# Blockchain and Cryptocurrencies

## Week 3 — Chapter 1.5: Basic Tools — Two Simple Cryptocurrencies

Prof. Dr. Peter Thiemann

Albert-Ludwigs-Universität Freiburg, Germany

SS 2020

# Contents

# Rules of Goofycoin

**Rule 1**

Goofy can create new coins whenever he wants. These new coins belong to him.

# Rules of Goofycoin

**Rule 1**

Goofy can create new coins whenever he wants. These new coins belong to him.

**Rule 2**

Whoever owns a coin can transfer it to someone else.

# Rules of Goofycoin

**Rule 1**

Goofy can create new coins whenever he wants. These new coins belong to him.

**Rule 2**

Whoever owns a coin can transfer it to someone else.

**Remark**

- Basis: Hash pointers and public key signatures
- Every entity in the Goofycoin world is identified by a public key (hash).
- Nobody but Goofy can create coins.

# Creation of a Goofycoin

## Recipe for Goofy

- generate a unique coin ID, say, unique_coin_id
- create the string "CreateCoin "+ unique_coin_id
- sign the resulting string

$\Rightarrow$ Goofy owns this Goofycoin

# Transfer of a Goofycoin

## Recipe for sender Goofy

Transfer from Goofy to Alice

- create a statement that says "pay [coin] to [Alice]"
  - [coin] is a hash pointer to the coin in question
  - [Alice] refers to Alice's public key
- Goofy signs the entire statement
- This statement now stands for the coin!

$\Rightarrow$ Alice owns this coin

# Transfer of a Goofycoin

## Recipe for sender Goofy

Transfer from Goofy to Alice

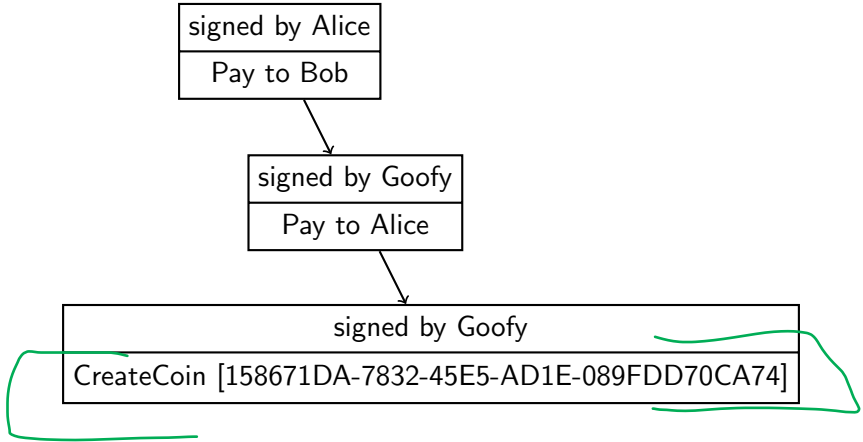
- create a statement that says "pay [coin] to [Alice]"
  - [coin] is a hash pointer to the coin in question
  - [Alice] refers to Alice's public key
- Goofy signs the entire statement
- This statement now stands for the coin!

$\Rightarrow$ Alice owns this coin

## General Recipe: Transfer from Alice to Bob

- sender Alice creates a statement "pay [coin] to [Bob]"
- sender Alice signs this statement
- Bob owns the resulting coin

# Illustration



# Validity Check

## Coin validation

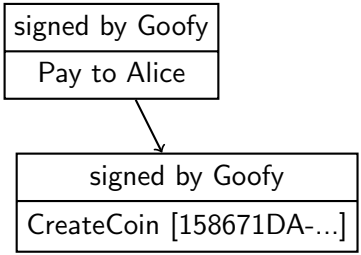
Bob receives a coin from Mr X.

It is valid if

- the coin has the form "CreateCoin" where
  - its signature verifies for Mr X,
  - and Mr X is Goofy; or
- the coin has the form "pay [coin] to [id]" where
  - [id] is Mr X,
  - the hash pointer [coin] is valid,
  - the signature of the coin verifies for the previous owner of the [coin], and
  - [coin] is validly received from its previous owner

# Problem with Goofycoin

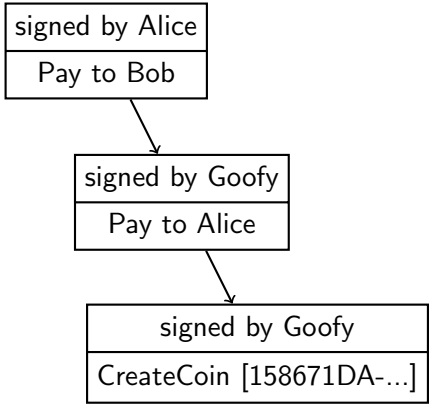
## Double Spending Attack



# Problem with Goofycoin

## Double Spending Attack

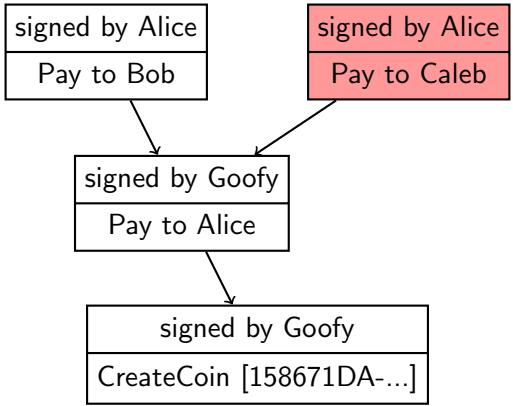
- suppose Alice signed her transfer to Bob



# Problem with Goofycoin

## Double Spending Attack

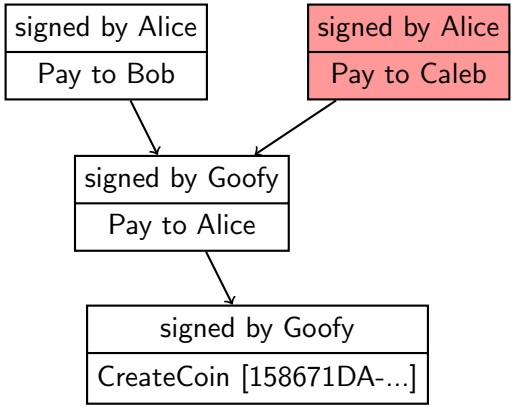
- suppose Alice signed her transfer to Bob
- afterwards she could sign another transfer for the same coin to Caleb



# Problem with Goofycoin

## Double Spending Attack

- suppose Alice signed her transfer to Bob
- afterwards she could sign another transfer for the same coin to Caleb
- the received coin validates for both, Bob and Caleb

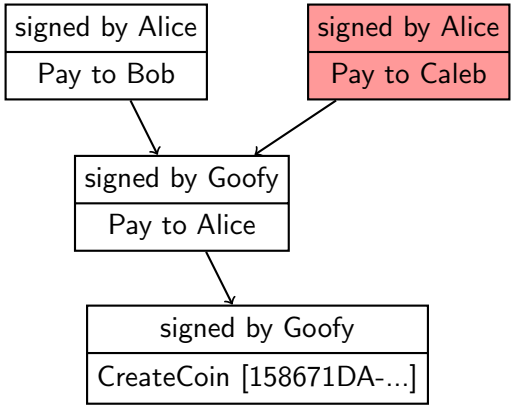


# Problem with Goofycoin

## Double Spending Attack

- suppose Alice signed her transfer to Bob
- afterwards she could sign another transfer for the same coin to Caleb
- the received coin validates for both, Bob and Caleb

$\Rightarrow$ Alice is spending the same coin twice



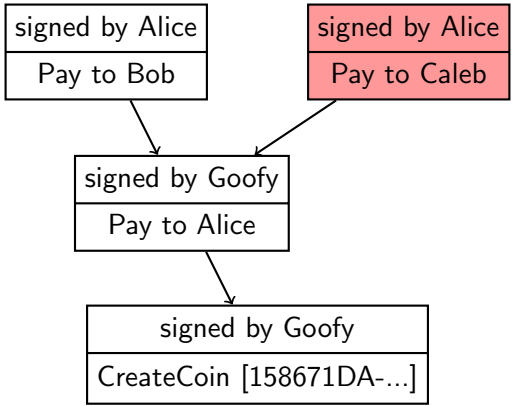
# Problem with Goofycoin

## Double Spending Attack

- suppose Alice signed her transfer to Bob
- afterwards she could sign another transfer for the same coin to Caleb
- the received coin validates for both, Bob and Caleb

$\Rightarrow$ Alice is spending the same coin twice

$\Rightarrow$ insecure! Goofycoin is unsuitable as a cryptocurrency!



# Contents

# Rules of Scroogecoin

**Rule 1**

Scrooge can create new coins whenever he wants. These coins belong to designated owners.

# Rules of Scroogecoin

**Rule 1**

Scrooge can create new coins whenever he wants. These coins belong to designated owners.

**Rule 2**

Scrooge is the guardian of an append-only transaction log (aka blockchain). He needs to sign off on any transaction.

# Rules of Scroogecoin

**Rule 1**

Scrooge can create new coins whenever he wants. These coins belong to designated owners.

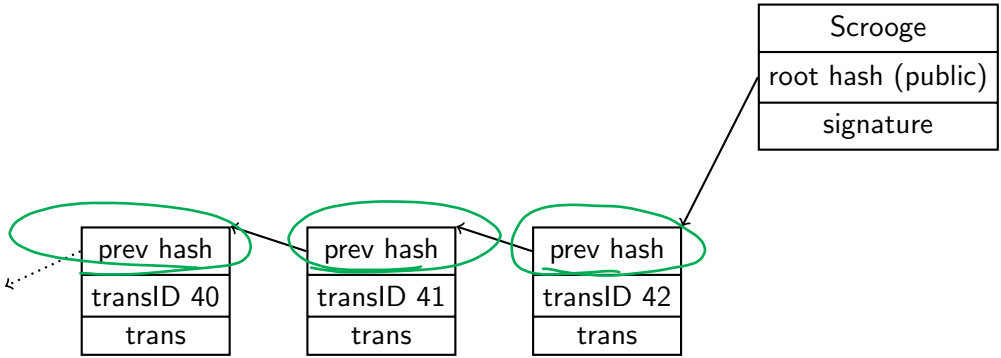
**Rule 2**

Scrooge is the guardian of an append-only transaction log (aka blockchain). He needs to sign off on any transaction.

**Remark**

- Basis: blockchain
- Every entity in Scroogecoin world identified by a public key
- Only Scrooge can create coins
- Only Scrooge can perform transactions

# The Scroogecoin Blockchain




- root hash signed by Scrooge
- each block contains a transaction, a transaction ID, and a hash pointer to the previous block
- everything (blockchain and signed root hash) public

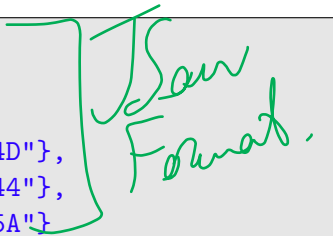
# Scroogecoin Transactions

**Two kinds of transactions**

- CreateCoins
- PayCoins

# Transaction CreateCoins

```
{"transID": 41,
 "type": "CreateCoins",
 "coins_created": [
  {"num": 0, "value": 3.2, "recipient": "C336F45C354D"},
  {"num": 1, "value": 1.4, "recipient": "96B68A43D344"},
  {"num": 2, "value": 7.1, "recipient": "0B7FD2428F5A"}
 ]}
```




- creates three coins named by coin IDs 41.0, 41.1, and 41.2
- coin ID is pair of transaction ID and coin number
- arbitrary coin values
- all recipients are different
- valid if signed by Scrooge

# Transaction PayCoins

```
{"transID": 42,
 "type": "PayCoins",
 "coins_consumed": ["41.2", "10.1", "10.0"],
 "coins_created": [
  {"num": 0, "value": 7.1, "recipient": "C336F45C354D"},
  {"num": 1, "value": 2.9, "recipient": "E9DD2D492273"},
],
 "signatures" : [
   "<signature for 41.2>",
   "<signature for 10.1>",
   "<signature for 10.0>"
]}
```

- consumes coins 41.2, 10.1, and 10.0
- creates (two) new coin with same total value as the consumed coins
- for each consumed coin, the owner needs to sign off on the transaction

# Validity of a Transaction

## CreateCoins

- Valid if signed by Scrooge

# Validity of a Transaction

## CreateCoins

- Valid if signed by Scrooge

## PayCoins

- All consumed coins are valid, i.e., they have been created by a preceding valid transaction in the blockchain
- None of the consumed coins has been consumed in a preceding transaction
- The total value of consumed coins is equal to the total value of the created coins
- The transaction is signed by all owners of consumed coins

# Validity of a Transaction

## CreateCoins

- Valid if signed by Scrooge

## PayCoins

- All consumed coins are valid, i.e., they have been created by a preceding valid transaction in the blockchain
- None of the consumed coins has been consumed in a preceding transaction
- The total value of consumed coins is equal to the total value of the created coins
- The transaction is signed by all owners of consumed coins

## Invariant of the blockchain

- Scrooge appends only valid transactions to the blockchain

# Validity of a Transaction

## CreateCoins

- Valid if signed by Scrooge

## PayCoins

- All consumed coins are valid, i.e., they have been created by a preceding valid transaction in the blockchain
- None of the consumed coins has been consumed in a preceding transaction
- The total value of consumed coins is equal to the total value of the created coins
- The transaction is signed by all owners of consumed coins

## Invariant of the blockchain

- Scrooge appends only valid transactions to the blockchain

## Guarantee of the blockchain

Everyone can check validity of a transaction and whether it is contained in the blockchain

# Properties of Scroogecoin

## Immutable coins

- coins cannot be changed, split, or combined
- they have to be recreated by a transaction

# Properties of Scroogecoin

## Immutable coins

- coins cannot be changed, split, or combined
- they have to be recreated by a transaction

## Problem: Scrooge

- single point of failure / centralized control
- cannot fake transactions because signatures cannot be faked
- can stop processing transactions for certain users
- can blackmail users to pay him for processing transactions
- can create arbitrarily many coins for anyone including himself
- can abandon the blockchain entirely

# Challenge for a cryptocurrency

## Can we decentralize?

- Can we get rid of a Scrooge-like entity?
- Can we work without a central authority while maintaining the advantages of the Scroogecoin?
- Is there a mechanism enabling all entities to agree on the current contents of the blockchain?
- Is there a decentralized mechanism to enable minting new coins in a reasonable way?

# Challenge for a cryptocurrency

## Can we decentralize?

- Can we get rid of a Scrooge-like entity?
- Can we work without a central authority while maintaining the advantages of the Scroogecoin?
- Is there a mechanism enabling all entities to agree on the current contents of the blockchain?
- Is there a decentralized mechanism to enable minting new coins in a reasonable way?

## One answer

- The ways of Bitcoin.

# Thanks!