Revenue ratio of a mining strategy on a public blockchain

by Cyril Grunspan

De Vinci Research Center, ESILV

Email: cyril.grunspan@devinci.fr

January 16, 2020

ESILV, engineering school with a research center and a scientific committee

ESILV, engineering school with a research center and a scientific committee

Bitcoin, whitepaper 2008, release 2009

ESILV, engineering school with a research center and a scientific committee

Bitcoin, whitepaper 2008, release 2009

Strange mathematical text written in word

ESILV, engineering school with a research center and a scientific committee

Bitcoin, whitepaper 2008, release 2009

Strange mathematical text written in word

First course on cryptocurrencies at ESILV (2015)

ESILV, engineering school with a research center and a scientific committee

Bitcoin, whitepaper 2008, release 2009

Strange mathematical text written in word

First course on cryptocurrencies at ESILV (2015)

Fin'tech program

ESILV, engineering school with a research center and a scientific committee

Bitcoin, whitepaper 2008, release 2009

Strange mathematical text written in word

First course on cryptocurrencies at ESILV (2015)

Fin'tech program

ESILV delivers proof of diplomas to students in the bitcoin blockchain

ESILV, engineering school with a research center and a scientific committee

Bitcoin, whitepaper 2008, release 2009

Strange mathematical text written in word

First course on cryptocurrencies at ESILV (2015)

Fin'tech program

ESILV delivers proof of diplomas to students in the bitcoin blockchain

Pierre Noizat, Jean-Paul Delahaye (articles, and blogs)

ESILV, engineering school with a research center and a scientific committee

Bitcoin, whitepaper 2008, release 2009

Strange mathematical text written in word

First course on cryptocurrencies at ESILV (2015)

Fin'tech program

ESILV delivers proof of diplomas to students in the bitcoin blockchain

Pierre Noizat, Jean-Paul Delahaye (articles, and blogs)

bitcoin.fr, bitcoinmagazine.com

ESILV, engineering school with a research center and a scientific committee

Bitcoin, whitepaper 2008, release 2009

Strange mathematical text written in word

First course on cryptocurrencies at ESILV (2015)

Fin'tech program

ESILV delivers proof of diplomas to students in the bitcoin blockchain

Pierre Noizat, Jean-Paul Delahaye (articles, and blogs)

bitcoin.fr, bitcoinmagazine.com

Mastering Bitcoin, Andreas Antonopoulous (2015)

ESILV, engineering school with a research center and a scientific committee

Bitcoin, whitepaper 2008, release 2009

Strange mathematical text written in word

First course on cryptocurrencies at ESILV (2015)

Fin'tech program

ESILV delivers proof of diplomas to students in the bitcoin blockchain

Pierre Noizat, Jean-Paul Delahaye (articles, and blogs)

bitcoin.fr, bitcoinmagazine.com

Mastering Bitcoin, Andreas Antonopoulous (2015)

10 articles on bitcoin, Ethereum, LN with Ricardo Pérez-Marco

Bitcoin, a P2P network, 10 000 nodes

Bitcoin, a P2P network, 10 000 nodes

1 bitcoin is 8 693,79 USD (January 25, 2020)

Bitcoin, a P2P network, 10 000 nodes

1 bitcoin is 8 693,79 USD (January 25, 2020)

Decentralized network

Bitcoin, a P2P network, 10 000 nodes

1 bitcoin is 8 693,79 USD (January 25, 2020)

Decentralized network

Bitcoin works without trust party

Bitcoin, a P2P network, 10 000 nodes

1 bitcoin is 8 693,79 USD (January 25, 2020)

Decentralized network

Bitcoin works without trust party

Many conferences on Bitcoin and blockchain already

Bitcoin, a P2P network, 10 000 nodes

1 bitcoin is 8 693,79 USD (January 25, 2020)

Decentralized network

Bitcoin works without trust party

Many conferences on Bitcoin and blockchain already

Satoshi Nakamoto, creator of Bitcoin

Bitcoin, a P2P network, 10 000 nodes

1 bitcoin is 8 693,79 USD (January 25, 2020)

Decentralized network

Bitcoin works without trust party

Many conferences on Bitcoin and blockchain already

Satoshi Nakamoto, creator of Bitcoin

Stuart Haber, Scott Stornetta, creators of the blockchain technology

Bitcoin, a P2P network, 10 000 nodes

1 bitcoin is 8 693,79 USD (January 25, 2020)

Decentralized network

Bitcoin works without trust party

Many conferences on Bitcoin and blockchain already

Satoshi Nakamoto, creator of Bitcoin

Stuart Haber, Scott Stornetta, creators of the blockchain technology

Innovation of Bitcoin is not in blockchain

Bitcoin, a P2P network, 10 000 nodes

1 bitcoin is 8 693,79 USD (January 25, 2020)

Decentralized network

Bitcoin works without trust party

Many conferences on Bitcoin and blockchain already

Satoshi Nakamoto, creator of Bitcoin

Stuart Haber, Scott Stornetta, creators of the blockchain technology

Innovation of Bitcoin is not in blockchain

Interesting mathematical object

“Smart contracts”

“Smart contracts”

financial transaction = input/output

“Smart contracts”

financial transaction = input/output

input = release script, output = blocking script

“Smart contracts”

financial transaction = input/output

input = release script, output = blocking script

Nick Szabo (2000)

“Smart contracts”

financial transaction = input/output

input = release script, output = blocking script

Nick Szabo (2000)

bitcoin, a programmable currency

“Smart contracts”

financial transaction = input/output

input = release script, output = blocking script

Nick Szabo (2000)

bitcoin, a programmable currency

Lightning Network (2017)

“Smart contracts”

financial transaction = input/output

input = release script, output = blocking script

Nick Szabo (2000)

bitcoin, a programmable currency

Lightning Network (2017)

Discreet Log contracts (2018)

"Smart contracts"

financial transaction = input/output

input = release script, output = blocking script

Nick Szabo (2000)

bitcoin, a programmable currency

Lightning Network (2017)

Discreet Log contracts (2018)

Possibly derivatives products with oracles and without a trust party

“Smart contracts”

financial transaction = input/output

input = release script, output = blocking script

Nick Szabo (2000)

bitcoin, a programmable currency

Lightning Network (2017)

Discreet Log contracts (2018)

Possibly derivatives products with oracles and without a trust party

Need to abandon ECDSA for Schnorr’s signatures

Mining process with Proof-of-Work

Mining process with Proof-of-Work

Cynthia Dwork and Moni Naor (1993)

Mining process with Proof-of-Work

Cynthia Dwork and Moni Naor (1993)

Rediscovered by Adam Back (1997)

Mining process with Proof-of-Work

Cynthia Dwork and Moni Naor (1993)

Rediscovered by Adam Back (1997)

Massive use of hashing functions

Mining process with Proof-of-Work

Cynthia Dwork and Moni Naor (1993)

Rediscovered by Adam Back (1997)

Massive use of hashing functions

Need to solve a cryptographic puzzle

Mining process with Proof-of-Work

Cynthia Dwork and Moni Naor (1993)

Rediscovered by Adam Back (1997)

Massive use of hashing functions

Need to solve a cryptographic puzzle

Activity of Mining

Mining process with Proof-of-Work

Cynthia Dwork and Moni Naor (1993)

Rediscovered by Adam Back (1997)

Massive use of hashing functions

Need to solve a cryptographic puzzle

Activity of Mining

Solution of the double spending problem

Mining process with Proof-of-Work

Cynthia Dwork and Moni Naor (1993)

Rediscovered by Adam Back (1997)

Massive use of hashing functions

Need to solve a cryptographic puzzle

Activity of Mining

Solution of the double spending problem

If Bob receives a payment from Alice, how can he be sure that she has not used the same money to make the same payment to Charlie or to herself?

Summary of Bitcoin

Summary of Bitcoin

Different and complementary databases:

- The mempool
- The set of UTXO
- The blockchain

Summary of Bitcoin

Different and complementary databases:

- The mempool
- The set of UTXO
- The blockchain

Bitcoin Adress = Hash of a PublicKey

Summary of Bitcoin

Different and complementary databases:

- The mempool
- The set of UTXO
- The blockchain

Bitcoin Adress = Hash of a PublicKey

Output Transaction = finite sequence (x_i, c_i)

Summary of Bitcoin

Different and complementary databases:

- The mempool
- The set of UTXO
- The blockchain

Bitcoin Adress = Hash of a PublicKey

Output Transaction = finite sequence (x_i, c_i)

$x_i =$ amount of bitcoins

Summary of Bitcoin

Different and complementary databases:

- The mempool
- The set of UTXO
- The blockchain

Bitcoin Adress = Hash of a PublicKey

Output Transaction = finite sequence (x_i, c_i)

$x_i =$ amount of bitcoins

$c_i =$ spending condition (a piece of code)

Summary of Bitcoin

Different and complementary databases:

- The mempool
- The set of UTXO
- The blockchain

Bitcoin Adress = Hash of a PublicKey

Output Transaction = finite sequence (x_i, c_i)

$x_i =$ amount of bitcoins

$c_i =$ spending condition (a piece of code)

Example: prove to be the owner of a given bitcoin adress

Summary of Bitcoin

Different and complementary databases:

- The mempool
- The set of UTXO
- The blockchain

Bitcoin Adress = Hash of a PublicKey

Output Transaction = finite sequence (x_i, c_i)

$x_i =$ amount of bitcoins

$c_i =$ spending condition (a piece of code)

Example: prove to be the owner of a given bitcoin adress

Input Transaction = finite sequence (λ_i, γ_i)

Summary of Bitcoin

Different and complementary databases:

- The mempool
- The set of UTXO
- The blockchain

Bitcoin Adress = Hash of a PublicKey

Output Transaction = finite sequence (x_i, c_i)

$x_i =$ amount of bitcoins

$c_i =$ spending condition (a piece of code)

Example: prove to be the owner of a given bitcoin adress

Input Transaction = finite sequence (λ_i, γ_i)

$\lambda_i =$ reference to a previous transaction (x_j, c_j)

Summary of Bitcoin

Different and complementary databases:

- The mempool
- The set of UTXO
- The blockchain

Bitcoin Adress = Hash of a PublicKey

Output Transaction = finite sequence (x_i, c_i)

$x_i =$ amount of bitcoins

$c_i =$ spending condition (a piece of code)

Example: prove to be the owner of a given bitcoin adress

Input Transaction = finite sequence (λ_i, γ_i)

$\lambda_i =$ reference to a previous transaction (x_j, c_j)

$\gamma_i =$ script that allows unlocking c_j (a piece of code)

Summary of Bitcoin

Different and complementary databases:

- The mempool
- The set of UTXO
- The blockchain

Bitcoin Adress = Hash of a PublicKey

Output Transaction = finite sequence (x_i, c_i)

$x_i =$ amount of bitcoins

$c_i =$ spending condition (a piece of code)

Example: prove to be the owner of a given bitcoin adress

Input Transaction = finite sequence (λ_i, γ_i)

$\lambda_i =$ reference to a previous transaction (x_j, c_j)

$\gamma_i =$ script that allows unlocking c_j (a piece of code)

UTXO = wealth of the world (in bitcoin)

Summary of Bitcoin

Different and complementary databases:

- The mempool
- The set of UTXO
- The blockchain

Bitcoin Adress = Hash of a PublicKey

Output Transaction = finite sequence (x_i, c_i)

$x_i =$ amount of bitcoins

$c_i =$ spending condition (a piece of code)

Example: prove to be the owner of a given bitcoin adress

Input Transaction = finite sequence (λ_i, γ_i)

$\lambda_i =$ reference to a previous transaction (x_j, c_j)

$\gamma_i =$ script that allows unlocking c_j (a piece of code)

UTXO = wealth of the world (in bitcoin)

blockchain = sequence of blocks limited in size (approximatly 1 MB)

Summary of Bitcoin

Different and complementary databases:

- The mempool
- The set of UTXO
- The blockchain

Bitcoin Adress = Hash of a PublicKey

Output Transaction = finite sequence (x_i, c_i)

$x_i =$ amount of bitcoins

$c_i =$ spending condition (a piece of code)

Example: prove to be the owner of a given bitcoin adress

Input Transaction = finite sequence (λ_i, γ_i)

$\lambda_i =$ reference to a previous transaction (x_j, c_j)

$\gamma_i =$ script that allows unlocking c_j (a piece of code)

UTXO = wealth of the world (in bitcoin)

blockchain = sequence of blocks limited in size (approximatly 1 MB)

Each block contains a proof of existence of a previous block

Summary of Bitcoin

Different and complementary databases:

- The mempool
- The set of UTXO
- The blockchain

Bitcoin Adress = Hash of a PublicKey

Output Transaction = finite sequence (x_i, c_i)

$x_i =$ amount of bitcoins

$c_i =$ spending condition (a piece of code)

Example: prove to be the owner of a given bitcoin adress

Input Transaction = finite sequence (λ_i, γ_i)

$\lambda_i =$ reference to a previous transaction (x_j, c_j)

$\gamma_i =$ script that allows unlocking c_j (a piece of code)

UTXO = wealth of the world (in bitcoin)

blockchain = sequence of blocks limited in size (approximatly 1 MB)

Each block contains a proof of existence of a previous block

Block = set of transactions + header

Summary of Bitcoin

Different and complementary databases:

- The mempool
- The set of UTXO
- The blockchain

Bitcoin Adress = Hash of a PublicKey

Output Transaction = finite sequence (x_i, c_i)

$x_i =$ amount of bitcoins

$c_i =$ spending condition (a piece of code)

Example: prove to be the owner of a given bitcoin adress

Input Transaction = finite sequence (λ_i, γ_i)

$\lambda_i =$ reference to a previous transaction (x_j, c_j)

$\gamma_i =$ script that allows unlocking c_j (a piece of code)

UTXO = wealth of the world (in bitcoin)

blockchain = sequence of blocks limited in size (approximatly 1 MB)

Each block contains a proof of existence of a previous block

Block = set of transactions + header

Mempool set = set of past transactions waiting to be validated in a block

Summary of Bitcoin

Different and complementary databases:

- The mempool
- The set of UTXO
- The blockchain

Bitcoin Adress = Hash of a PublicKey

Output Transaction = finite sequence (x_i, c_i)

$x_i =$ amount of bitcoins

$c_i =$ spending condition (a piece of code)

Example: prove to be the owner of a given bitcoin adress

Input Transaction = finite sequence (λ_i, γ_i)

$\lambda_i =$ reference to a previous transaction (x_j, c_j)

$\gamma_i =$ script that allows unlocking c_j (a piece of code)

UTXO = wealth of the world (in bitcoin)

blockchain = sequence of blocks limited in size (approximatly 1 MB)

Each block contains a proof of existence of a previous block

Block = set of transactions + header

Mempool set = set of past transactions waiting to be validated in a block

UTXO set updated everytime a new block is added to the blockchain

Summary of Bitcoin

Different and complementary databases:

- The mempool
- The set of UTXO
- The blockchain

Bitcoin Adress = Hash of a PublicKey

Output Transaction = finite sequence (x_i, c_i)

$x_i =$ amount of bitcoins

$c_i =$ spending condition (a piece of code)

Example: prove to be the owner of a given bitcoin adress

Input Transaction = finite sequence (λ_i, γ_i)

$\lambda_i =$ reference to a previous transaction (x_j, c_j)

$\gamma_i =$ script that allows unlocking c_j (a piece of code)

UTXO = wealth of the world (in bitcoin)

blockchain = sequence of blocks limited in size (approximatly 1 MB)

Each block contains a proof of existence of a previous block

Block = set of transactions + header

Mempool set = set of past transactions waiting to be validated in a block

UTXO set updated everytime a new block is added to the blockchain

This procedure allows transfering bitcoins

Summary of Bitcoin

Different and complementary databases:

- The mempool
- The set of UTXO
- The blockchain

Bitcoin Adress = Hash of a PublicKey

Output Transaction = finite sequence (x_i, c_i)

$x_i =$ amount of bitcoins

$c_i =$ spending condition (a piece of code)

Example: prove to be the owner of a given bitcoin adress

Input Transaction = finite sequence (λ_i, γ_i)

$\lambda_i =$ reference to a previous transaction (x_j, c_j)

$\gamma_i =$ script that allows unlocking c_j (a piece of code)

UTXO = wealth of the world (in bitcoin)

blockchain = sequence of blocks limited in size (approximatly 1 MB)

Each block contains a proof of existence of a previous block

Block = set of transactions + header

Mempool set = set of past transactions waiting to be validated in a block

UTXO set updated everytime a new block is added to the blockchain

This procedure allows transfering bitcoins

"Don't trust, verify!"

Summary of Bitcoin

Different and complementary databases:

- The mempool
- The set of UTXO
- The blockchain

Bitcoin Adress = Hash of a PublicKey

Output Transaction = finite sequence (x_i, c_i)

$x_i =$ amount of bitcoins

$c_i =$ spending condition (a piece of code)

Example: prove to be the owner of a given bitcoin adress

Input Transaction = finite sequence (λ_i, γ_i)

$\lambda_i =$ reference to a previous transaction (x_j, c_j)

$\gamma_i =$ script that allows unlocking c_j (a piece of code)

UTXO = wealth of the world (in bitcoin)

blockchain = sequence of blocks limited in size (approximatly 1 MB)

Each block contains a proof of existence of a previous block

Block = set of transactions + header

Mempool set = set of past transactions waiting to be validated in a block

UTXO set updated everytime a new block is added to the blockchain

This procedure allows transfering bitcoins

"Don't trust, verify!"

A block newly created gives bitcoins to the miner: transaction fees and coinbase (monetary creation)

Summary of Bitcoin

Different and complementary databases:

- The mempool
- The set of UTXO
- The blockchain

Bitcoin Adress = Hash of a PublicKey

Output Transaction = finite sequence (x_i, c_i)

$x_i =$ amount of bitcoins

$c_i =$ spending condition (a piece of code)

Example: prove to be the owner of a given bitcoin adress

Input Transaction = finite sequence (λ_i, γ_i)

$\lambda_i =$ reference to a previous transaction (x_j, c_j)

$\gamma_i =$ script that allows unlocking c_j (a piece of code)

UTXO = wealth of the world (in bitcoin)

blockchain = sequence of blocks limited in size (approximatly 1 MB)

Each block contains a proof of existence of a previous block

Block = set of transactions + header

Mempool set = set of past transactions waiting to be validated in a block

UTXO set updated everytime a new block is added to the blockchain

This procedure allows transfering bitcoins

“Don’t trust, verify!”

A block newly created gives bitcoins to the miner: transaction fees and coinbase (monetary creation)

Official blockchain = chain with the most proof of work (the longest in practise)

Summary of Bitcoin

Different and complementary databases:

- The mempool
- The set of UTXO
- The blockchain

Bitcoin Adress = Hash of a PublicKey

Output Transaction = finite sequence (x_i, c_i)

$x_i =$ amount of bitcoins

$c_i =$ spending condition (a piece of code)

Example: prove to be the owner of a given bitcoin adress

Input Transaction = finite sequence (λ_i, γ_i)

$\lambda_i =$ reference to a previous transaction (x_j, c_j)

$\gamma_i =$ script that allows unlocking c_j (a piece of code)

UTXO = wealth of the world (in bitcoin)

blockchain = sequence of blocks limited in size (approximatly 1 MB)

Each block contains a proof of existence of a previous block

Block = set of transactions + header

Mempool set = set of past transactions waiting to be validated in a block

UTXO set updated everytime a new block is added to the blockchain

This procedure allows transfering bitcoins

"Don't trust, verify!"

A block newly created gives bitcoins to the miner: transaction fees and coinbase (monetary creation)

Official blockchain = chain with the most proof of work (the longest in practise)

In case of competition, a node selects the first chain he has received

Summary of Bitcoin

Different and complementary databases:

- The mempool
- The set of UTXO
- The blockchain

Bitcoin Adress = Hash of a PublicKey

Output Transaction = finite sequence (x_i, c_i)

$x_i =$ amount of bitcoins

$c_i =$ spending condition (a piece of code)

Example: prove to be the owner of a given bitcoin adress

Input Transaction = finite sequence (λ_i, γ_i)

$\lambda_i =$ reference to a previous transaction (x_j, c_j)

$\gamma_i =$ script that allows unlocking c_j (a piece of code)

UTXO = wealth of the world (in bitcoin)

blockchain = sequence of blocks limited in size (approximatly 1 MB)

Each block contains a proof of existence of a previous block

Block = set of transactions + header

Mempool set = set of past transactions waiting to be validated in a block

UTXO set updated everytime a new block is added to the blockchain

This procedure allows transfering bitcoins

"Don't trust, verify!"

A block newly created gives bitcoins to the miner: transaction fees and coinbase (monetary creation)

Official blockchain = chain with the most proof of work (the longest in practise)

In case of competition, a node selects the first chain he has received

Brute force necessary to solve a cryptographic puzzle and validate a block

Summary of Bitcoin

Different and complementary databases:

- The mempool
- The set of UTXO
- The blockchain

Bitcoin Adress = Hash of a PublicKey

Output Transaction = finite sequence (x_i, c_i)

$x_i =$ amount of bitcoins

$c_i =$ spending condition (a piece of code)

Example: prove to be the owner of a given bitcoin adress

Input Transaction = finite sequence (λ_i, γ_i)

$\lambda_i =$ reference to a previous transaction (x_j, c_j)

$\gamma_i =$ script that allows unlocking c_j (a piece of code)

UTXO = wealth of the world (in bitcoin)

blockchain = sequence of blocks limited in size (approximatly 1 MB)

Each block contains a proof of existence of a previous block

Block = set of transactions + header

Mempool set = set of past transactions waiting to be validated in a block

UTXO set updated everytime a new block is added to the blockchain

This procedure allows transfering bitcoins

"Don't trust, verify!"

A block newly created gives bitcoins to the miner: transaction fees and coinbase (monetary creation)

Official blockchain = chain with the most proof of work (the longest in practise)

In case of competition, a node selects the first chain he has received

Brute force necessary to solve a cryptographic puzzle and validate a block

Probability of success to be first to find a new block = relative hashrate

Summary of Bitcoin

Different and complementary databases:

- The mempool
- The set of UTXO
- The blockchain

Bitcoin Adress = Hash of a PublicKey

Output Transaction = finite sequence (x_i, c_i)

$x_i =$ amount of bitcoins

$c_i =$ spending condition (a piece of code)

Example: prove to be the owner of a given bitcoin adress

Input Transaction = finite sequence (λ_i, γ_i)

$\lambda_i =$ reference to a previous transaction (x_j, c_j)

$\gamma_i =$ script that allows unlocking c_j (a piece of code)

UTXO = wealth of the world (in bitcoin)

blockchain = sequence of blocks limited in size (approximatly 1 MB)

Each block contains a proof of existence of a previous block

Block = set of transactions + header

Mempool set = set of past transactions waiting to be validated in a block

UTXO set updated everytime a new block is added to the blockchain

This procedure allows transfering bitcoins

"Don't trust, verify!"

A block newly created gives bitcoins to the miner: transaction fees and coinbase (monetary creation)

Official blockchain = chain with the most proof of work (the longest in practise)

In case of competition, a node selects the first chain he has received

Brute force necessary to solve a cryptographic puzzle and validate a block

Probability of success to be first to find a new block = relative hashrate

Time of validation $\boldsymbol{T}$ is memoryless: $\mathbb{P}[\boldsymbol{T} > s + t | \boldsymbol{T} > s] = \mathbb{P}[\boldsymbol{T} > t]$.

Summary of Bitcoin

Different and complementary databases:

- The mempool
- The set of UTXO
- The blockchain

Bitcoin Adress = Hash of a PublicKey

Output Transaction = finite sequence (x_i, c_i)

$x_i =$ amount of bitcoins

$c_i =$ spending condition (a piece of code)

Example: prove to be the owner of a given bitcoin adress

Input Transaction = finite sequence (λ_i, γ_i)

$\lambda_i =$ reference to a previous transaction (x_j, c_j)

$\gamma_i =$ script that allows unlocking c_j (a piece of code)

UTXO = wealth of the world (in bitcoin)

blockchain = sequence of blocks limited in size (approximatly 1 MB)

Each block contains a proof of existence of a previous block

Block = set of transactions + header

Mempool set = set of past transactions waiting to be validated in a block

UTXO set updated everytime a new block is added to the blockchain

This procedure allows transfering bitcoins

"Don't trust, verify!"

A block newly created gives bitcoins to the miner: transaction fees and coinbase (monetary creation)

Official blockchain = chain with the most proof of work (the longest in practise)

In case of competition, a node selects the first chain he has received

Brute force necessary to solve a cryptographic puzzle and validate a block

Probability of success to be first to find a new block = relative hashrate

Time of validation $\boldsymbol{T}$ is memoryless: $\mathbb{P}[\boldsymbol{T} > s + t | \boldsymbol{T} > s] = \mathbb{P}[\boldsymbol{T} > t]$.

Random variable $\boldsymbol{T}$ follows an exponential law with parameter $\frac{q}{\tau_0}$

Summary of Bitcoin

Different and complementary databases:

- The mempool
- The set of UTXO
- The blockchain

Bitcoin Adress = Hash of a PublicKey

Output Transaction = finite sequence (x_i, c_i)

$x_i =$ amount of bitcoins

$c_i =$ spending condition (a piece of code)

Example: prove to be the owner of a given bitcoin adress

Input Transaction = finite sequence (λ_i, γ_i)

$\lambda_i =$ reference to a previous transaction (x_j, c_j)

$\gamma_i =$ script that allows unlocking c_j (a piece of code)

UTXO = wealth of the world (in bitcoin)

blockchain = sequence of blocks limited in size (approximatly 1 MB)

Each block contains a proof of existence of a previous block

Block = set of transactions + header

Mempool set = set of past transactions waiting to be validated in a block

UTXO set updated everytime a new block is added to the blockchain

This procedure allows transfering bitcoins

"Don't trust, verify!"

A block newly created gives bitcoins to the miner: transaction fees and coinbase (monetary creation)

Official blockchain = chain with the most proof of work (the longest in practise)

In case of competition, a node selects the first chain he has received

Brute force necessary to solve a cryptographic puzzle and validate a block

Probability of success to be first to find a new block = relative hashrate

Time of validation $\boldsymbol{T}$ is memoryless: $\mathbb{P}[\boldsymbol{T} > s + t | \boldsymbol{T} > s] = \mathbb{P}[\boldsymbol{T} > t]$.

Random variable $\boldsymbol{T}$ follows an exponential law with parameter $\frac{q}{\tau_0}$

$\tau_0 = 600$ seconds (interblock time), $q =$ miner's relative hashrate

Summary of Bitcoin

Different and complementary databases:

- The mempool
- The set of UTXO
- The blockchain

Bitcoin Adress = Hash of a PublicKey

Output Transaction = finite sequence (x_i, c_i)

$x_i =$ amount of bitcoins

$c_i =$ spending condition (a piece of code)

Example: prove to be the owner of a given bitcoin adress

Input Transaction = finite sequence (λ_i, γ_i)

$\lambda_i =$ reference to a previous transaction (x_j, c_j)

$\gamma_i =$ script that allows unlocking c_j (a piece of code)

UTXO = wealth of the world (in bitcoin)

blockchain = sequence of blocks limited in size (approximatly 1 MB)

Each block contains a proof of existence of a previous block

Block = set of transactions + header

Mempool set = set of past transactions waiting to be validated in a block

UTXO set updated everytime a new block is added to the blockchain

This procedure allows transfering bitcoins

"Don't trust, verify!"

A block newly created gives bitcoins to the miner: transaction fees and coinbase (monetary creation)

Official blockchain = chain with the most proof of work (the longest in practise)

In case of competition, a node selects the first chain he has received

Brute force necessary to solve a cryptographic puzzle and validate a block

Probability of success to be first to find a new block = relative hashrate

Time of validation $\boldsymbol{T}$ is memoryless: $\mathbb{P}[\boldsymbol{T} > s + t | \boldsymbol{T} > s] = \mathbb{P}[\boldsymbol{T} > t]$.

Random variable $\boldsymbol{T}$ follows an exponential law with parameter $\frac{q}{\tau_0}$

$\tau_0 = 600$ seconds (interblock time), $q =$ miner's relative hashrate

Number of blocks validated by a miner = Poisson process

1. Alice sends a transaction to herself registred in a secret block (value v)

1. Alice sends a transaction to herself registred in a secret block (value v)
2. She keeps mining on it

1. Alice sends a transaction to herself registred in a secret block (value v)
2. She keeps mining on it
3. In parallel, she publicly sends this transaction to Bob

1. Alice sends a transaction to herself registred in a secret block (value v)
2. She keeps mining on it
3. In parallel, she publicly sends this transaction to Bob
4. Bob waits for z confirmations in the blockchain (Satoshi recommends $z = 6, 7$)

1. Alice sends a transaction to herself registred in a secret block (value v)
2. She keeps mining on it
3. In parallel, she publicly sends this transaction to Bob
4. Bob waits for z confirmations in the blockchain (Satoshi recommends $z = 6, 7$)
5. If Alice has not been able to produce a chain of blocks greater than z, she keeps mining until she catches up and overtake the official blockchain by one block

1. Alice sends a transaction to herself registred in a secret block (value v)
2. She keeps mining on it
3. In parallel, she publicly sends this transaction to Bob
4. Bob waits for z confirmations in the blockchain (Satoshi recommends $z = 6, 7$)
5. If Alice has not been able to produce a chain of blocks greater than z, she keeps mining until she catches up and overtake the official blockchain by one block
6. If successful, she makes all her blocks public and forces the network to accept her blocks. Bob now owns an illegal transaction...

1. Alice sends a transaction to herself registred in a secret block (value v)
2. She keeps mining on it
3. In parallel, she publicly sends this transaction to Bob
4. Bob waits for z confirmations in the blockchain (Satoshi recommends $z = 6, 7$)
5. If Alice has not been able to produce a chain of blocks greater than z, she keeps mining until she catches up and overtake the official blockchain by one block
6. If successful, she makes all her blocks public and forces the network to accept her blocks. Bob now owns an illegal transaction...
7. If successful, Alice has won all the published blocks plus v

1. Alice sends a transaction to herself registred in a secret block (value v)
2. She keeps mining on it
3. In parallel, she publicly sends this transaction to Bob
4. Bob waits for z confirmations in the blockchain (Satoshi recommends $z = 6, 7$)
5. If Alice has not been able to produce a chain of blocks greater than z, she keeps mining until she catches up and overtake the official blockchain by one block
6. If successful, she makes all her blocks public and forces the network to accept her blocks. Bob now owns an illegal transaction...
7. If successful, Alice has won all the published blocks plus v
8. Otherwise, her revenue is 0.

Theorem 1. *(Grunspan-Pérez-Marco 2017) The probability of success of the attack is* $P(z) = I_{4pq}(z, \frac{1}{2})$ *where* q *is Alice's relative hasrate and* $p = 1 - q$.

Theorem 3. *(Grunspan-Pérez-Marco 2017) The probability of success of the attack is* $P(z) = I_{4pq}\left(z, \frac{1}{2}\right)$ *where* q *is Alice's relative hasrate and* $p = 1 - q$.

Regularized Incomplete Beta function $I_x(a,b) = \frac{\Gamma(a+b)}{\Gamma(a)\,\Gamma(b)} \int_0^x t^{a-1}\,(1-t)^{b-1}\,\mathrm{dt}$.

Theorem 5. *(Grunspan-Pérez-Marco 2017) The probability of success of the attack is* $P(z) = I_{4pq}\left(z, \frac{1}{2}\right)$ *where* q *is Alice's relative hasrate and* $p = 1 - q$*.*

Regularized Incomplete Beta function $I_x(a,b) = \frac{\Gamma(a+b)}{\Gamma(a)\,\Gamma(b)} \int_0^x t^{a-1}\,(1-t)^{b-1}\,\mathrm{d}t$.

Proof. Let $\boldsymbol{S}_z =$ date when the honest miners have mined z blocks (Gamma law), $\boldsymbol{X} =$ number of blocks mined by Alice between 0 and $\boldsymbol{S}_z$ (negative binomial law)

$$\begin{aligned}
\forall j, \quad \mathbb{P}[\boldsymbol{X} = j] &= p^z q^j \binom{z+j-1}{j} \\
P(z) &= \sum_{j=0}^{z-1} \mathbb{P}[\boldsymbol{X} = j] q_{z-j} + \sum_{j=z}^{\infty} \mathbb{P}[\boldsymbol{X} = j] \\
\forall i, j, n, \quad \mathbb{P}[\boldsymbol{Z}_{n+1} = i \,|\, \boldsymbol{Z}_n = j] &= p\,\mathbf{1}_{i=j+1} + q\,\mathbf{1}_{i=j-1} \\
q_k &= \left(\frac{q}{p}\right)^k
\end{aligned}$$

$$
\begin{aligned}
\sum_{j=0}^{z-1} \mathbb{P}[\boldsymbol{X} = j] &= I_p(z\,,z) \\
\sum_{j=0}^{z-1} \mathbb{P}[\boldsymbol{X} = j]\left(\frac{q}{p}\right)^{z-j} &= I_q(z\,,z) \\
I_p(z\,,z) - I_q(z\,,z) &= 1 \\
I_q(z\,,z) &= \frac{1}{2}\, I_{4pq}\!\left(z\,,\frac{1}{2}\right)
\end{aligned}
$$

$\square$

Theorem 7. *(Grunspan-Pérez-Marco 2017) The probability of success of the attack is* $P(z) = I_{4pq}\left(z, \frac{1}{2}\right)$ *where* q *is Alice's relative hasrate and* $p = 1 - q$.

Regularized Incomplete Beta function $I_x(a,b) = \frac{\Gamma(a+b)}{\Gamma(a)\,\Gamma(b)} \int_0^x t^{a-1}\,(1-t)^{b-1}\,\mathrm{dt}$.

Proof. Let $\boldsymbol{S}_z =$ date when the honest miners have mined z blocks (Gamma law), $\boldsymbol{X} =$ number of blocks mined by Alice between 0 and $\boldsymbol{S}_z$ (negative binomial law)

$$\begin{aligned}
\forall j, \quad \mathbb{P}[\boldsymbol{X} = j] &= p^z q^j \begin{pmatrix} z+j-1 \\ j \end{pmatrix} \\
P(z) &= \sum_{j=0}^{z-1} \mathbb{P}[\boldsymbol{X} = j] q_{z-j} + \sum_{j=z}^{\infty} \mathbb{P}[\boldsymbol{X} = j] \\
\forall i, j, n, \quad \mathbb{P}[\boldsymbol{Z}_{n+1} = i | \boldsymbol{Z}_n = j] &= p\,\mathbf{1}_{i=j+1} + q\,\mathbf{1}_{i=j-1} \\
q_k &= \left(\frac{q}{p}\right)^k
\end{aligned}$$

$$
\begin{aligned}
\sum_{j=0}^{z-1} \mathbb{P}[\boldsymbol{X}=j] &= I_p(z\,,z) \\
\sum_{j=0}^{z-1} \mathbb{P}[\boldsymbol{X}=j]\left(\frac{q}{p}\right)^{z-j} &= I_q(z\,,z) \\
I_p(z\,,z) - I_q(z\,,z) &= 1 \\
I_q(z\,,z) &= \frac{1}{2}\, I_{4pq}\!\left(z\,,\frac{1}{2}\right)
\end{aligned}
$$

$\square$

Corollary 8. *(GPM, 2017) When* $z \to \infty$, $P(z) \sim \frac{s^z}{\sqrt{\pi\,(1-s)\,z}}$ *with* $s = 4\,p\,q$.

Theorem 9. *(Grunspan-Pérez-Marco 2017) The probability of success of the attack is $P(z)=I_{4pq}\left(z,\frac{1}{2}\right)$ where q is Alice's relative hasrate and $p=1-q$.*

Regularized Incomplete Beta function $I_x(a,b)=\frac{\Gamma(a+b)}{\Gamma(a)\,\Gamma(b)}\int_0^x t^{a-1}\,(1-t)^{b-1}\,\mathrm{d}t$.

Proof. Let $\boldsymbol{S}_z=$ date when the honest miners have mined z blocks (Gamma law), $\boldsymbol{X}=$ number of blocks mined by Alice between 0 and $\boldsymbol{S}_z$ (negative binomial law)

$$\begin{aligned}
\forall j,\quad \mathbb{P}[\boldsymbol{X}=j] &= p^z q^j \begin{pmatrix} z+j-1 \\ j \end{pmatrix} \\
P(z) &= \sum_{j=0}^{z-1} \mathbb{P}[\boldsymbol{X}=j]q_{z-j} + \sum_{j=z}^{\infty} \mathbb{P}[\boldsymbol{X}=j] \\
\forall i,j,n,\quad \mathbb{P}[\boldsymbol{Z}_{n+1}=i|\boldsymbol{Z}_n=j] &= p\,\mathbf{1}_{i=j+1}+q\,\mathbf{1}_{i=j-1} \\
q_k &= \left(\frac{q}{p}\right)^k
\end{aligned}$$

$$
\begin{aligned}
\sum_{j=0}^{z-1} \mathbb{P}[\boldsymbol{X}=j] &= I_p(z\,,z) \\
\sum_{j=0}^{z-1} \mathbb{P}[\boldsymbol{X}=j]\left(\frac{q}{p}\right)^{z-j} &= I_q(z\,,z) \\
I_p(z\,,z) - I_q(z\,,z) &= 1 \\
I_q(z\,,z) &= \frac{1}{2}\, I_{4pq}\!\left(z\,,\frac{1}{2}\right)
\end{aligned}
$$

□

Corollary 10. *(GPM, 2017) When* $z \to \infty$, $P(z) \sim \frac{s^z}{\sqrt{\pi\,(1-s)\,z}}$ *with* $s = 4\,p\,q$.

Note. Pre-mining 1 block is implicit in Satoshi's white paper.

Insufficient result: Alice could simply pre-mine $z + 1$ blocks!

Insufficient result: Alice could simply pre-mine $z+1$ blocks!

Random walk $\tilde{\boldsymbol{Z}}$ on $\mathbb{Z}$ (partially absorbing at 0) reaches all states of $\mathbb{N}$:

$$\begin{aligned}
\forall n, \quad \mathbb{P}\big[\tilde{\boldsymbol{Z}}_{n+1}=j\,|\,\tilde{\boldsymbol{Z}}_n=0\big] &= p\,\mathbf{1}_{j=0}+q\,\mathbf{1}_{j=1} \\
\forall n, j, i\neq 0, \quad \mathbb{P}\big[\tilde{\boldsymbol{Z}}_{n+1}=j\,|\,\tilde{\boldsymbol{Z}}_n=i\big] &= p\,\mathbf{1}_{j=i-1}+q\,\mathbf{1}_{j=i+1}
\end{aligned}$$

Insufficient result: Alice could simply pre-mine $z+1$ blocks!

Random walk $\tilde{\boldsymbol{Z}}$ on $\mathbb{Z}$ (partially absorbing at 0) reaches all states of $\mathbb{N}$:

$$\begin{aligned} \forall n, \quad \mathbb{P}\big[\tilde{\boldsymbol{Z}}_{n+1}=j\,|\,\tilde{\boldsymbol{Z}}_n=0\big] \;&=\; p\,\mathbf{1}_{j=0}+q\,\mathbf{1}_{j=1} \\ \forall n,j,i\neq 0, \quad \mathbb{P}\big[\tilde{\boldsymbol{Z}}_{n+1}=j\,|\,\tilde{\boldsymbol{Z}}_n=i\big] \;&=\; p\,\mathbf{1}_{j=i-1}+q\,\mathbf{1}_{j=i+1} \end{aligned}$$

Execution time very high...

Insufficient result: Alice could simply pre-mine $z+1$ blocks!

Random walk $\tilde{\boldsymbol{Z}}$ on $\mathbb{Z}$ (partially absorbing at 0) reaches all states of $\mathbb{N}$:

$$\begin{aligned}\forall n, \quad \mathbb{P}\big[\tilde{\boldsymbol{Z}}_{n+1}=j\,|\,\tilde{\boldsymbol{Z}}_n=0\big] &= p\,\mathbf{1}_{j=0}+q\,\mathbf{1}_{j=1}\\ \forall n,j,i\neq 0, \quad \mathbb{P}\big[\tilde{\boldsymbol{Z}}_{n+1}=j\,|\,\tilde{\boldsymbol{Z}}_n=i\big] &= p\,\mathbf{1}_{j=i-1}+q\,\mathbf{1}_{j=i+1}\end{aligned}$$

Execution time very high...

Importance of the duration time of the attack

Insufficient result: Alice could simply pre-mine $z+1$ blocks!

Random walk $\tilde{\boldsymbol{Z}}$ on $\mathbb{Z}$ (partially absorbing at 0) reaches all states of $\mathbb{N}$:

$$\begin{aligned} \forall n, \quad \mathbb{P}\big[\tilde{\boldsymbol{Z}}_{n+1}=j\,|\,\tilde{\boldsymbol{Z}}_n=0\big] &= p\,\mathbf{1}_{j=0}+q\,\mathbf{1}_{j=1} \\ \forall n,j,i\neq 0, \quad \mathbb{P}\big[\tilde{\boldsymbol{Z}}_{n+1}=j\,|\,\tilde{\boldsymbol{Z}}_n=i\big] &= p\,\mathbf{1}_{j=i-1}+q\,\mathbf{1}_{j=i+1} \end{aligned}$$

Execution time very high...

Importance of the duration time of the attack

Double spending: a risk that cannot be cancelled...

Insufficient result: Alice could simply pre-mine $z+1$ blocks!

Random walk $\tilde{\boldsymbol{Z}}$ on $\mathbb{Z}$ (partially absorbing at 0) reaches all states of $\mathbb{N}$:

$$\begin{aligned} \forall n, \quad \mathbb{P}\big[\tilde{\boldsymbol{Z}}_{n+1}=j\,|\,\tilde{\boldsymbol{Z}}_n=0\big] &= p\,\mathbf{1}_{j=0}+q\,\mathbf{1}_{j=1} \\ \forall n, j, i\neq 0, \quad \mathbb{P}\big[\tilde{\boldsymbol{Z}}_{n+1}=j\,|\,\tilde{\boldsymbol{Z}}_n=i\big] &= p\,\mathbf{1}_{j=i-1}+q\,\mathbf{1}_{j=i+1} \end{aligned}$$

Execution time very high...

Importance of the duration time of the attack

Double spending: a risk that cannot be cancelled...

Incentive to launch a double spend attack?

Insufficient result: Alice could simply pre-mine $z+1$ blocks!

Random walk $\tilde{\boldsymbol{Z}}$ on $\mathbb{Z}$ (partially absorbing at 0) reaches all states of $\mathbb{N}$:

$$\begin{aligned} \forall n, \quad \mathbb{P}\big[\tilde{\boldsymbol{Z}}_{n+1}=j\,|\,\tilde{\boldsymbol{Z}}_n=0\big] &= p\,\mathbf{1}_{j=0}+q\,\mathbf{1}_{j=1} \\ \forall n,j,i\neq 0, \quad \mathbb{P}\big[\tilde{\boldsymbol{Z}}_{n+1}=j\,|\,\tilde{\boldsymbol{Z}}_n=i\big] &= p\,\mathbf{1}_{j=i-1}+q\,\mathbf{1}_{j=i+1} \end{aligned}$$

Execution time very high...

Importance of the duration time of the attack

Double spending: a risk that cannot be cancelled...

Incentive to launch a double spend attack?

Importance of the value of the double spend

Threshold $\bar{v}$ (depending on q) = benchmark for network's security

Threshold $\bar{v}$ (depending on q) = benchmark for network's security

Indicator for the robustness of the network

Threshold $\bar{v}$ (depending on q) = benchmark for network's security

Indicator for the robustness of the network

Compare profitability of a rogue strategy with the honest one

Threshold $\bar{v}$ (depending on q) = benchmark for network's security

Indicator for the robustness of the network

Compare profitability of a rogue strategy with the honest one

Nakamoto's strategy is not viable: probability of total ruin >0

Threshold $\bar{v}$ (depending on q) = benchmark for network's security

Indicator for the robustness of the network

Compare profitability of a rogue strategy with the honest one

Nakamoto's strategy is not viable: probability of total ruin >0

Give-up threshold A

Threshold $\bar{v}$ (depending on q) = benchmark for network's security

Indicator for the robustness of the network

Compare profitability of a rogue strategy with the honest one

Nakamoto's strategy is not viable: probability of total ruin >0

Give-up threshold A

Alice stops her attack if her delay with the official blockchain is greater than A.

Threshold $\bar{v}$ (depending on q) = benchmark for network's security

Indicator for the robustness of the network

Compare profitability of a rogue strategy with the honest one

Nakamoto's strategy is not viable: probability of total ruin >0

Give-up threshold A

Alice stops her attack if her delay with the official blockchain is greater than A.

Definition 17. *A strategy is a stopping time which determines the end of the attack*

Threshold $\bar{v}$ (depending on q) = benchmark for network's security

Indicator for the robustness of the network

Compare profitability of a rogue strategy with the honest one

Nakamoto's strategy is not viable: probability of total ruin >0

Give-up threshold A

Alice stops her attack if her delay with the official blockchain is greater than A.

Definition 18. *A strategy is a stopping time which determines the end of the attack*

PnL per unit of time = PnL_t

Threshold $\bar{v}$ (depending on q) = benchmark for network's security

Indicator for the robustness of the network

Compare profitability of a rogue strategy with the honest one

Nakamoto's strategy is not viable: probability of total ruin >0

Give-up threshold A

Alice stops her attack if her delay with the official blockchain is greater than A.

Definition 19. *A strategy is a stopping time which determines the end of the attack*

PnL per unit of time = PnL_t

Alice repeats her strategy n times (n attack cycles)

Alice's PnL_t is $\mathrm{PnL}_t = \dfrac{\boldsymbol{R}_1 - \boldsymbol{C}_1 + \ldots + \boldsymbol{R}_n - \boldsymbol{C}_n}{\boldsymbol{T}_1 + \ldots + \boldsymbol{T}_n}$

Alice's PnL_t is $\mathrm{PnL}_t = \frac{\boldsymbol{R}_1 - \boldsymbol{C}_1 + \ldots + \boldsymbol{R}_n - \boldsymbol{C}_n}{\boldsymbol{T}_1 + \ldots + \boldsymbol{T}_n}$

$\boldsymbol{R}_i$ (resp. $\boldsymbol{C}_i$) is Alice's revenue (resp. cost of mining) after a i-th attack cycle

Alice's PnL_t is $\mathrm{PnL}_t = \frac{\boldsymbol{R}_1 - \boldsymbol{C}_1 + \ldots + \boldsymbol{R}_n - \boldsymbol{C}_n}{\boldsymbol{T}_1 + \ldots + \boldsymbol{T}_n}$

$\boldsymbol{R}_i$ (resp. $\boldsymbol{C}_i$) is Alice's revenue (resp. cost of mining) after a i-th attack cycle

Random variables $\boldsymbol{R}_i$ (resp. $\boldsymbol{C}_i$) are iid. We assume $\mathbb{E}[\boldsymbol{T}] < \infty$

Alice's PnL_t is $\mathrm{PnL}_t = \frac{\boldsymbol{R}_1 - \boldsymbol{C}_1 + \ldots + \boldsymbol{R}_n - \boldsymbol{C}_n}{\boldsymbol{T}_1 + \ldots + \boldsymbol{T}_n}$

$\boldsymbol{R}_i$ (resp. $\boldsymbol{C}_i$) is Alice's revenue (resp. cost of mining) after a i-th attack cycle

Random variables $\boldsymbol{R}_i$ (resp. $\boldsymbol{C}_i$) are iid. We assume $\mathbb{E}[\boldsymbol{T}] < \infty$

Strong law of numbers: $\mathrm{PnL}_\infty = \boldsymbol{\Gamma} - \boldsymbol{\Upsilon}$ with

$$\begin{aligned}\boldsymbol{\Gamma} &= \frac{\mathbb{E}[\boldsymbol{R}]}{\mathbb{E}[\boldsymbol{T}]}\\ \boldsymbol{\Upsilon} &= \frac{\mathbb{E}[\boldsymbol{C}]}{\mathbb{E}[\boldsymbol{T}]}\end{aligned}$$

Alice's PnL_t is $\mathrm{PnL}_t = \frac{\boldsymbol{R}_1 - \boldsymbol{C}_1 + \ldots + \boldsymbol{R}_n - \boldsymbol{C}_n}{\boldsymbol{T}_1 + \ldots + \boldsymbol{T}_n}$

$\boldsymbol{R}_i$ (resp. $\boldsymbol{C}_i$) is Alice's revenue (resp. cost of mining) after a i-th attack cycle

Random variables $\boldsymbol{R}_i$ (resp. $\boldsymbol{C}_i$) are iid. We assume $\mathbb{E}[\boldsymbol{T}] < \infty$

Strong law of numbers: $\mathrm{PnL}_\infty = \Gamma - \Upsilon$ with

$$\begin{aligned} \Gamma &= \frac{\mathbb{E}[\boldsymbol{R}]}{\mathbb{E}[\boldsymbol{T}]} \\ \Upsilon &= \frac{\mathbb{E}[\boldsymbol{C}]}{\mathbb{E}[\boldsymbol{T}]} \end{aligned}$$

Γ = revenue ratio (computable), Υ = cost ratio (difficult to evaluate)

Alice's PnL_t is $\mathrm{PnL}_t = \frac{\boldsymbol{R}_1 - \boldsymbol{C}_1 + \ldots + \boldsymbol{R}_n - \boldsymbol{C}_n}{\boldsymbol{T}_1 + \ldots + \boldsymbol{T}_n}$

$\boldsymbol{R}_i$ (resp. $\boldsymbol{C}_i$) is Alice's revenue (resp. cost of mining) after a i-th attack cycle

Random variables $\boldsymbol{R}_i$ (resp. $\boldsymbol{C}_i$) are iid. We assume $\mathbb{E}[\boldsymbol{T}] < \infty$

Strong law of numbers: $\mathrm{PnL}_\infty = \Gamma - \Upsilon$ with

$$\begin{aligned} \Gamma &= \frac{\mathbb{E}[\boldsymbol{R}]}{\mathbb{E}[\boldsymbol{T}]} \\ \Upsilon &= \frac{\mathbb{E}[\boldsymbol{C}]}{\mathbb{E}[\boldsymbol{T}]} \end{aligned}$$

Γ = revenue ratio (computable), Υ = cost ratio (difficult to evaluate)

Important remark: Υ is independent of the chosen strategy.

1. Start of the attack cycle (goto 2).

Description of the (A, k)-double spend strategy

1. Start of the attack cycle (goto 2).
2. The attacker mines honestly on top of the official blockchain k blocks with a transaction that returns the payment funds to an address he controls (goto 3).

Description of the (A, k)-double spend strategy

1. Start of the attack cycle (goto 2).
2. The attacker mines honestly on top of the official blockchain k blocks with a transaction that returns the payment funds to an address he controls (goto 3).
3. When he succeeds in pre-mining k blocks lading the honest miners, he keeps his fork secret, sends the purchasing transaction to the vendor, and keeps up mining on his secret fork (goto 4).

1. Start of the attack cycle (goto 2).
2. The attacker mines honestly on top of the official blockchain k blocks with a transaction that returns the payment funds to an address he controls (goto 3).
3. When he succeeds in pre-mining k blocks lading the honest miners, he keeps his fork secret, sends the purchasing transaction to the vendor, and keeps up mining on his secret fork (goto 4).
4. If the delay with the official blockchain gets larger than A then the attacker gives-up and the double spend attack fails (goto 6).

1. Start of the attack cycle (goto 2).
2. The attacker mines honestly on top of the official blockchain k blocks with a transaction that returns the payment funds to an address he controls (goto 3).
3. When he succeeds in pre-mining k blocks lading the honest miners, he keeps his fork secret, sends the purchasing transaction to the vendor, and keeps up mining on his secret fork (goto 4).
4. If the delay with the official blockchain gets larger than A then the attacker gives-up and the double spend attack fails (goto 6).
5. If the secret fork of the attacker gets longer than the official blockchain that has added z confirmations to the vendor transaction, then the attacker releases his fork and the double spend is successful (goto 6).

Description of the (A, k)-double spend strategy

1. Start of the attack cycle (goto 2).
2. The attacker mines honestly on top of the official blockchain k blocks with a transaction that returns the payment funds to an address he controls (goto 3).
3. When he succeeds in pre-mining k blocks lading the honest miners, he keeps his fork secret, sends the purchasing transaction to the vendor, and keeps up mining on his secret fork (goto 4).
4. If the delay with the official blockchain gets larger than A then the attacker gives-up and the double spend attack fails (goto 6).
5. If the secret fork of the attacker gets longer than the official blockchain that has added z confirmations to the vendor transaction, then the attacker releases his fork and the double spend is successful (goto 6).
6. End of the attack cycle (goto 1)

Asumption: $A > z$ and $k = 1$ (pre-mining of 1 block)

Asumption: $A > z$ and $k = 1$ (pre-mining of 1 block)

Nakamoto's strategy is $(\infty, 1)$-strategy.

Asumption: $A > z$ and $k = 1$ (pre-mining of 1 block)

Nakamoto's strategy is $(\infty, 1)$-strategy.

Proposition 24. *The probability of success is* $P_A(z) = \frac{I_{4pq}\left(z, \frac{1}{2}\right) - \lambda^{A+1}}{1 - \lambda^{A+1}}$ *with* $\lambda = \frac{q}{p}$.

Asumption: $A > z$ and $k = 1$ (pre-mining of 1 block)

Nakamoto's strategy is $(\infty, 1)$-strategy.

Proposition 26. *The probability of success is* $P_A(z) = \frac{I_{4pq}\left(z, \frac{1}{2}\right) - \lambda^{A+1}}{1-\lambda^{A+1}}$ *with* $\lambda = \frac{q}{p}$.

Lemma 27. *Let* $\tilde{N}$ *and* $\tilde{N}'$ *be two Poisson processes with parameters* α *and* α' *and* $X, Y \in \mathbb{R}_+$. *Let* $\tilde{T}_{X,Y} = Inf\,\{t \in \mathbb{R}_+; (\tilde{N}(t) = \tilde{N}'(t) + X) \vee (\tilde{N}'(t) = \tilde{N}(t) + Y)\}$.

Then,

$$\mathbb{E}[\tilde{T}_{X,Y}] = \frac{X+Y}{\alpha - \alpha'}\left(\frac{1-\lambda^Y}{1-\lambda^{X+Y}} - \frac{Y}{X+Y}\right)$$

Proposition 28. *We have:*

$$\mathbb{E}[\boldsymbol{T}_A] = \frac{z}{2\,p}\, I_{4pq}(z, 1/2) + \frac{A+1}{p\,(1-\lambda)^2\,[A+1]}\, I_{(p-q)^2}(1/2, z) - \frac{p^{z-1}\,q^z}{p\,(1-\lambda)\,B(z,z)} + \frac{1}{q}$$

Proposition 29. *We have:*

$$\mathbb{E}[\boldsymbol{T}_A] = \frac{z}{2\,p}\, I_{4pq}\,(z, 1/2) + \frac{A+1}{p\,(1-\lambda)^2\,[A+1]}\, I_{(p-q)^2}\,(1/2, z) - \frac{p^{z-1}\,q^z}{p\,(1-\lambda)\,B(z,z)} + \frac{1}{q}$$

Proof. We have:

$$\boldsymbol{T}_A \;=\; \mathbf{S}_z + 1_{\mathbf{N}'(\mathbf{S}_z)<z} \cdot \tilde{T}_{A+1-(z-\mathbf{N}'(\mathbf{S}_z)),z-\mathbf{N}'(\mathbf{S}_z)}$$

and $\tilde{N}(t) = \tilde{N}(\mathbf{S}_z + t) - z$ (resp. $\tilde{N}'(t) = \tilde{N}'(\mathbf{S}_z + t) - \tilde{N}'(\mathbf{S}_z)$) is a Poisson process (Markov property). □

Theorem 30. *Let* $(\boldsymbol{X}_n)_{n\in\mathbb{N}}$ *be a simple biased random walk on* $\{0,\ldots,M\}$*:*

$$P(i,j)=\mathbb{P}[\boldsymbol{X}_{n+1}=j\,|\,\boldsymbol{X}_n=i]=p\,\mathbf{1}_{i=j-1}+q\,\mathbf{1}_{i=j+1}$$

for $(i,j)\in[1,M-1]\times[0,M]$ *with absorbing bounds:* $P(0,0)=P(M,M)=1$. *For* $(m,k)\in\mathbb{N}^2$, *let* $\nu_{m,k}$ *be the stopping time* $\nu_{m,k}=\operatorname{Inf}\{n;\boldsymbol{X}_n=k\,|\,\boldsymbol{X}_0=m\}$ *and* $\nu_m=\nu_{m,0}\wedge\nu_{m,M}$. *Then, ((?) = Stern's formula)*

$$\mathbb{E}[\nu_m] = \frac{M}{p-q}\left(\frac{1-\lambda^m}{1-\lambda^M}-\frac{m}{M}\right) \tag{1}$$

$$\mathbb{P}[\nu_m=\nu_{m,0}] = \frac{\lambda^m-\lambda^M}{1-\lambda^M} \tag{2}$$

$$\mathbb{E}[\nu_m|\nu_m=\nu_{m,0}] = \frac{m\,\lambda^m-(2M-m)\,\lambda^M+(2M-m)\,\lambda^{M+m}-m\,\lambda^{2M}}{p\,(1-\lambda)\,(\lambda^m-\lambda^M)\,(1-\lambda^M)} \tag{3}$$

Corollary 31. *We have:* $\lim_{M \to \infty} \mathbb{E}[\nu_m | \nu_m = \nu_{m,0}] = \frac{m}{p - q}$

Corollary 33. *We have:* $\lim_{M\to\infty} \mathbb{E}[\nu_m | \nu_m = \nu_{m,0}] = \frac{m}{p-q}$

Alice's delay = biased random walk on $\{0, ..., A\}$.

Corollary 35. *We have:* $\lim_{M \to \infty} \mathbb{E}[\nu_m | \nu_m = \nu_{m,0}] = \frac{m}{p-q}$

Alice's delay = biased random walk on $\{0, ..., A\}$.

A step to the left = a block mined by Alice

Corollary 37. *We have:* $\lim_{M\to\infty} \mathbb{E}[\nu_m|\nu_m = \nu_{m,0}] = \frac{m}{p-q}$

Alice's delay = biased random walk on $\{0, \ldots, A\}$.

A step to the left = a block mined by Alice

Proposition 38. *The expected revenue per cycle is*

$$\frac{\mathbb{E}[\mathbf{R}_A]}{b} = \frac{q\,z}{2\,p}\; I_{4pq}\;(z,\; 1\,/\,2) - \frac{(A+1)\,\lambda^{A+1}}{p\,(1-\lambda)^3\,[A+1]^2}\; I_{(p-q)^2}\;(1\,/\,2,\; z) + \frac{2-\lambda+\lambda^{A+2}}{(1-\lambda)^2\,[A+1]}\,\frac{p^{z-1}\,q^z}{B(z,z)} + P_A(z)\,(v+1) \quad \text{with}\,[A+1] = \frac{1-\lambda^{A+1}}{1-\lambda}.$$

Corollary 39. *When $q \longrightarrow 0$, the minimal amount to make profitable a Nakamoto double spend with $z > 1$ confirmations is asymptotically $\bar{v} \geqslant \frac{q^{-z}}{2\binom{2z-1}{z}} b$ (b is the coinbase)*

Corollary 40. *When* $q \longrightarrow 0$*, the minimal amount to make profitable a Nakamoto double spend with* $z > 1$ *confirmations is asymptotically* $\bar{v} \geqslant \frac{q^{-z}}{2\binom{2z-1}{z}} b$ *(*b *is the coinbase)*

When $q = 0.01, \bar{v}(0.01) \approx 50$ coinbases $\approx$ 5 385 643 USD

Corollary 41. *When $q \longrightarrow 0$, the minimal amount to make profitable a Nakamoto double spend with $z > 1$ confirmations is asymptotically $\bar{v} \geqslant \frac{q^{-z}}{2\binom{2z-1}{z}} b$ (b is the coinbase)*

When $q = 0.01, \bar{v}(0.01) \approx 50$ coinbases ≈ 5 385 643 USD

Optimal strategy: $\bar{v}(0.01) = 49.2513$.

Corollary 42. *When* $q \longrightarrow 0$*, the minimal amount to make profitable a Nakamoto double spend with* $z > 1$ *confirmations is asymptotically* $\bar{v} \geqslant \frac{q^{-z}}{2\binom{2z-1}{z}} b$ *(*b *is the coinbase)*

When $q = 0.01, \bar{v}(0.01) \approx 50$ coinbases ≈ 5 385 643 USD

Optimal strategy: $\bar{v}(0.01) = 49.2513$.

For reasonable values of v, 1 or 2 confirmations are enough i.e.,

Corollary 43. *When $q \longrightarrow 0$, the minimal amount to make profitable a Nakamoto double spend with $z > 1$ confirmations is asymptotically $\bar{v} \geqslant \frac{q^{-z}}{2\binom{2z-1}{z}} b$ (b is the coinbase)*

When $q = 0.01, \bar{v}(0.01) \approx 50$ coinbases ≈ 5 385 643 USD

Optimal strategy: $\bar{v}(0.01) = 49.2513$.

For reasonable values of v, 1 or 2 confirmations are enough i.e.,

We have $\mathbf{\Gamma}_A(q) < \mathbf{\Gamma}_H$ for $z = 1$ or 2 and reasonable values of q.

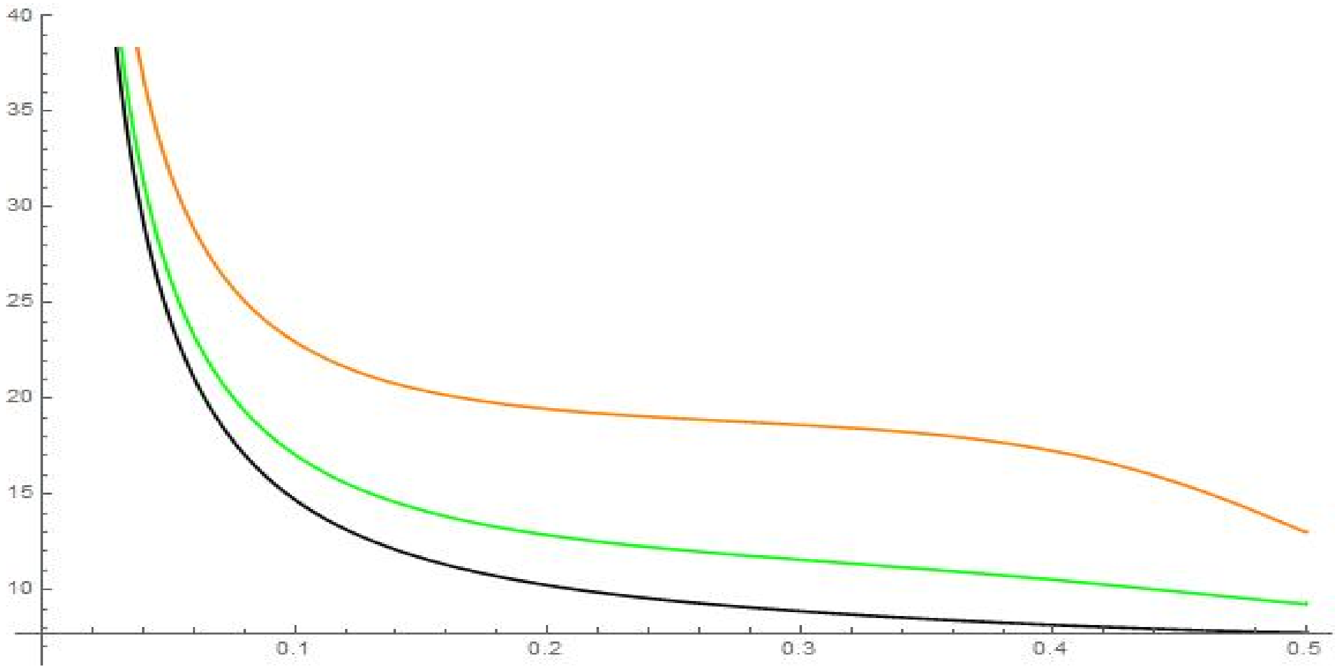

Figure 1. Mean cycle Time of the $(A, 1)$-Nakamoto strategy with $z = 2$ and $A = 3$ (black), 5 (green), 10 (orange). X-axis: relative hashrate, Y-axis: duration time (1 is 10 minutes).

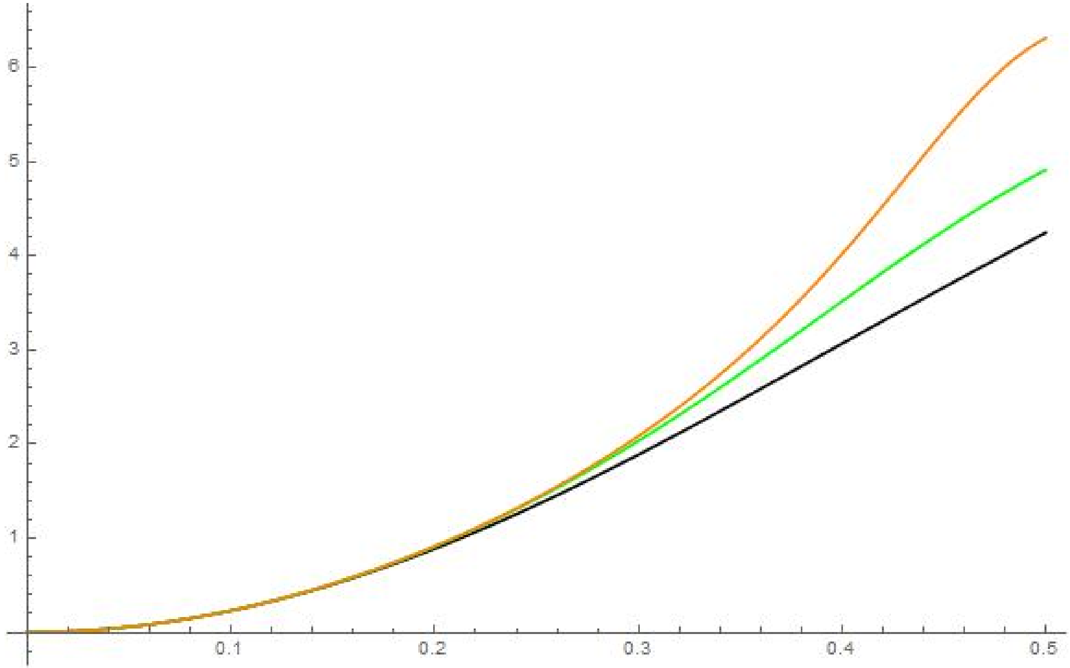

Figure 2. Mean revenue of the $(A,1)$-Nakamoto strategy by attack cycle with $z=2$, $v=1$ and $A=3$ (black), 5 (green), 10 (orange). X-axis: relative hashrate, Y-axis: revenue (1 is 1 coinbase).

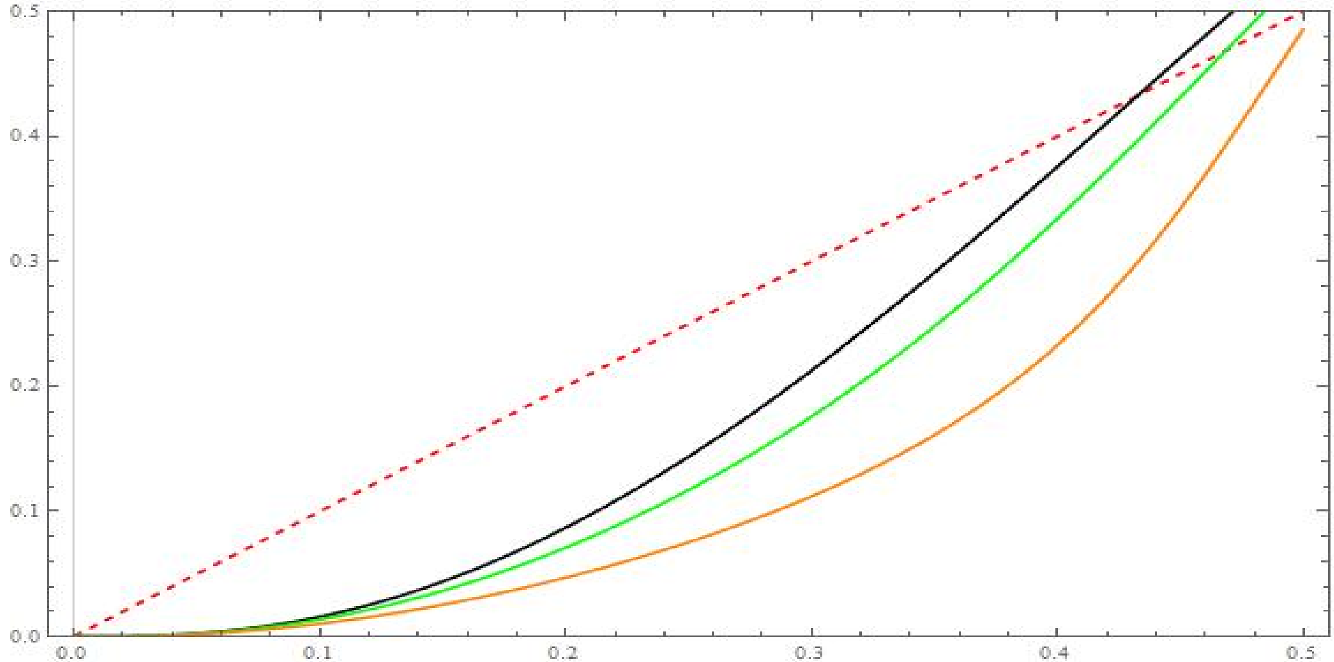

Figure 3. Revenue ratio of the $(A,1)$-Nakamoto strategy with $z=2$, $v=1$ and $A=3$ (black), 5 (green), 10 (orange). X-axis: relative hashrate, Y-axis: revenue ratio (1 is $\frac{b}{\tau_0}$ wit $b=1$ coinbase and $\tau_0=600$ seconds).