Smart Contracts in Blockchain

An implementation-agnostic definition

Prof. Marco Comuzzi
Department of Industrial Engineering
Ulsan National Institute of Science and Technology (UNIST)
mcomuzzi@unist.ac.kr

1.
THE CASE FOR SMART CONTRACTS

What is blockchain?

A “distributed state machine”

A system allowing nodes to maintain the values of a set of variables defining the “state” of a system, without the need for a central intermediary

This is enabled by the 5 ingredients of the blockchain recipe

Marco Comuzzi mcomuzzi@unistac.kr

To sum up … what is blockchain?

Network

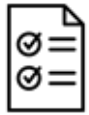
[peer-to-peer network]

Data structure

[distributed ledger]

Protocol

[consensus mechanism].



Hashing

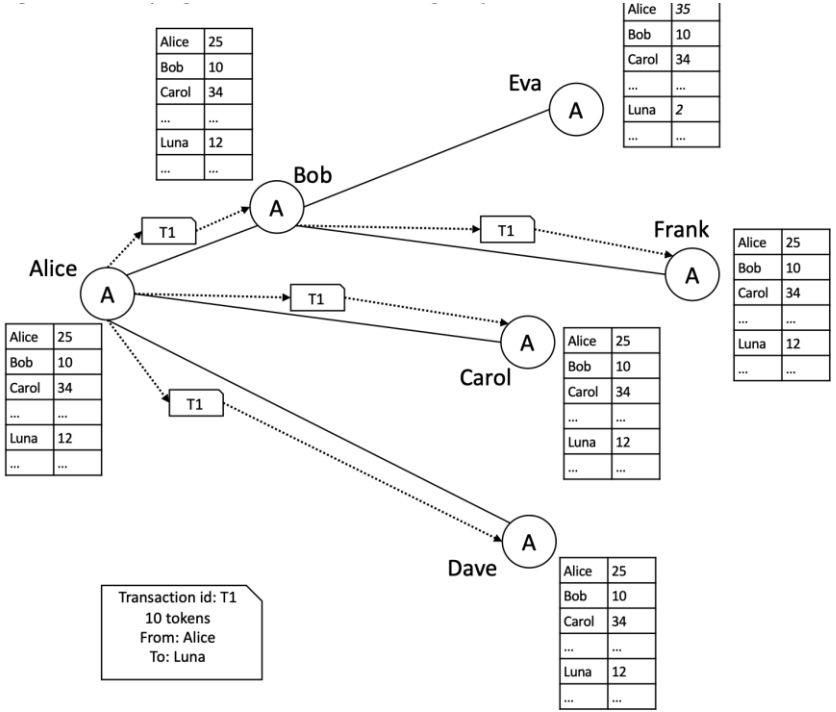
[Immutability of ledger]

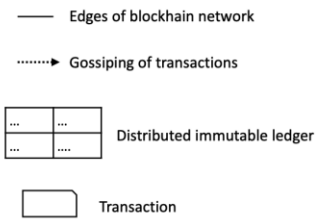
Digital signatures

[identity of nodes]

Marco Comuzzi mcomuzzi@unistac.kr

Ledger as state of the system: cryptocurrency





The state of the system is the balance of all nodes

Transactions are transfers of cryptocurrency units between nodes

Marco Comuzzi mcomuzzi@unistac.kr

Ledger as transaction list: BC4C

Content of the ledger (transactions)

Transaction id	Transaction	Originator	Timestamp
0	Genesis of BC4C	BC4C	22-04-19 12:00:17
1	CreateNewGame[AliceCarol1, Carol]	Alice	22-04-20 09:00:03
2	CreateNewGame[BobCarol1, Carol]	Bob	22-04-20 09:00:45
3	ConfirmGameCreation[BobCarol1]	Carol	22-04-20 09:01:23
4	ConfirmGameCreation[AliceCarol1]	Carol	22-04-20 09:01:28
5	Move[BobCarol1, 1, pawn_h2, h3, false]	Bob	22-04-20 09:02:34
6	CreateNewGame[DaveAlice1, Alice]	Dave	22-04-20 09:02:48
7	Move[AliceCarol1, 1, pawn_d2, d4, false]	Alice	22-04-20 09:03:01
8	ConfirmMove[AliceCarol1, 1]	Carol	22-04-20 09:03:55
9	Move[AliceCarol, 2, pawn_a2, a3, false]	Carol	22-04-20 09:04:26

State of the System (games currently playing)

AliceCarol1

BobCarol1

The state of the system is the current state of games

Transactions are the moves of the games

Is that all?

The "state" in these examples is very simple

Most importantly, states update are "static":
state updated only by nodes when issuing new transactions

No "business logic" associated with the state update

Can we extend blockchain with logic that manipulates the value of the state variables based on the input provided by transactions?

Marco Comuzzi mcomuzzi@unistac.kr

Motivating Example: BC4C-CGM

Let's extend the BC4C with cryptocurrency ("CGM")

Nodes have also a CGM balance, CGM tokens can be exchanged by nodes like any other cryptocurrency

[it is not important to know how CGM have been created]

To make chess games more competitive, players in a game must pay a deposit when a game starts, the winner of a game will get their deposit back, and the one of the counterpart.

Marco Comuzzi mcomuzzi@unistac.kr


MUVI
E-SPORTS
THE FUTURE IS HERE

2.
DEFINING SMART CONTRACTS

From BC4C to BC4C-CGM

Can we design BC4C-CGM using what we know thus far?

Not really…

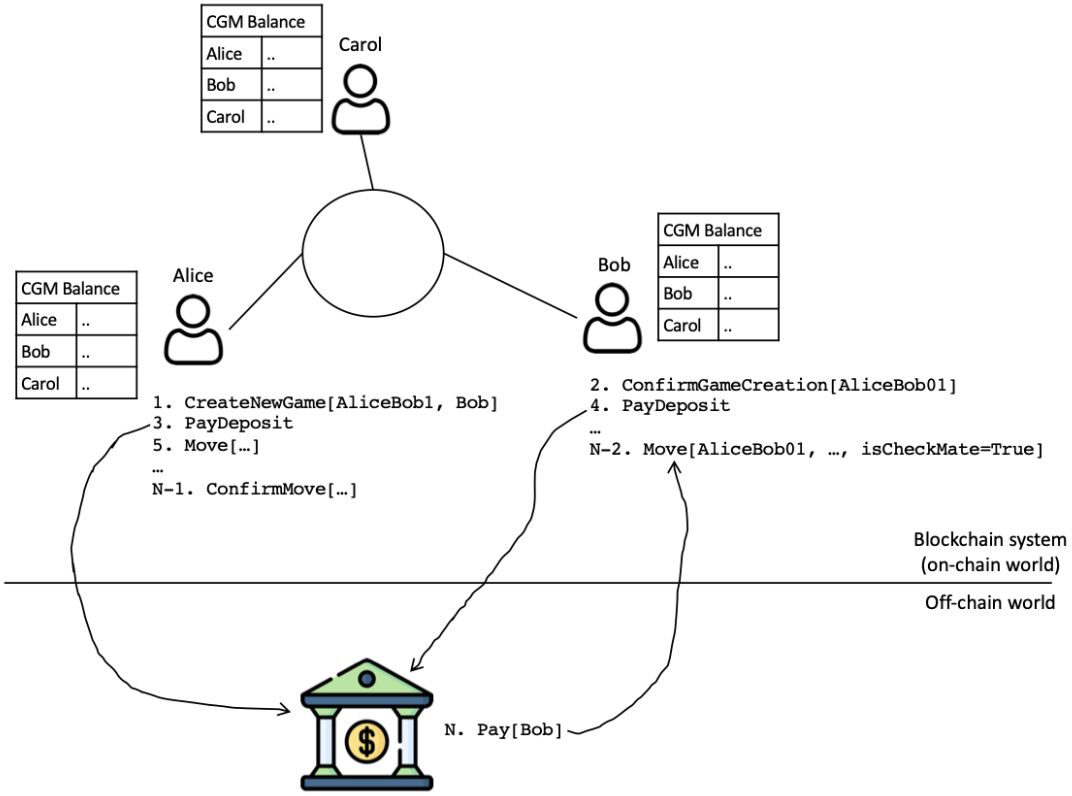
Who or where do we store the deposits while a game is being played?

How can we "force" the losers to pay their deposit to the winner?

It seems like we need an intermediary…

Marco Comuzzi mcomuzzi@unistac.kr

Off-chain design of BC4C-CGM



If we can convince the players to play the deposits to a bank (off-chain)…

… the bank can hold the deposits and pay it back to the winner when a game ends

OK, but not blockchain ☺

What if...

... instead of a bank, we write a simple computer program that players can call only once before a game starts to pay the deposits

A game cannot start if both players have not paid their deposit

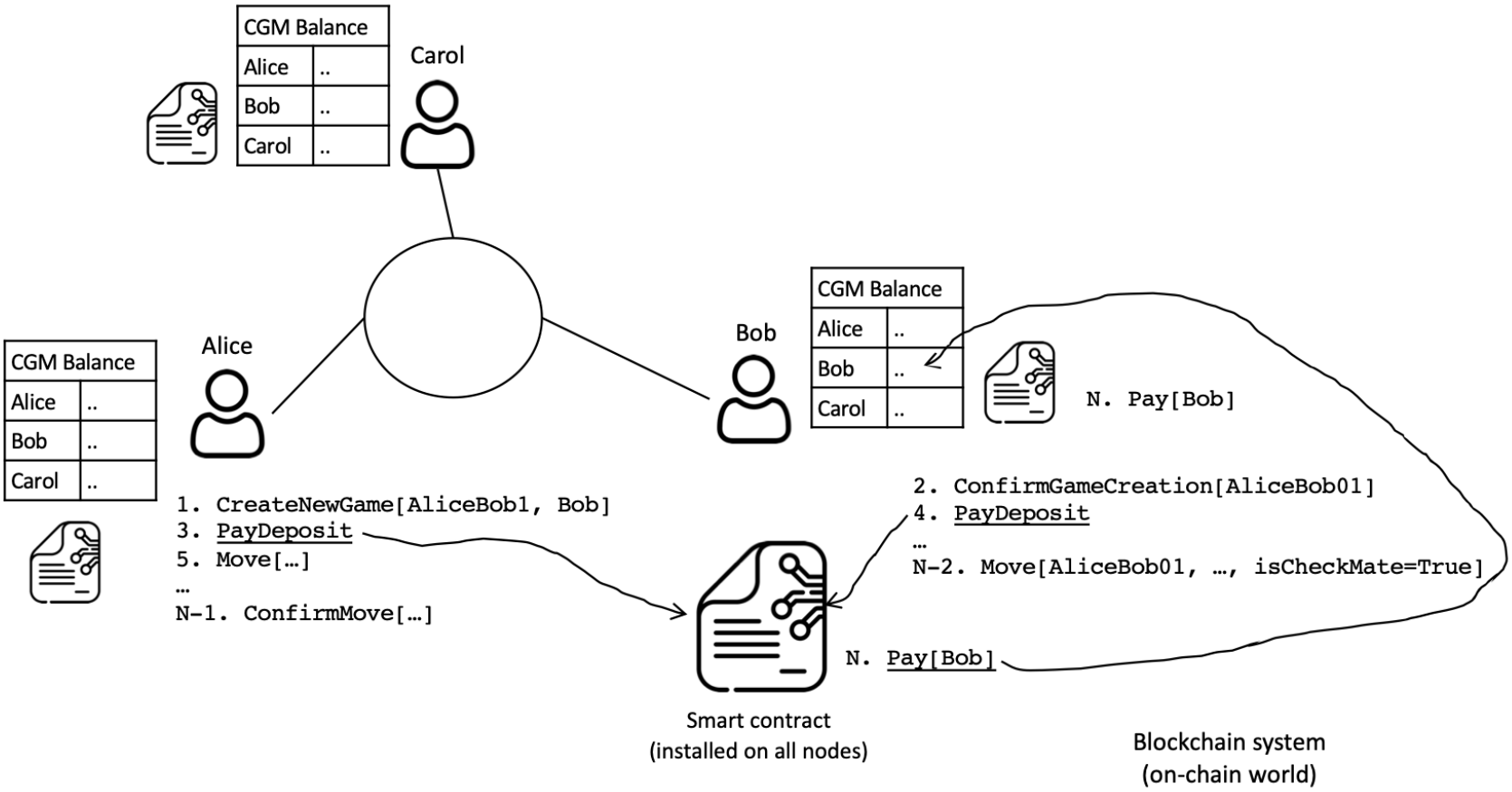
Every node installs this program locally, transactions can trigger the execution of this program

The program monitors the state of the blockchain and, when a game ends, pays the deposits back to the winner

Note:
pay a deposit = modify the value of blockchain state variables (balances of the players)

(This computer program is a “smart contract”)

Marco Comuzzi mcomuzzi@unistac.kr


CGM Balance
Alice ..
Bob ..
Carol ..
Carol
Alice
Bob
CGM Balance
Alice ..
Bob ..
Carol ..
CGM Balance
Alice ..
Bob ..
Carol ..
N. Pay[Bob]
1. CreateNewGame[AliceBob1, Bob]
3. PayDeposit
5. Move[…]
…
N-1. ConfirmMove[…]
2. ConfirmGameCreation[AliceBob01]
4. PayDeposit
…
N-2. Move[AliceBob01, …, isCheckMate=True]
N. Pay[Bob]
Smart contract
(installed on all nodes)
Blockchain system
(on-chain world)

Welcome smart contracts

Simply stated, a smart contract is a computer program that can manipulate the state of a blockchain

Smart contracts do not have to be legally binding “contracts”

Smart contracts are not very “smart” (actually, they have a lot of limitations!)

Smart contracts

Business logic

Are there any limits to the logic that can be implemented by a smart contract?

Data

Which type of data can be handled by smart contracts?

Properties

What properties are specific to the fact that smart contracts operate within a blockchain?

Marco Comuzzi mcomuzzi@unistac.kr

What kind of logic can be implemented in a smart contract?

Stateless

Given an input, always returns the same output

STATELESS

```
calcDiscount(price)
 discount = price * 0.2
 return discount
```

Stateful

Have an internal state, so the output depends on the input and the internal state

STATEFUL

```
Initial state:
calls = 0

calcDiscount(price)
 if calls < 10
  discount = price * 0.2
 else
  discount 0
 calls = calls + 1
 return discount
```

BC4C-CGM Smart Contract

A stateful smart contract

The state of the SC is the list of games for which a deposit has been paid by the players

If a game is not registered, an error is returned if one tries to pay a deposit (i.e., "the game does not exist!").
E.g. `payDeposit(BobAlice01)`

If a game is already registered, then the smart contract accepts the deposit and updates its own state (recording the fact that a player has paid the deposit). Deposit can be paid only once, so an error is returned if a player tries to pay a deposit more than once

E.g. `payDeposit(AliceCarol02)`

State
AliceCarol01(Alice: paid, Carol: paid)
BobDave01(Bob: paid, Dave: paid)
BobAlice01(Bob: paid, Alice: paid)
AliceCarol02(Carol: paid)

register(gameID)
payDeposit(gameID)
recordOutcome(gameID,winner)

Marco Comuzzi mcomuzzi@unistac.kr

To sum up

Smart contracts are computer programs written is some computer language understood by all the nodes of a blockchain system.

Smart contracts expose different functions, which may be invoked by the nodes of a blockchain system.

Smart contracts may maintain an internal state, which can be modified by the node invocations.

The execution of a smart contract does at least one of the following:

- Modifies the internal state of the smart contract.
- Returns a value as the result of an internal computation of the smart contract.
- Modifies the state of the blockchain system.

Marco Comuzzi mcomuzzi@unistac.kr

Which data can be handled by a smart contract?

ON-CHAIN DATA

State
AliceCarol01(Alice: paid, Carol: paid)
BobDave01(Bob: paid, Dave: paid)
BobAlice01(Bob: paid, Alice: paid)
AliceCarol02(Carol: paid)

register(gameID)
payDeposit(gameID)
recordOutcome(gameID,winner)

Input parameters. Provided as data of a transaction that invokes the smart contract
(similar to function invocation parameters in any programming language)

Internal state variables. Define the state of the SC and persist beyond individual invocations
(similar to "global variables" in a procedural programming language)

Blockchain state variables. For example, the balance of every user in a cryptocurrency of BC4C-CGM.

Marco Comuzzi mcomuzzi@unistac.kr

Functions, parameters, variables, …

Is there any difference between smart contracts and any other computer program?

Marco Comuzzi mcomuzzi@unistac.kr

Immutability of smart contracts

Smart contracts can modify the state of the blockchain, so they also must be immutable:

1.The code of a smart contract should be available to all the nodes of a blockchain system. In this way, every node can invoke it whenever needed.

2. The nodes must not be able to modify the code of a smart contract. Once it is deployed in a blockchain system, for instance by one of its users, nobody should be able to modify the code of a smart contract, let alone delete it.

3. Smart contracts must enable consistent state updates by all the nodes in a blockchain network.

1.The code of a smart contract should be available to all the nodes of a blockchain system. In this way, every node can invoke it whenever needed.

The code of a smart contract can be distributed using a transaction (e.g. Ethereum)

Or it can be part of the blockchain client (e.g. private blockchains)

Marco Comuzzi mcomuzzi@unistac.kr

2. The nodes must not be able to modify the code of a smart contract. Once it is deployed in a blockchain system, for instance by one of its users, nobody should be able to modify the code of a smart contract, let alone delete it.

Follows from the property 1

3. Smart contracts must enable consistent state updates by all the nodes in a blockchain network.

Nodes that received and installed a smart contract can execute any transaction invoking it to update the state of the blockchain

Is it that easy?

Can smart contracts contain logic that uses random numbers?

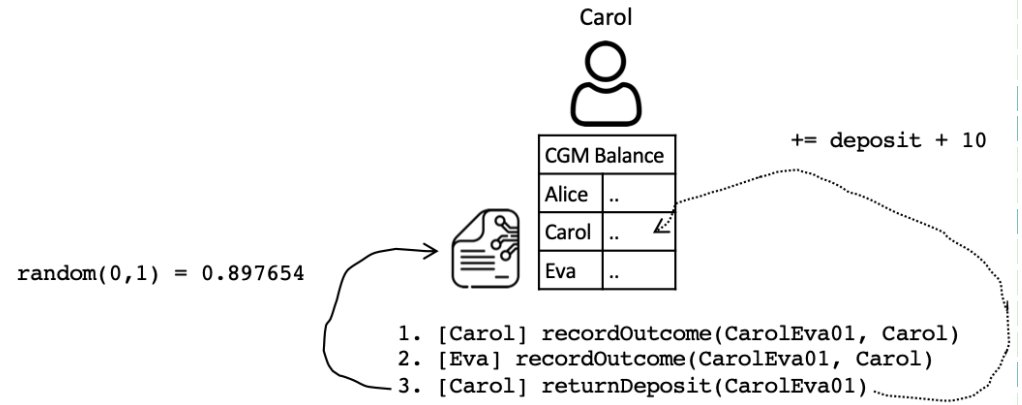
How can nodes replicate consistently the state updates if every invocation of a smart contract may lead to a different output?

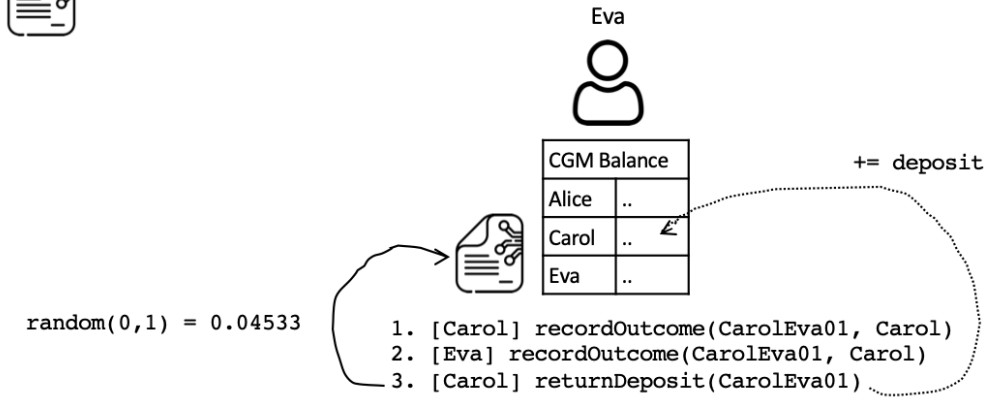
Marco Comuzzi mcomuzzi@unistac.kr

Variation of BC4C-CGM

State
…
CarolEva01(Carol: paid, Eva: paid, winner: ?)
…

```
register(gameID)
payDeposit(gameID)
recordOutcome(gameID,winner)
returnDeposit(gameID)
 …
 if random(0,1) > 0.8
  deposit = deposit + 10
 … return deposit to winner …
```





Marco Comuzzi mcomuzzi@unistac.kr

Can smart contracts use off-chain data?

How can we guarantee that nodes receive the same off-chain data every time they execute a smart contract?

Marco Comuzzi mcomuzzi@unistac.kr

Smart contracts and off-chain data: BC4C-CGM+

Let's imagine that CGM tokens have become very popular, and they can be exchanged for "fiat" currencies, like USD, EUR or KRW at online "crypto-exchanges".

So, the price of CGM tokens in fiat may fluctuate over time

We want to guarantee that players are never paid more than the equivalent of 100 USD when they win a game.

When a game terminates, the value in USD of the CGM token deposits is checked:

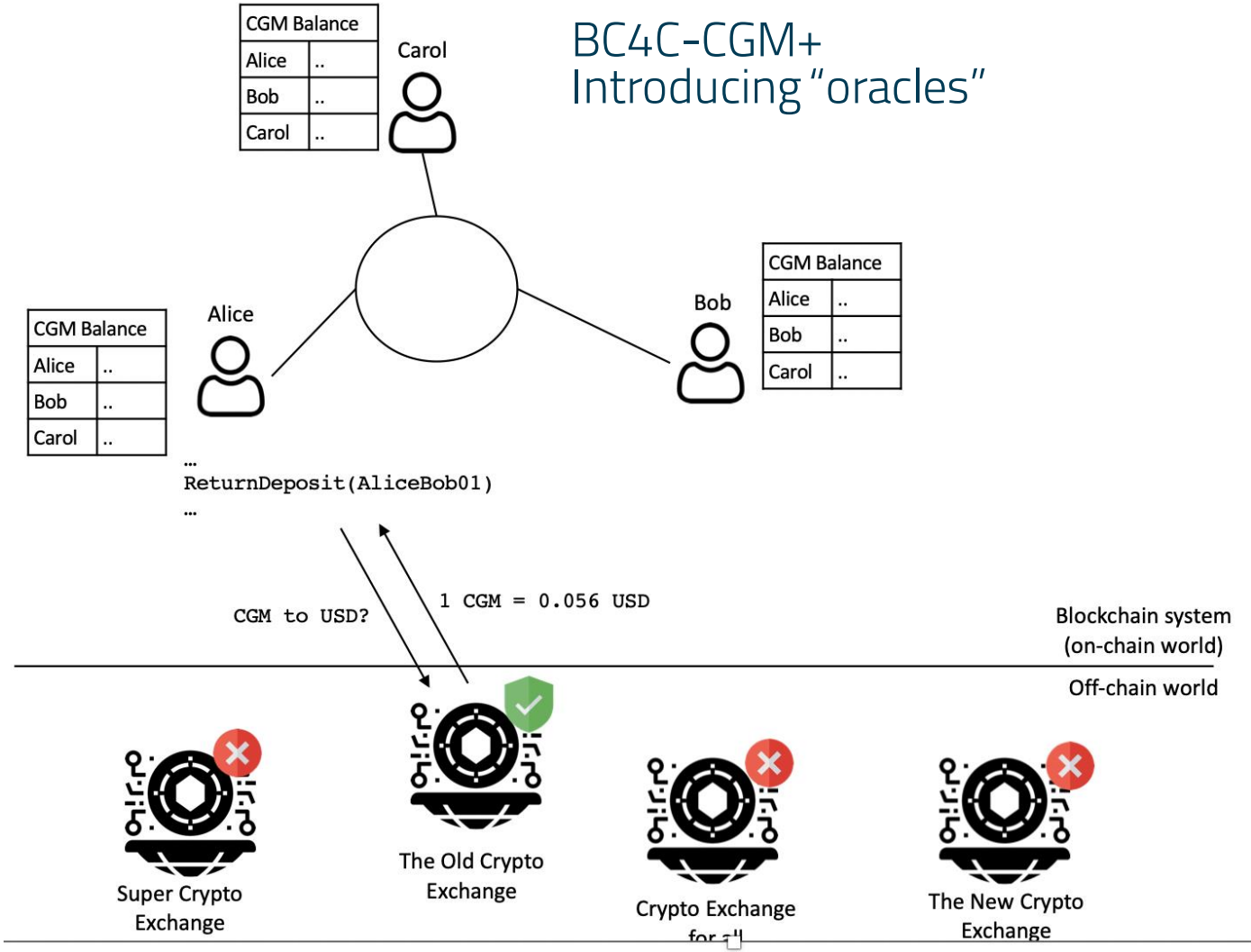
If deposits <= 100 USD, then they are returned to the winner in full

If deposits > 100 USD, then only the equivalent of 100 USD are paid to the winner, the remainder is split equally to the players.

The smart contract must know the exchange rate of CGM and USD (off-chain data)

Marco Comuzzi mcomuzzi@unistac.kr

BC4C-CGM+
Introducing "oracles"



Smart contracts, off-chain data, and oracles

1.All the players, i.e., the nodes of the blockchain system, trust that the crypto-exchange behaves faithfully, reporting to the smart contract a correct CGM token price whenever required. In practice, this cannot be enforced through the design of the technology. The best we can do is to choose a popular crypto-exchange that all the nodes can trust.

2.We create a mechanism to inject this information into the smart contract whenever necessary. Most importantly, this mechanism must ensure that every smart contract invocation to determine the amount of deposit paid back for a given terminated game receives the same CGM price from the crypto-exchange. Only in this way, in fact, it is possible to guarantee that all the nodes will update consistently the state of the BC4C-CGM+ blockchain system.

Marco Comuzzi mcomuzzi@unistac.kr

THANKS!

https://sites.google.com/site/marcocomuzziphd

http://iel.unist.ac.kr/

You can find me at:

@dr_bsad

mcomuzzi@unist.ac.kr