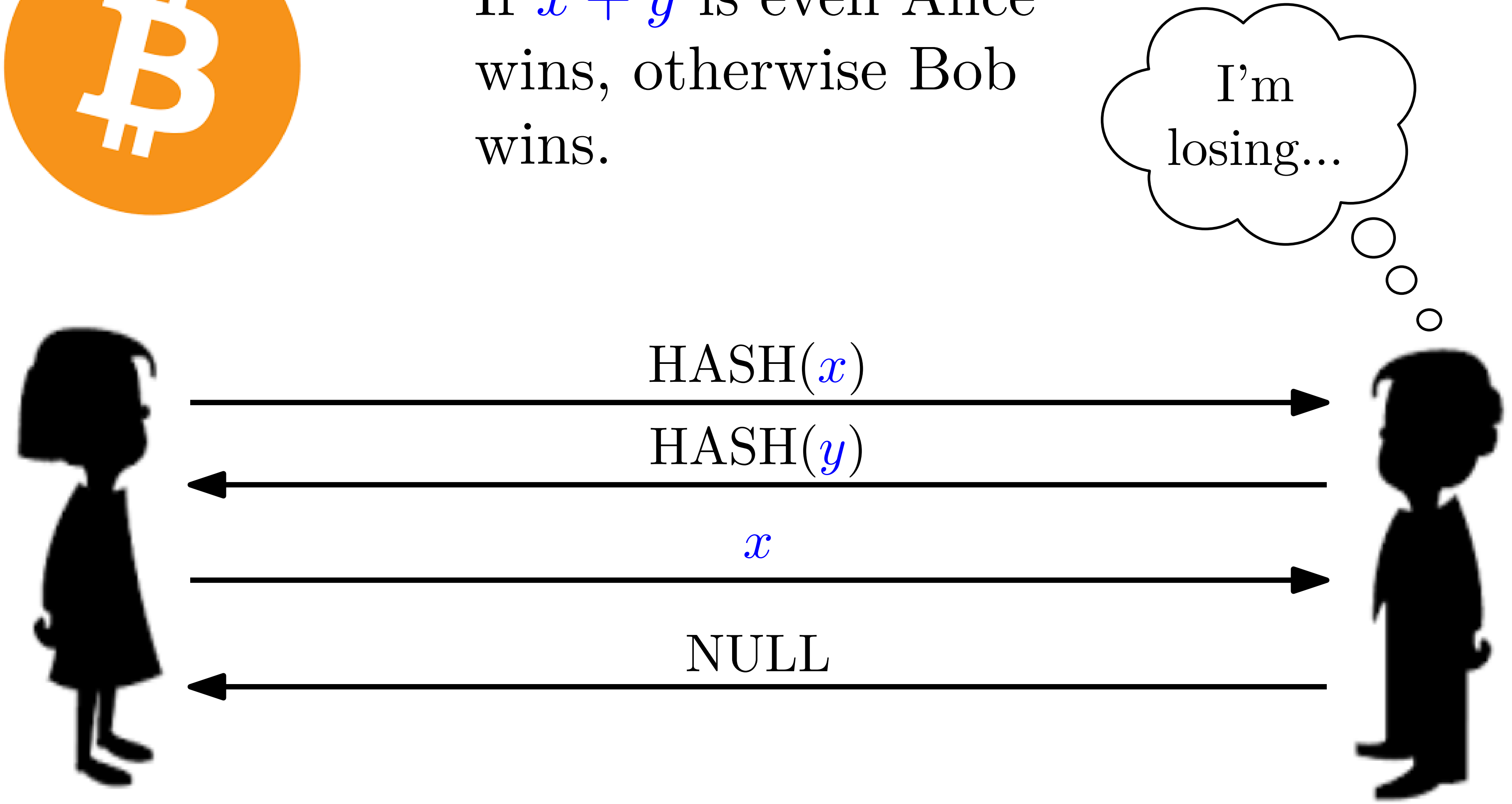


Alice chooses a random number x

Bob chooses a random number y

Lottery I

If $x + y$ is even Alice wins, otherwise Bob wins.

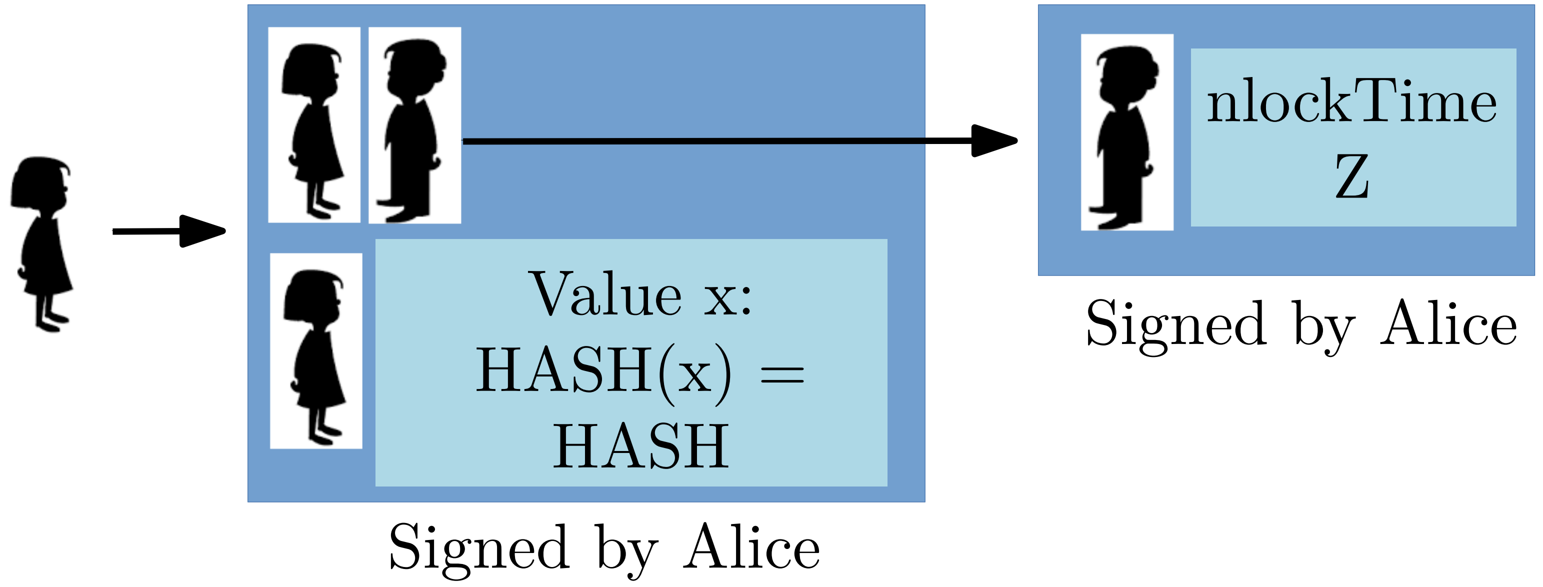


Alice chooses a random number x

Bob chooses a random number y

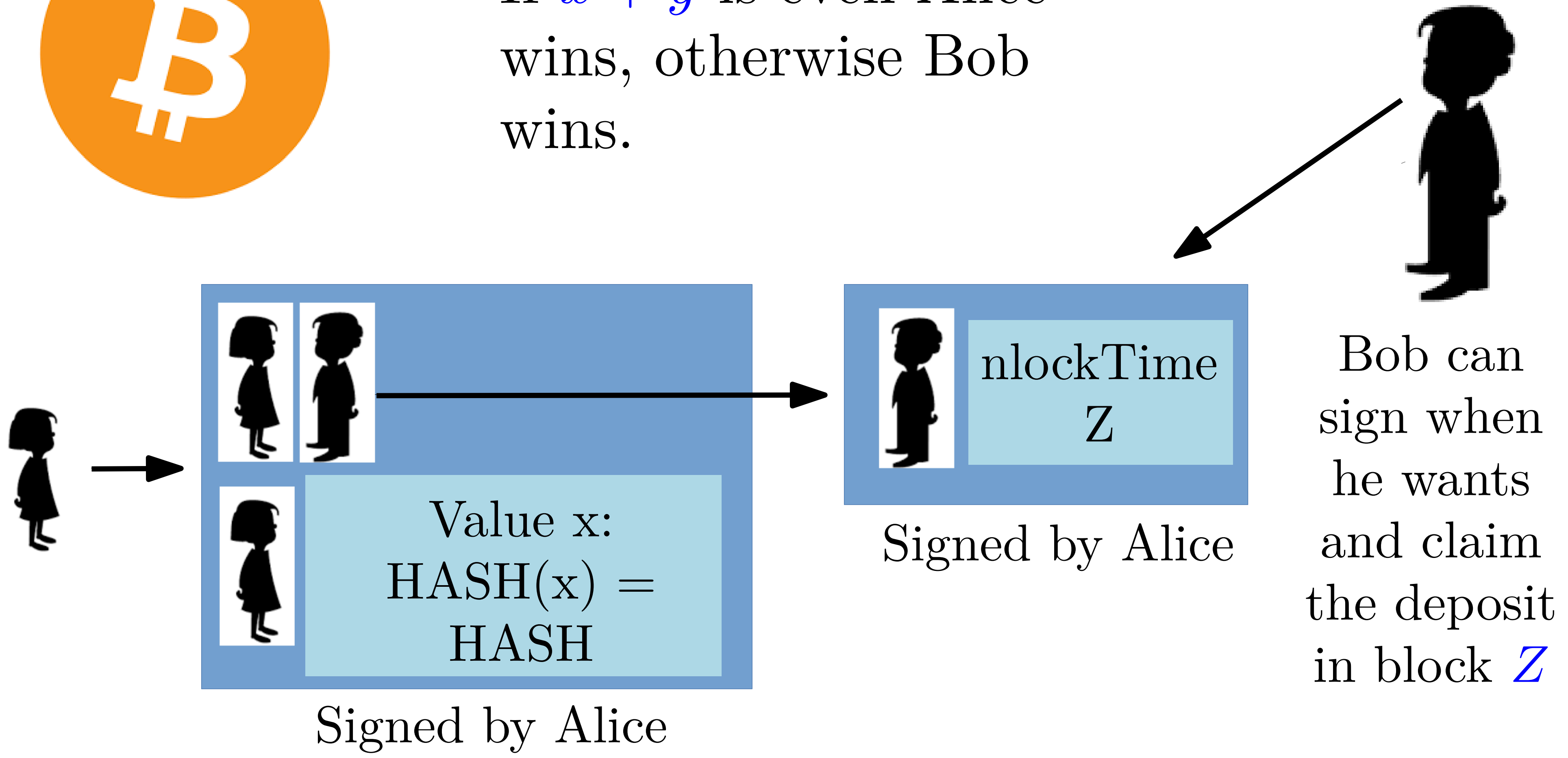
Lottery II

If $x + y$ is even Alice wins, otherwise Bob wins.



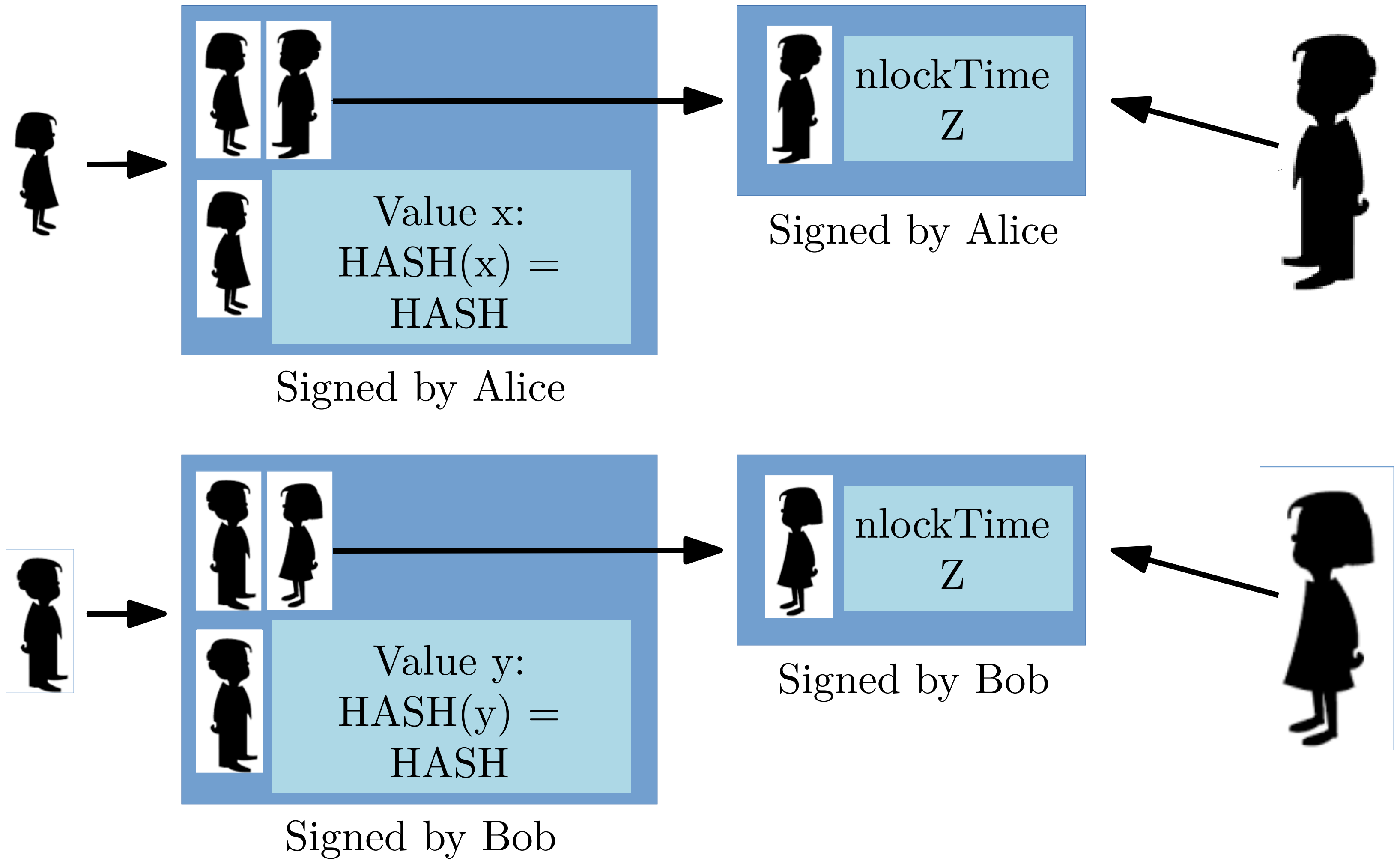
Lottery II

If $x + y$ is even Alice wins, otherwise Bob wins.

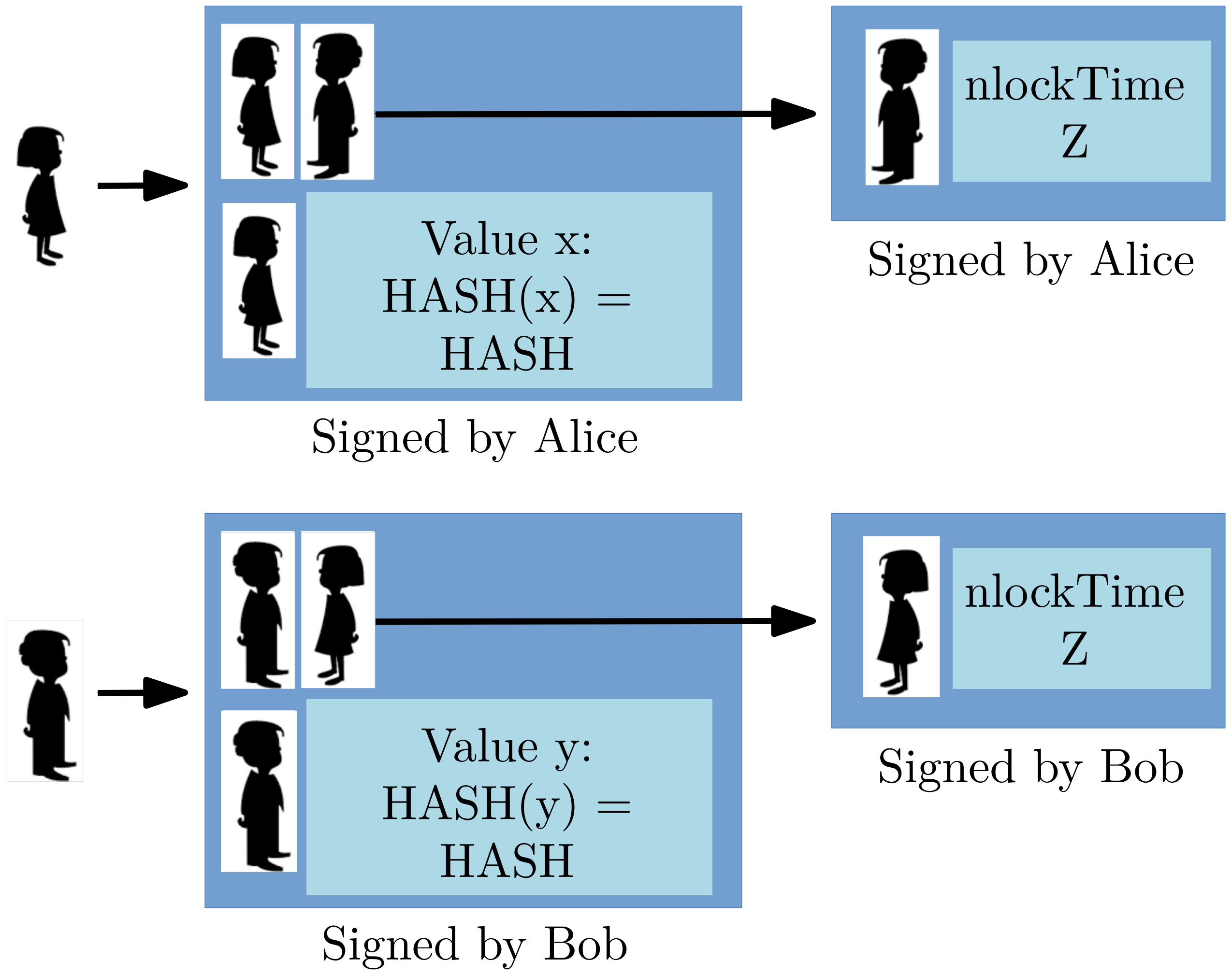


Bob can sign when he wants and claim the deposit in block Z

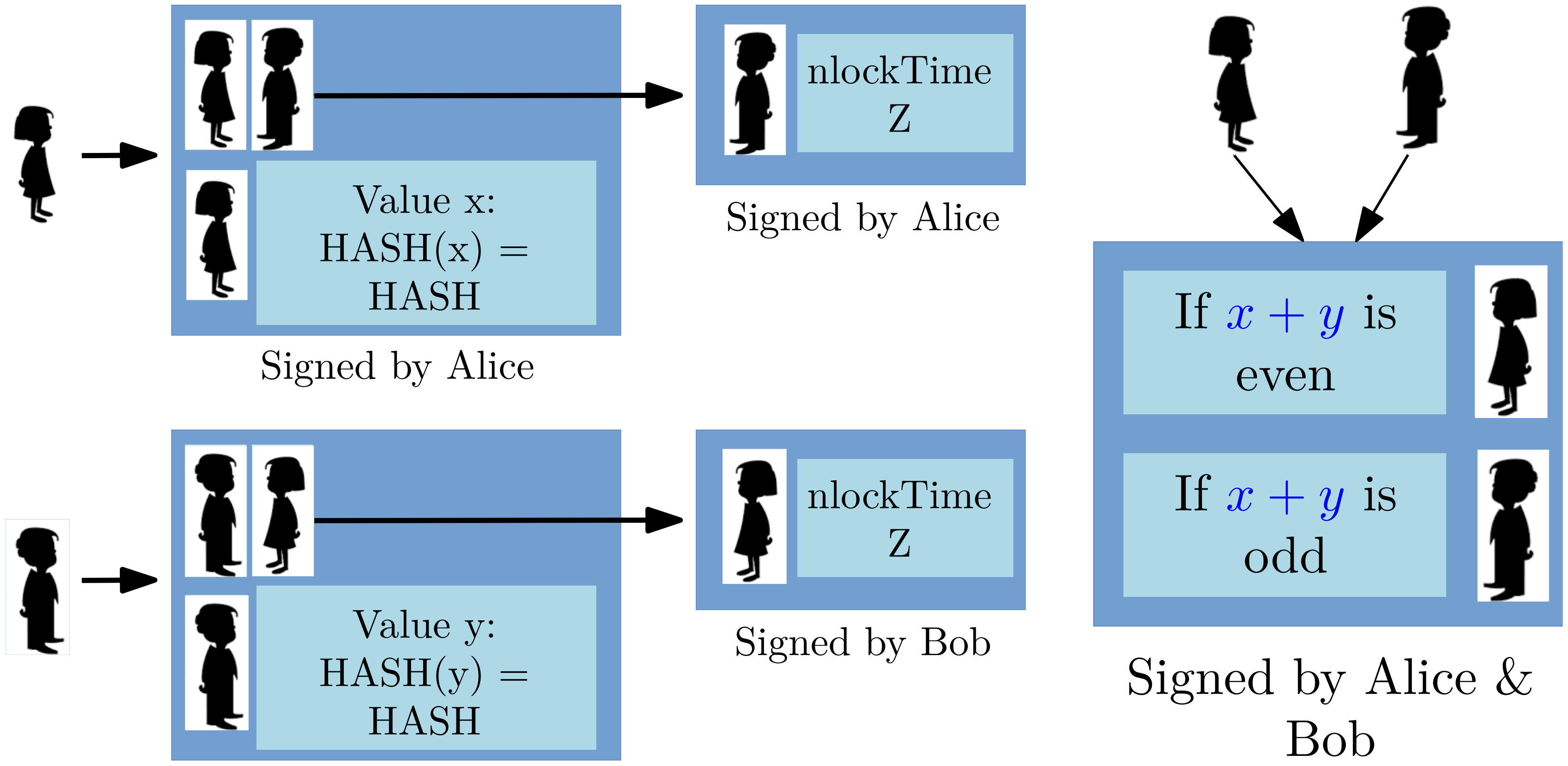
Lottery II


Value x:
HASH(x) =
HASH
Signed by Alice
nlockTime
Z
Signed by Alice
Value y:
HASH(y) =
HASH
Signed by Bob
nlockTime
Z
Signed by Bob

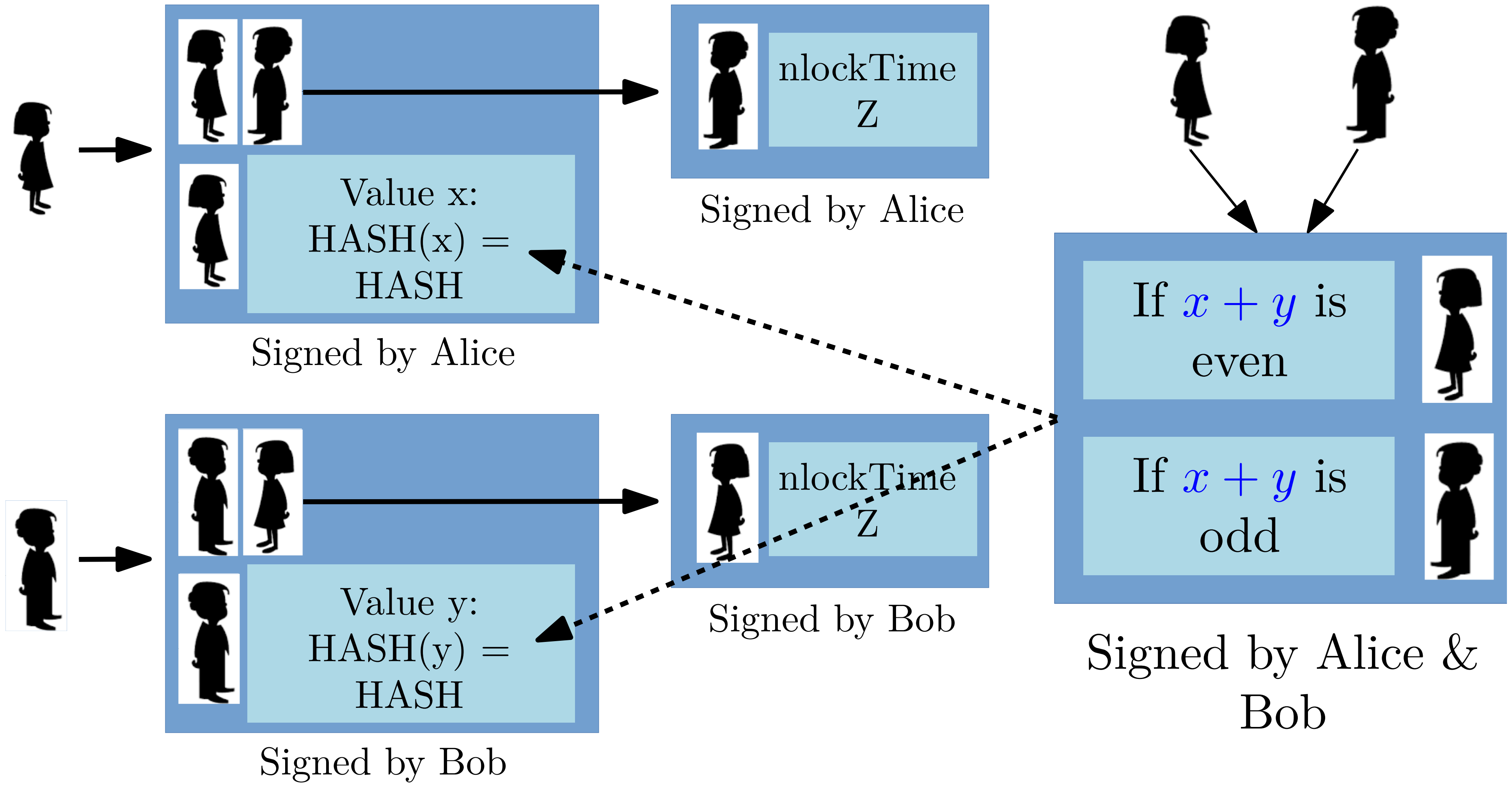
Lottery II


nlockTime
Z
Signed by Alice
Value x:
HASH(x) =
HASH
Signed by Alice
nlockTime
Z
Signed by Bob
Value y:
HASH(y) =
HASH
Signed by Bob

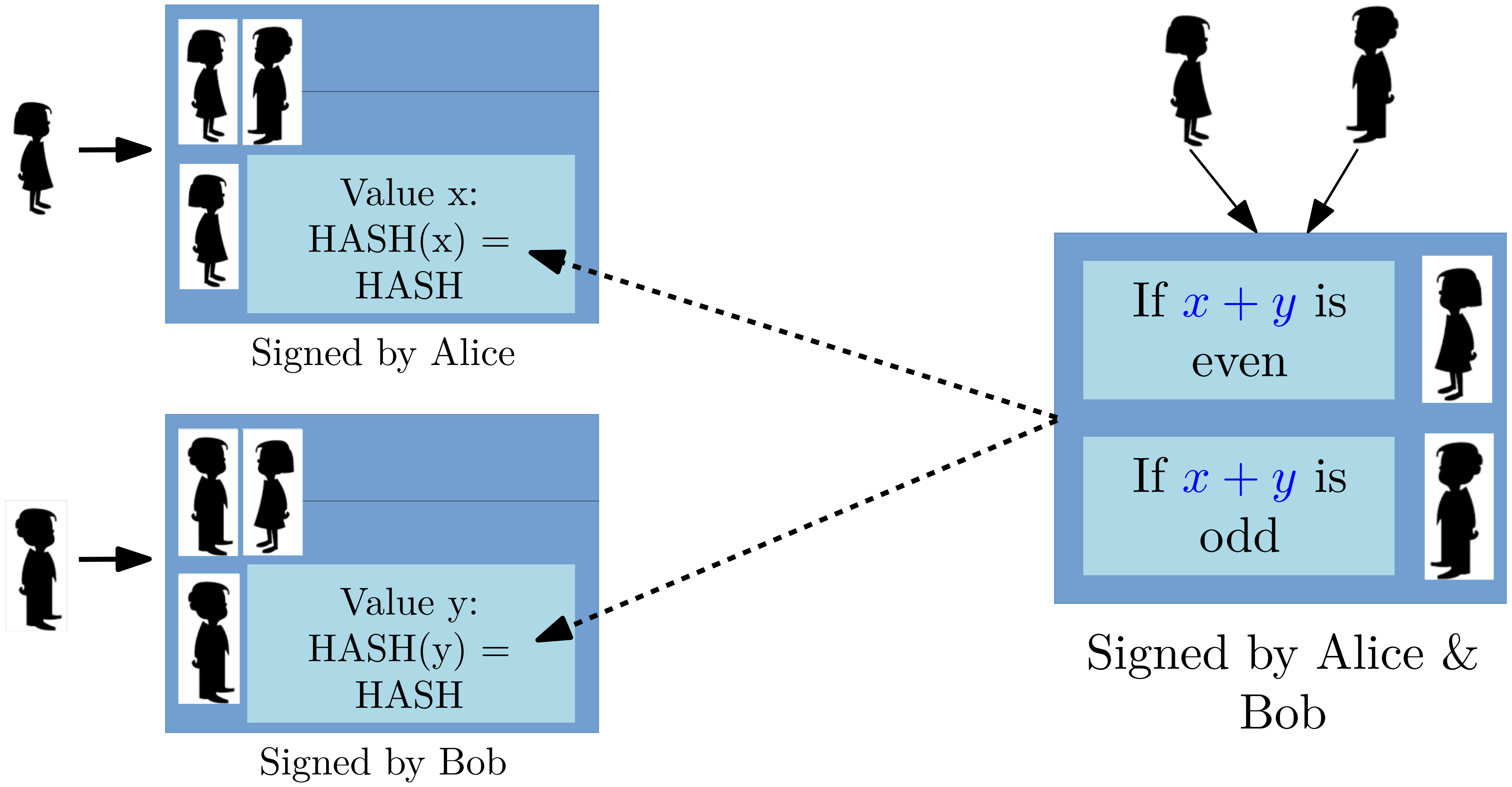
Lottery II


Value x:
HASH(x) =
HASH
Signed by Alice
nlockTime
Z
Signed by Alice
Value y:
HASH(y) =
HASH
Signed by Bob
nlockTime
Z
Signed by Bob
If $x + y$ is
even
If $x + y$ is
odd
Signed by Alice &
Bob

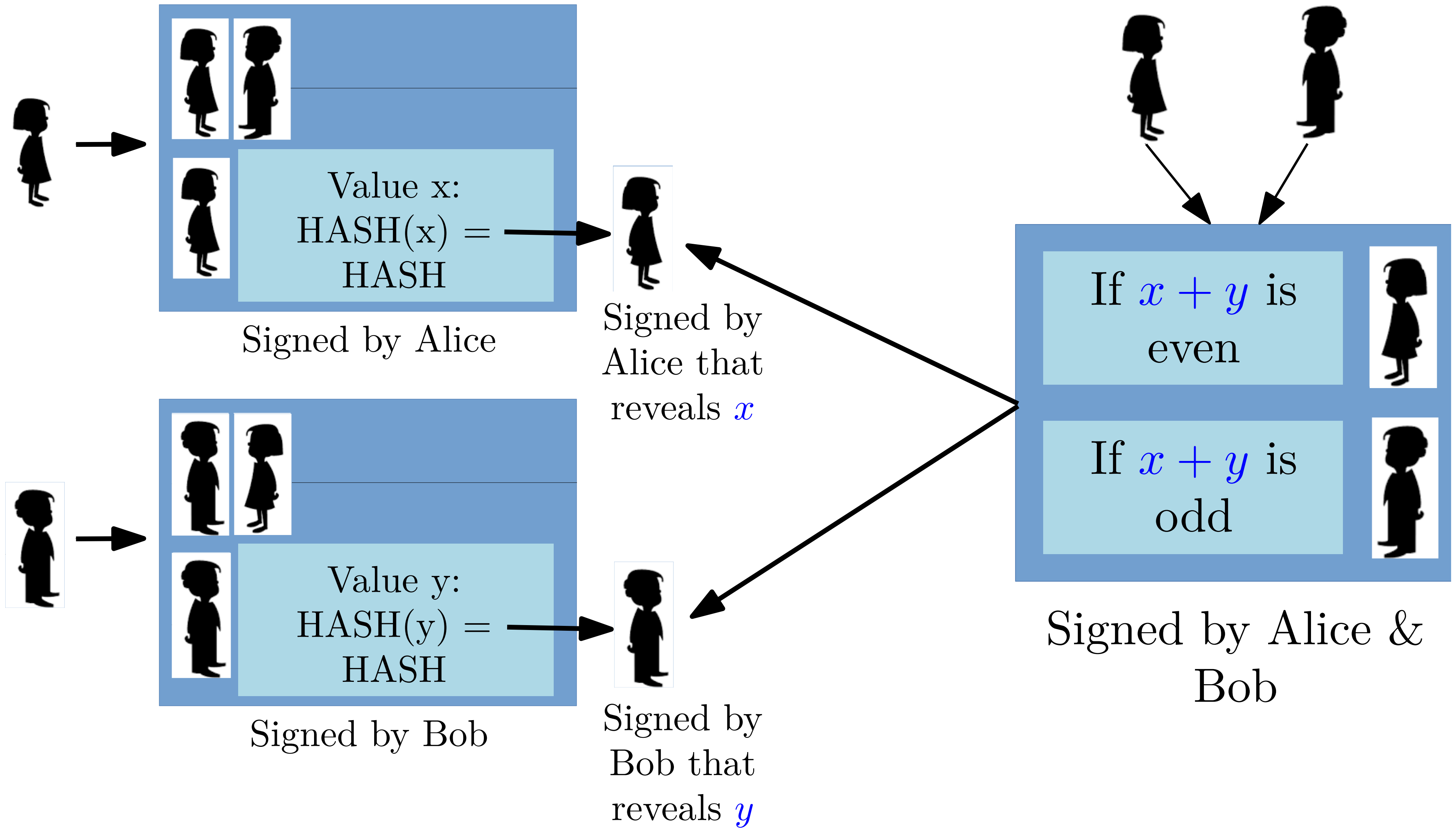
Lottery II


Value x:
HASH(x) =
HASH
Signed by Alice
nlockTime
Z
Signed by Alice
Value y:
HASH(y) =
HASH
Signed by Bob
nlockTime
Z
Signed by Bob
If $x + y$ is even
If $x + y$ is odd
Signed by Alice & Bob

Lottery II


Value x:
HASH(x) =
HASH
Signed by Alice
Value y:
HASH(y) =
HASH
Signed by Bob
If $x + y$ is even
If $x + y$ is odd
Signed by Alice & Bob

Lottery II


Value x:
HASH(x) = HASH
Signed by Alice
Signed by Alice that reveals x
Value y:
HASH(y) = HASH
Signed by Bob
Signed by Bob that reveals y
If $x + y$ is even
If $x + y$ is odd
Signed by Alice & Bob

Lottery with n participants

Every participant has to provide a deposit to each other participant.
Hence, to bet 1 BTC, n BTC have to be *employed*.
“Secure Multiparty Computations on Bitcoin”

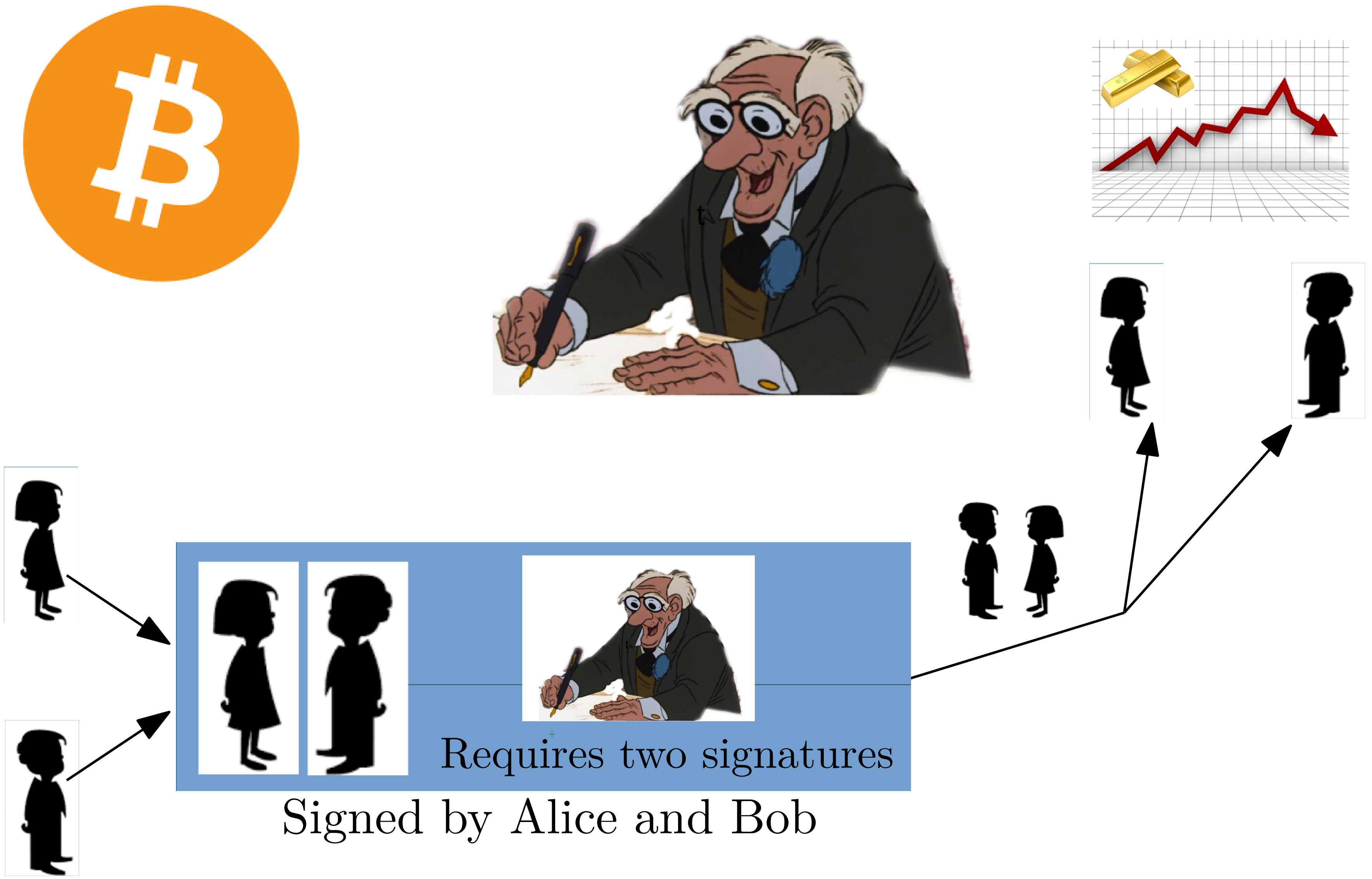
Lottery with n participants

Every participant has to provide a deposit to each other participant.
Hence, to bet 1 BTC, n BTC have to be *employed*.
"Secure Multiparty Computations on Bitcoin"

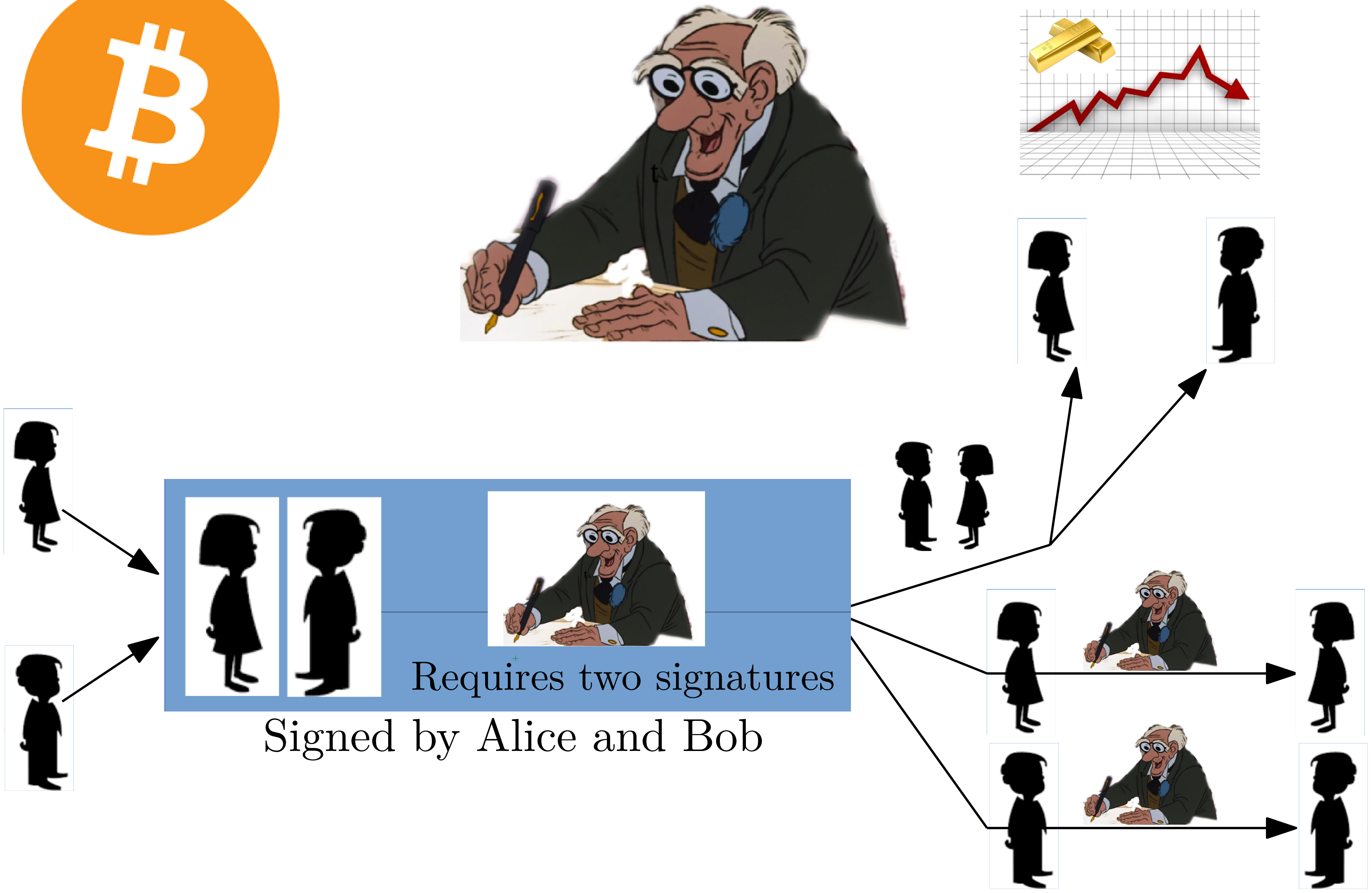
The latter limitation has recently been overcome in
"Constant-deposit multiparty lotteries on Bitcoin"

Betting on external events

Betting on external events



Betting on external events



Ethereum

Turing-Complete Language

Ethereum

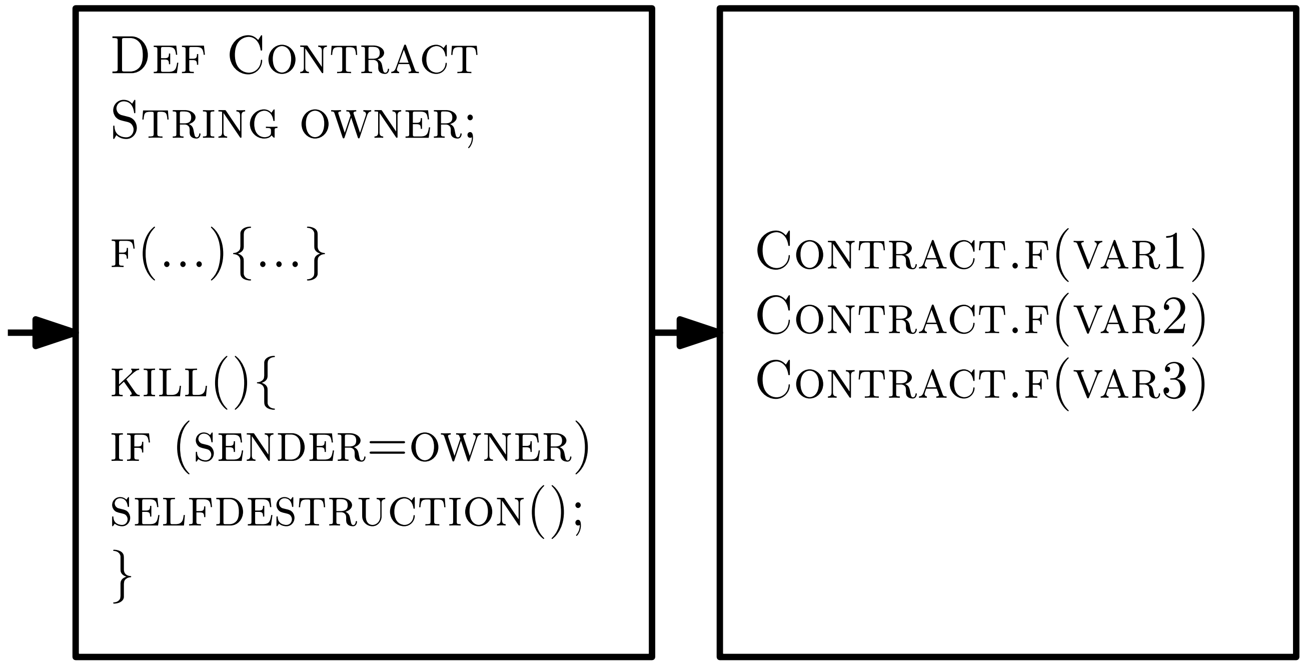
Almost Turing-Complete Language

The execution (on behalf of the miners) of the transactions/contracts costs ETHER, proportionally to the number of instructions which are executed. When someone creates a contract, he/she also specifies how many ETHER he/she is willing to pay.

Smart Contracts

```
Def Contract
String owner;

f(...){...}

kill(){
if (sender=owner)
selfdestruction();
}
```

Smart Contracts

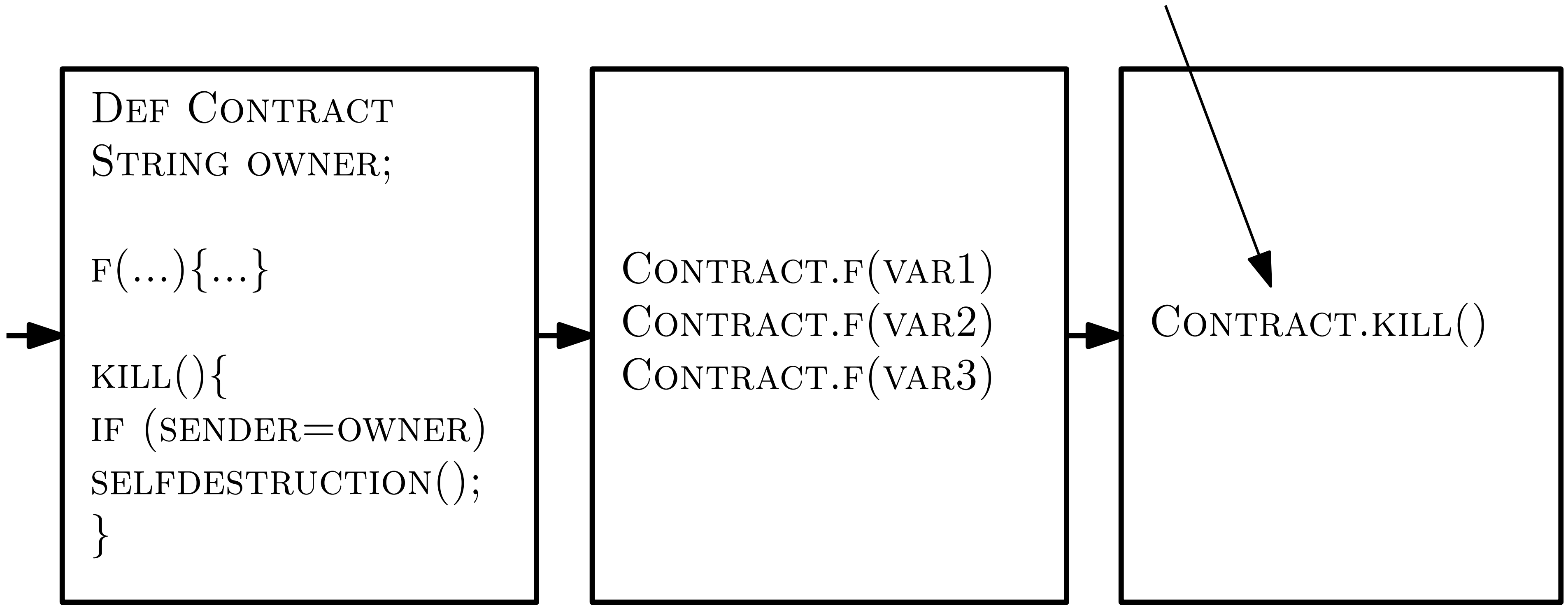


```
Def Contract
String owner;

f(...){...}

kill(){
if (sender=owner)
selfdestruction();
}
```

```
Contract.f(var1)
Contract.f(var2)
Contract.f(var3)
```

Smart Contracts

Transactions have to be signed with the public key

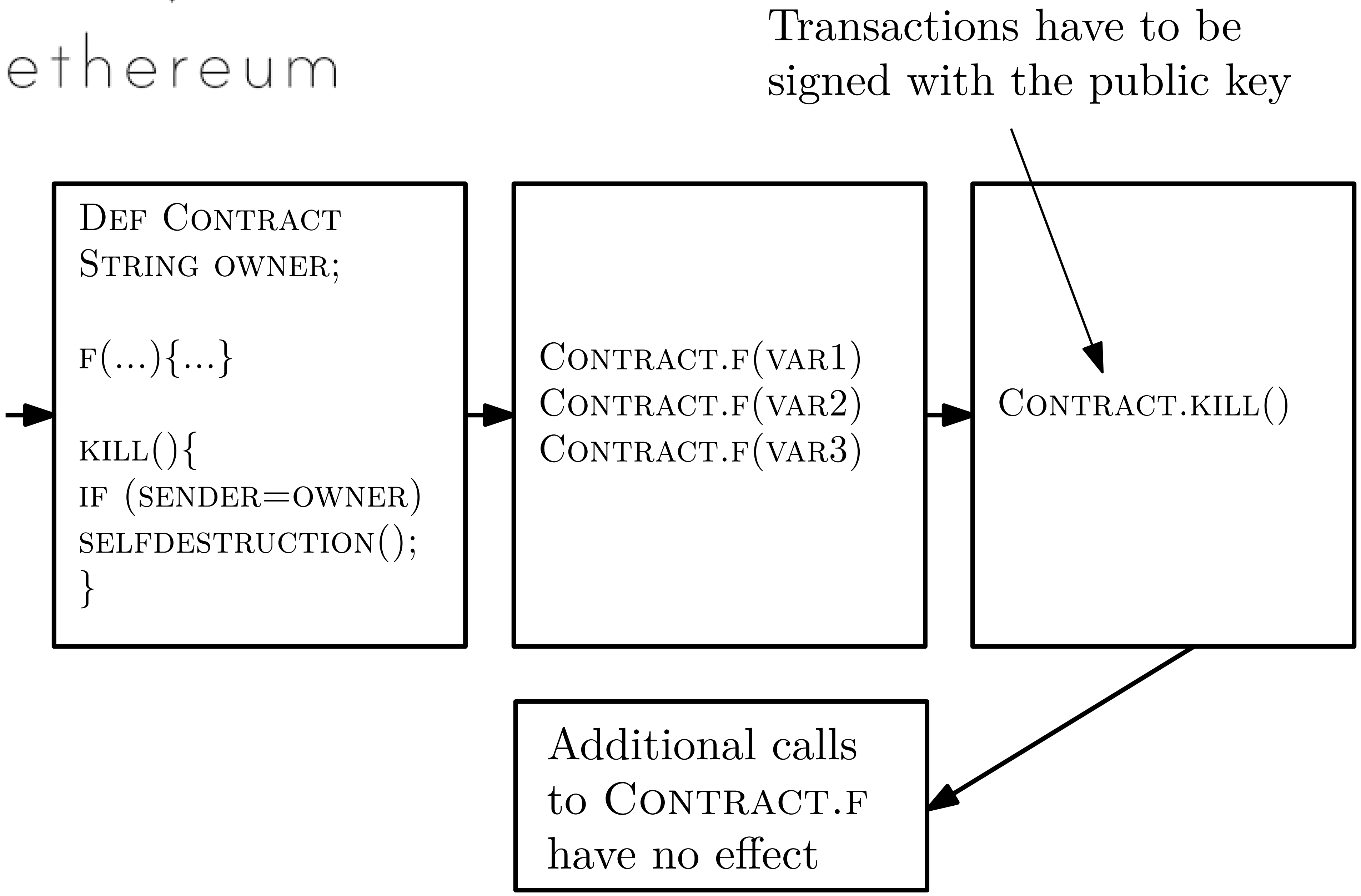


```
Def Contract
String owner;

f(...){...}

kill(){
if (sender=owner)
selfdestruction();
}
```

```
Contract.f(var1)
Contract.f(var2)
Contract.f(var3)
```

```
Contract.kill()
```

Smart Contracts



```
Def Contract
String owner;

f(...){...}

kill(){
if (sender=owner)
selfdestruction();
}
```

```
Contract.f(var1)
Contract.f(var2)
Contract.f(var3)
```

```
Contract.kill()
```

Transactions have to be signed with the public key
Additional calls to Contract.f have no effect

The DAO

A digital Decentralized Autonomous Organization

```
Def TheDAO
buyShare(...){...}
VoteProject(...){...}
SellShare(...){...}
```

The DAO

A digital Decentralized
Autonomous Organization

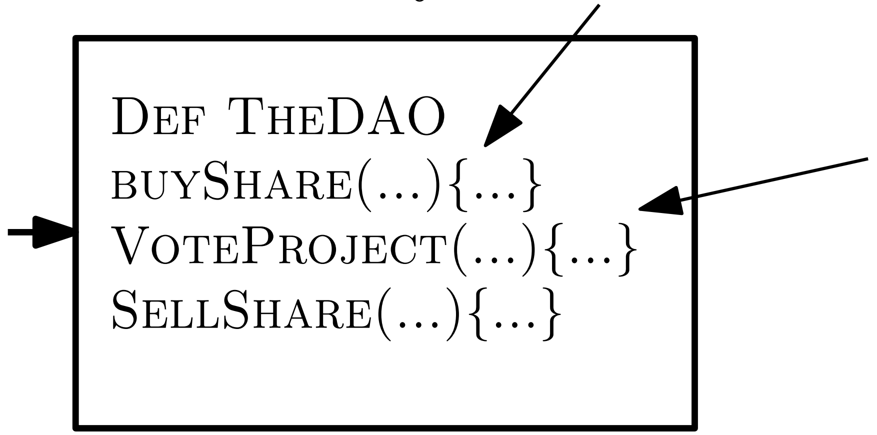
Using ethereum, it is possible
to buy shares of the DAO.

```
DEF THEDAO
BUYSHARE(...){...}
VOTEPROJECT(...){...}
SELLSHARE(...){...}
```

The DAO

A digital Decentralized Autonomous Organization

Using ethereum, it is possible to buy shares of the DAO.



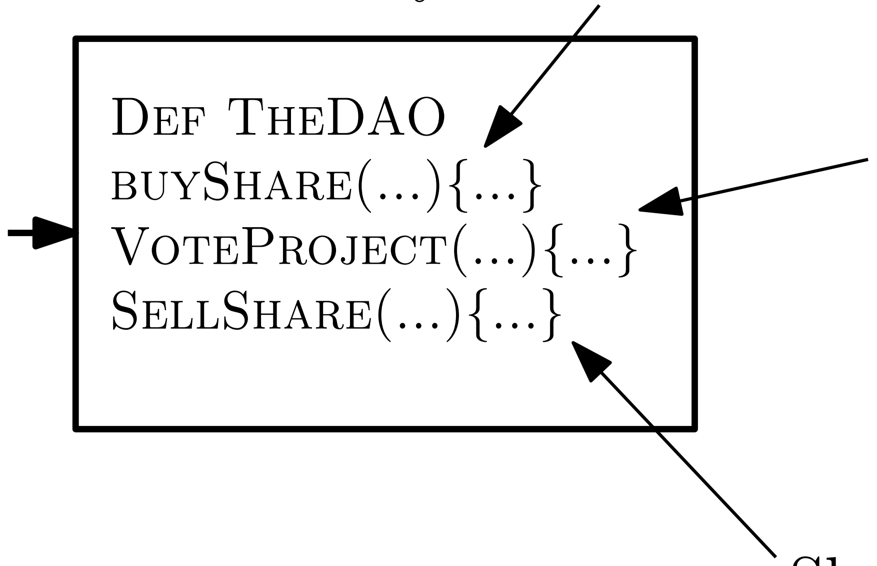
```
DEF THEDAO
BUYSHARE(...){...}
VOTEPROJECT(...){...}
SELLSHARE(...){...}
```

Shares of the DAO can be used to vote which projects should be financed.

The DAO

A digital Decentralized Autonomous Organization

Using ethereum, it is possible to buy shares of the DAO.



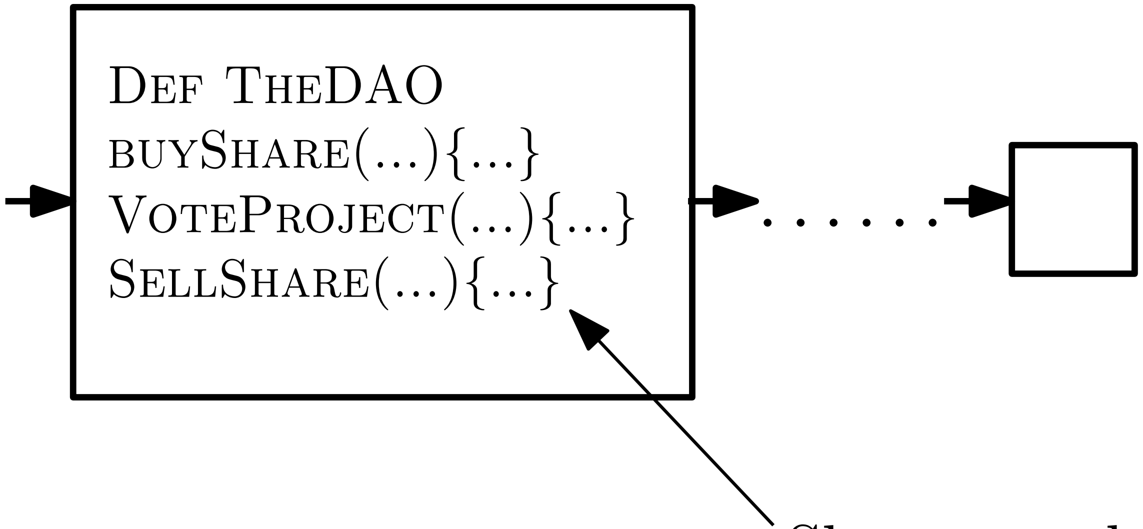
```
DEF THEDAO
BUYSHARE(...){...}
VOTEPROJECT(...){...}
SELLSHARE(...){...}
```

Shares of the DAO can be used to vote which projects should be financed.

Shares can be returned in exchange for ethereum.

The DAO

A digital Decentralized Autonomous Organization

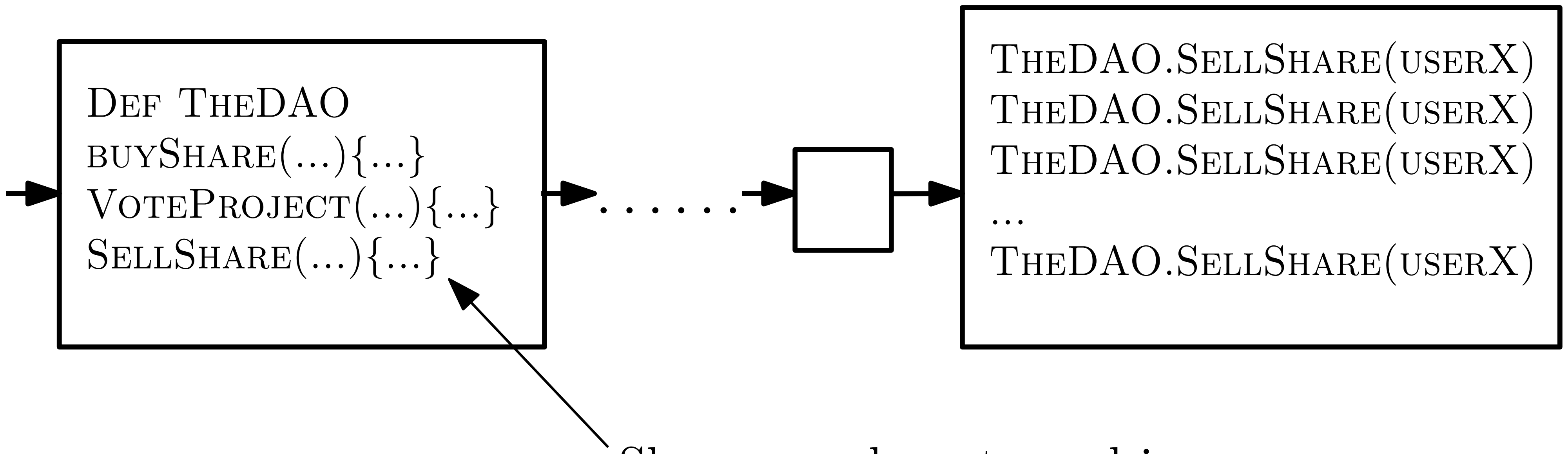


```
DEF THEDAO
BUYSHARE(...){...}
VOTEPROJECT(...){...}
SELLSHARE(...){...}
```

Shares can be returned in exchange for ethereum.

The DAO

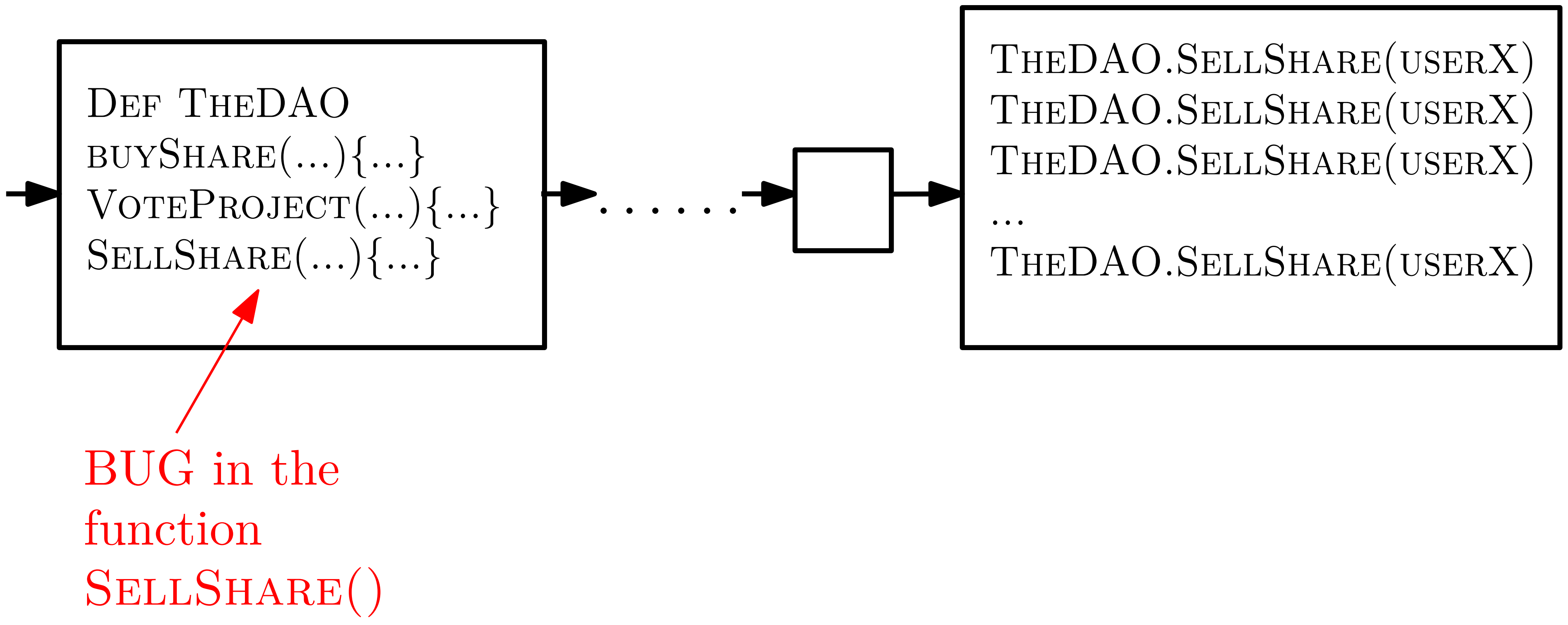
A digital Decentralized Autonomous Organization



Shares can be returned in exchange for ethereum.

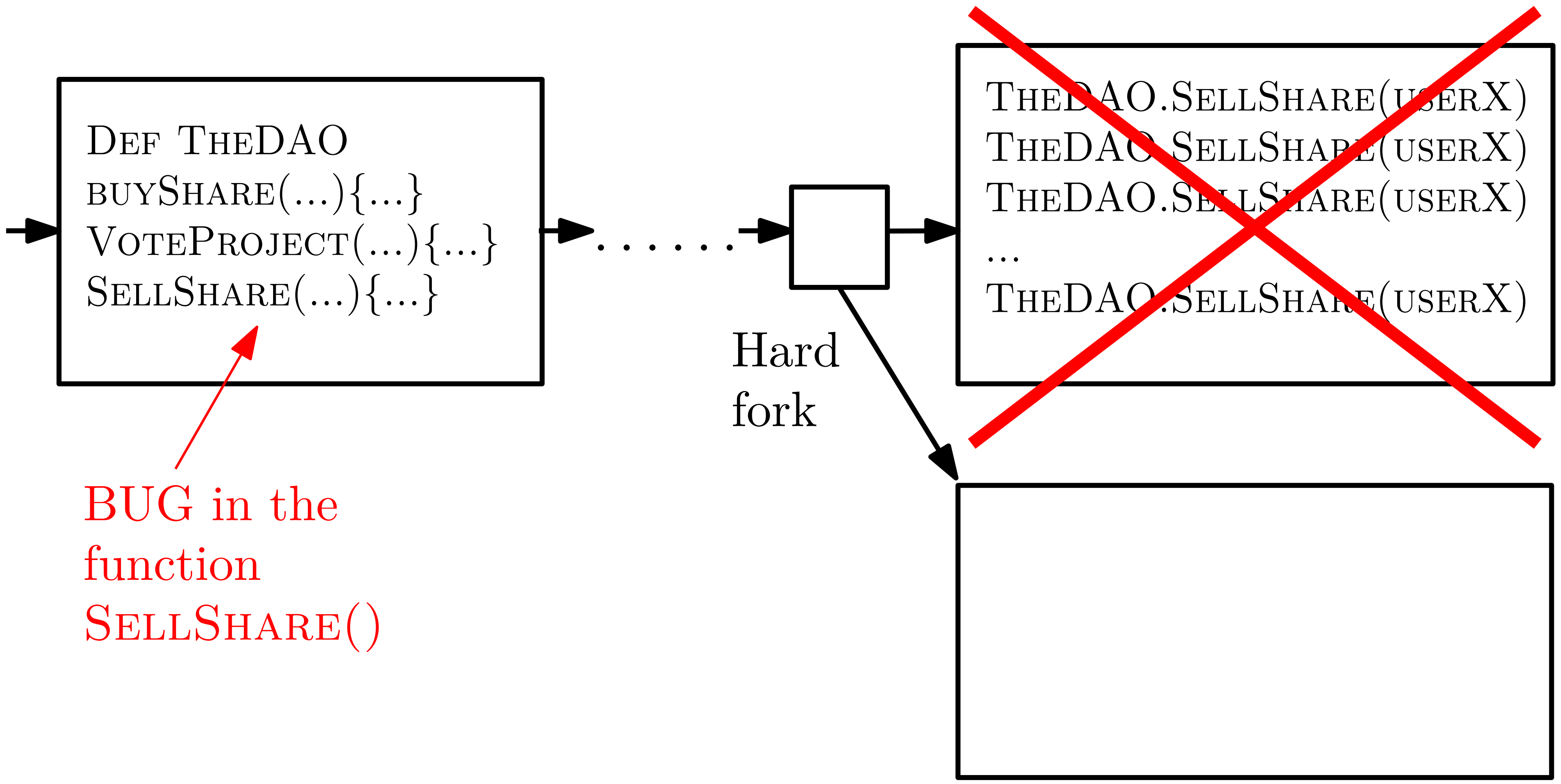
The DAO

A digital Decentralized Autonomous Organization



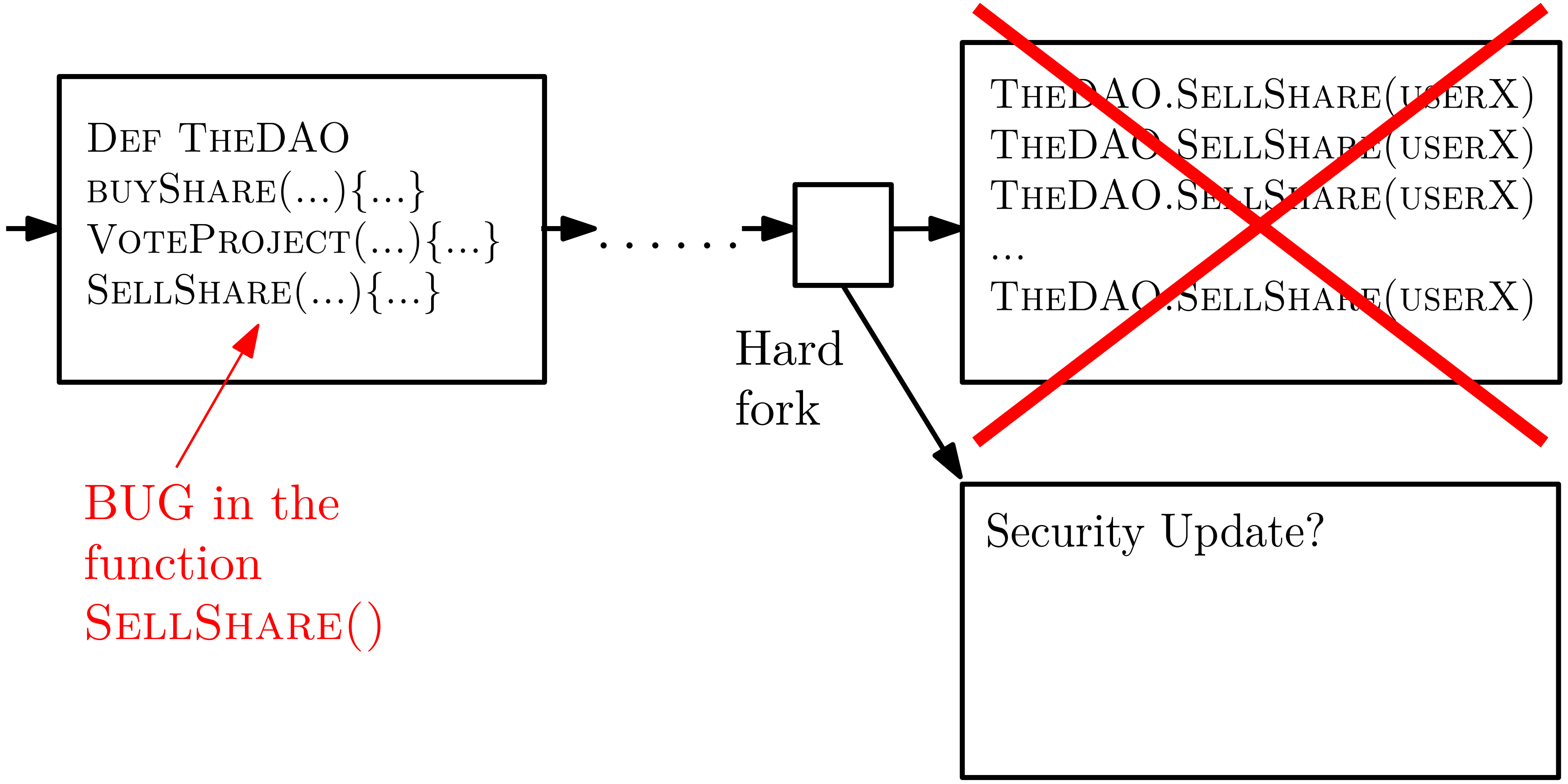
The DAO

A digital Decentralized Autonomous Organization



The DAO

A digital Decentralized Autonomous Organization

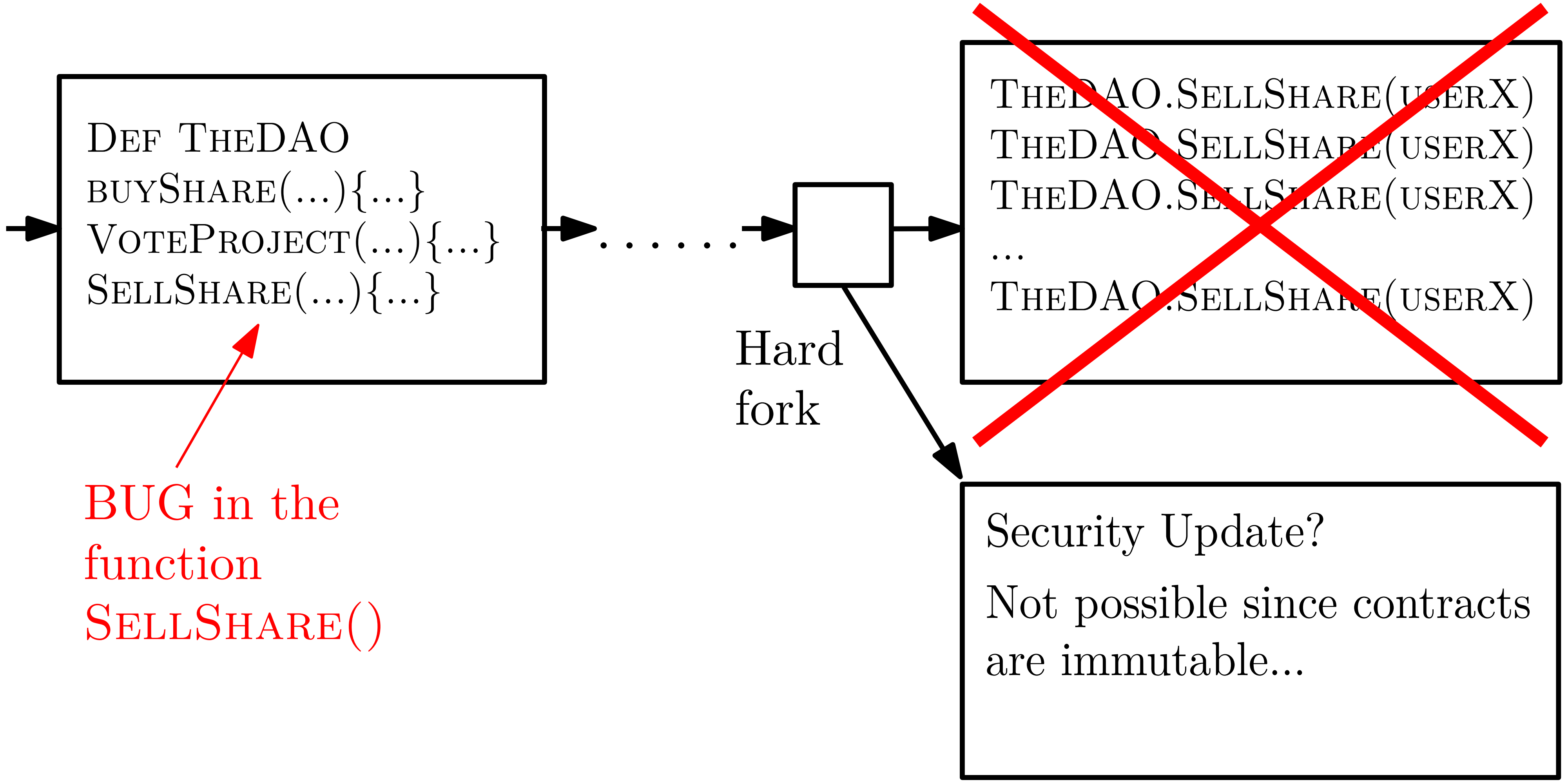


```
DEF THEDAO
BUYSHARE(...){...}
VOTEPROJECT(...){...}
SELLSHARE(...){...}
```

~~THEDAO.SELLSHARE(USERX)~~
~~THEDAO.SELLSHARE(USERX)~~
~~THEDAO.SELLSHARE(USERX)~~
~~...~~
~~THEDAO.SELLSHARE(USERX)~~

The DAO

A digital Decentralized Autonomous Organization



```
DEF THEDAO
BUYSHARE(...){...}
VOTEPROJECT(...){...}
SELLSHARE(...){...}
```

BUG in the function SELLSHARE()

~~THEDAO.SELLSHARE(USERX)~~
~~THEDAO.SELLSHARE(USERX)~~
~~THEDAO.SELLSHARE(USERX)~~
~~...~~
~~THEDAO.SELLSHARE(USERX)~~

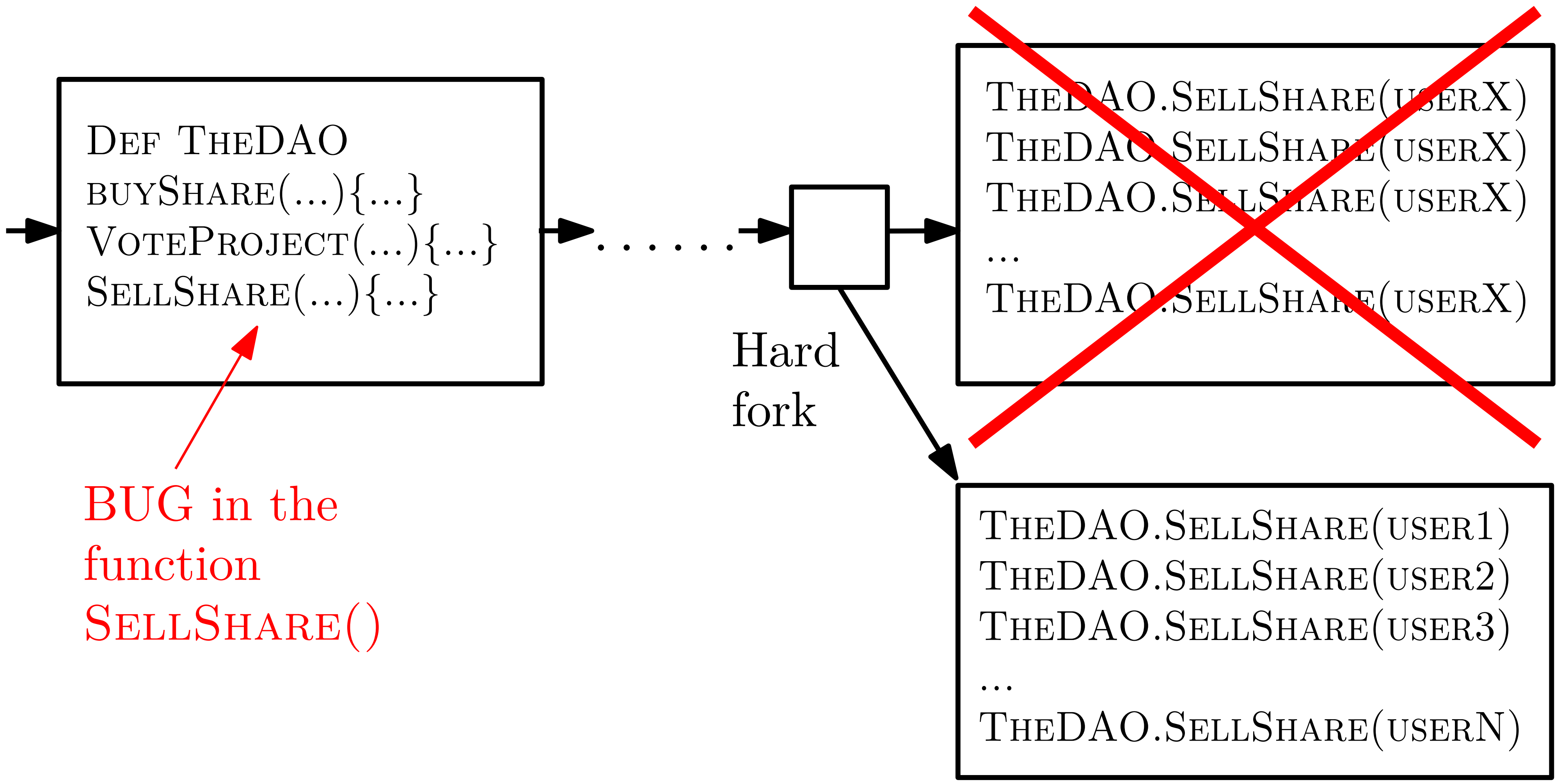
Security Update?

Not possible since contracts are immutable...

The DAO

A digital Decentralized Autonomous Organization

ethereum



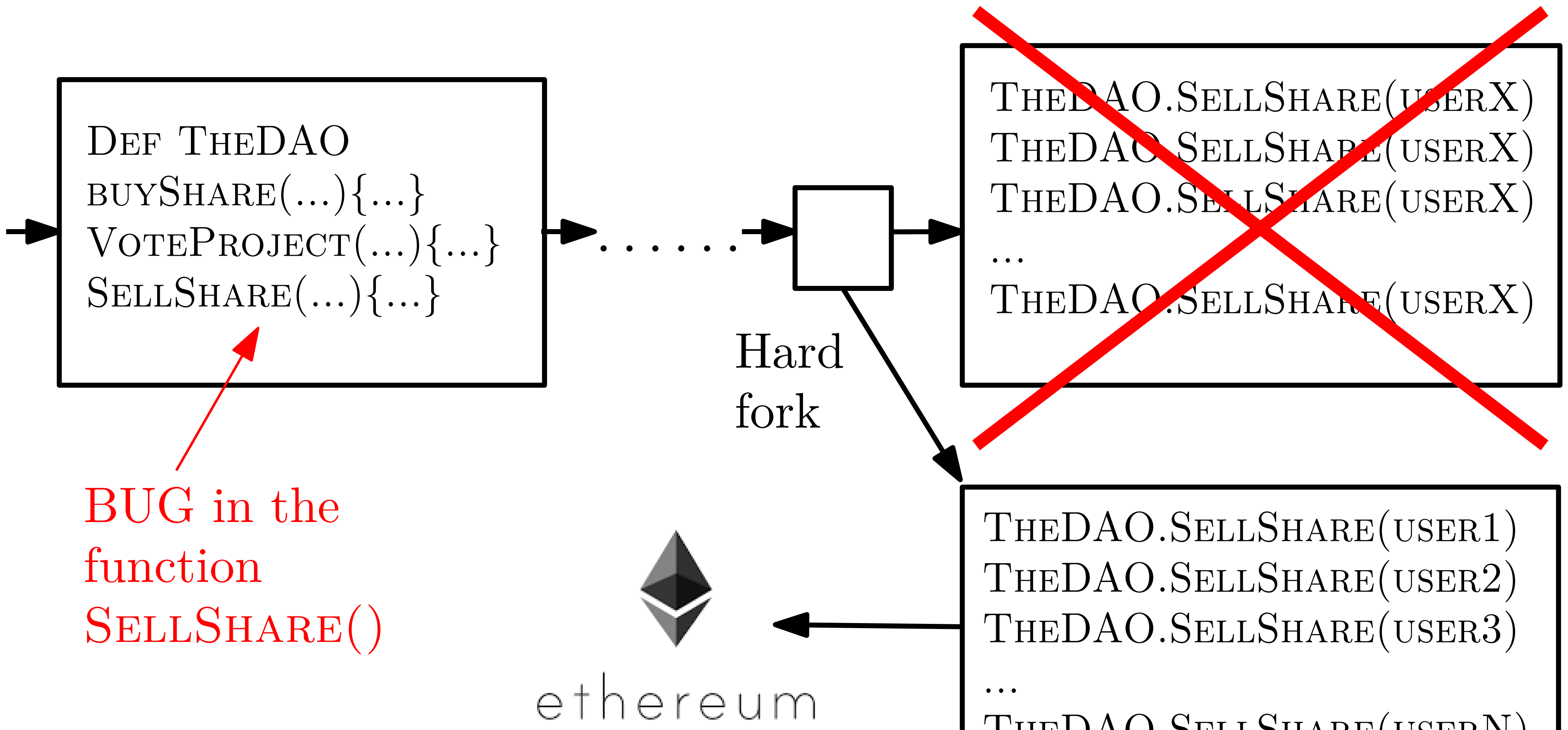
```
Def TheDAO
buyShare(...){...}
VoteProject(...){...}
SellShare(...){...}
```

```
~~TheDAO.SellShare(userX)~~
~~TheDAO.SellShare(userX)~~
~~TheDAO.SellShare(userX)~~
~~...~~
~~TheDAO.SellShare(userX)~~
```

```
TheDAO.SellShare(user1)
TheDAO.SellShare(user2)
TheDAO.SellShare(user3)
...
TheDAO.SellShare(userN)
```

The DAO

A digital Decentralized Autonomous Organization



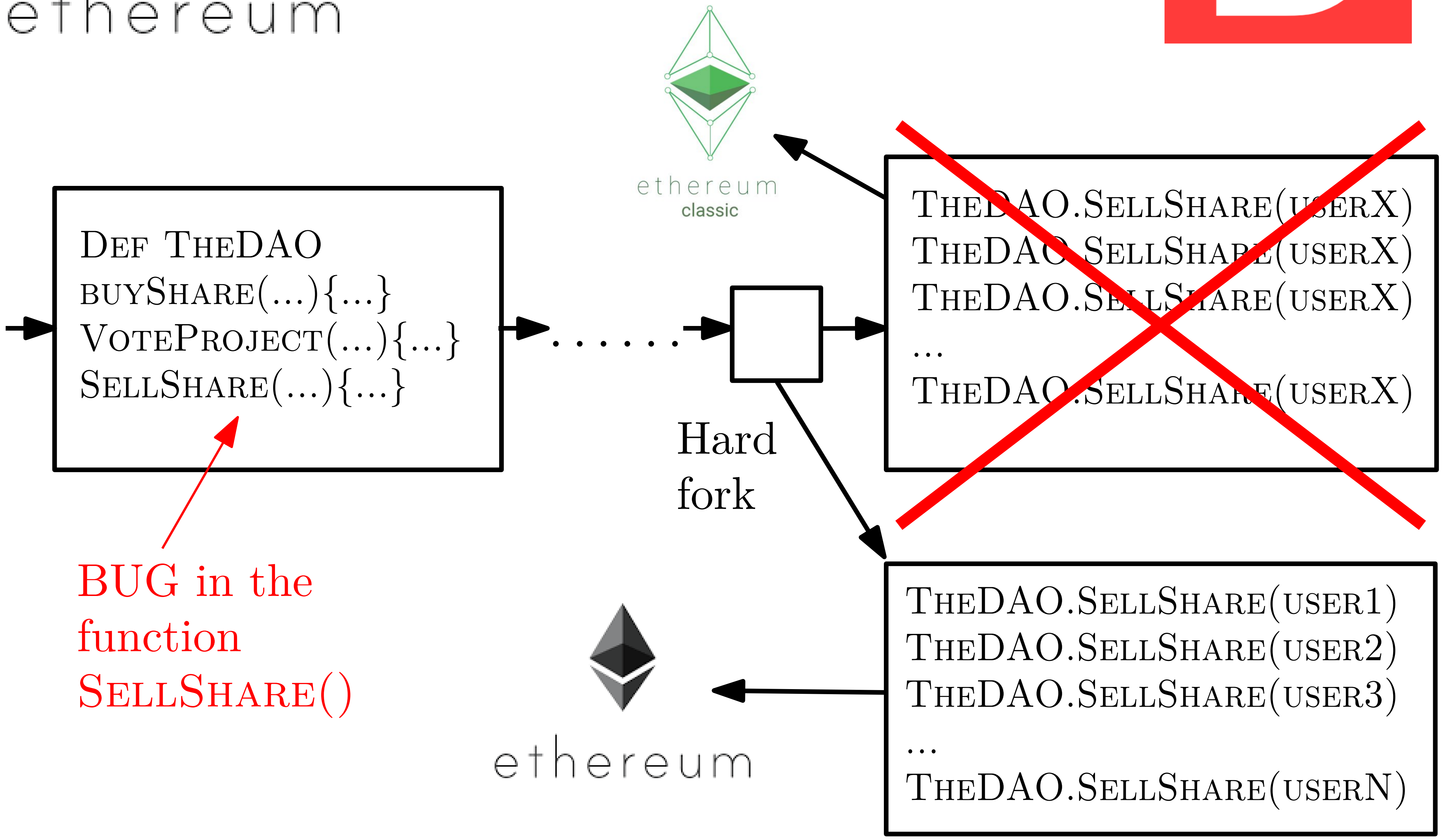
```
DEF THEDAO
BUYSHARE(...){...}
VOTEPROJECT(...){...}
SELLSHARE(...){...}
```

```
THEDAO.SELLSHARE(USERX)
THEDAO.SELLSHARE(USERX)
THEDAO.SELLSHARE(USERX)
...
THEDAO.SELLSHARE(USERX)
```

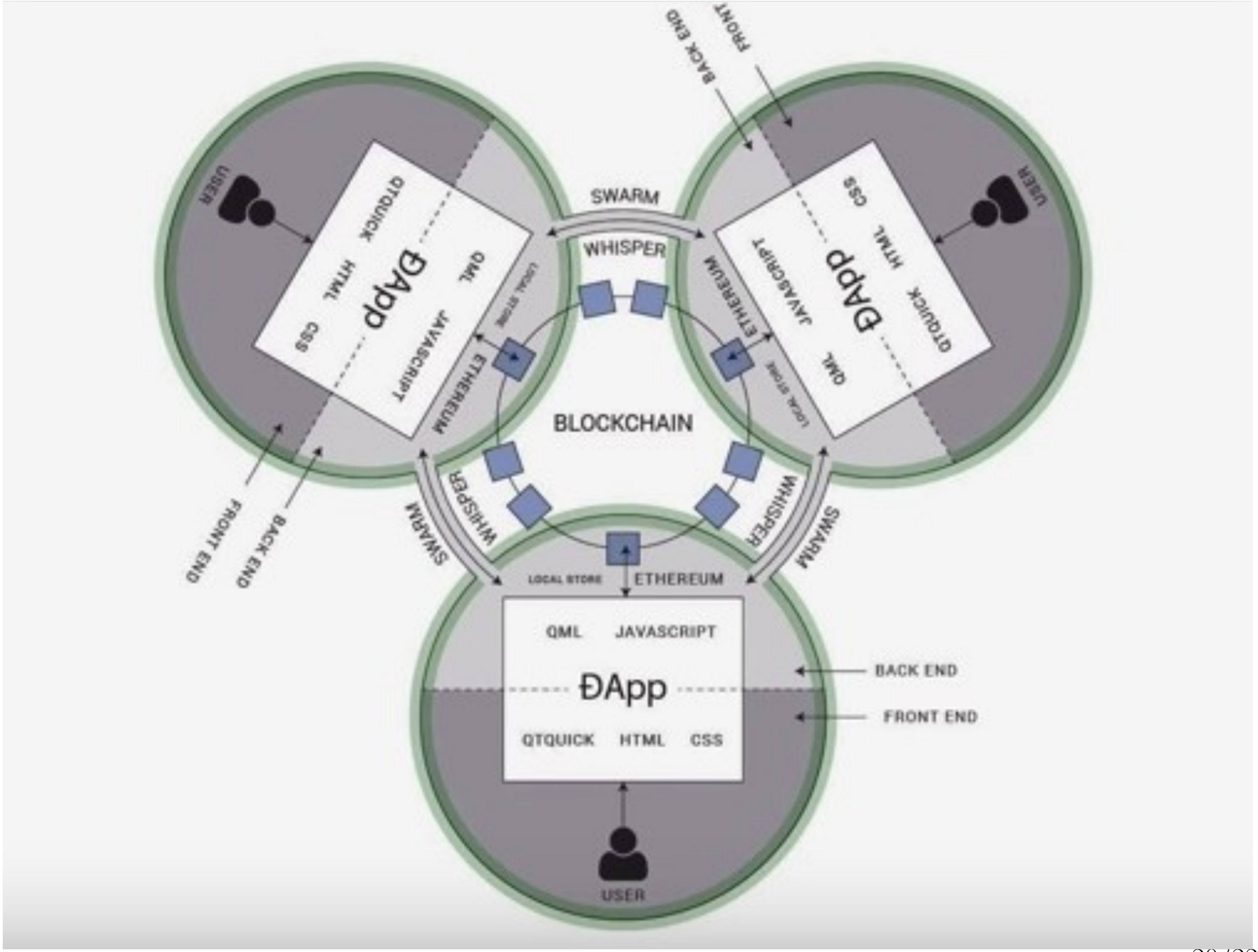
```
THEDAO.SELLSHARE(USER1)
THEDAO.SELLSHARE(USER2)
THEDAO.SELLSHARE(USER3)
...
THEDAO.SELLSHARE(USERN)
```

The DAO

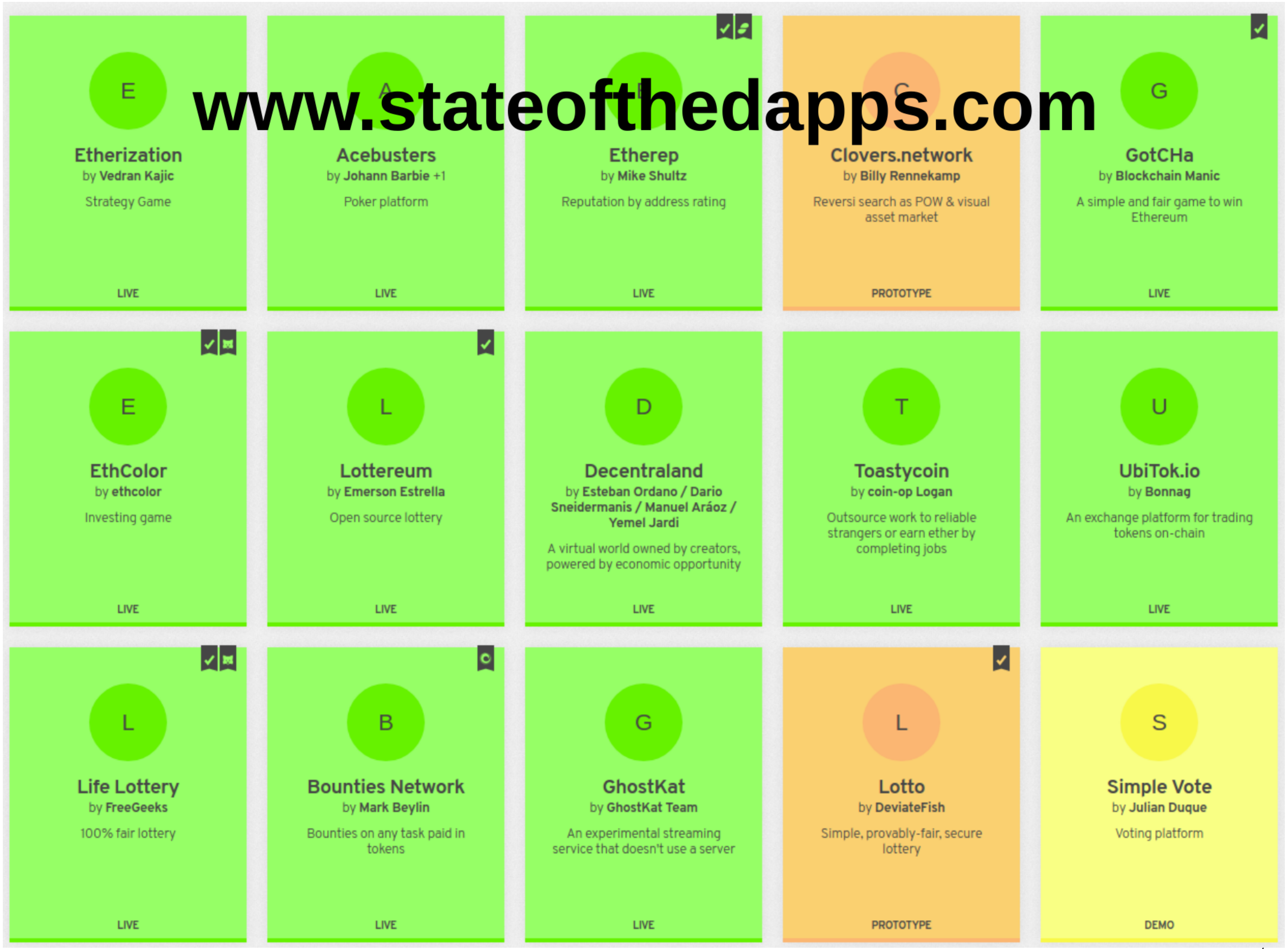
A digital Decentralized Autonomous Organization



A Dystopic Scenario


SWARM
WHISPER
BLOCKCHAIN
ETHEREUM
LOCAL STORE
ĐApp
QML
JAVASCRIPT
QTQUICK
HTML
CSS
BACK END
FRONT END
USER

www.stateofthedapps.com



Etherization
by **Vedran Kajic**
Strategy Game
LIVE

Acebusters
by **Johann Barbie** +1
Poker platform
LIVE

Etherep
by **Mike Shultz**
Reputation by address rating
LIVE

Clovers.network
by **Billy Rennekamp**
Reversi search as POW & visual asset market
PROTOTYPE

GotCHa
by **Blockchain Manic**
A simple and fair game to win Ethereum
LIVE

EthColor
by **ethcolor**
Investing game
LIVE

Lottereum
by **Emerson Estrella**
Open source lottery
LIVE

Decentraland
by **Esteban Ordano / Dario Sneidermanis / Manuel Aráoz / Yemel Jardi**
A virtual world owned by creators, powered by economic opportunity
LIVE

Toastycoin
by **coin-op Logan**
Outsource work to reliable strangers or earn ether by completing jobs
LIVE

UbiTok.io
by **Bonnag**
An exchange platform for trading tokens on-chain
LIVE

Life Lottery
by **FreeGeeks**
100% fair lottery
LIVE

Bounties Network
by **Mark Beylin**
Bounties on any task paid in tokens
LIVE

GhostKat
by **GhostKat Team**
An experimental streaming service that doesn't use a server
LIVE

Lotto
by **DeviateFish**
Simple, provably-fair, secure lottery
PROTOTYPE

Simple Vote
by **Julian Duque**
Voting platform
DEMO

Thank You!