universität innsbruck

Blockchain-Daten analysieren mit

BlockSci

"A high-performance tool for blockchain science and exploration"

Martin Plattner, Universität Innsbruck
martin@mplattner.at

Vielen Dank an Malte Möser für die Mithilfe und das Zurverfügungstellen von Unterlagen.

Agenda

1. **Blockchain-Analyse:** Motivation, verfügbare Daten, Herausforderungen
2. **BlockSci:** Überblick, Design, Funktionsweise
3. **Demo:** BlockSci Grundlagen
4. **Hands-On:** Arbeiten Sie selbst mit BlockSci
5. **Abschluss:** Fazit

Blockchain-Analyse

Motivation, verfügbare Daten, Herausforderungen

Blockchain-Analyse: Wozu?

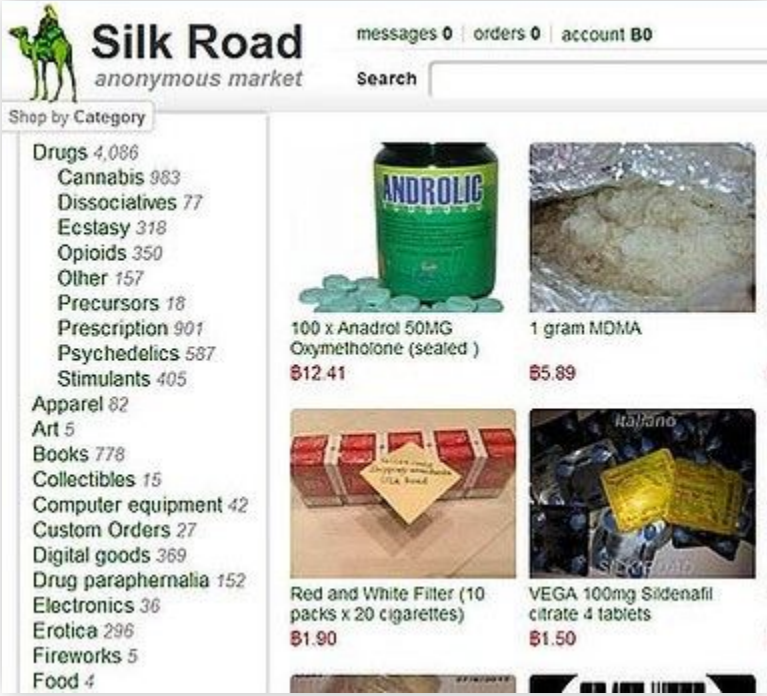
- Vermehrte Nutzung von Blockchain-Systemen
- Relevant u.a. für
  - Forensik, Strafverfolgung
  - Forschung: Ökonomie, Technik, Nutzerverhalten, Privacy
- Öffentliche Transaktionshistorie ermöglicht Analysen
  - Bitcoin: etwa 300 Gigabyte

Blockchain-Analyse: Wozu?

- Vermehrte Nutzung von Blockchain-Systemen
- Relevant u.a. für
  - Forensik, Strafverfolgung
  - Forschung: Ökonomie, Technik, Nutzerverhalten, Privacy
- Öffentliche Transaktionshistorie ermöglicht Analysen
  - Bitcoin: etwa 300 Gigabyte

Blockchain-Analyse: Wozu?

- Vermehrte Nutzung von Blockchain-Systemen
- Relevant u.a. für
  - Forensik, Strafverfolgung
  - Forschung: Ökonomie, Technik, Nutzerverhalten, Privacy
- Öffentliche Transaktionshistorie ermöglicht Analysen
  - Bitcoin: etwa 300 Gigabyte

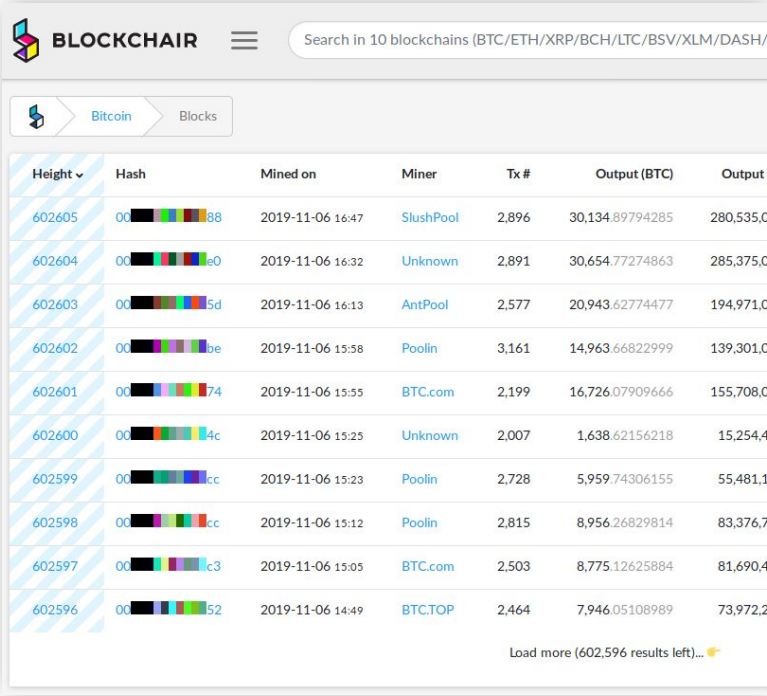


Blockchain-Analyse: Wozu?

- Vermehrte Nutzung von Blockchain-Systemen
- Relevant u.a. für
  - Forensik, Strafverfolgung
  - Forschung: Ökonomie, Technik, Nutzerverhalten, Privacy
- Öffentliche Transaktionshistorie ermöglicht Analysen
  - Bitcoin: etwa 300 Gigabyte

Blockchain-Analyse: Wozu?

- Vermehrte Nutzung von Blockchain-Systemen
- Relevant u.a. für
  - Forensik, Strafverfolgung
  - Forschung: Ökonomie, Technik, Nutzerverhalten, Privacy
- Öffentliche Transaktionshistorie ermöglicht Analysen
  - Bitcoin: etwa 300 Gigabyte

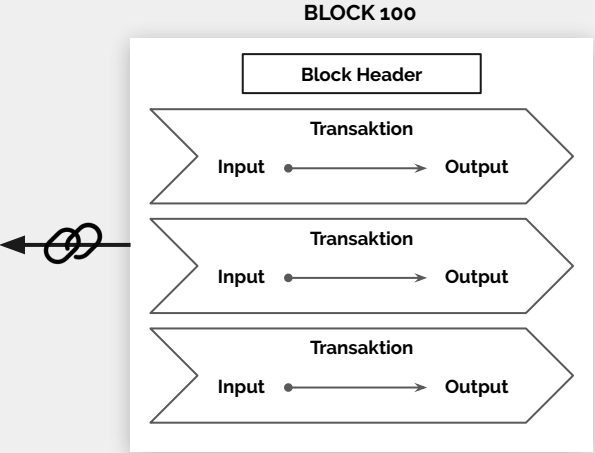


Quelle: blockchair.com

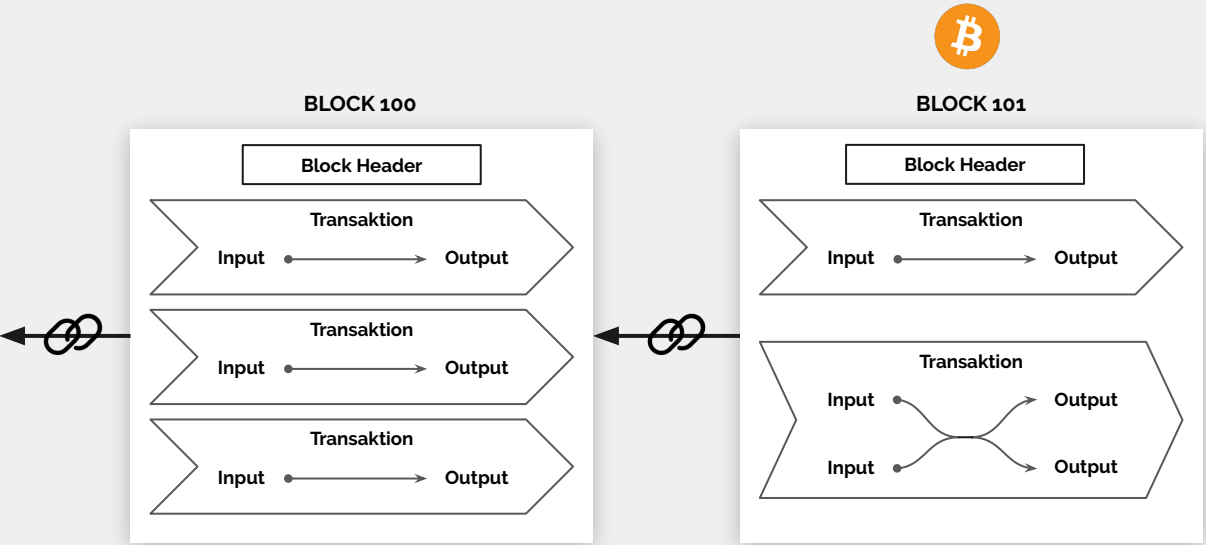
Blockchain-Analyse: Verfügbare Daten

Blockchain-Analyse: Verfügbare Daten

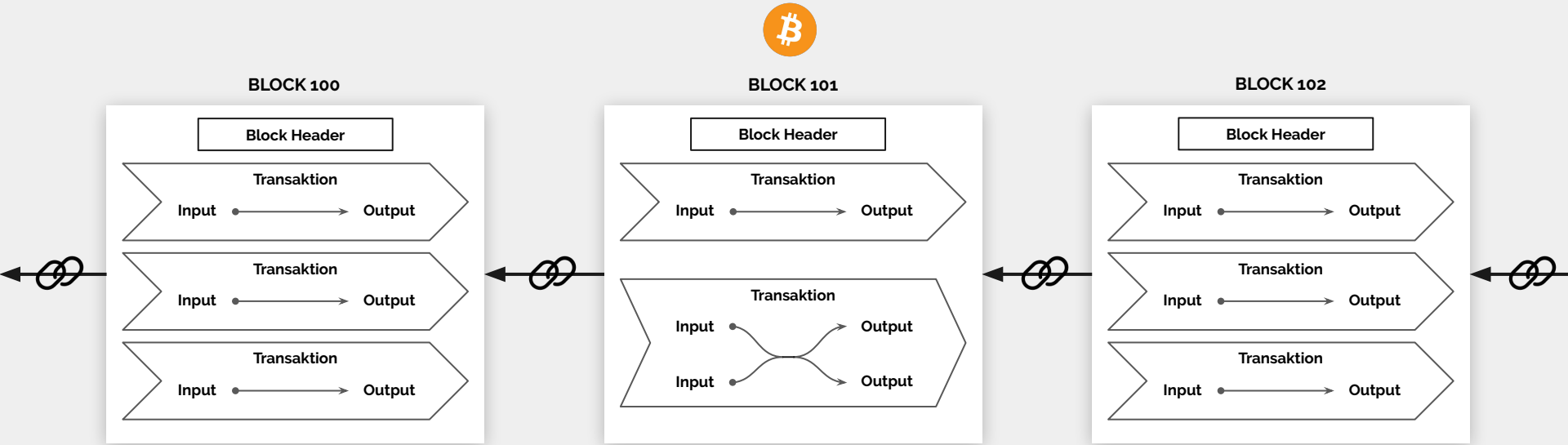
Blockchain-Analyse: Verfügbare Daten



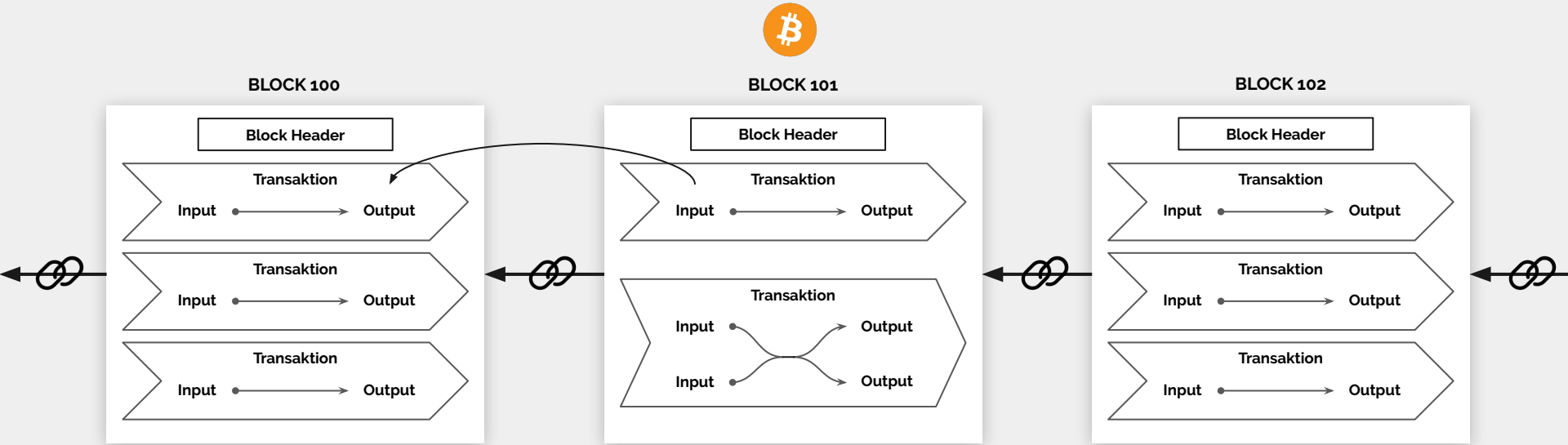
Blockchain-Analyse: Verfügbare Daten



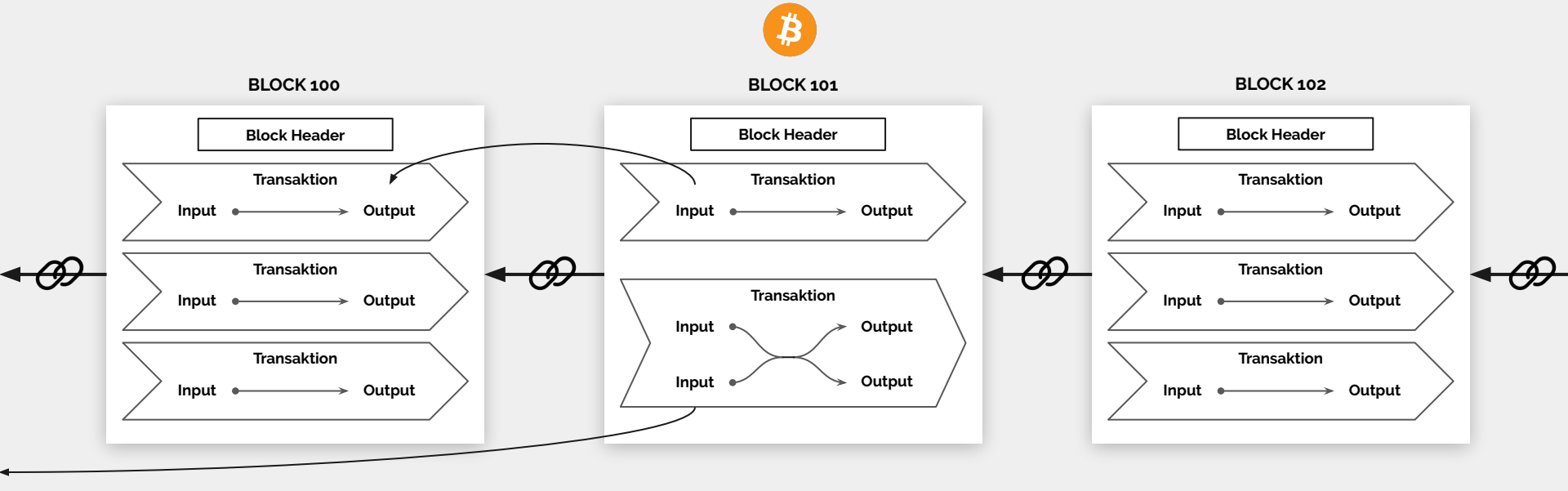
Blockchain-Analyse: Verfügbare Daten



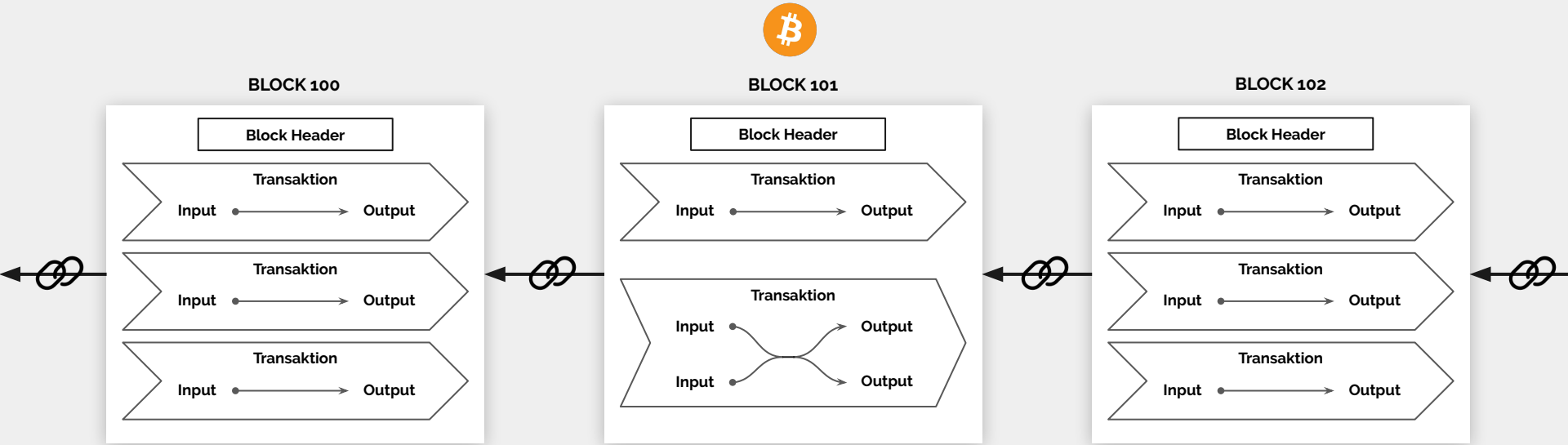
Blockchain-Analyse: Verfügbare Daten



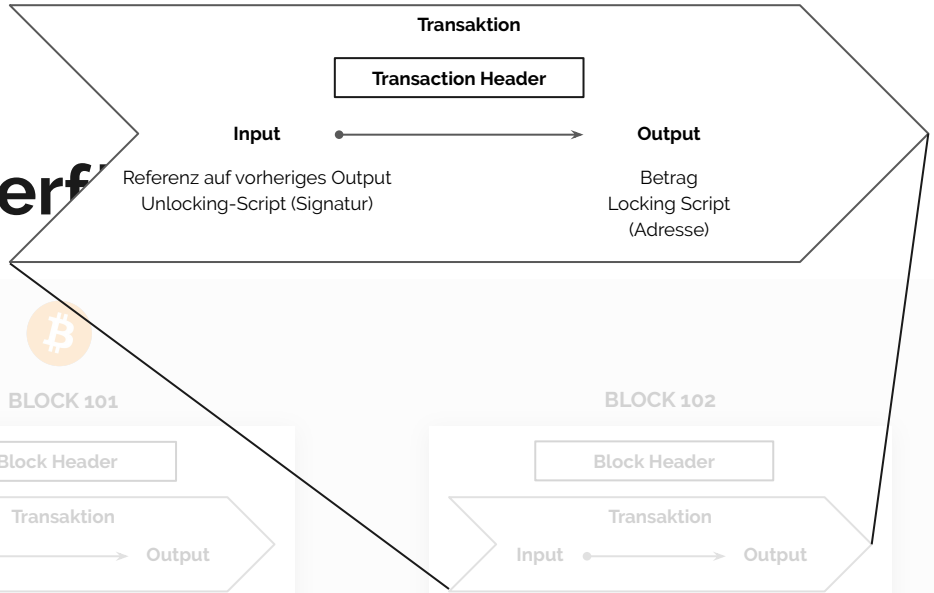
Blockchain-Analyse: Verfügbare Daten

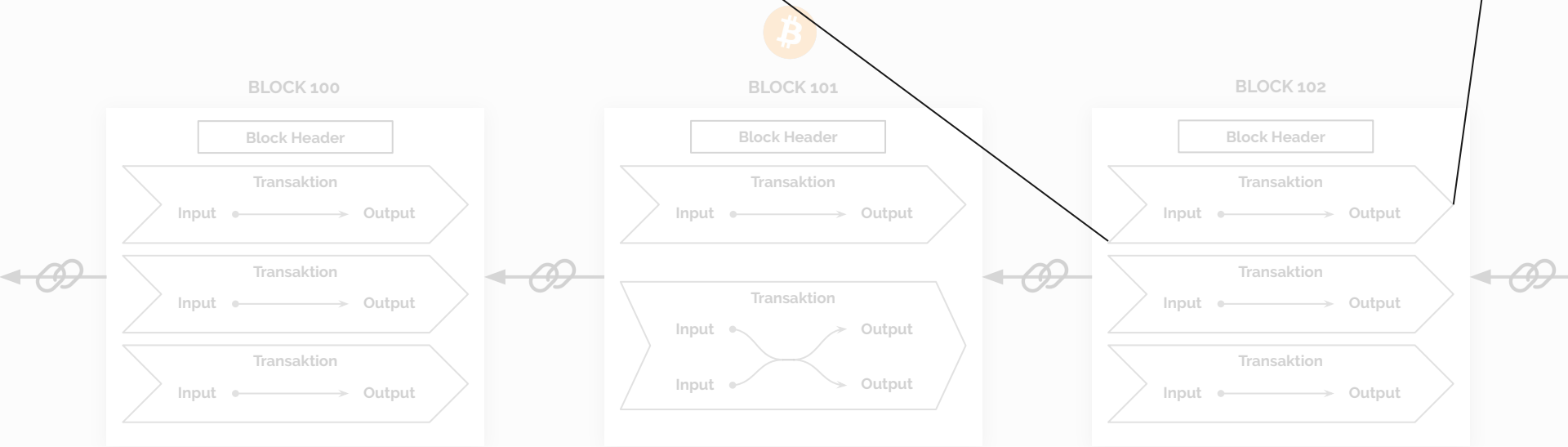


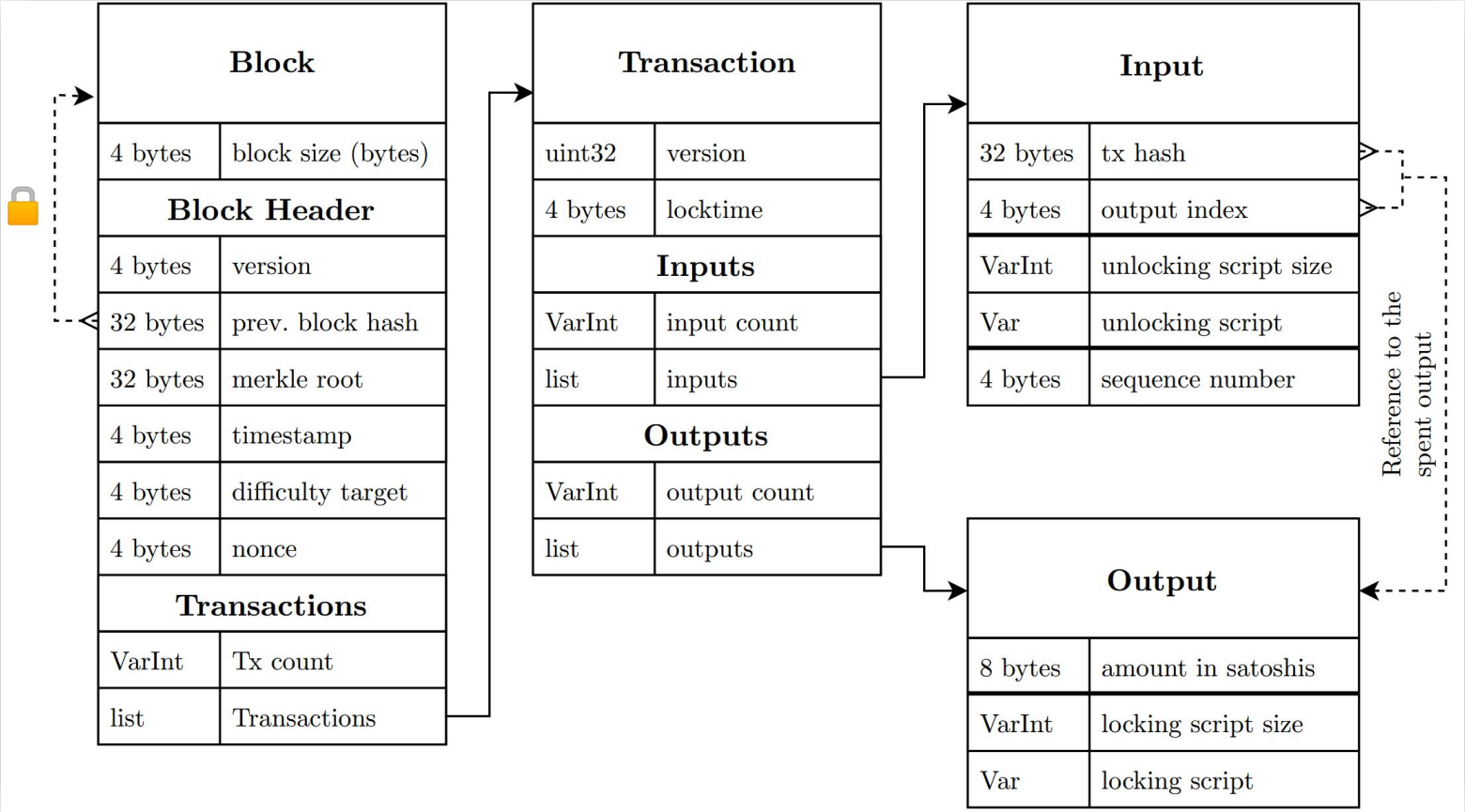
Blockchain-Analyse: Verfügbare Daten



Blockchain-Analyse: Verf






Block
4 bytes
block size (bytes)
Block Header
4 bytes
version
32 bytes
prev. block hash
32 bytes
merkle root
4 bytes
timestamp
4 bytes
difficulty target
4 bytes
nonce
Transactions
VarInt
Tx count
list
Transactions
Transaction
uint32
version
4 bytes
locktime
Inputs
VarInt
input count
list
inputs
Outputs
VarInt
output count
list
outputs
Input
32 bytes
tx hash
4 bytes
output index
VarInt
unlocking script size
Var
unlocking script
4 bytes
sequence number
Output
8 bytes
amount in satoshis
VarInt
locking script size
Var
locking script
Reference to the spent output

Blockchain-Analyse: Herausforderungen

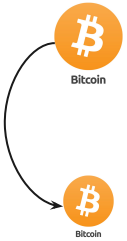
- Performance und Skalierbarkeit
  - Bitcoin: 300 GB Daten, 500 Mio. Transaktionen, 900 Mio. Adressen
- Benutzerfreundlichkeit
  - Fundierte Analysen setzen flexibles Interface voraus
- Reproduzierbarkeit
  - Insbesondere für Wissenschaft und Strafverfolgung wichtig

Blockchain-Analyse: Herausforderungen

Bitcoin

300 GB

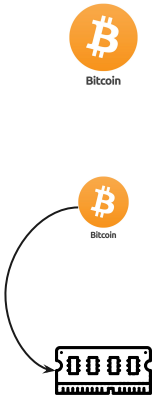
Blockchain-Analyse: Herausforderungen



300 GB

150 GB

Blockchain-Analyse: Herausforderungen



300 GB

150 GB

128 GB

Blockchain-Analyse: Herausfor

300 GB

150 GB

128 GB

AMD Ryzen™ 9 3900 12-Core

Simultaneous Multithreading

Arbeitsspeicher:	128 GB DDR4 ECC
Festplatten:	2 x 1,92 TB NVMe SSD Datacenter Edition
Anbindung:	1 GBit/s-Port
Bandbreite garantiert:	1 GBit/s
Backup Space:	100 GB
Traffic:	Unbegrenzt *
Verfügbare Standorte:	

99,96 €

monatlich + einmalig Setup: 105,91 €

Preis inkl. 19 % USt.

Verfügbarkeit prüfen ›

JETZT KONFIGURIEREN

Blockchain-Analyse: Herausforderungen

300 GB

150 GB

128 GB

Region: US East (Ohio)

	vCPU	ECU	Memory (GiB)	Instance Storage (GB)	Linux/UNIX Usage
General Purpose - Current Generation					
m5.4xlarge	16	70	64 GiB	EBS Only	$0.768 per Hour
m5.8xlarge	32	128	128 GiB	EBS Only	$1.536 per Hour
m5.12xlarge	48	168	192 GiB	EBS Only	$2.304 per Hour
m5.16xlarge	64	256	256 GiB	EBS Only	$3.072 per Hour
m5.24xlarge	96	337	384 GiB	EBS Only	$4.608 per Hour

Blockchain-Analyse: Herausforderungen

Bitcoin

300 GB

Bitcoin

150 GB

128 GB

3904 GB

Memory Optimized - Current Generation

x1.16xlarge	64	174.5	976 GiB	1 x 1920 SSD	$6.669 per Hour
x1.32xlarge	128	349	1,952 GiB	2 x 1920 SSD	$13.338 per Hour
x1e.xlarge	4	12	122 GiB	1 x 120 SSD	$0.834 per Hour
x1e.2xlarge	8	23	244 GiB	1 x 240 SSD	$1.668 per Hour
x1e.4xlarge	16	47	488 GiB	1 x 480 SSD	$3.336 per Hour
x1e.8xlarge	32	91	976 GiB	1 x 960 SSD	$6.672 per Hour
x1e.16xlarge	64	179	1,952 GiB	1 x 1920 SSD	$13.344 per Hour
x1e.32xlarge	128	340	3,904 GiB	2 x 1920 SSD	$26.688 per Hour

Blockchain-Analyse: Herausforderungen

300 GB

150 GB

128 GB

3904 GB

Fazit: Die Blockchain ist “klein”.

BlockSci

Überblick, Design, Funktionsweise

BlockSci: Überblick

Training Demo - Jupyter Notebook - Mozilla Firefox

Training Demo - Jupyter

localhost:8888/n 133%

jupyter Training Demo (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

```
In [1]: import blocksci

In [2]: btc = blocksci.Blockchain("/mnt/data/blocksci/bitcoin/595303-root-v0.6-0

In [3]: address = btc.address_from_string("1BvBMSEYstWetqTFn5Au4m4GFg7xJaNVN2")

In [4]: address.balance() / 1000000000

Out[4]: 0.060332617
```

BlockSci: Überblick

- Sehr performantes Blockchain Analysewerkzeug (Open-Source)

Why GitHub? Team Enterprise Explore Marketplace Pricing Search / Sign in Sign up

citp / **BlockSci**

Watch 70 Star 1k Fork 196

Code Issues 77 Pull requests 14 Actions Projects 2 Wiki Security 0 Insights

A high-performance tool for blockchain science and exploration https://arxiv.org/pdf/1709.02489.pdf

bitcoin blockchain analysis blocksci

1,136 commits **5** branches **0** packages **6** releases **8** contributors GPL-3.0

Branch: **v0.6** New pull request Find file Clone or download

This branch is 369 commits ahead, 28 commits behind master. Pull request Compare

mplattner and **maltemoeser** fix: random benchmarks don't work w/o m argument ... Latest commit 289a2f7 12 days ago

.circleci	ci: invalidate caches, upgrade cmake	10 days ago
.github/ISSUE_TEMPLATE	Add custom issue templates	2 years ago
.travis	test: add performance tests for Python interface	2 months ago
Notebooks	Refactor to separate out internal components into static library that…	2 years ago
benchmark	fix: random benchmarks don't work w/o m argument	10 days ago

Research mit BlockSci

Steven Goldfeder, Harry Kalodner, Dillon Reisman, and Arvind Narayanan. "When the cookie meets the blockchain: Privacy risks of web payments via cryptocurrencies". In: *Proceedings on Privacy Enhancing Technologies* 2018.4 (2018), pp. 179–199.

Yazan Boshmaf, Husam Al Jawaheri, and Mashael Al Sabah. "BlockTag: design and applications of a tagging system for blockchain analysis". In: *IFIP International Conference on ICT Systems Security and Privacy Protection*. Springer. 2019, pp. 299–313.

Marc Jourdan, Sebastien Blandin, Laura Wynter, and Pralhad Deshpande. "Characterizing entities in the Bitcoin blockchain". In: *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE. 2018, pp. 55–62.

Michael Fröwis, Thilo Gottschalk, Bernhard Haslhofer, Christian Rückert, and Paulina Pesch. "Safeguarding the Evidential Value of Forensic Cryptocurrency Investigations". In: *arXiv preprint arXiv:1906.12221* (2019).

Danny Yuxing Huang, Maxwell Matthaios Aliapoulios, Vector Guo Li, Luca Invernizzi, Elie Bursztein, Kylie McRoberts, Jonathan Levin, Kirill Levchenko, Alex C Snoeren, and Damon McCoy. "Tracking ransomware end-to-end". In: *2018 IEEE Symposium on Security and Privacy (S&P)*. IEEE. 2018, pp. 618–631.

Yury Zhauniarovich, Yazan Boshmaf, Husam Al Jawaheri, and Mashael Al Sabah. "Characterizing Bitcoin donations to open source software on GitHub". In: *CoRR* abs/1907.04002 (2019). URL: http://arxiv.org/abs/1907.04002.

BlockSci: Überblick

- Sehr performantes Blockchain Analysewerkzeug (Open-Source)
  - Iteration aller Transaktionen in wenigen Sekunden
  - Geringe Systemvoraussetzungen: 64 GB RAM, 1 TB SSD

BlockSci: Überblick

- Sehr performantes Blockchain Analysewerkzeug (Open-Source)
  - Iteration aller Transaktionen in wenigen Sekunden
  - Geringe Systemvoraussetzungen: 64 GB RAM, 1 TB SSD
- Abfragen und Analysen über Python-Interface
  - Erlaubt Zugriff auf Blöcke, Transaktionen, Inputs, Outputs, Adressen etc.

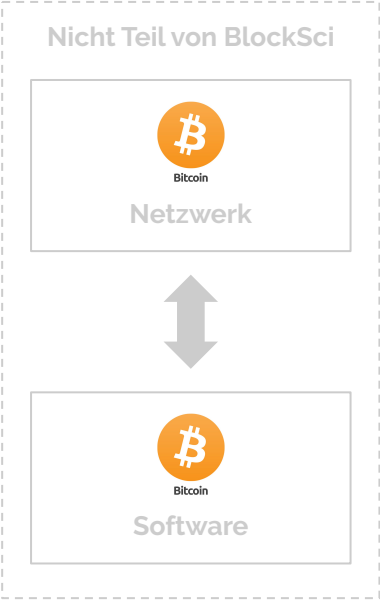
BlockSci: Überblick

- Sehr performantes Blockchain Analysewerkzeug (Open-Source)
  - Iteration aller Transaktionen in wenigen Sekunden
  - Geringe Systemvoraussetzungen: 64 GB RAM, 1 TB SSD
- Abfragen und Analysen über Python-Interface
  - Erlaubt Zugriff auf Blöcke, Transaktionen, Inputs, Outputs, Adressen etc.
- Unterstützt Bitcoin-Datenstruktur
  - daher auch Bitcoin Cash, Bitcoin SV, Dash, ZCash u.a.

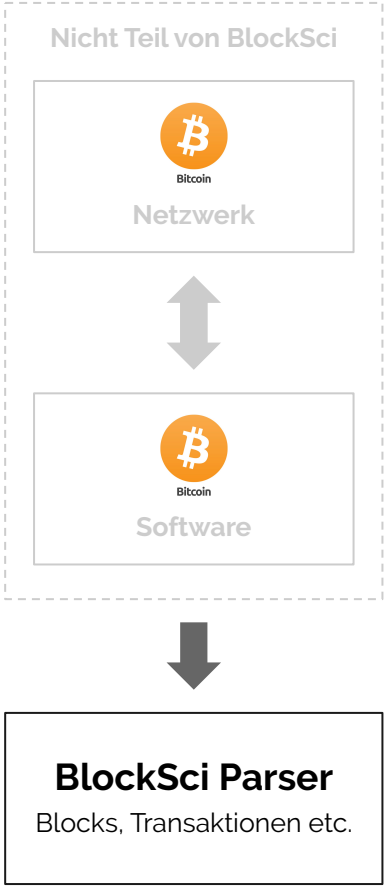
BlockSci: Design-Entscheidungen

- Domain-spezifische in-memory Datenbank
- Analytische Datenbank, im Gegensatz zu transaktioneller Datenbank
- Vertikale Skalierung: Läuft auf *einer* Maschine, *nicht* verteilt (d.h. keine “distributed database”)
- Viele Optimierungen, z.T. sehr Hardware-nah, programmiert in C++
- Effizientes Datenformat

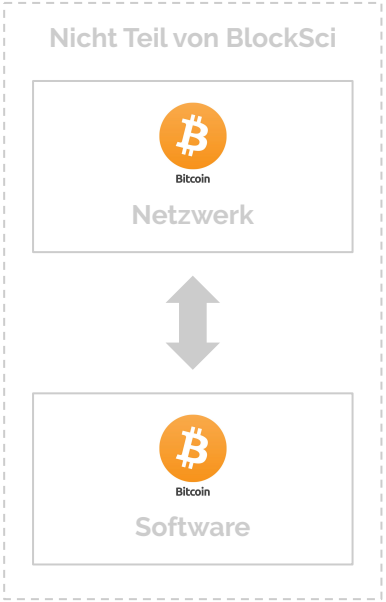
BlockSci: Funktionsweise

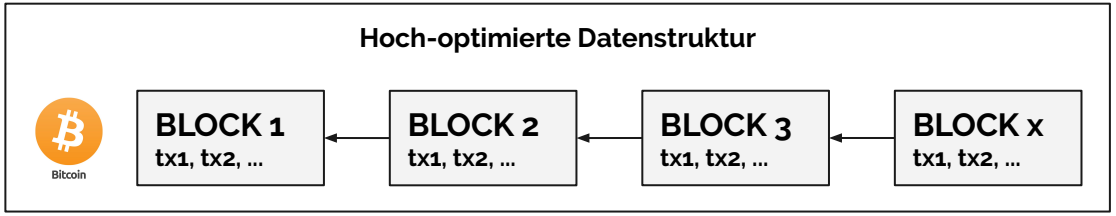


BlockSci: Funktionsweise



BlockSci: Funktionsweise





BlockSci: Funktionsweise

Hoch-optimierte Datenstruktur

BLOCK 1
tx1, tx2, ...

BLOCK 2
tx1, tx2, ...

BLOCK 3
tx1, tx2, ...

BLOCK x
tx1, tx2, ...

Für Analysen: Laden in den Arbeitsspeicher

Hoch-optimierte Datenstruktur

BLOCK 1
tx1, tx2, ...

BLOCK 2
tx1, tx2, ...

BLOCK 3
tx1, tx2, ...

BLOCK x
tx1, tx2, ...

BlockSci Parser
Blocks, Transaktionen etc.

Nicht Teil von BlockSci

Netzwerk

Software

BlockSci: Funktionsweise

Hoch-optimierte Datenstruktur

BLOCK 1
tx1, tx2, ...

BLOCK 2
tx1, tx2, ...

BLOCK 3
tx1, tx2, ...

BLOCK x
tx1, tx2, ...

Für Analysen: Laden in den Arbeitsspeicher

Hoch-optimierte Datenstruktur

BLOCK 1
tx1, tx2, ...

BLOCK 2
tx1, tx2, ...

BLOCK 3
tx1, tx2, ...

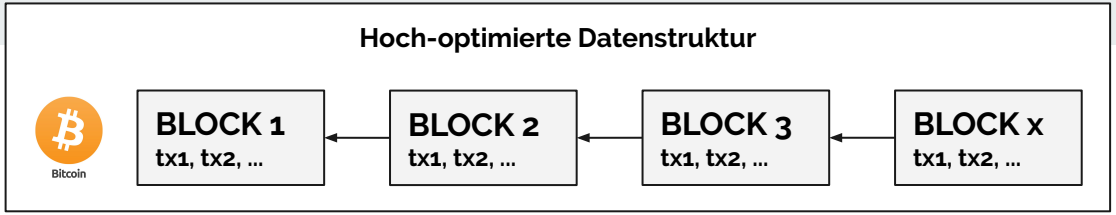
BLOCK x
tx1, tx2, ...

BlockSci Parser
Blocks, Transaktionen etc.

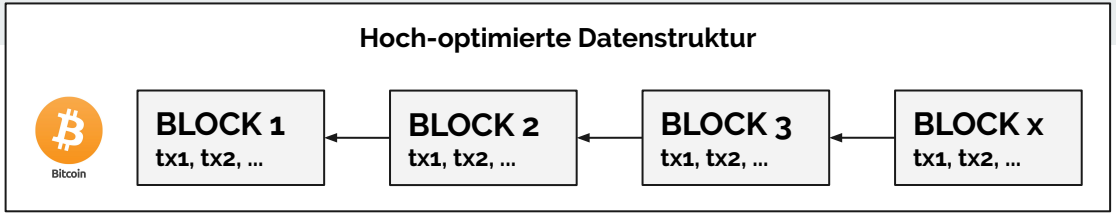
Nicht Teil von BlockSci

Netzwerk

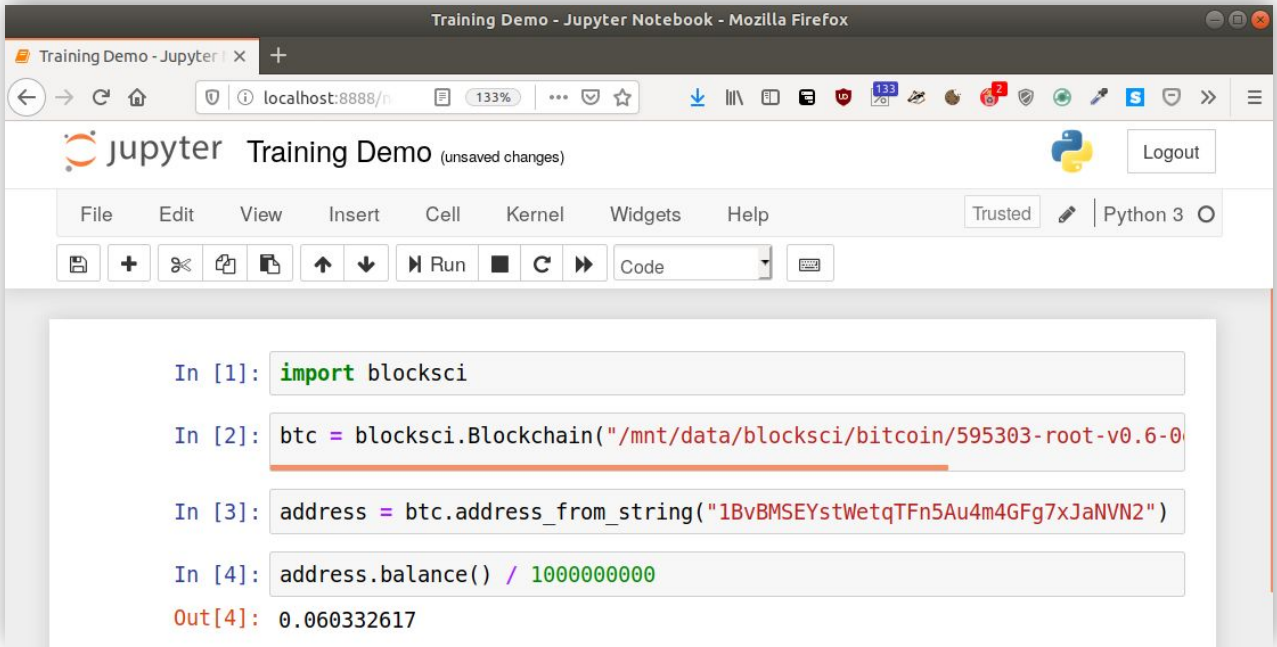
Software

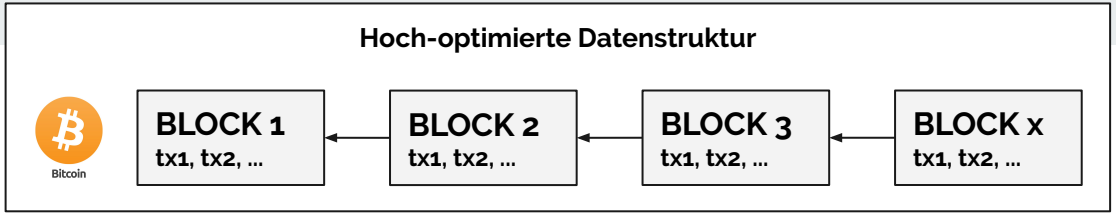


BlockSci: Funktionsweise

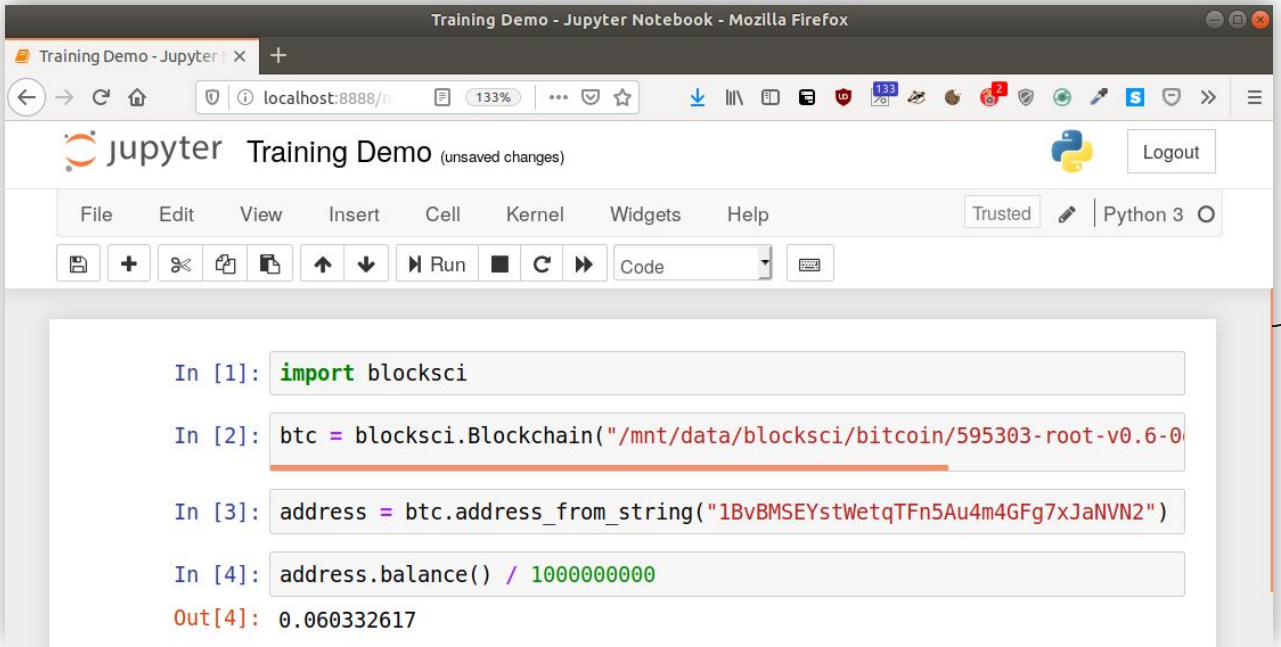


BlockSci: Funktionsweise





BlockSci: Funktionsweise



BlockSci: Funktionsweise

Hoch-optimierte Datenstruktur

BLOCK 1
tx1, tx2, ...

BLOCK 2
tx1, tx2, ...

BLOCK 3
tx1, tx2, ...

BLOCK x
tx1, tx2, ...

Für Analysen: Laden in den Arbeitsspeicher

Hoch-optimierte Datenstruktur

BLOCK 1
tx1, tx2, ...

BLOCK 2
tx1, tx2, ...

BLOCK 3
tx1, tx2, ...

BLOCK x
tx1, tx2, ...

BlockSci Parser
Blocks, Transaktionen etc.

Nicht Teil von BlockSci

Netzwerk

Software

BlockSci: Funktionsweise

Hoch-optimierte Datenstruktur

Bitcoin

blockFile	RawBlock | RawBlock | RawBlock | RawBlock | RawBlock, ...
txFile	RawTransaction | RawTransaction | RawTransaction, ...

Für Analysen: Laden in den Arbeitsspeicher

Hoch-optimierte Datenstruktur

Bitcoin

blockFile	RawBlock | RawBlock | RawBlock | RawBlock | RawBlock, ...
txFile	RawTransaction | RawTransaction | RawTransaction, ...

BlockSci Parser

Blocks, Transaktionen etc.

Nicht Teil von BlockSci

Bitcoin

Netzwerk

Bitcoin

Software

BlockSci: Funktionsweise

Nicht Teil von BlockSci

```
struct RawBlock {
    uint32_t firstTxIndex;      // offset: 0 bytes
    uint32_t numTxes;           // offset: 4 bytes
    uint32_t height;            // offset: 8 bytes
    uint256 hash;               // offset: 12 bytes
    int32_t version;            // offset: 44 bytes
    uint32_t timestamp;         // offset: 48 bytes
    uint32_t bits;              // offset: 52 bytes
    uint32_t nonce;             // offset: 56 bytes
    uint32_t realSize;          // offset: 60 bytes
    uint32_t baseSize;          // offset: 64 bytes
    uint64_t coinbaseOffset;    // offset: 68 bytes
                                // total size: 76 bytes

} // padded size: 80 bytes
```

Hoch-optimierte Datenstruktur

Bitcoin

blockFile: RawBlock | RawBlock | RawBlock | RawBlock | RawBlock, ...

txFile: RawTransaction | RawTransaction | RawTransaction, ...

Für Analysen: Laden in den Arbeitsspeicher

Hoch-optimierte Datenstruktur

Bitcoin

blockFile: RawBlock | RawBlock | RawBlock | RawBlock | RawBlock, ...

txFile: RawTransaction | RawTransaction | RawTransaction, ...

BlockSci Parser

Blocks, Transaktionen etc.

Exkurs I

BlockSci's Datenformat im Hex-Editor

block.dat ×

Edit As: Hex Run Script Run Template: block.dat.bt

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000h:	06	22	6E	46	11	1A	0B	59	CA	AF	12	60	43	EB	5B	BF
0010h:	28	C3	4F	3A	5E	33	2A	1F	C7	B2	B7	3C	F1	88	91	0F
0020h:	00	00	00	00	00	00	00	00	00	00	00	00	01	00	00	00
0030h:	00	00	00	00	01	00	00	00	01	00	00	00	01	00	00	00
0040h:	DA	E5	49	4D	FF	FF	7F	20	02	00	00	00	1D	01	00	00
0050h:	1D	01	00	00	00	00	00	C8	A5	94	09	48	4F	D0	EA	FD
0060h:	03	5B	45	DF	90	60	16	C7	8B	F8	98	38	82	C1	EE	D9
0070h:	D9	BD	E9	70	02	06	FD	66	51	00	00	00	00	00	00	00
0080h:	01	00	00	00	01	00	00	00	00	00	00	00	02	00	00	00
0090h:	02	00	00	00	00	00	00	20	D8	D8	89	5B	FF	FF	7F	20
00A0h:	01	00	00	00	FC	00	00	00	D8	00	00	00	00	00	00	C8
00B0h:	3E	42	73	08	1D	22	78	49	64	86	E6	1E	E4	33	28	71
00C0h:	BB	94	3A	EF	56	63	DA	49	65	51	B0	1E	2D	D3	A4	29
00D0h:	58	00	00	00	00	00	00	00	02	00	00	00	01	00	00	00
00E0h:	00	00	00	00	02	00	00	00	03	00	00	00	00	00	00	20
00F0h:	D9	D8	89	5B	FF	FF	7F	20	00	00	00	00	FC	00	00	00
0100h:	D8	00	00	00	00	00	00	C8	2D	74	10	E8	D7	BF	4F	C7
0110h:	F3	AB	DD	D7	E0	C0	8B	6F	94	69	5E	04	E7	99	AD	19
0120h:	0E	48	CF	15	79	02	90	56	5F	00	00	00	00	00	00	00
0130h:	03	00	00	00	01	00	00	00	00	00	00	00	02	00	00	00
0140h:	04	00	00	00	00	00	00	20	D9	D8	89	5B	FF	FF	7F	20
0150h:	00	00	00	00	FC	00	00	00	D8	00	00	00	00	00	00	C8
0160h:	9C	C5	9D	CD	BD	53	1A	48	FC	8D	AE	4B	7F	FA	15	C6
0170h:	CE	E8	EE	8C	6A	12	9B	06	B4	C6	CD	FF	0B	9D	B0	11
0180h:	66	00	00	00	00	00	00	00	04	00	00	00	01	00	00	00
0190h:	00	00	00	00	02	00	00	00	05	00	00	00	00	00	00	20
01A0h:	DA	D8	89	5B	FF	FF	7F	20	00	00	00	00	FC	00	00	00
01B0h:	D8	00	00	00	00	00	00	C8	6B	37	80	FC	F8	58	CE	05
01C0h:	91	70	07	D3	2F	85	69	CE	C6	D1	4D	F2	37	73	0D	76
01D0h:	1F	B5	28	67	41	FC	37	65	6D	00	00	00	00	00	00	00
01E0h:	05	00	00	00	01	00	00	00	00	00	00	00	02	00	00	00
01F0h:	06	00	00	00	00	00	00	20	DA	D8	89	5B	FF	FF	7F	20
0200h:	00	00	00	00	FC	00	00	00	D8	00	00	00	00	00	00	C8

Variables

Name	Value	Start	Size	Color	
struct RawBlock blocks[80]		0h	7040	Fg:	Bg:
struct RawBlock blocks[0]		0h	88	Fg:	Bg:
int64 hash[4]		0h	32	Fg:	Bg:
uint64 coinbaseOffset	0	20h	8	Fg:	Bg:
uint32 firstTxIndex	0	28h	4	Fg:	Bg:
uint32 txCount	1	2Ch	4	Fg:	Bg:
uint32 inputCount	0	30h	4	Fg:	Bg:
uint32 outputCount	1	34h	4	Fg:	Bg:
uint32 height	1	38h	4	Fg:	Bg:
int32 version	1	3Ch	4	Fg:	Bg:
uint32 timestamp	1296688602	40h	4	Fg:	Bg:
uint32 bits	545259519	44h	4	Fg:	Bg:
uint32 nonce	2	48h	4	Fg:	Bg:
uint32 realSize	285	4Ch	4	Fg:	Bg:
uint32 baseSize	285	50h	4	Fg:	Bg:
uint32 padding	3355443200	54h	4	Fg:	Bg:
struct RawBlock blocks[1]		58h	88	Fg:	Bg:
int64 hash[4]		58h	32	Fg:	Bg:
uint64 coinbaseOffset	81	78h	8	Fg:	Bg:
uint32 firstTxIndex	1	80h	4	Fg:	Bg:
uint32 txCount	1	84h	4	Fg:	Bg:
uint32 inputCount	0	88h	4	Fg:	Bg:
uint32 outputCount	2	8Ch	4	Fg:	Bg:
uint32 height	2	90h	4	Fg:	Bg:
int32 version	536870912	94h	4	Fg:	Bg:
uint32 timestamp	1535760600	98h	4	Fg:	Bg:
uint32 bits	545259519	9Ch	4	Fg:	Bg:
uint32 nonce	1	A0h	4	Fg:	Bg:
uint32 realSize	252	A4h	4	Fg:	Bg:
uint32 baseSize	216	A8h	4	Fg:	Bg:
uint32 padding	3355443200	ACh	4	Fg:	Bg:
struct RawBlock blocks[2]		B0h	88	Fg:	Bg:
struct RawBlock blocks[3]		108h	88	Fg:	Bg:
struct RawBlock blocks[4]		160h	88	Fg:	Bg:
struct RawBlock blocks[5]		1B8h	88	Fg:	Bg:
struct RawBlock blocks[6]		210h	88	Fg:	Bg:
struct RawBlock blocks[7]		268h	88	Fg:	Bg:
struct RawBlock blocks[8]		2C0h	88	Fg:	Bg:
struct RawBlock blocks[9]		318h	88	Fg:	Bg:
struct RawBlock blocks[10]		370h	88	Fg:	Bg:
struct RawBlock blocks[11]		3C8h	88	Fg:	Bg:
struct RawBlock blocks[12]		420h	88	Fg:	Bg:
struct RawBlock blocks[13]		478h	88	Fg:	Bg:

block.dat ×

Edit As: Hex Run Script Run Template: block.dat.bt

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000h:	06	22	6E	46	11	1A	0B	59	CA	AF	12	60	43	EB	5B	BF
0010h:	28	C3	4F	3A	5E	33	2A	1F	C7	B2	B7	3C	F1	88	91	0F
0020h:	00	00	00	00	00	00	00	00	00	00	00	00	01	00	00	00
0030h:	00	00	00	00	01	00	00	00	01	00	00	00	01	00	00	00
0040h:	DA	E5	49	4D	FF	FF	7F	20	02	00	00	00	1D	01	00	00
0050h:	1D	01	00	00	00	00	00	C8	A5	94	09	48	4F	D0	EA	FD
0060h:	03	5B	45	DF	90	60	16	C7	8B	F8	98	38	82	C1	EE	D9
0070h:	D9	BD	E9	70	02	06	FD	66	51	00	00	00	00	00	00	00
0080h:	01	00	00	00	01	00	00	00	00	00	00	00	02	00	00	00
0090h:	02	00	00	00	00	00	00	20	D8	D8	89	5B	FF	FF	7F	20
00A0h:	01	00	00	00	FC	00	00	00	D8	00	00	00	00	00	00	C8
00B0h:	3E	42	73	08	1D	22	78	49	64	86	E6	1E	E4	33	28	71
00C0h:	BB	94	3A	EF	56	63	DA	49	65	51	B0	1E	2D	D3	A4	29
00D0h:	58	00	00	00	00	00	00	00	02	00	00	00	01	00	00	00
00E0h:	00	00	00	00	02	00	00	00	03	00	00	00	00	00	00	20
00F0h:	D9	D8	89	5B	FF	FF	7F	20	00	00	00	00	FC	00	00	00
0100h:	D8	00	00	00	00	00	00	C8	2D	74	10	E8	D7	BF	4F	C7
0110h:	F3	AB	DD	D7	E0	C0	8B	6F	94	69	5E	04	E7	99	AD	19
0120h:	0E	48	CF	15	79	02	90	56	5F	00	00	00	00	00	00	00
0130h:	03	00	00	00	01	00	00	00	00	00	00	00	02	00	00	00
0140h:	04	00	00	00	00	00	00	20	D9	D8	89	5B	FF	FF	7F	20
0150h:	00	00	00	00	FC	00	00	00	D8	00	00	00	00	00	00	C8
0160h:	9C	C5	9D	CD	BD	53	1A	48	FC	8D	AE	4B	7F	FA	15	C6
0170h:	CE	E8	EE	8C	6A	12	9B	06	B4	C6	CD	FF	0B	9D	B0	11
0180h:	66	00	00	00	00	00	00	00	04	00	00	00	01	00	00	00
0190h:	00	00	00	00	02	00	00	00	05	00	00	00	00	00	00	20
01A0h:	DA	D8	89	5B	FF	FF	7F	20	00	00	00	00	FC	00	00	00
01B0h:	D8	00	00	00	00	00	00	C8	6B	37	80	FC	F8	58	CE	05
01C0h:	91	70	07	D3	2F	85	69	CE	C6	D1	4D	F2	37	73	0D	76
01D0h:	1F	B5	28	67	41	FC	37	65	6D	00	00	00	00	00	00	00
01E0h:	05	00	00	00	01	00	00	00	00	00	00	00	02	00	00	00
01F0h:	06	00	00	00	00	00	00	20	DA	D8	89	5B	FF	FF	7F	20
0200h:	00	00	00	00	FC	00	00	00	D8	00	00	00	00	00	00	C8

Variables

Name	Value	Start	Size	Color
struct RawBlock blocks[80]		0h	7040	Fg: Bg:
struct RawBlock blocks[0]		0h	88	Fg: Bg:
int64 hash[4]		0h	32	Fg: Bg:
uint64 coinbaseOffset	0	20h	8	Fg: Bg:
uint32 firstTxIndex	0	28h	4	Fg: Bg:
uint32 txCount	1	2Ch	4	Fg: Bg:
uint32 inputCount	0	30h	4	Fg: Bg:
uint32 outputCount	1	34h	4	Fg: Bg:
uint32 height	1	38h	4	Fg: Bg:
int32 version	1	3Ch	4	Fg: Bg:
uint32 timestamp	1296688602	40h	4	Fg: Bg:
uint32 bits	545259519	44h	4	Fg: Bg:
uint32 nonce	2	48h	4	Fg: Bg:
uint32 realSize	285	4Ch	4	Fg: Bg:
uint32 baseSize	285	50h	4	Fg: Bg:
uint32 padding	3355443200	54h	4	Fg: Bg:
struct RawBlock blocks[1]		58h	88	Fg: Bg:
int64 hash[4]		58h	32	Fg: Bg:
uint64 coinbaseOffset	81	78h	8	Fg: Bg:
uint32 firstTxIndex	1	80h	4	Fg: Bg:
uint32 txCount	1	84h	4	Fg: Bg:
uint32 inputCount	0	88h	4	Fg: Bg:
uint32 outputCount	2	8Ch	4	Fg: Bg:
uint32 height	2	90h	4	Fg: Bg:
int32 version	536870912	94h	4	Fg: Bg:
uint32 timestamp	1535760600	98h	4	Fg: Bg:
uint32 bits	545259519	9Ch	4	Fg: Bg:
uint32 nonce	1	A0h	4	Fg: Bg:
uint32 realSize	252	A4h	4	Fg: Bg:
uint32 baseSize	216	A8h	4	Fg: Bg:
uint32 padding	3355443200	ACh	4	Fg: Bg:
struct RawBlock blocks[2]		B0h	88	Fg: Bg:
struct RawBlock blocks[3]		108h	88	Fg: Bg:
struct RawBlock blocks[4]		160h	88	Fg: Bg:
struct RawBlock blocks[5]		1B8h	88	Fg: Bg:
struct RawBlock blocks[6]		210h	88	Fg: Bg:
struct RawBlock blocks[7]		268h	88	Fg: Bg:
struct RawBlock blocks[8]		2C0h	88	Fg: Bg:
struct RawBlock blocks[9]		318h	88	Fg: Bg:
struct RawBlock blocks[10]		370h	88	Fg: Bg:
struct RawBlock blocks[11]		3C8h	88	Fg: Bg:
struct RawBlock blocks[12]		420h	88	Fg: Bg:
struct RawBlock blocks[13]		478h	88	Fg: Bg:

block.dat ×

Edit As: Hex Run Script Run Template: block.dat.bt

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000h:	06	22	6E	46	11	1A	0B	59	CA	AF	12	60	43	EB	5B	BF
0010h:	28	C3	4F	3A	5E	33	2A	1F	C7	B2	B7	3C	F1	88	91	0F
0020h:	00	00	00	00	00	00	00	00	00	00	00	00	01	00	00	00
0030h:	00	00	00	00	01	00	00	00	01	00	00	00	01	00	00	00
0040h:	DA	E5	49	4D	FF	FF	7F	20	02	00	00	00	1D	01	00	00
0050h:	1D	01	00	00	00	00	00	C8	A5	94	09	48	4F	D0	EA	FD
0060h:	03	5B	45	DF	90	60	16	C7	8B	F8	98	38	82	C1	EE	D9
0070h:	D9	BD	E9	70	02	06	FD	66	51	00	00	00	00	00	00	00
0080h:	01	00	00	00	01	00	00	00	00	00	00	00	02	00	00	00
0090h:	02	00	00	00	00	00	00	20	D8	D8	89	5B	FF	FF	7F	20
00A0h:	01	00	00	00	FC	00	00	00	D8	00	00	00	00	00	00	C8
00B0h:	3E	42	73	08	1D	22	78	49	64	86	E6	1E	E4	33	28	71
00C0h:	BB	94	3A	EF	56	63	DA	49	65	51	B0	1E	2D	D3	A4	29
00D0h:	58	00	00	00	00	00	00	00	02	00	00	00	01	00	00	00
00E0h:	00	00	00	00	02	00	00	00	03	00	00	00	00	00	00	20
00F0h:	D9	D8	89	5B	FF	FF	7F	20	00	00	00	00	FC	00	00	00
0100h:	D8	00	00	00	00	00	00	C8	2D	74	10	E8	D7	BF	4F	C7
0110h:	F3	AB	DD	D7	E0	C0	8B	6F	94	69	5E	04	E7	99	AD	19
0120h:	0E	48	CF	15	79	02	90	56	5F	00	00	00	00	00	00	00
0130h:	03	00	00	00	01	00	00	00	00	00	00	00	02	00	00	00
0140h:	04	00	00	00	00	00	00	20	D9	D8	89	5B	FF	FF	7F	20
0150h:	00	00	00	00	FC	00	00	00	D8	00	00	00	00	00	00	C8
0160h:	9C	C5	9D	CD	BD	53	1A	48	FC	8D	AE	4B	7F	FA	15	C6
0170h:	CE	E8	EE	8C	6A	12	9B	06	B4	C6	CD	FF	0B	9D	B0	11
0180h:	66	00	00	00	00	00	00	00	04	00	00	00	01	00	00	00
0190h:	00	00	00	00	02	00	00	00	05	00	00	00	00	00	00	20
01A0h:	DA	D8	89	5B	FF	FF	7F	20	00	00	00	00	FC	00	00	00
01B0h:	D8	00	00	00	00	00	00	C8	6B	37	80	FC	F8	58	CE	05
01C0h:	91	70	07	D3	2F	85	69	CE	C6	D1	4D	F2	37	73	0D	76
01D0h:	1F	B5	28	67	41	FC	37	65	6D	00	00	00	00	00	00	00
01E0h:	05	00	00	00	01	00	00	00	00	00	00	00	02	00	00	00
01F0h:	06	00	00	00	00	00	00	20	DA	D8	89	5B	FF	FF	7F	20
0200h:	00	00	00	00	FC	00	00	00	D8	00	00	00	00	00	00	C8

Variables

Name	Value	Start	Size	Color
struct RawBlock blocks[80]		0h	7040	Fg: Bg:
struct RawBlock blocks[0]		0h	88	Fg: Bg:
int64 hash[4]		0h	32	Fg: Bg:
uint64 coinbaseOffset	0	20h	8	Fg: Bg:
uint32 firstTxIndex	0	28h	4	Fg: Bg:
uint32 txCount	1	2Ch	4	Fg: Bg:
uint32 inputCount	0	30h	4	Fg: Bg:
uint32 outputCount	1	34h	4	Fg: Bg:
uint32 height	1	38h	4	Fg: Bg:
int32 version	1	3Ch	4	Fg: Bg:
uint32 timestamp	1296688602	40h	4	Fg: Bg:
uint32 bits	545259519	44h	4	Fg: Bg:
uint32 nonce	2	48h	4	Fg: Bg:
uint32 realSize	285	4Ch	4	Fg: Bg:
uint32 baseSize	285	50h	4	Fg: Bg:
uint32 padding	3355443200	54h	4	Fg: Bg:
struct RawBlock blocks[1]		58h	88	Fg: Bg:
int64 hash[4]		58h	32	Fg: Bg:
uint64 coinbaseOffset	81	78h	8	Fg: Bg:
uint32 firstTxIndex	1	80h	4	Fg: Bg:
uint32 txCount	1	84h	4	Fg: Bg:
uint32 inputCount	0	88h	4	Fg: Bg:
uint32 outputCount	2	8Ch	4	Fg: Bg:
uint32 height	2	90h	4	Fg: Bg:
int32 version	536870912	94h	4	Fg: Bg:
uint32 timestamp	1535760600	98h	4	Fg: Bg:
uint32 bits	545259519	9Ch	4	Fg: Bg:
uint32 nonce	1	A0h	4	Fg: Bg:
uint32 realSize	252	A4h	4	Fg: Bg:
uint32 baseSize	216	A8h	4	Fg: Bg:
uint32 padding	3355443200	ACh	4	Fg: Bg:
struct RawBlock blocks[2]		B0h	88	Fg: Bg:
struct RawBlock blocks[3]		108h	88	Fg: Bg:
struct RawBlock blocks[4]		160h	88	Fg: Bg:
struct RawBlock blocks[5]		1B8h	88	Fg: Bg:
struct RawBlock blocks[6]		210h	88	Fg: Bg:
struct RawBlock blocks[7]		268h	88	Fg: Bg:
struct RawBlock blocks[8]		2C0h	88	Fg: Bg:
struct RawBlock blocks[9]		318h	88	Fg: Bg:
struct RawBlock blocks[10]		370h	88	Fg: Bg:
struct RawBlock blocks[11]		3C8h	88	Fg: Bg:
struct RawBlock blocks[12]		420h	88	Fg: Bg:
struct RawBlock blocks[13]		478h	88	Fg: Bg:

block.dat ×

Edit As: Hex Run Script Run Template: block.dat.bt

```
0000h: 06 22 6E 46 11 1A 0B 59 CA AF 12 60 43
0010h: 28 C3 4F 3A 5E 33 2A 1F C7 B2 B7 3C
0020h: 00 00 00 00 00 00 00 00 00 00 00 00
0030h: 00 00 00 00 01 00 00 00 01 00 00 0
0040h: DA E5 49 4D FF FF 7F 20 02 00 00
0050h: 1D 01 00 00 00 00 00 C8 A5 94 0
0060h: 03 5B 45 DF 90 60 16 C7 8B F8
0070h: D9 BD E9 70 02 06 FD 66 51 00
0080h: 01 00 00 00 01 00 00 00 00
0090h: 02 00 00 00 00 00 00 20 D8
00A0h: 01 00 00 00 FC 00 00 00 D8
00B0h: 3E 42 73 08 1D 22 78 49 64 8
00C0h: BB 94 3A EF 56 63 DA 49 65 51
00D0h: 58 00 00 00 00 00 00 00 02 00
00E0h: 00 00 00 00 02 00 00 00 03 00 00
00F0h: D9 D8 89 5B FF FF 7F 20 00 00 00
0100h: D8 00 00 00 00 00 00 C8 2D 74 10 E
0110h: F3 AB DD D7 E0 C0 8B 6F 94 69 5E 04
0120h: 0E 48 CF 15 79 02 90 56 5F 00 00 00
0130h: 03 00 00 00 01 00 00 00 00 00 00 00 02
0140h: 04 00 00 00 00 00 00 20 D9 D8 89 5B FF
0150h: 00 00 00 00 FC 00 00 00 D8 00 00 00 00 00
0160h: 9C C5 9D CD BD 53 1A 48 FC 8D AE 4B 7F FA 15 C6
0170h: CE E8 EE 8C 6A 12 9B 06 B4 C6 CD FF 0B 9D B0 11
0180h: 66 00 00 00 00 00 00 00 04 00 00 00 01 00 00 00
0190h: 00 00 00 00 02 00 00 00 05 00 00 00 00 00 00 20
01A0h: DA D8 89 5B FF FF 7F 20 00 00 00 00 FC 00 00 00
01B0h: D8 00 00 00 00 00 00 C8 6B 37 80 FC F8 58 CE 05
01C0h: 91 70 07 D3 2F 85 69 CE C6 D1 4D F2 37 73 0D 76
01D0h: 1F B5 28 67 41 FC 37 65 6D 00 00 00 00 00 00 00
01E0h: 05 00 00 00 01 00 00 00 00 00 00 00 02 00 00 00
01F0h: 06 00 00 00 00 00 00 20 DA D8 89 5B FF FF 7F 20
0200h: 00 00 00 00 FC 00 00 00 D8 00 00 00 00 00 00 C8
```

```
struct RawBlock {
    uint32_t firstTxIndex;     // offset: 0 bytes
    uint32_t numTxes;          // offset: 4 bytes
    uint32_t height;           // offset: 8 bytes
    uint256 hash;              // offset: 12 bytes
    int32_t version;           // offset: 44 bytes
    uint32_t timestamp;        // offset: 48 bytes
    uint32_t bits;             // offset: 52 bytes
    uint32_t nonce;            // offset: 56 bytes
    uint32_t realSize;         // offset: 60 bytes
    uint32_t baseSize;         // offset: 64 bytes
    uint64_t coinbaseOffset;   // offset: 68 bytes
                               // total size: 76 bytes
} // padded size: 80 bytes
```

Variables

Name	Value	Start	Size	Color
uint32 baseSize	216	A8h	4	Fg: Bg:
uint32 padding	3355443200	ACh	4	Fg: Bg:
struct RawBlock blocks[2]		B0h	88	Fg: Bg:
struct RawBlock blocks[3]		108h	88	Fg: Bg:
struct RawBlock blocks[4]		160h	88	Fg: Bg:
struct RawBlock blocks[5]		1B8h	88	Fg: Bg:
struct RawBlock blocks[6]		210h	88	Fg: Bg:
struct RawBlock blocks[7]		268h	88	Fg: Bg:
struct RawBlock blocks[8]		2C0h	88	Fg: Bg:
struct RawBlock blocks[9]		318h	88	Fg: Bg:
struct RawBlock blocks[10]		370h	88	Fg: Bg:
struct RawBlock blocks[11]		3C8h	88	Fg: Bg:
struct RawBlock blocks[12]		420h	88	Fg: Bg:
struct RawBlock blocks[13]		478h	88	Fg: Bg:

block.dat ×

Edit As: Hex Run Script Run Template: block.dat.bt

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000h:	06	22	6E	46	11	1A	0B	59	CA	AF	12	60	43	EB	5B	BF
0010h:	28	C3	4F	3A	5E	33	2A	1F	C7	B2	B7	3C	F1	88	91	0F
0020h:	00	00	00	00	00	00	00	00	00	00	00	00	01	00	00	00
0030h:	00	00	00	00	01	00	00	00	01	00	00	00	01	00	00	00
0040h:	DA	E5	49	4D	FF	FF	7F	20	02	00	00	00	1D	01	00	00
0050h:	1D	01	00	00	00	00	00	C8	A5	94	09	48	4F	D0	EA	FD
0060h:	03	5B	45	DF	90	60	16	C7	8B	F8	98	38	82	C1	EE	D9
0070h:	D9	BD	E9	70	02	06	FD	66	51	00	00	00	00	00	00	00
0080h:	01	00	00	00	01	00	00	00	00	00	00	00	02	00	00	00
0090h:	02	00	00	00	00	00	00	20	D8	D8	89	5B	FF	FF	7F	20
00A0h:	01	00	00	00	FC	00	00	00	D8	00	00	00	00	00	00	C8
00B0h:	3E	42	73	08	1D	22	78	49	64	86	E6	1E	E4	33	28	71
00C0h:	BB	94	3A	EF	56	63	DA	49	65	51	B0	1E	2D	D3	A4	29
00D0h:	58	00	00	00	00	00	00	00	02	00	00	00	01	00	00	00
00E0h:	00	00	00	00	02	00	00	00	03	00	00	00	00	00	00	20
00F0h:	D9	D8	89	5B	FF	FF	7F	20	00	00	00	00	FC	00	00	00
0100h:	D8	00	00	00	00	00	00	C8	2D	74	10	E8	D7	BF	4F	C7
0110h:	F3	AB	DD	D7	E0	C0	8B	6F	94	69	5E	04	E7	99	AD	19
0120h:	0E	48	CF	15	79	02	90	56	5F	00	00	00	00	00	00	00
0130h:	03	00	00	00	01	00	00	00	00	00	00	00	02	00	00	00
0140h:	04	00	00	00	00	00	00	20	D9	D8	89	5B	FF	FF	7F	20
0150h:	00	00	00	00	FC	00	00	00	D8	00	00	00	00	00	00	C8
0160h:	9C	C5	9D	CD	BD	53	1A	48	FC	8D	AE	4B	7F	FA	15	C6
0170h:	CE	E8	EE	8C	6A	12	9B	06	B4	C6	CD	FF	0B	9D	B0	11
0180h:	66	00	00	00	00	00	00	00	04	00	00	00	01	00	00	00
0190h:	00	00	00	00	02	00	00	00	05	00	00	00	00	00	00	20
01A0h:	DA	D8	89	5B	FF	FF	7F	20	00	00	00	00	FC	00	00	00
01B0h:	D8	00	00	00	00	00	00	C8	6B	37	80	FC	F8	58	CE	05
01C0h:	91	70	07	D3	2F	85	69	CE	C6	D1	4D	F2	37	73	0D	76
01D0h:	1F	B5	28	67	41	FC	37	65	6D	00	00	00	00	00	00	00
01E0h:	05	00	00	00	01	00	00	00	00	00	00	00	02	00	00	00
01F0h:	06	00	00	00	00	00	00	20	DA	D8	89	5B	FF	FF	7F	20
0200h:	00	00	00	00	FC	00	00	00	D8	00	00	00	00	00	00	C8

Variables

Name	Value	Start	Size	Color
struct RawBlock blocks[80]		0h	7040	Fg: Bg:
struct RawBlock blocks[0]		0h	88	Fg: Bg:
int64 hash[4]		0h	32	Fg: Bg:
uint64 coinbaseOffset	0	20h	8	Fg: Bg:
uint32 firstTxIndex	0	28h	4	Fg: Bg:
uint32 txCount	1	2Ch	4	Fg: Bg:
uint32 inputCount	0	30h	4	Fg: Bg:
uint32 outputCount	1	34h	4	Fg: Bg:
uint32 height	1	38h	4	Fg: Bg:
int32 version	1	3Ch	4	Fg: Bg:
uint32 timestamp	1296688602	40h	4	Fg: Bg:
uint32 bits	545259519	44h	4	Fg: Bg:
uint32 nonce	2	48h	4	Fg: Bg:
uint32 realSize	285	4Ch	4	Fg: Bg:
uint32 baseSize	285	50h	4	Fg: Bg:
uint32 padding	3355443200	54h	4	Fg: Bg:
struct RawBlock blocks[1]		58h	88	Fg: Bg:
int64 hash[4]		58h	32	Fg: Bg:
uint64 coinbaseOffset	81	78h	8	Fg: Bg:
uint32 firstTxIndex	1	80h	4	Fg: Bg:
uint32 txCount	1	84h	4	Fg: Bg:
uint32 inputCount	0	88h	4	Fg: Bg:
uint32 outputCount	2	8Ch	4	Fg: Bg:
uint32 height	2	90h	4	Fg: Bg:
int32 version	536870912	94h	4	Fg: Bg:
uint32 timestamp	1535760600	98h	4	Fg: Bg:
uint32 bits	545259519	9Ch	4	Fg: Bg:
uint32 nonce	1	A0h	4	Fg: Bg:
uint32 realSize	252	A4h	4	Fg: Bg:
uint32 baseSize	216	A8h	4	Fg: Bg:
uint32 padding	3355443200	ACh	4	Fg: Bg:
struct RawBlock blocks[2]		B0h	88	Fg: Bg:
struct RawBlock blocks[3]		108h	88	Fg: Bg:
struct RawBlock blocks[4]		160h	88	Fg: Bg:
struct RawBlock blocks[5]		1B8h	88	Fg: Bg:
struct RawBlock blocks[6]		210h	88	Fg: Bg:
struct RawBlock blocks[7]		268h	88	Fg: Bg:
struct RawBlock blocks[8]		2C0h	88	Fg: Bg:
struct RawBlock blocks[9]		318h	88	Fg: Bg:
struct RawBlock blocks[10]		370h	88	Fg: Bg:
struct RawBlock blocks[11]		3C8h	88	Fg: Bg:
struct RawBlock blocks[12]		420h	88	Fg: Bg:
struct RawBlock blocks[13]		478h	88	Fg: Bg:

block.dat ×

Edit As: Hex Run Script Run Template: block.dat.bt

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000h:	06	22	6E	46	11	1A	0B	59	CA	AF	12	60	43	EB	5B	BF
0010h:	28	C3	4F	3A	5E	33	2A	1F	C7	B2	B7	3C	F1	88	91	0F
0020h:	00	00	00	00	00	00	00	00	00	00	00	00	01	00	00	00
0030h:	00	00	00	00	01	00	00	00	01	00	00	00	01	00	00	00
0040h:	DA	E5	49	4D	FF	FF	7F	20	02	00	00	00	1D	01	00	00
0050h:	1D	01	00	00	00	00	00	C8	A5	94	09	48	4F	D0	EA	FD
0060h:	03	5B	45	DF	90	60	16	C7	8B	F8	98	38	82	C1	EE	D9
0070h:	D9	BD	E9	70	02	06	FD	66	51	00	00	00	00	00	00	00
0080h:	01	00	00	00	01	00	00	00	00	00	00	00	02	00	00	00
0090h:	02	00	00	00	00	00	00	20	D8	D8	89	5B	FF	FF	7F	20
00A0h:	01	00	00	00	FC	00	00	00	D8	00	00	00	00	00	00	C8
00B0h:	3E	42	73	08	1D	22	78	49	64	86	E6	1E	E4	33	28	71
00C0h:	BB	94	3A	EF	56	63	DA	49	65	51	B0	1E	2D	D3	A4	29
00D0h:	58	00	00	00	00	00	00	00	02	00	00	00	01	00	00	00
00E0h:	00	00	00	00	02	00	00	00	03	00	00	00	00	00	00	20
00F0h:	D9	D8	89	5B	FF	FF	7F	20	00	00	00	00	FC	00	00	00
0100h:	D8	00	00	00	00	00	00	C8	2D	74	10	E8	D7	BF	4F	C7
0110h:	F3	AB	DD	D7	E0	C0	8B	6F	94	69	5E	04	E7	99	AD	19
0120h:	0E	48	CF	15	79	02	90	56	5F	00	00	00	00	00	00	00
0130h:	03	00	00	00	01	00	00	00	00	00	00	00	02	00	00	00
0140h:	04	00	00	00	00	00	00	20	D9	D8	89	5B	FF	FF	7F	20
0150h:	00	00	00	00	FC	00	00	00	D8	00	00	00	00	00	00	C8
0160h:	9C	C5	9D	CD	BD	53	1A	48	FC	8D	AE	4B	7F	FA	15	C6
0170h:	CE	E8	EE	8C	6A	12	9B	06	B4	C6	CD	FF	0B	9D	B0	11
0180h:	66	00	00	00	00	00	00	00	04	00	00	00	01	00	00	00
0190h:	00	00	00	00	02	00	00	00	05	00	00	00	00	00	00	20
01A0h:	DA	D8	89	5B	FF	FF	7F	20	00	00	00	00	FC	00	00	00
01B0h:	D8	00	00	00	00	00	00	C8	6B	37	80	FC	F8	58	CE	05
01C0h:	91	70	07	D3	2F	85	69	CE	C6	D1	4D	F2	37	73	0D	76
01D0h:	1F	B5	28	67	41	FC	37	65	6D	00	00	00	00	00	00	00
01E0h:	05	00	00	00	01	00	00	00	00	00	00	00	02	00	00	00
01F0h:	06	00	00	00	00	00	00	20	DA	D8	89	5B	FF	FF	7F	20
0200h:	00	00	00	00	FC	00	00	00	D8	00	00	00	00	00	00	C8

Variables

Name	Value	Start	Size	Color
struct RawBlock blocks[80]		0h	7040	Fg: Bg:
struct RawBlock blocks[0]		0h	88	Fg: Bg:
int64 hash[4]		0h	32	Fg: Bg:
uint64 coinbaseOffset	0	20h	8	Fg: Bg:
uint32 firstTxIndex	0	28h	4	Fg: Bg:
uint32 txCount	1	2Ch	4	Fg: Bg:
uint32 inputCount	0	30h	4	Fg: Bg:
uint32 outputCount	1	34h	4	Fg: Bg:
uint32 height	1	38h	4	Fg: Bg:
int32 version	1	3Ch	4	Fg: Bg:
uint32 timestamp	1296688602	40h	4	Fg: Bg:
uint32 bits	545259519	44h	4	Fg: Bg:
uint32 nonce	2	48h	4	Fg: Bg:
uint32 realSize	285	4Ch	4	Fg: Bg:
uint32 baseSize	285	50h	4	Fg: Bg:
uint32 padding	3355443200	54h	4	Fg: Bg:
struct RawBlock blocks[1]		58h	88	Fg: Bg:
int64 hash[4]		58h	32	Fg: Bg:
uint64 coinbaseOffset	81	78h	8	Fg: Bg:
uint32 firstTxIndex	1	80h	4	Fg: Bg:
uint32 txCount	1	84h	4	Fg: Bg:
uint32 inputCount	0	88h	4	Fg: Bg:
uint32 outputCount	2	8Ch	4	Fg: Bg:
uint32 height	2	90h	4	Fg: Bg:
int32 version	536870912	94h	4	Fg: Bg:
uint32 timestamp	1535760600	98h	4	Fg: Bg:
uint32 bits	545259519	9Ch	4	Fg: Bg:
uint32 nonce	1	A0h	4	Fg: Bg:
uint32 realSize	252	A4h	4	Fg: Bg:
uint32 baseSize	216	A8h	4	Fg: Bg:
uint32 padding	3355443200	ACh	4	Fg: Bg:
struct RawBlock blocks[2]		B0h	88	Fg: Bg:
struct RawBlock blocks[3]		108h	88	Fg: Bg:
struct RawBlock blocks[4]		160h	88	Fg: Bg:
struct RawBlock blocks[5]		1B8h	88	Fg: Bg:
struct RawBlock blocks[6]		210h	88	Fg: Bg:
struct RawBlock blocks[7]		268h	88	Fg: Bg:
struct RawBlock blocks[8]		2C0h	88	Fg: Bg:
struct RawBlock blocks[9]		318h	88	Fg: Bg:
struct RawBlock blocks[10]		370h	88	Fg: Bg:
struct RawBlock blocks[11]		3C8h	88	Fg: Bg:
struct RawBlock blocks[12]		420h	88	Fg: Bg:
struct RawBlock blocks[13]		478h	88	Fg: Bg:

block.dat ×

Edit As: Hex Run Script Run Template: block.dat.bt

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000h:	06	22	6E	46	11	1A	0B	59	CA	AF	12	60	43	EB	5B	BF
0010h:	28	C3	4F	3A	5E	33	2A	1F	C7	B2	B7	3C	F1	88	91	0F
0020h:	00	00	00	00	00	00	00	00	00	00	00	00	01	00	00	00
0030h:	00	00	00	00	01	00	00	00	01	00	00	00	01	00	00	00
0040h:	DA	E5	49	4D	FF	FF	7F	20	02	00	00	00	1D	01	00	00
0050h:	1D	01	00	00	00	00	00	C8	A5	94	09	48	4F	D0	EA	FD
0060h:	03	5B	45	DF	90	60	16	C7	8B	F8	98	38	82	C1	EE	D9
0070h:	D9	BD	E9	70	02	06	FD	66	51	00	00	00	00	00	00	00
0080h:	01	00	00	00	01	00	00	00	00	00	00	00	02	00	00	00
0090h:	02	00	00	00	00	00	00	20	D8	D8	89	5B	FF	FF	7F	20
00A0h:	01	00	00	00	FC	00	00	00	D8	00	00	00	00	00	00	C8
00B0h:	3E	42	73	08	1D	22	78	49	64	86	E6	1E	E4	33	28	71
00C0h:	BB	94	3A	EF	56	63	DA	49	65	51	B0	1E	2D	D3	A4	29
00D0h:	58	00	00	00	00	00	00	00	02	00	00	00	01	00	00	00
00E0h:	00	00	00	00	02	00	00	00	03	00	00	00	00	00	00	20
00F0h:	D9	D8	89	5B	FF	FF	7F	20	00	00	00	00	FC	00	00	00
0100h:	D8	00	00	00	00	00	00	C8	2D	74	10	E8	D7	BF	4F	C7
0110h:	F3	AB	DD	D7	E0	C0	8B	6F	94	69	5E	04	E7	99	AD	19
0120h:	0E	48	CF	15	79	02	90	56	5F	00	00	00	00	00	00	00
0130h:	03	00	00	00	01	00	00	00	00	00	00	00	02	00	00	00
0140h:	04	00	00	00	00	00	00	20	D9	D8	89	5B	FF	FF	7F	20
0150h:	00	00	00	00	FC	00	00	00	D8	00	00	00	00	00	00	C8
0160h:	9C	C5	9D	CD	BD	53	1A	48	FC	8D	AE	4B	7F	FA	15	C6
0170h:	CE	E8	EE	8C	6A	12	9B	06	B4	C6	CD	FF	0B	9D	B0	11
0180h:	66	00	00	00	00	00	00	00	04	00	00	00	01	00	00	00
0190h:	00	00	00	00	02	00	00	00	05	00	00	00	00	00	00	20
01A0h:	DA	D8	89	5B	FF	FF	7F	20	00	00	00	00	FC	00	00	00
01B0h:	D8	00	00	00	00	00	00	C8	6B	37	80	FC	F8	58	CE	05
01C0h:	91	70	07	D3	2F	85	69	CE	C6	D1	4D	F2	37	73	0D	76
01D0h:	1F	B5	28	67	41	FC	37	65	6D	00	00	00	00	00	00	00
01E0h:	05	00	00	00	01	00	00	00	00	00	00	00	02	00	00	00
01F0h:	06	00	00	00	00	00	00	20	DA	D8	89	5B	FF	FF	7F	20
0200h:	00	00	00	00	FC	00	00	00	D8	00	00	00	00	00	00	C8

Variables

Name	Value	Start	Size	Color
struct RawBlock blocks[80]		0h	7040	Fg: Bg:
struct RawBlock blocks[0]		0h	88	Fg: Bg:
int64 hash[4]		0h	32	Fg: Bg:
uint64 coinbaseOffset	0	20h	8	Fg: Bg:
uint32 firstTxIndex	0	28h	4	Fg: Bg:
uint32 txCount	1	2Ch	4	Fg: Bg:
uint32 inputCount	0	30h	4	Fg: Bg:
uint32 outputCount	1	34h	4	Fg: Bg:
uint32 height	1	38h	4	Fg: Bg:
int32 version	1	3Ch	4	Fg: Bg:
uint32 timestamp	1296688602	40h	4	Fg: Bg:
uint32 bits	545259519	44h	4	Fg: Bg:
uint32 nonce	2	48h	4	Fg: Bg:
uint32 realSize	285	4Ch	4	Fg: Bg:
uint32 baseSize	285	50h	4	Fg: Bg:
uint32 padding	3355443200	54h	4	Fg: Bg:
struct RawBlock blocks[1]		58h	88	Fg: Bg:
int64 hash[4]		58h	32	Fg: Bg:
uint64 coinbaseOffset	81	78h	8	Fg: Bg:
uint32 firstTxIndex	1	80h	4	Fg: Bg:
uint32 txCount	1	84h	4	Fg: Bg:
uint32 inputCount	0	88h	4	Fg: Bg:
uint32 outputCount	2	8Ch	4	Fg: Bg:
uint32 height	2	90h	4	Fg: Bg:
int32 version	536870912	94h	4	Fg: Bg:
uint32 timestamp	1535760600	98h	4	Fg: Bg:
uint32 bits	545259519	9Ch	4	Fg: Bg:
uint32 nonce	1	A0h	4	Fg: Bg:
uint32 realSize	252	A4h	4	Fg: Bg:
uint32 baseSize	216	A8h	4	Fg: Bg:
uint32 padding	3355443200	ACh	4	Fg: Bg:
struct RawBlock blocks[2]		B0h	88	Fg: Bg:
struct RawBlock blocks[3]		108h	88	Fg: Bg:
struct RawBlock blocks[4]		160h	88	Fg: Bg:
struct RawBlock blocks[5]		1B8h	88	Fg: Bg:
struct RawBlock blocks[6]		210h	88	Fg: Bg:
struct RawBlock blocks[7]		268h	88	Fg: Bg:
struct RawBlock blocks[8]		2C0h	88	Fg: Bg:
struct RawBlock blocks[9]		318h	88	Fg: Bg:
struct RawBlock blocks[10]		370h	88	Fg: Bg:
struct RawBlock blocks[11]		3C8h	88	Fg: Bg:
struct RawBlock blocks[12]		420h	88	Fg: Bg:
struct RawBlock blocks[13]		478h	88	Fg: Bg:

BlockSci: Funktionsweise

Hoch-optimierte Datenstruktur

Bitcoin

blockFile	RawBlock | RawBlock | RawBlock | RawBlock | RawBlock, ...
txFile	RawTransaction | RawTransaction | RawTransaction, ...

Für Analysen: Laden in den Arbeitsspeicher

Hoch-optimierte Datenstruktur

Bitcoin

blockFile	RawBlock | RawBlock | RawBlock | RawBlock | RawBlock, ...
txFile	RawTransaction | RawTransaction | RawTransaction, ...

BlockSci Parser

Blocks, Transaktionen etc.

Nicht Teil von BlockSci

Bitcoin

Netzwerk

Bitcoin

Software

BlockSci: Funktionsweise

Hoch-optimierte Datenstruktur

blockFile | RawBlock | RawBlock | RawBlock | RawBlock | RawBlock, ...

txFile | RawTransaction | RawTransaction | RawTransaction | RawTransaction, ...

Für Analysen: Laden in den Arbeitsspeicher

Hoch-optimierte Datenstruktur

blockFile | RawBlock | RawBlock | RawBlock | RawBlock | RawBlock, ...

txFile | RawTransaction | RawTransaction | RawTransaction | RawTransaction, ...

```
struct RawTransaction {
    uint32_t realSize;     // offset: 0 bytes
    uint32_t baseSize;     // offset: 4 bytes
    uint32_t locktime;     // offset: 8 bytes
    uint16_t inputCount;   // offset: 12 bytes
    uint16_t outputCount;  // offset: 14 bytes
};  // total size: 16 bytes

// followed by all inputs and outputs of that transaction

struct Inout {
    private:
        uint32_t linkedTxNum;
        uint32_t toAddressNum;
        uint64_t other;
};
```

Nicht Teil von BlockSci

Netzwerk

Speicherung

BlockSci Parser

Blocks, Transaktionen etc.

BlockSci: Funktionsweise

Hoch-optimierte Datenstruktur

Bitcoin

blockFile	RawBlock | RawBlock | RawBlock | RawBlock | RawBlock, ...
txFile	RawTransaction | RawTransaction | RawTransaction, ...

Für Analysen: Laden in den Arbeitsspeicher

Hoch-optimierte Datenstruktur

Bitcoin

blockFile	RawBlock | RawBlock | RawBlock | RawBlock | RawBlock, ...
txFile	RawTransaction | RawTransaction | RawTransaction, ...

BlockSci Parser

Blocks, Transaktionen etc.

Nicht Teil von BlockSci

Bitcoin

Netzwerk

Bitcoin

Software

BlockSci: Funktionsweise

Nicht Teil von BlockSci

```
struct RawBlock {
    uint32_t firstTxIndex;      // offset: 0 bytes
    uint32_t numTxes;           // offset: 4 bytes
    uint32_t height;            // offset: 8 bytes
    uint256 hash;               // offset: 12 bytes
    int32_t version;            // offset: 44 bytes
    uint32_t timestamp;         // offset: 48 bytes
    uint32_t bits;              // offset: 52 bytes
    uint32_t nonce;             // offset: 56 bytes
    uint32_t realSize;          // offset: 60 bytes
    uint32_t baseSize;          // offset: 64 bytes
    uint64_t coinbaseOffset;    // offset: 68 bytes
                                // total size: 76 bytes

} // padded size: 80 bytes

while (blockFile.hasBlock()) {
    txArray.add(new RawBlock(blockFile.readBlock());
}
```

Hoch-optimierte Datenstruktur

blockFile: RawBlock | RawBlock | RawBlock | RawBlock | RawBlock, ...

txFile: RawTransaction | RawTransaction | RawTransaction, ...

Für Analysen: Laden in der Hoch-optimierten Struktur

Hoch-optimierte Datenstruktur

blockFile: RawBlock | RawBlock | RawBlock | RawBlock | RawBlock, ...

txFile: RawTransaction | RawTransaction | RawTransaction, ...

BlockSci Parser

Blocks, Transaktionen etc.

BlockSci: Funktionsweise

Nicht Teil von BlockSci

Hoch-optimierte Datenstruktur

blockFile | RawBlock | RawBlock | RawBlock | RawBlock | RawBlock, ...

txFile | RawTransaction | RawTransaction | RawTransaction, ...

```
struct RawBlock {
    uint32_t firstTxIndex;      // offset: 0 bytes
    uint32_t numTxes;           // offset: 4 bytes
    uint32_t height;            // offset: 8 bytes
    uint256 hash;               // offset: 12 bytes
    int32_t version;            // offset: 44 bytes
    uint32_t timestamp;         // offset: 48 bytes
    uint32_t bits;              // offset: 52 bytes
    uint32_t nonce;             // offset: 56 bytes
    uint32_t realSize;          // offset: 60 bytes
    uint32_t baseSize;          // offset: 64 bytes
    uint64_t coinbaseOffset;    // offset: 68 bytes
                                // total size: 76 bytes

} // padded size: 80 bytes
```

~~while (blockFile.hasBlock()) {~~
~~txArray.add(new RawBlock(blockFile.readBlock());~~
~~}~~

Für Analysen: Laden in der hochoptimierten Struktur

Hoch-optimierte Datenstruktur

blockFile | RawBlock | RawBlock | RawBlock | RawBlock | RawBlock, ...

txFile | RawTransaction | RawTransaction | RawTransaction, ...

BlockSci Parser

Blocks, Transaktionen etc.

Exkurs II

Memory-Mapped Files: `mmap()`

NAME

top

mmap, munmap - map or unmap files or devices into memory

SYNOPSIS

top

```
#include <sys/mman.h>

void *mmap(void *addr, size_t length, int prot, int flags,
           int fd, off_t offset);
int munmap(void *addr, size_t length);
```

See NOTES for information on feature test macro requirements.

DESCRIPTION

top

mmap() creates a new mapping in the virtual address space of the calling process. The starting address for the new mapping is specified in *addr*. The *length* argument specifies the length of the mapping (which must be greater than 0).

If *addr* is NULL, then the kernel chooses the (page-aligned) address at which to create the mapping; this is the most portable method of creating a new mapping. If *addr* is not NULL, then the kernel takes it as a hint about where to place the mapping; on Linux, the kernel will pick a nearby page boundary (but always above or equal to the value specified by */proc/sys/vm/mmap_min_addr*) and attempt to create the mapping there. If another mapping already exists there, the kernel picks a new address that may or may not depend on the hint. The address of the new mapping is returned as the result of the call.

The contents of a file mapping (as opposed to an anonymous mapping; see **MAP_ANONYMOUS** below), are initialized using *length* bytes starting at offset *offset* in the file (or other object) referred to by the file descriptor *fd*. *offset* must be a multiple of the page size as returned by *sysconf(_SC_PAGE_SIZE)*.

NAME

top

mmap, munmap - map or unmap files or devices into memory

SYNOPSIS

top

```
#include <sys/mman.h>

void *mmap(void *addr, size_t length, int prot, int flags,
           int fd, off_t offset);
int munmap(void *addr, size_t length);
```

See NOTES for information on feature test macro requirements.

DESCRIPTION

top

mmap() creates a new mapping in the virtual address space of the calling process. The starting address for the new mapping is specified in *addr*. The *length* argument specifies the length of the mapping (which must be greater than 0).

If *addr* is NULL, then the kernel chooses the (page-aligned) address at which to create the mapping; this is the most portable method of creating a new mapping. If *addr* is not NULL, then the kernel takes it as a hint about where to place the mapping; on Linux, the kernel will pick a nearby page boundary (but always above or equal to the value specified by */proc/sys/vm/mmap_min_addr*) and attempt to create the mapping there. If another mapping already exists there, the kernel picks a new address that may or may not depend on the hint. The address of the new mapping is returned as the result of the call.

The contents of a file mapping (as opposed to an anonymous mapping; see **MAP_ANONYMOUS** below), are initialized using *length* bytes starting at offset *offset* in the file (or other object) referred to by the file descriptor *fd*. *offset* must be a multiple of the page size as returned by *sysconf(_SC_PAGE_SIZE)*.

NAME

top

mmap, munmap - map or unmap files or devices into memory

SYNOPSIS

top

```
#include <sys/mman.h>

void *mmap(void *addr, size_t length, int prot, int flags,
           int fd, off_t offset);
int munmap(void *addr, size_t length);
```

See NOTES for information on feature test macro requirements.

DESCRIPTION

top

mmap() creates a new mapping in the virtual address space of the calling process. The starting address for the new mapping is specified in *addr*. The *length* argument specifies the length of the mapping (which must be greater than 0).

If *addr* is NULL, then the kernel chooses the (page-aligned) address at which to create the mapping; this is the most portable method of creating a new mapping. If *addr* is not NULL, then the kernel takes it as a hint about where to place the mapping; on Linux, the kernel will pick a nearby page boundary (but always above or equal to the value specified by */proc/sys/vm/mmap_min_addr*) and attempt to create the mapping there. If another mapping already exists there, the kernel picks a new address that may or may not depend on the hint. The address of the new mapping is returned as the result of the call.

The contents of a file mapping (as opposed to an anonymous mapping; see **MAP_ANONYMOUS** below), are initialized using *length* bytes starting at offset *offset* in the file (or other object) referred to by the file descriptor *fd*. *offset* must be a multiple of the page size as returned by *sysconf(_SC_PAGE_SIZE)*.

NAME top

mmap, munmap - map or unmap files or devices into memory

SYNOPSIS top

```
#include <sys/mman.h>

void *mmap(void *addr, size_t length, int prot, int flags,
           int fd, off_t offset);
int munmap(void *addr, size_t length);
```

See NOTES for information on feature test macro requirements.

DESCRIPTION top

mmap() creates a new mapping in the virtual address space of the calling process. The starting address for the new mapping is specified in *addr*. The *length* argument specifies the length of the mapping (which must be greater than 0).

If *addr* is NULL, then the kernel chooses the (page-aligned) address at which to create the mapping; this is the most portable method of creating a new mapping. If *addr* is not NULL, then the kernel takes it as a hint about where to place the mapping; on Linux, the kernel will pick a nearby page boundary (but always above or equal to the value specified by */proc/sys/vm/mmap_min_addr*) and attempt to create the mapping there. If another mapping already exists there, the kernel picks a new address that may or may not depend on the hint. The address of the new mapping is returned as the result of the call.

The contents of a file mapping (as opposed to an anonymous mapping; see **MAP_ANONYMOUS** below), are initialized using *length* bytes starting at offset *offset* in the file (or other object) referred to by the file descriptor *fd*. *offset* must be a multiple of the page size as returned by *sysconf(_SC_PAGE_SIZE)*.

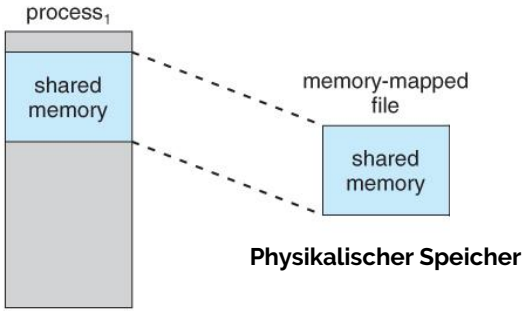
Für Analysen: Laden in den Arbeitsspeicher

Hoch-optimierte Datenstruktur

blockFile	**RawBlock | RawBlock | RawBlock | RawBlock | RawBlock, ...**
txFile	**RawTransaction | RawTransaction | RawTransaction, ...**

Quelle: https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/9_VirtualMemory.html

BlockSci Instanz



`mmap(blockFile)`

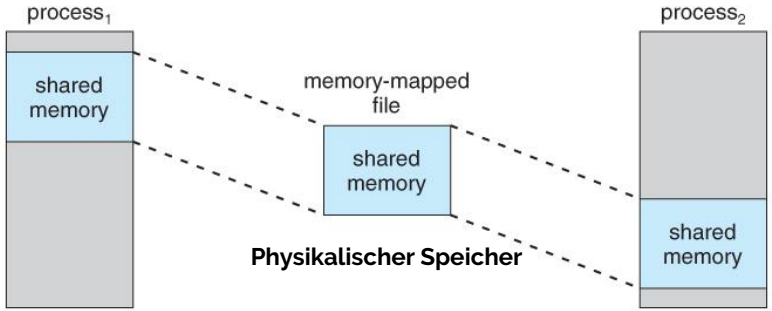
Hoch-optimierte Datenstruktur

blockFile	RawBlock | RawBlock | RawBlock | RawBlock | RawBlock, ...
txFile	RawTransaction | RawTransaction | RawTransaction, ...

Quelle: https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/9_VirtualMemory.html

BlockSci Instanz 1

BlockSci Instanz 2



`mmap(blockFile)`

Hoch-optimierte Datenstruktur

blockFile	**RawBlock | RawBlock | RawBlock | RawBlock | RawBlock, ...**
txFile	**RawTransaction | RawTransaction | RawTransaction, ...**

BlockSci: Funktionsweise

Hoch-optimierte Datenstruktur

Bitcoin

blockFile	RawBlock | RawBlock | RawBlock | RawBlock | RawBlock, ...
txFile	RawTransaction | RawTransaction | RawTransaction, ...

Für Analysen: Laden in den Arbeitsspeicher

Hoch-optimierte Datenstruktur

Bitcoin

blockFile	RawBlock | RawBlock | RawBlock | RawBlock | RawBlock, ...
txFile	RawTransaction | RawTransaction | RawTransaction, ...

BlockSci Parser

Blocks, Transaktionen etc.

Nicht Teil von BlockSci

Bitcoin

Netzwerk

Bitcoin

Software

BlockSci: Funktionsweise

Nicht Teil von BlockSci

```
struct RawBlock {
    uint32_t firstTxIndex;      // offset: 0 bytes
    uint32_t numTxes;           // offset: 4 bytes
    uint32_t height;            // offset: 8 bytes
    uint256 hash;               // offset: 12 bytes
    int32_t version;            // offset: 44 bytes
    uint32_t timestamp;         // offset: 48 bytes
    uint32_t bits;              // offset: 52 bytes
    uint32_t nonce;             // offset: 56 bytes
    uint32_t realSize;          // offset: 60 bytes
    uint32_t baseSize;          // offset: 64 bytes
    uint64_t coinbaseOffset;    // offset: 68 bytes
                                // total size: 76 bytes
} // padded size: 80 bytes

RawBlock* firstBlock = mmap("/btc/chain/block.dat");
```

Hoch-optimierte Datenstruktur

blockFile: RawBlock | RawBlock | RawBlock | RawBlock | RawBlock, ...

txFile: RawTransaction | RawTransaction | RawTransaction, ...

Für Analysen: Laden in den Arbeitsspeicher

Hoch-optimierte Datenstruktur

blockFile: RawBlock | RawBlock | RawBlock | RawBlock | RawBlock, ...

txFile: RawTransaction | RawTransaction | RawTransaction, ...

BlockSci Parser

Blocks, Transaktionen etc.

Nicht Teil von BlockSci

block.dat ×

Edit As: Hex Run Script Run Template: block.dat.bt

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000h:	06	22	6E	46	11	1A	0B	59	CA	AF	12	60	43	EB	5B	BF
0010h:	28	C3	4F	3A	5E	33	2A	1F	C7	B2	B7	3C	F1	88	91	0F
0020h:	00	00	00	00	00	00	00	00	00	00	00	00	01	00	00	00
0030h:	00	00	00	00	01	00	00	00	01	00	00	00	01	00	00	00
0040h:	DA	E5	49	4D	FF	FF	7F	20	02	00	00	00	1D	01	00	00
0050h:	1D	01	00	00	00	00	00	C8	A5	94	09	48	4F	D0	EA	FD
0060h:	03	5B	45	DF	90	60	16	C7	8B	F8	98	38	82	C1	EE	D9
0070h:	D9	BD	E9	70	02	06	FD	66	51	00	00	00	00	00	00	00
0080h:	01	00	00	00	01	00	00	00	00	00	00	00	02	00	00	00
0090h:	02	00	00	00	00	00	00	20	D8	D8	89	5B	FF	FF	7F	20
00A0h:	01	00	00	00	FC	00	00	00	D8	00	00	00	00	00	00	C8

```
struct RawBlock {
    uint32_t firstTxIndex;      // offset: 0 bytes
    uint32_t numTxes;           // offset: 4 bytes
    uint32_t height;            // offset: 8 bytes
    uint256 hash;               // offset: 12 bytes
    int32_t version;            // offset: 44 bytes
    uint32_t timestamp;         // offset: 48 bytes
    uint32_t bits;              // offset: 52 bytes
    uint32_t nonce;             // offset: 56 bytes
    uint32_t realSize;          // offset: 60 bytes
    uint32_t baseSize;          // offset: 64 bytes
    uint64_t coinbaseOffset;    // offset: 68 bytes
                                // total size: 76 bytes

} // padded size: 80 bytes

RawBlock* firstBlock = mmap("/btc/chain/block.dat");
```

Für Analysen: Laden

Hoch-optimierte Datenstruktur

Bitcoin

blockFile	RawBlock | RawBlock | RawBlock | RawBlock | RawBlock, ...
txFile	RawTransaction | RawTransaction | RawTransaction, ...

BlockSci Parser

Blocks, Transaktionen etc.

BlockSci: Funktionsweise

Nicht Teil von BlockSci

Hoch-optimierte Datenstruktur

blockFile | RawBlock | RawBlock | RawBlock | RawBlock | RawBlock, ...

txFile | RawTransaction | RawTransaction | RawTransaction, ...

Für Analysen: Laden in den Arbeitsspeicher

```
struct RawBlock {
    uint32_t firstTxIndex;      // offset: 0 bytes
    uint32_t numTxes;           // offset: 4 bytes
    uint32_t height;            // offset: 8 bytes
    uint256 hash;               // offset: 12 bytes
    int32_t version;            // offset: 44 bytes
    uint32_t timestamp;         // offset: 48 bytes
    uint32_t bits;              // offset: 52 bytes
    uint32_t nonce;             // offset: 56 bytes
    uint32_t realSize;          // offset: 60 bytes
    uint32_t baseSize;          // offset: 64 bytes
    uint64_t coinbaseOffset;    // offset: 68 bytes
                                // total size: 76 bytes

} // padded size: 80 bytes

RawBlock* firstBlock = mmap("/btc/chain/block.dat");
RawBlock* laterBlock = firstBlock + 10000; // offset: 10000 * 80
std::cout << "height: " << laterBlock->height;
```

Hoch-optimierte Datenstruktur

blockFile | RawBlock | RawBlock | RawBlock | RawBlock | RawBlock, ...

txFile | RawTransaction | RawTransaction | RawTransaction, ...

BlockSci Parser

Blocks, Transaktionen etc.

BlockSci: Funktionsweise

Nicht Teil von BlockSci

Hoch-optimierte Datenstruktur

blockFile | RawBlock | RawBlock | RawBlock | RawBlock | RawBlock, ...

txFile | RawTransaction | RawTransaction | RawTransaction, ...

Für Analysen: Laden in den Arbeitsspeicher

```
struct RawBlock {
    uint32_t firstTxIndex;      // offset: 0 bytes
    uint32_t numTxes;           // offset: 4 bytes
    uint32_t height;            // offset: 8 bytes
    uint256 hash;               // offset: 12 bytes
    int32_t version;            // offset: 44 bytes
    uint32_t timestamp;         // offset: 48 bytes
    uint32_t bits;              // offset: 52 bytes
    uint32_t nonce;             // offset: 56 bytes
    uint32_t realSize;          // offset: 60 bytes
    uint32_t baseSize;          // offset: 64 bytes
    uint64_t coinbaseOffset;    // offset: 68 bytes
                                // total size: 76 bytes

} // padded size: 80 bytes

RawBlock* firstBlock = mmap("/btc/chain/block.dat");
RawBlock* laterBlock = firstBlock + 10000; // offset: 10000 * 80
std::cout << "height: " << laterBlock->height;
```

Hoch-optimierte Datenstruktur

blockFile | RawBlock | RawBlock | RawBlock | RawBlock | RawBlock, ...

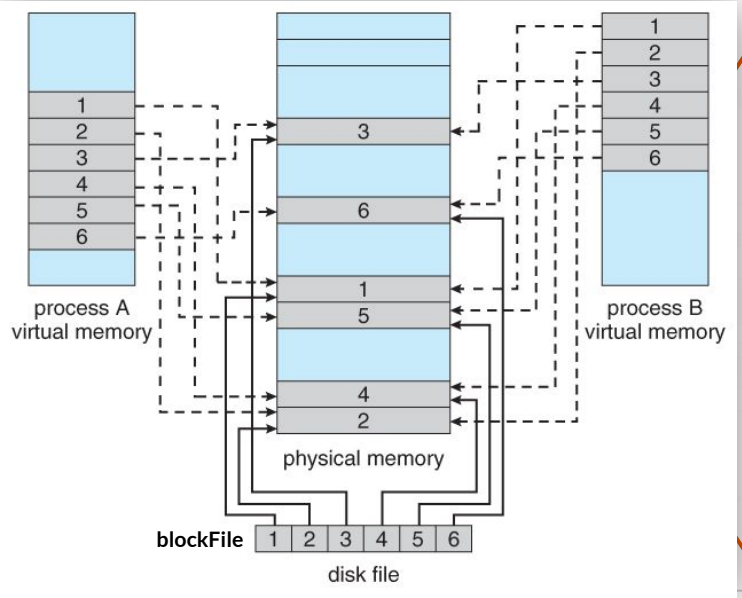
txFile | RawTransaction | RawTransaction | RawTransaction, ...

BlockSci Parser

Blocks, Transaktionen etc.

Quelle: https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/9_VirtualMemory.html

Nicht Teil von BlockSci




```
struct RawBlock {
    uint32_t firstTxIndex;     // offset: 0 bytes
    uint32_t numTxes;          // offset: 4 bytes
    uint32_t height;           // offset: 8 bytes
    uint256 hash;              // offset: 12 bytes
    int32_t version;           // offset: 44 bytes
    uint32_t timestamp;        // offset: 48 bytes
    uint32_t bits;             // offset: 52 bytes
    uint32_t nonce;            // offset: 56 bytes
    uint32_t realSize;         // offset: 60 bytes
    uint32_t baseSize;         // offset: 64 bytes
    uint64_t coinbaseOffset;   // offset: 68 bytes
                               // total size: 76 bytes
} // padded size: 80 bytes

RawBlock* firstBlock = mmap("/btc/chain/block.dat");
RawBlock* laterBlock = firstBlock + 10000; // offset: 10000 * 80
std::cout << "height: " << laterBlock->height;
```

Hoch-optimierte Datenstruktur

blockFile | RawBlock | RawBlock | RawBlock | RawBlock | RawBlock, ...

txFile | RawTransaction | RawTransaction | RawTransaction, ...

BlockSci Parser

Blocks, Transaktionen etc.

BlockSci: Funktionsweise

Hoch-optimierte Datenstruktur

blockFile	RawBlock | RawBlock | RawBlock | RawBlock | RawBlock, ...
txFile	RawTransaction | RawTransaction | RawTransaction, ...

Für Analysen: `mmap()` in den Arbeitsspeicher

Hoch-optimierte Datenstruktur

blockFile	RawBlock | RawBlock | RawBlock | RawBlock | RawBlock, ...
txFile	RawTransaction | RawTransaction | RawTransaction, ...

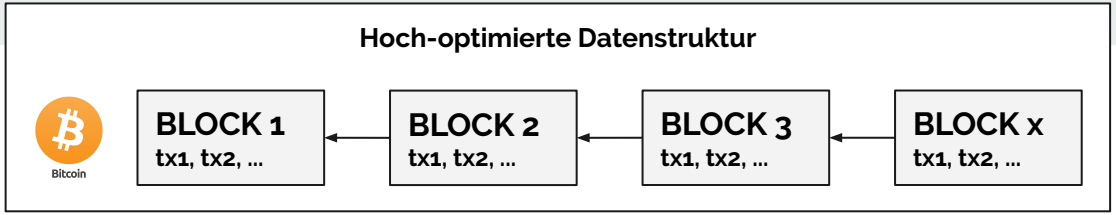
BlockSci Parser

Blocks, Transaktionen etc.

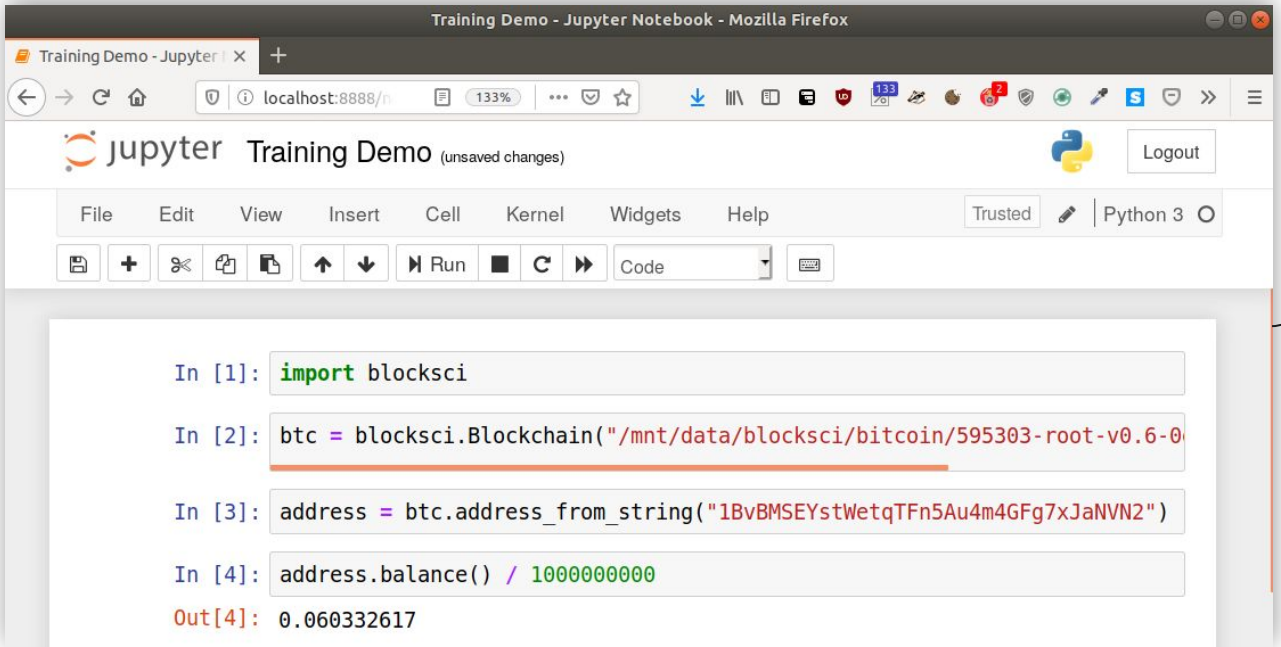
Nicht Teil von BlockSci

Netzwerk

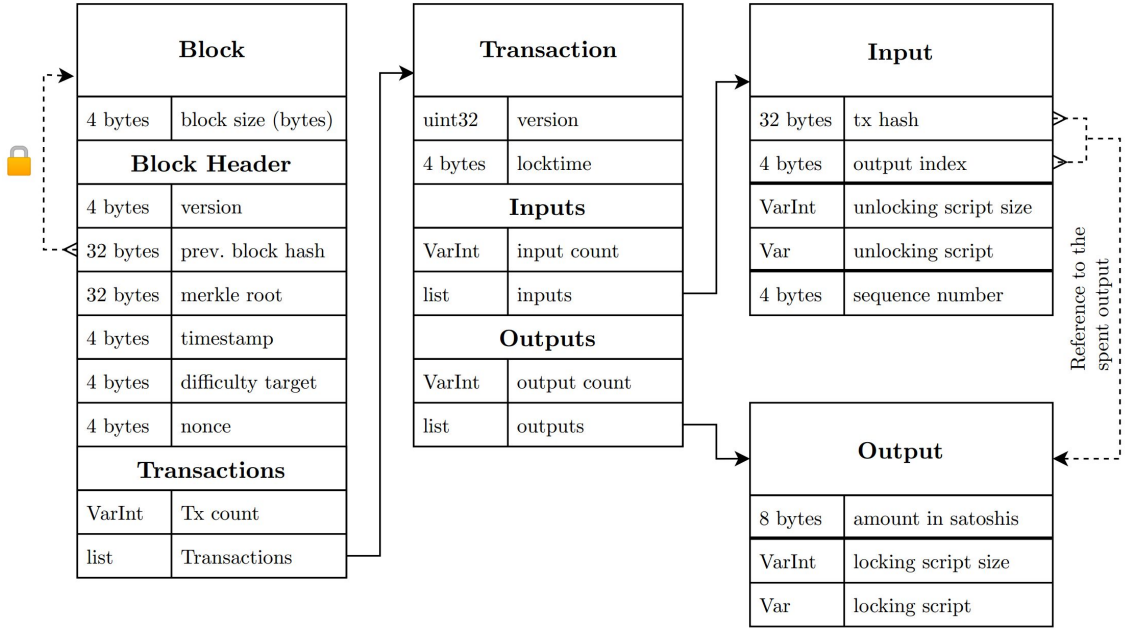
Software



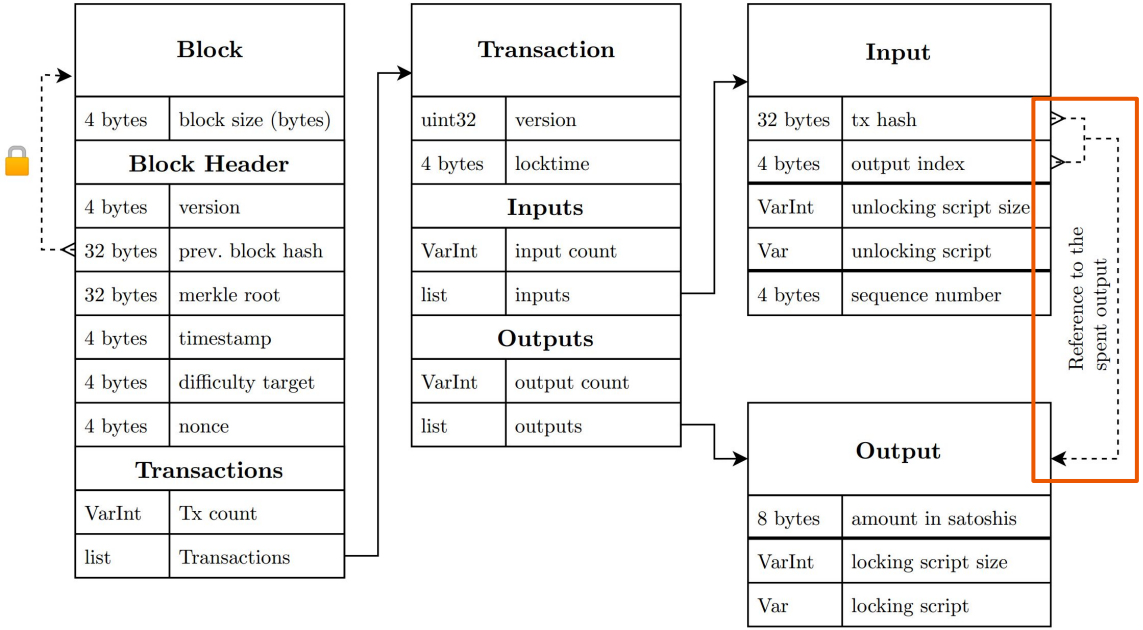
BlockSci: Funktionsweise



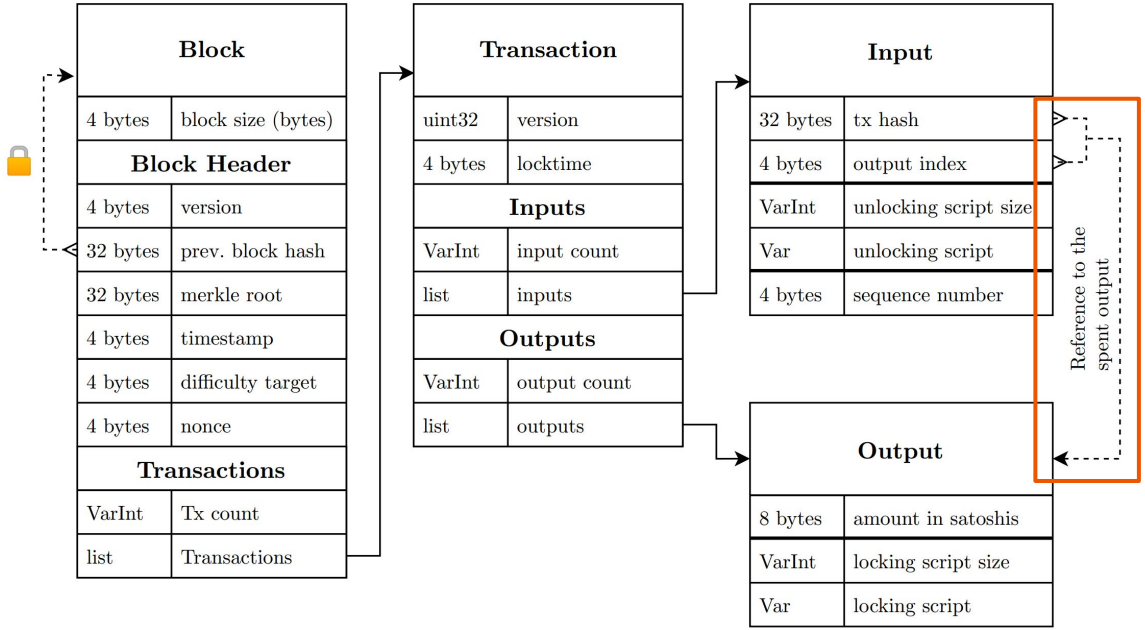
BlockSci: Weitere Optimierungen



BlockSci: Weitere Optimierungen

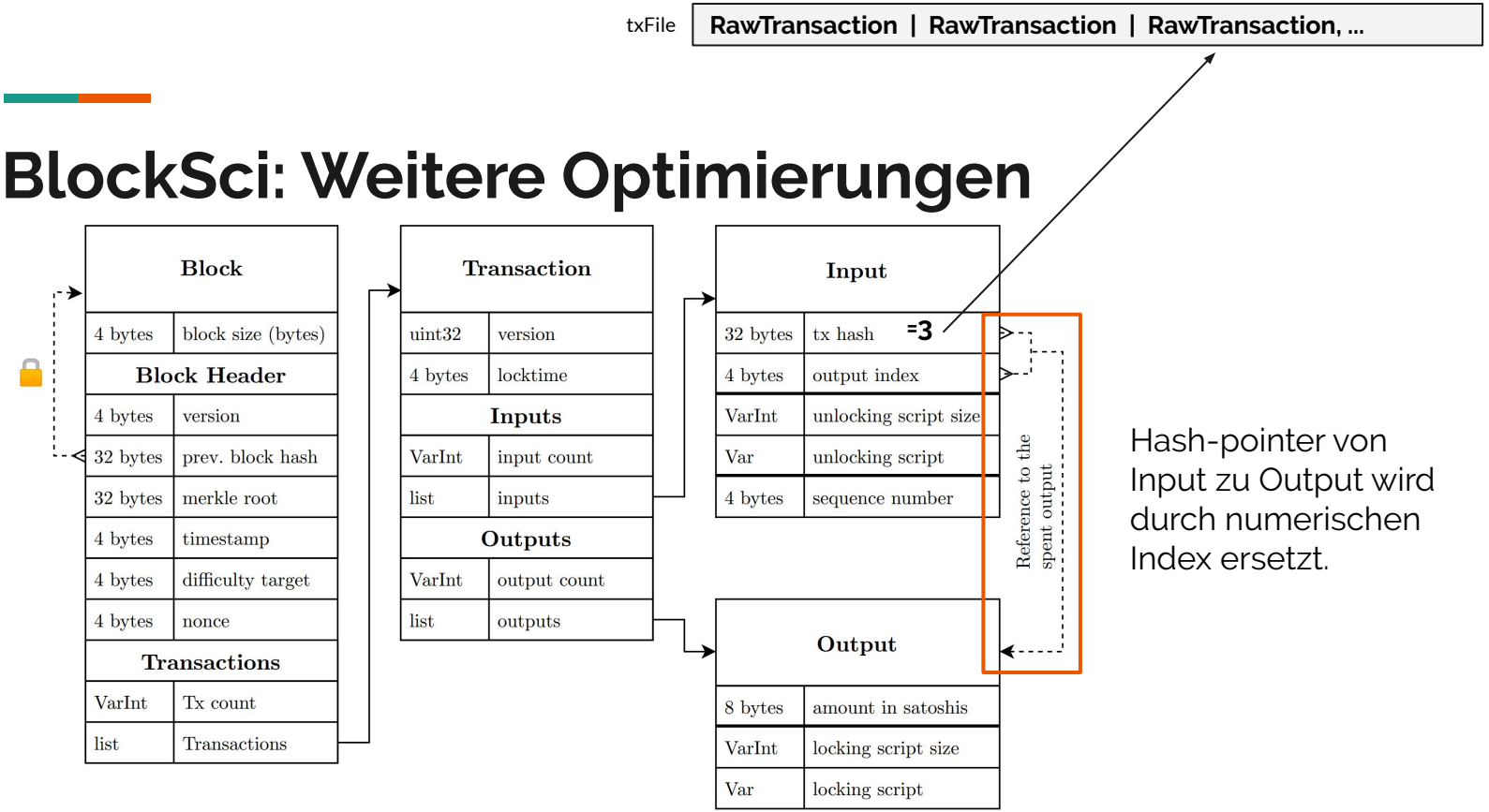


BlockSci: Weitere Optimierungen



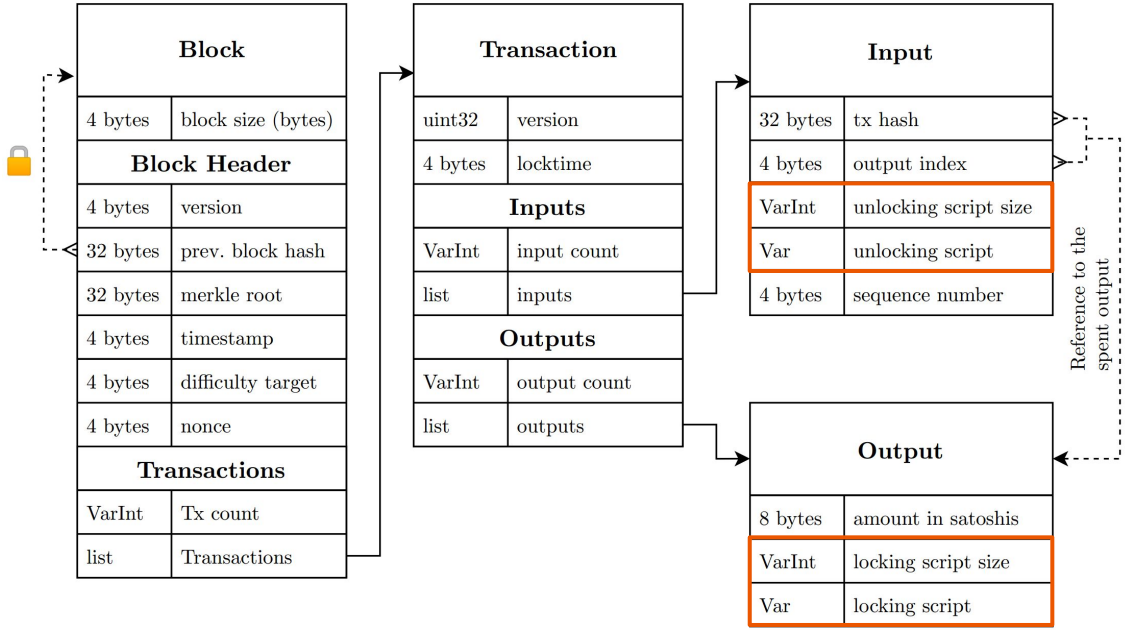
Hash-pointer von Input zu Output wird durch numerischen Index ersetzt.

BlockSci: Weitere Optimierungen



Hash-pointer von Input zu Output wird durch numerischen Index ersetzt.

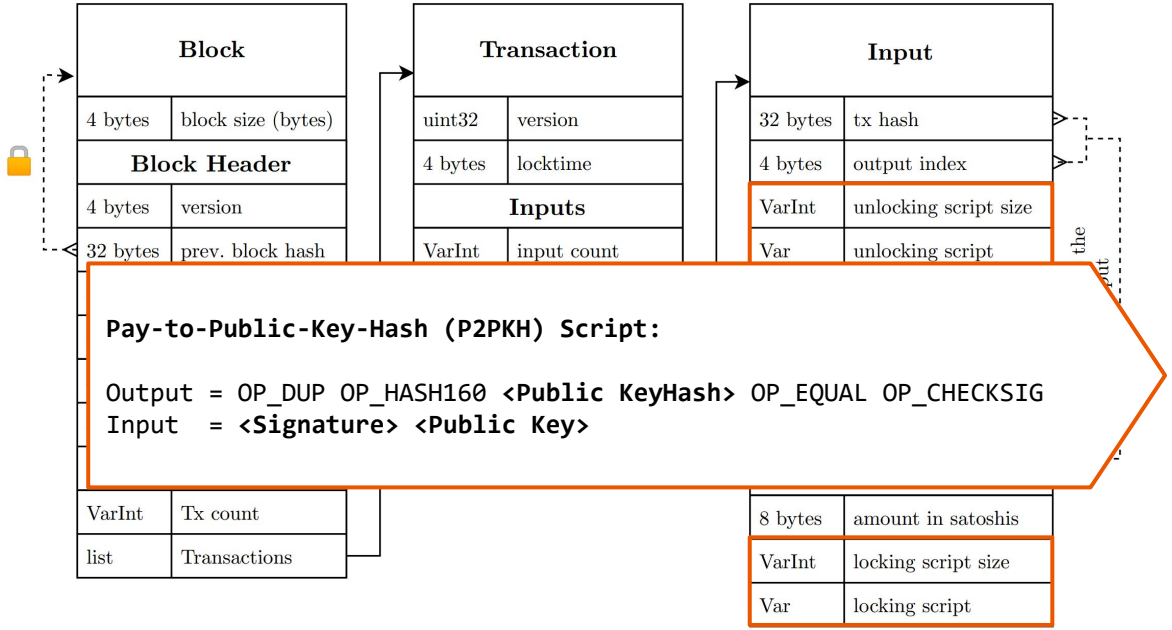
BlockSci: Weitere Optimierungen



Scripts werden de-dupliziert sowie irrelevante Daten entfernt.

z.B. Signaturen sowie opcodes von Standardtypen.

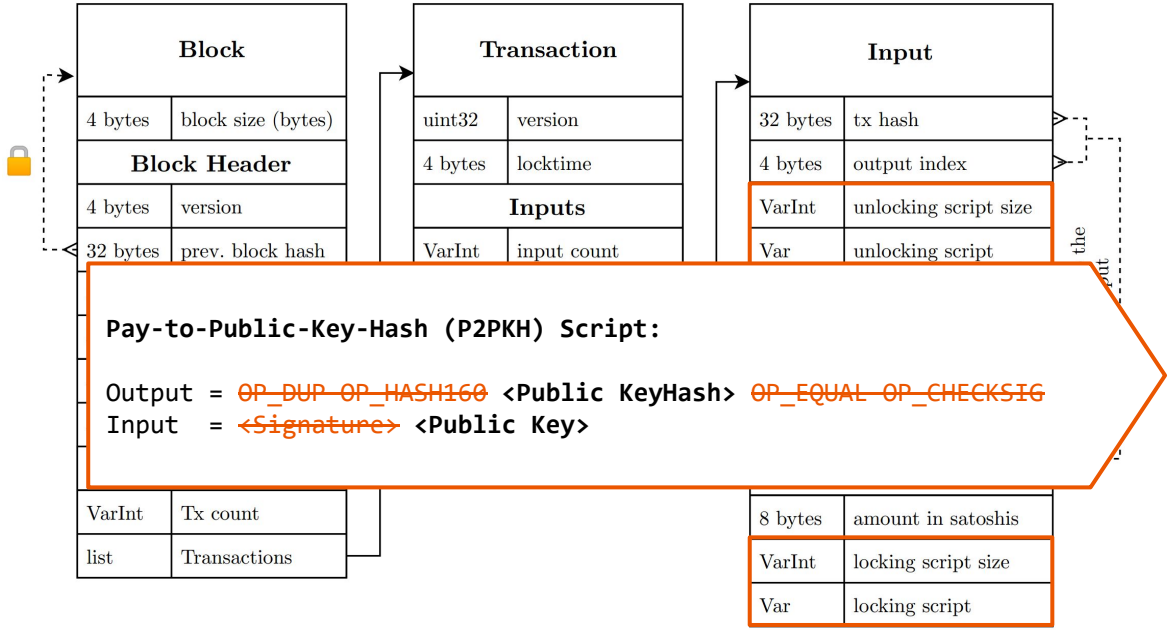
BlockSci: Weitere Optimierungen



Scripts werden de-dupliziert sowie irrelevante Daten entfernt.

z.B. Signaturen sowie opcodes von Standardtypen.

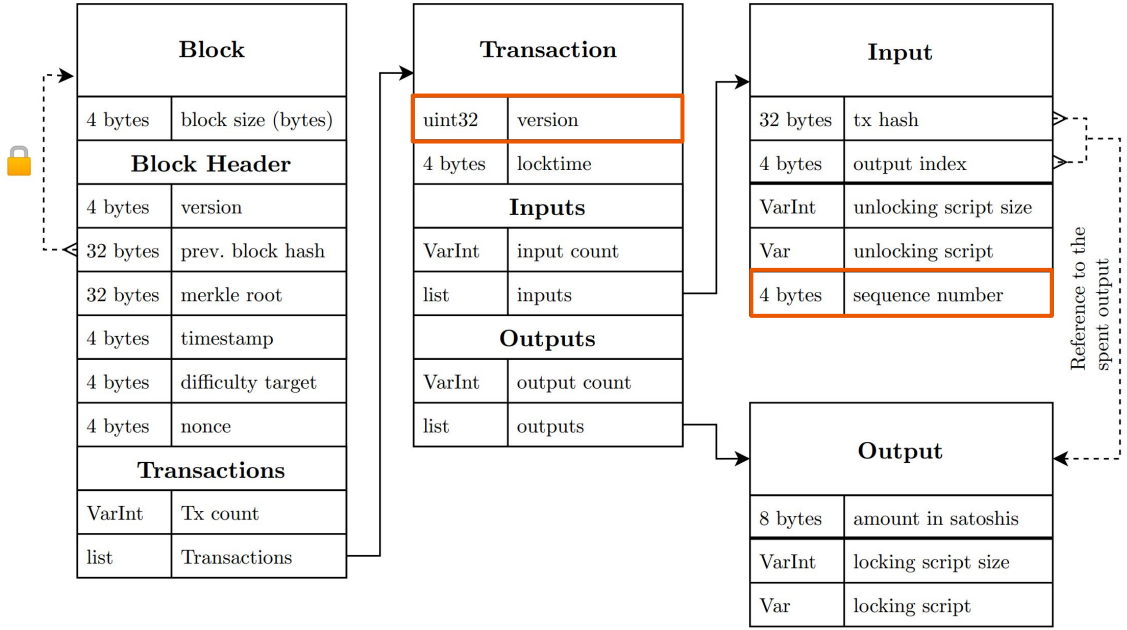
BlockSci: Weitere Optimierungen



Scripts werden de-dupliziert sowie irrelevante Daten entfernt.

z.B. Signaturen sowie opcodes von Standardtypen.

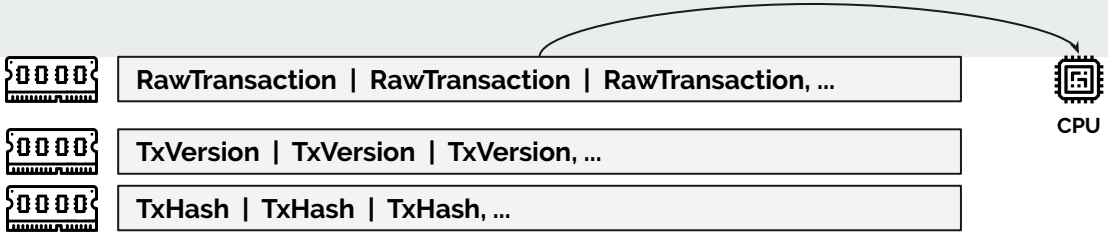
BlockSci: Weitere Optimierungen



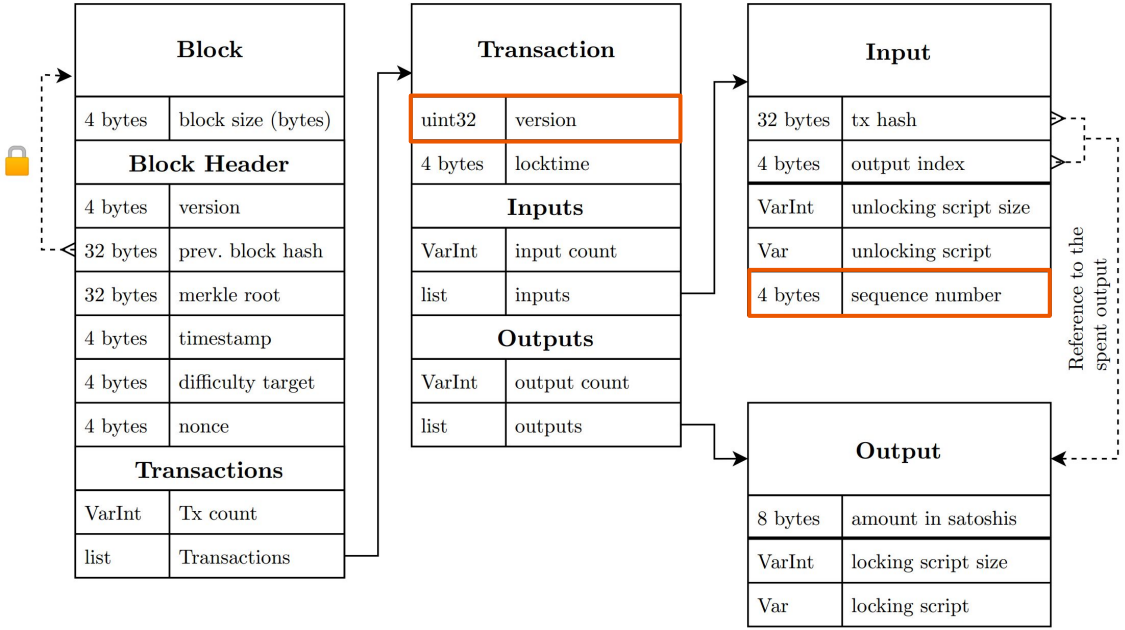
Selten benötigte Werte werden in separaten Memory-mapped files gespeichert.

→ Optimiert Speicherzugriff sowie CPU Caching

Beispiele:
`Transaction.version`
`Transaction.hash`
`Input.sequenceNum`



BlockSci: Weitere Optimierungen



Selten benötigte Werte werden in separaten Memory-mapped files gespeichert.

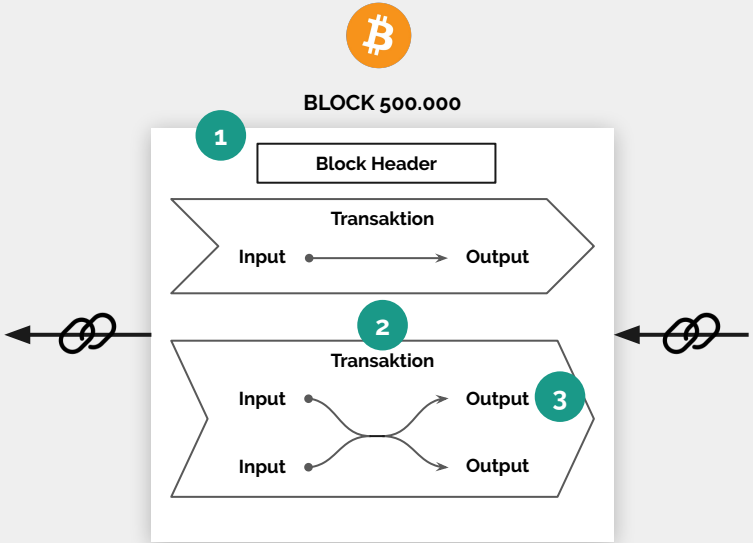
→ Optimiert Speicherzugriff sowie CPU Caching

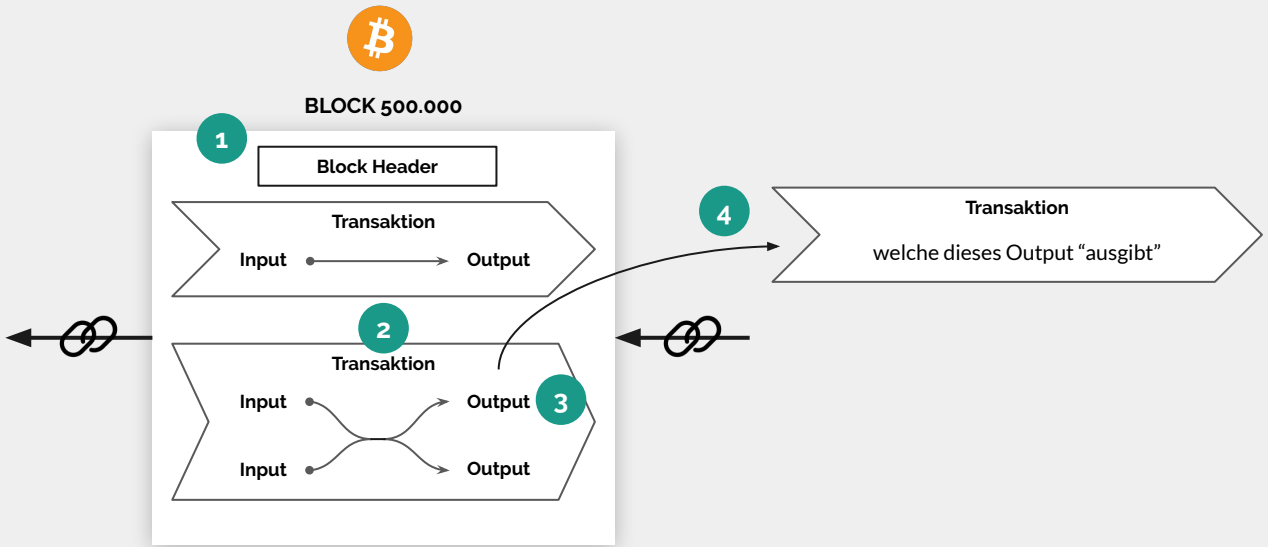
Beispiele:

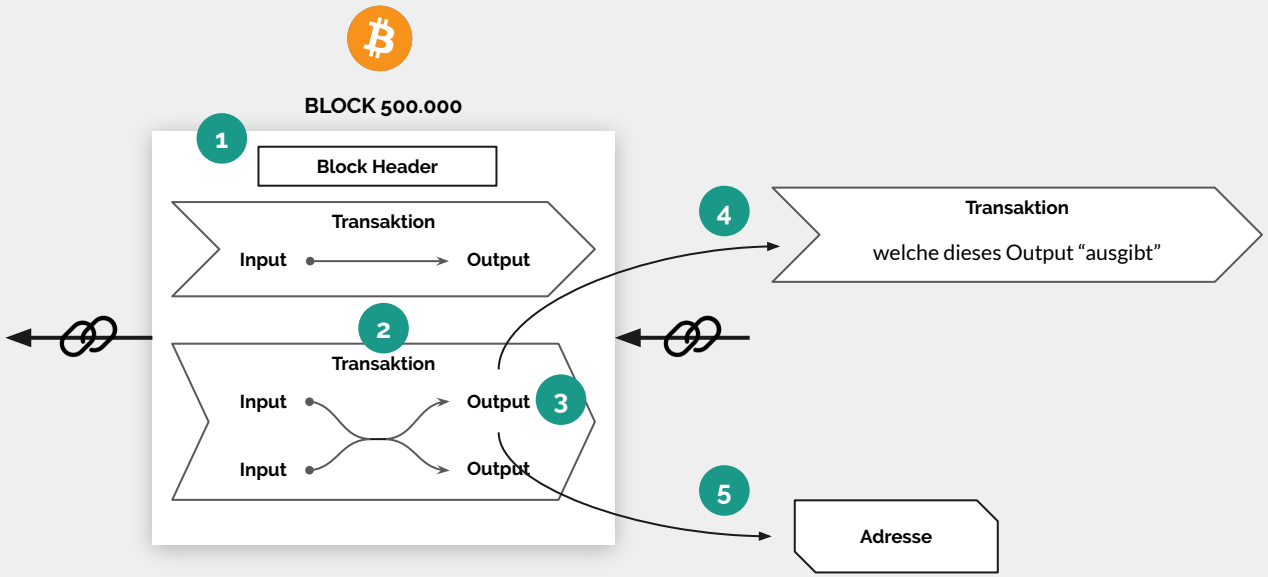
```
Transaction.version
Transaction.hash
Input.sequenceNum
```

Demo I

BlockSci Grundlagen


BLOCK 500.000
1
Block Header
Transaktion
Input
Output
2
Transaktion
Input
Input
Output
3
Output


BLOCK 500.000
1
Block Header
Transaktion
Input
Output
2
Transaktion
Input
Input
Output
Output
3
4
Transaktion
welche dieses Output “ausgibt”


BLOCK 500.000
1
Block Header
Transaktion
Input
Output
2
Transaktion
Input
Input
Output
Output
3
4
Transaktion
welche dieses Output “ausgibt”
5
Adresse

Demo II

BlockSci API Cheat Sheet

Hands-On

Arbeiten Sie selbst mit BlockSci

Hands-On: Ablauf

- Vorab: Setup, Verhaltensregeln, Zugangsdaten
- Alle loggen sich ein und melden sich bei Problemen
- Vertrautmachen mit der Umgebung
- Hausübung (“Übungszettel”)
  - Insgesamt sind 100 Punkte (%) in 4 Aufgaben zu erreichen.
  - Abgabefrist: 14.05.2020 um 23:59
  - Als Abgabe gilt die Datei (Notebook) in der Form so wie sie zur Abgabefrist am Server vorliegt.
    - Machen Sie zur Sicherheit ein Backup bei Ihnen lokal. (File → Download As → Notebook)

Hands-On: Setup

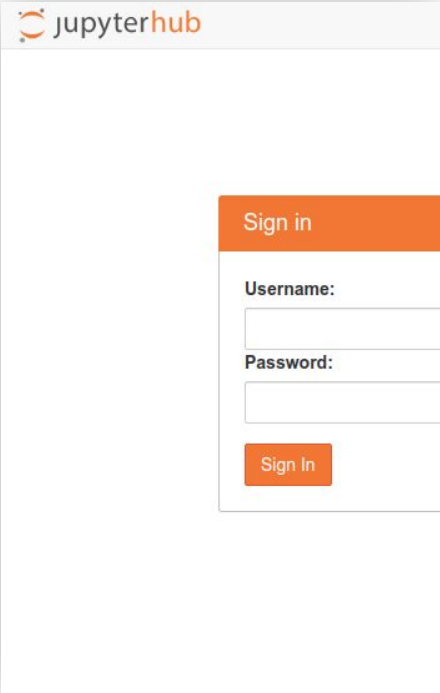
- BlockSci: Daten, Library und Python API
- Jupyter Notebook: Interaktive Python Shell
- JupyterHub
  - Individuelle, isolierte Notebook-Instanz pro Teilnehmer
- Server: 256 GB RAM, 1TB SSD, 6-Kern CPU
  - Limitiert auf 64 GB und 1 CPU-Kern pro Nutzer

Hands-On: Hinweise & Verhaltensregeln

- Eine Datei (Notebook) sollte nicht zeitgleich mehrfach geöffnet werden - Datenverlust möglich
- **Jegliche Form von Missbrauch sowie Angriffe gegen die Infrastruktur sind verboten!**

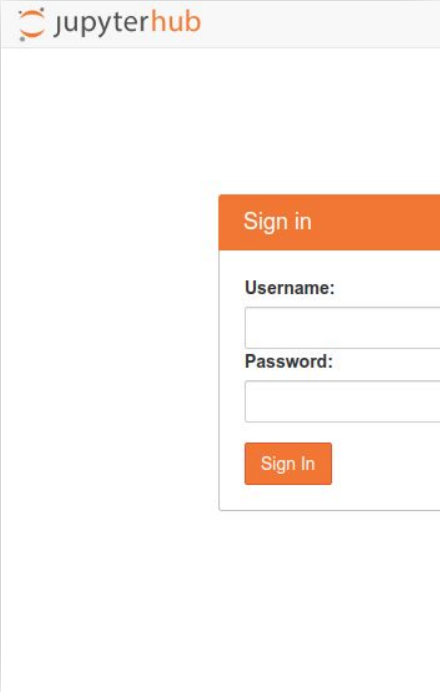
Hands-On: Zugangsdaten

- Username: Ihr Nachname wie in LFU:online bzw. OLAT angegeben
  - In Kleinbuchstaben, Umlaute ersetzt (ü → ue)
- Passwort: Eingabe des ersten Logins wird automatisch als Passwort gesetzt
  - Bitte vergeben Sie ein sicheres Passwort,
  - merken Sie es sich,
  - und geben Sie die Zugangsdaten nicht weiter.



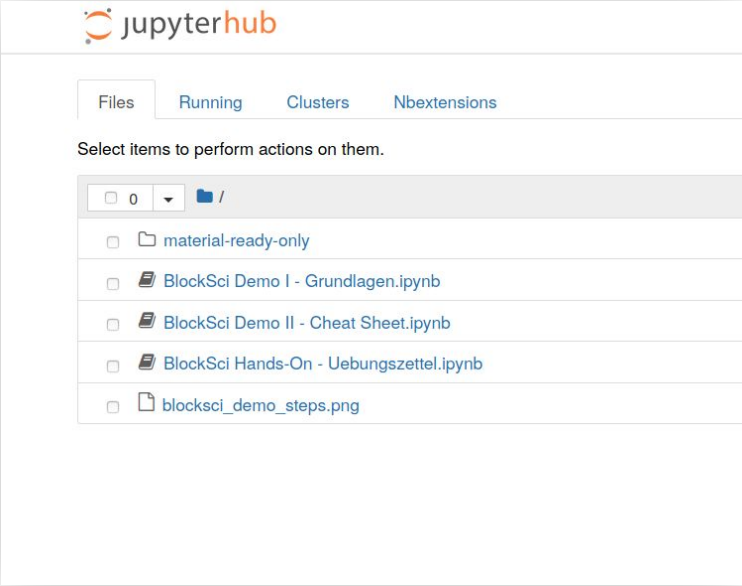
Hands-On: Zugangsdaten

- Username: Ihr Nachname wie in LFU:online bzw. OLAT angegeben
  - In Kleinbuchstaben, Umlaute ersetzt (ü → ue)
- Passwort: Eingabe des ersten Logins wird automatisch als Passwort gesetzt
  - Bitte vergeben Sie ein sicheres Passwort,
  - merken Sie es sich,
  - und geben Sie die Zugangsdaten nicht weiter.
- **Jupyter Notebook Zugang:** https://blocksci.mplattner.at/
- Dokumentation: https://mplattner.at/blocksci-docs/



Hands-On: Startbildschirm

- **Jupyter Notebook Zugang:** https://blocksci.mplattner.at/
- Dokumentation: https://mplattner.at/blocksci-docs/
- Vorhandenes Material
  - **Grundlagen:** Demo wie im Stream
  - **Cheat Sheet:** API Referenz relevanter Klassen
  - **Übungszettel:** Hausübung
  - **material-ready-only/:** gleiche Dateien, jedoch ready-only



Hands-On

Arbeiten Sie selbst mit BlockSci

Abschluss

Zusammenfassung und Fazit

Zusammenfassung

- **BlockSci:** Hoch-performante Blockchain Analyseplattform (Open-Source)
- **Architektur:** In-Memory Datenbank, Serialisierte C++ structs, Memory-Mapped Files mit `mmap()`
- **Demo und Hands-On:** Benutzerfreundliches, flexibles Interface
- **Hausübung ("Übungszettel"):** Abgabe bis 14.5 um 23:59
- **Bei Interesse:** BlockSci Zugang steht noch bis Freitag, den 15. Mai für Sie zur Verfügung

Vielen Dank

Zeit für Fragen, Anregungen etc.

Martin Plattner
martin@mplattner.at

universität
innsbruck