Data Integrity for Librarians, Archivists, and Criminals: What We Can Steal from Bitcoin, BitTorrent, and Usenet

Jeffrey Spies

Center for Open Science | SHARE | UVA

@jeffspies

Presented at CNI 2017/04/03

Librarians and archivists care about data integrity

- Persistence
- Preservation
- Decentralization
- Distribution
- Robustness
- Fault-tolerance
- Inclusivity

If someone removed the Oxford commas from a document (e.g., the title of my talk), how would I know?

Direct Comparison

Data Integrity for Librarians, Archivists, and Criminals
Data Integrity for Librarians, Archivists and Criminals

What if direct comparison is impossible or impractical?

- I send you the file, so you don't have direct access to my version
- I want to track the integrity of a file over time

What metadata could I store and/or share that would represent "sameness"? How about file size?

Data Integrity for Librarians, Archivists, and Criminals

Compare: 56 bytes (i.e., 56 characters)

```
Data Integrity for Librarians, Archivists, and Criminals
Data Integrity for Librarians, Archivists and Criminals
```

Except….

```
Data Integrity for Librarians, Archivists  and Criminals
```

Hash functions deterministically map input data to output that is generally smaller in size.

```
md5 -s “Data Integrity for Librarians, Archivists, and Criminals: What We Can Steal from Bitcoin, BitTorrent, and Usenet”
```

23c1d6085d85ae07378da9861e792c34

Hash functions map arbitrarily sized input data to output that is of fixed size.

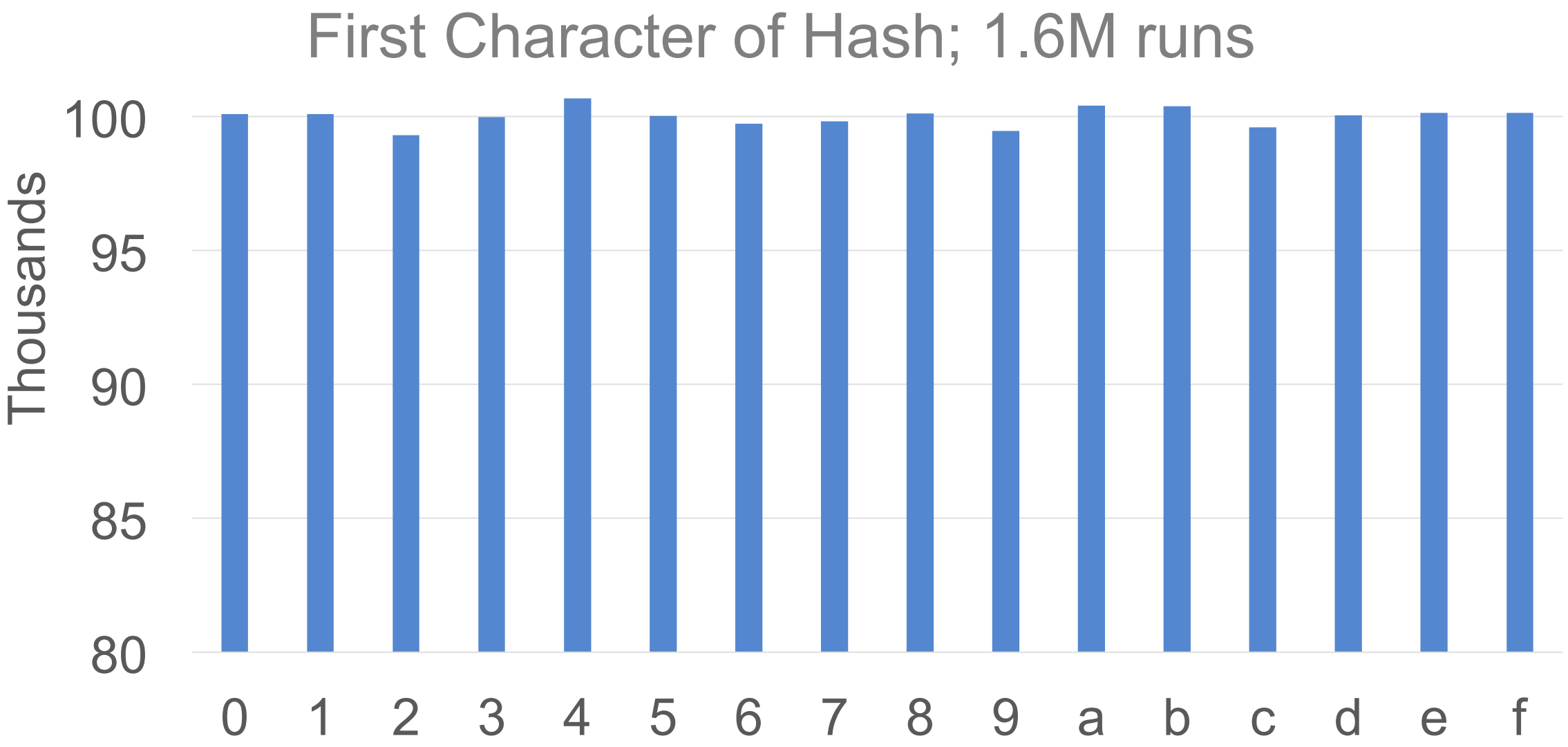
```
md5 -s “Data Integrity for Librarians, Archivists, and Criminals: What We Can Steal from Bitcoin, BitTorrent, and Usenet”
```

23c1d6085d85ae07378da9861e792c34

```
md5 -s "Data Integrity for Librarians, Archivists, and Criminals”
```

386c5555b5aa07c0069596e88b119cc8

Hash functions map input uniformly over an output range.



Cryptographic hash functions are non-invertable and collision-resistant; for a given input, the output is practically unique.

```
md5 -s "Data Integrity for Librarians, Archivists, and Criminals”
```

386c5555b5aa07c0069596e88b119cc8

```
md5 -s "Data Integrity for Librarians, Archivists, and Criminal”
```

40b3d06cd7454e204787bf8eb685d086

If someone removed the Oxford commas from a document (e.g., the title of my talk), how I know?

```
md5 -s “Data Integrity for Librarians, Archivists, and Criminals:
What We Can Steal from Bitcoin, BitTorrent, and Usenet”
```

23c1d6085d85ae07378da9861e792c34

```
md5 -s “Data Integrity for Librarians, Archivists and Criminals: What
We Can Steal from Bitcoin, BitTorrent and Usenet”
```

6eed93a3b7dc829f38065518b346ee72

We did it!

23c1d6085d85ae07378da9861e792c34 != 6eed93a3b7dc829f38065518b346ee72

Enough with the commas: why does this matter?

- Data integrity is important
- Media is prone to data degradation (or data decay, data rot)
- Unintentional corruption is rare, but…
- We're working with such large scales of data and accessing it so quickly that even low probability errors are common
- Examples
  - NetApp found more than 400,000 silent data corruptions across 1.5M drives over 41 months—30,000 were not detected by hardware RAID
  - CERN found 128 MB of data permanently corrupted across 97 PB of data over 6 months

Cryptographic hash functions

- MD5
  - 128 bits, 32 char hex digest
  - Only recommended to recognize unintentional corruption
- SHA1
  - 160 bits, 40 char hex digest
  - No longer recommended for collision resistance
- SHA2
  - SHA-256 – 256 bits, 64 char hex digest
  - SHA-512 – 512 bits, 128 char hex digest

```
/storage/mary/data.csv, 150MB, 23c1d
/storage/john/data.csv, 50MB, 6e88b
/storage/chris/marysdata.csv, 150MB, 23c1d
```

```
/storage/23c1d, 150MB
/storage/6e88b, 50MB
```

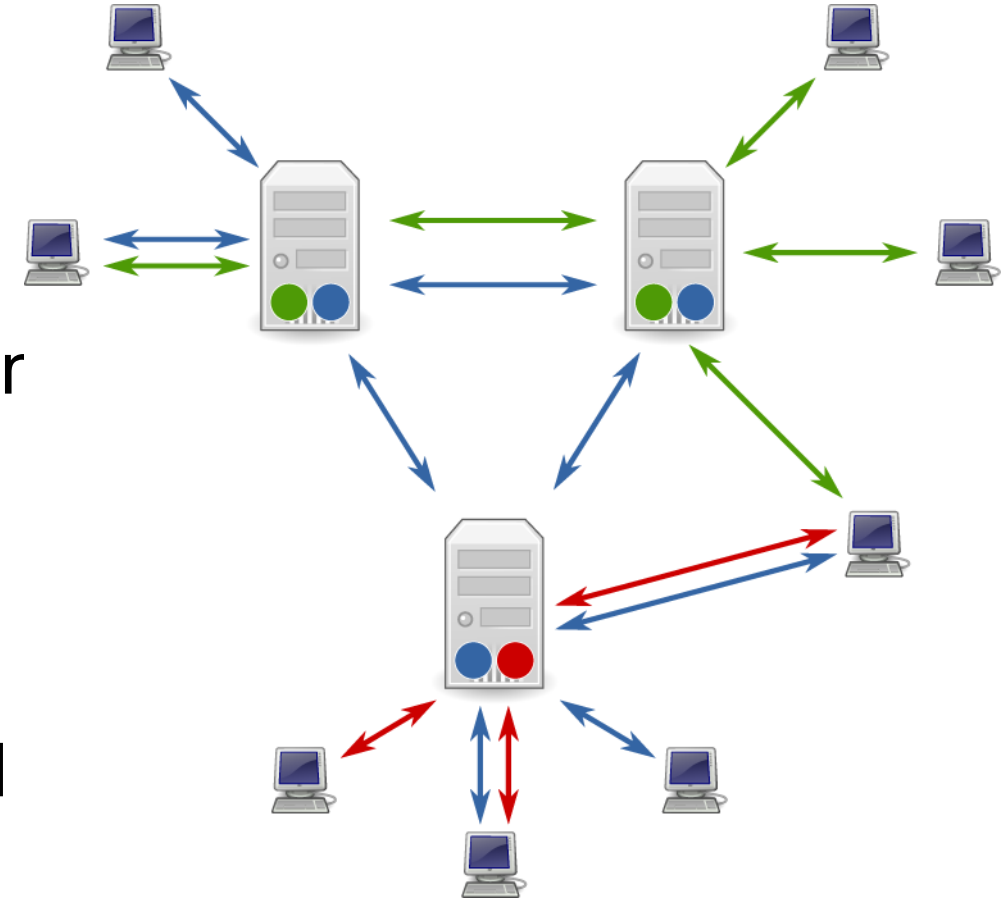
Content-addressable storage

Takeaways

- Pairing hashes with downloads
- Teach people to check those hashes using tools like *md5*, *md5sum*, *sha1*, *sha1sum*, *sha256*, *sha256sum*, or Python's *hashlib*.
- Content-addressable storage

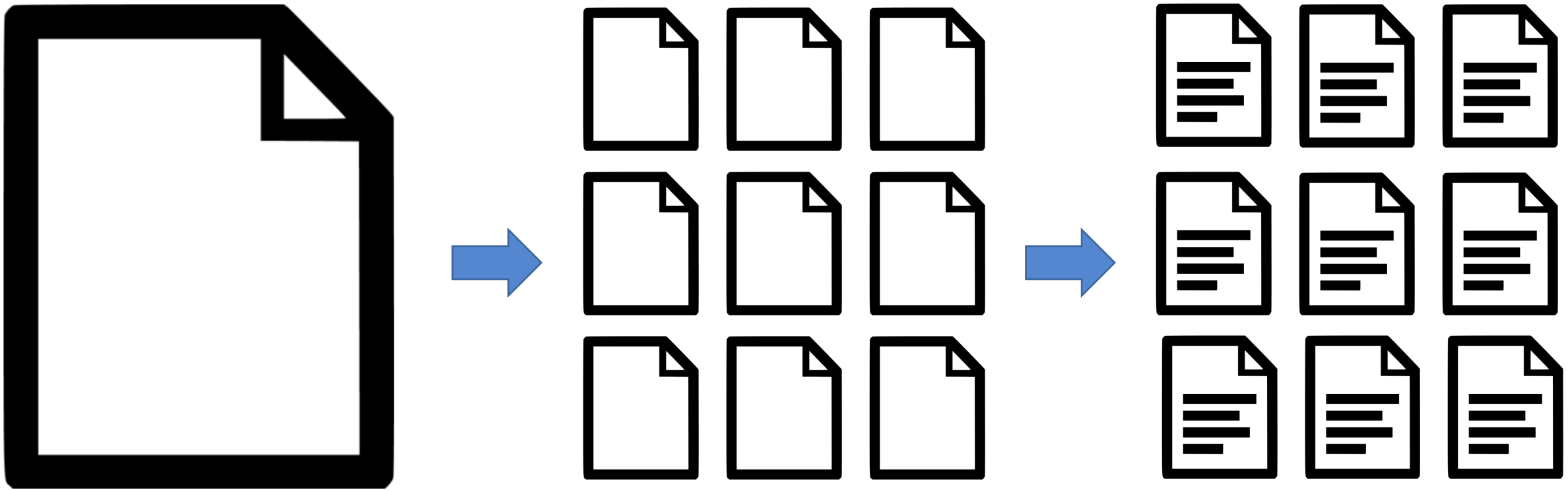
Usenet

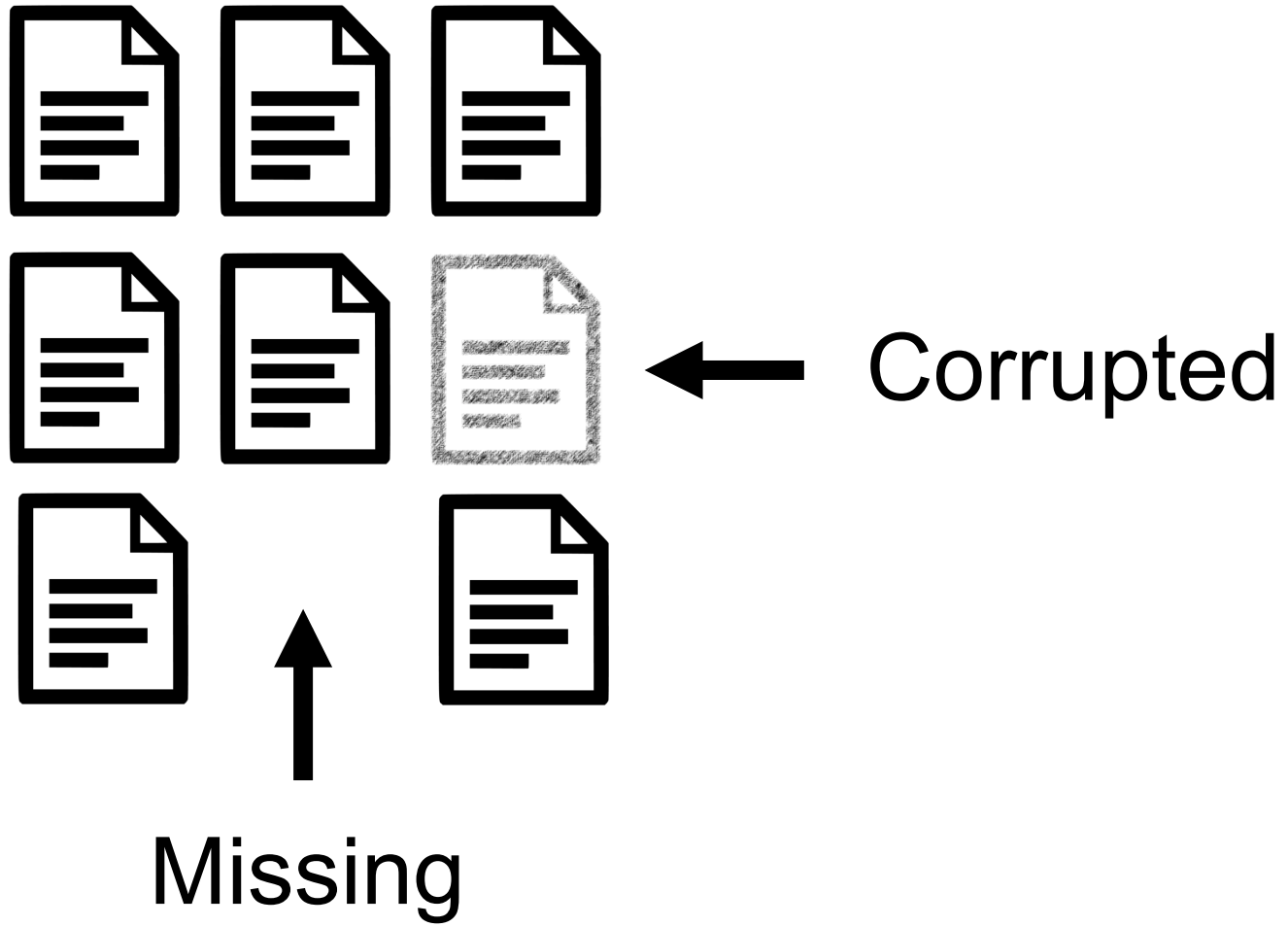
- Established in 1980
- Users can read and post messages or topics called newsgroups
- Like BBSes & Internet forums
- No central server or admin
- Over time, people used it to store and share files



Challenges to sharing files on Usenet

- Not designed to transmit binary files
- Message size limits


Split
Encode


Corrupted
Missing

Parity Computation

Julie R; CC-BY-SA

Redundant **A**rray of **I**ndependent **D**isks (RAID)

XOR(0, 1) ≡ 1
XOR(1, 0) ≡ 1
XOR(0, 0) ≡ 0
XOR(1, 1) ≡ 0




1 1 0 1 0 1 0 0 1




1 1 0 1 0 1 0 0 1

XOR (110,
101,
001) = 0 1 0

1 1 0 1 0 1 0 0 1

XOR (110,
101,
001) = 0 1 0

1 1 0

1 0 1

0 0 1

0 1 0

1 1 0

1 0 1

0 0 1

0 1 0



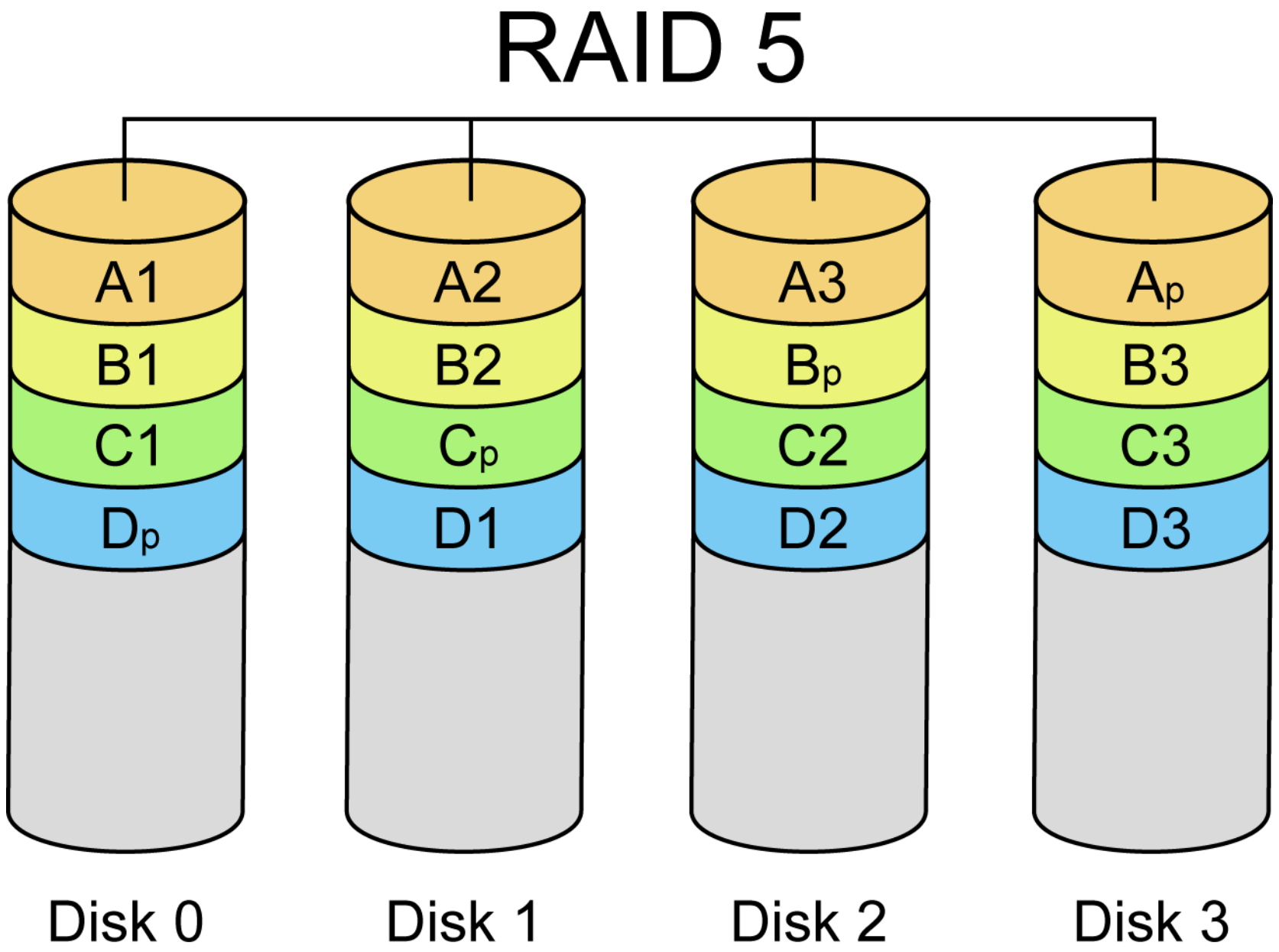

1 1 0 1 0 1 0 0 1 0 1 0

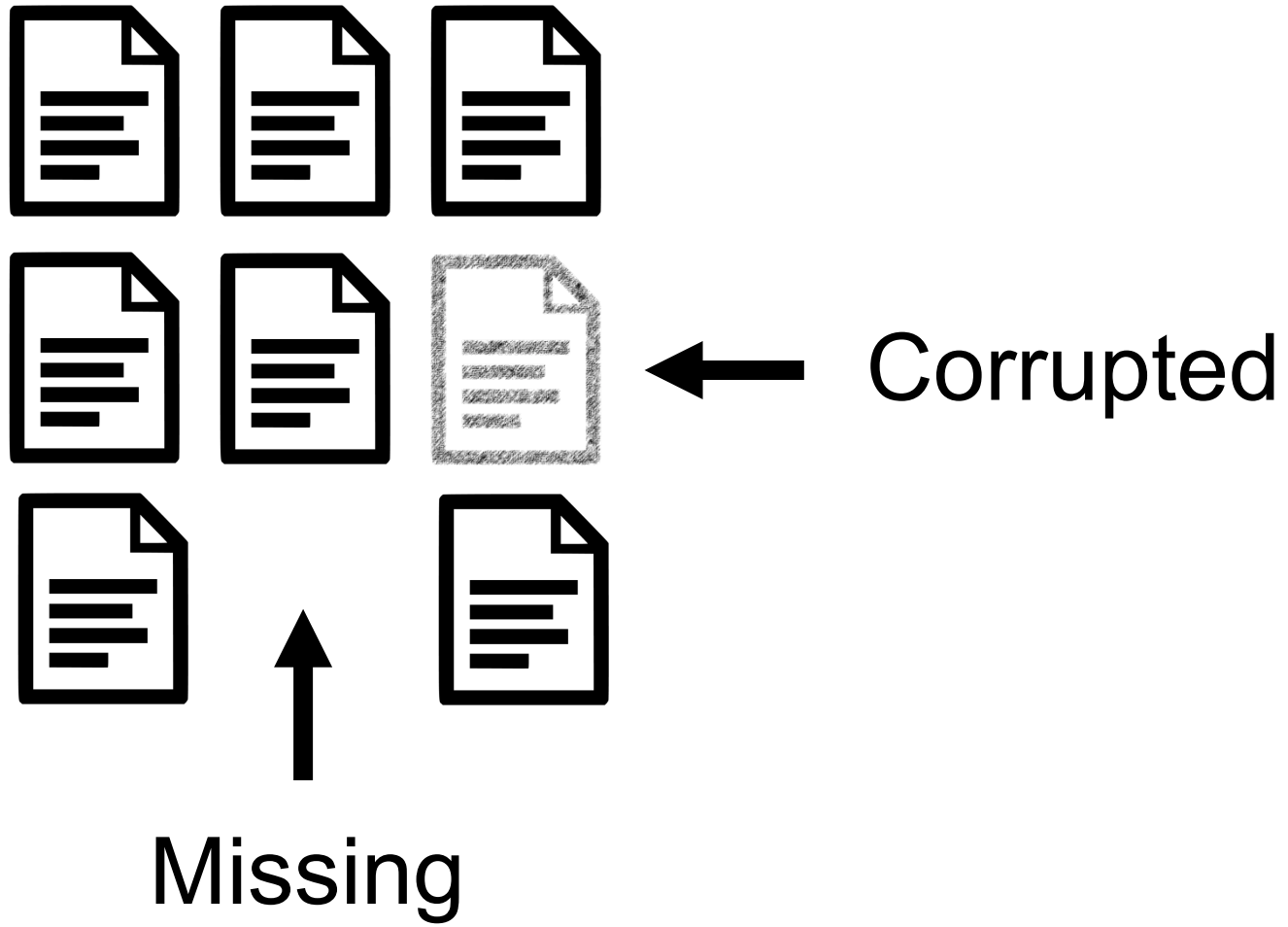
XOR (110,
101,
010) =

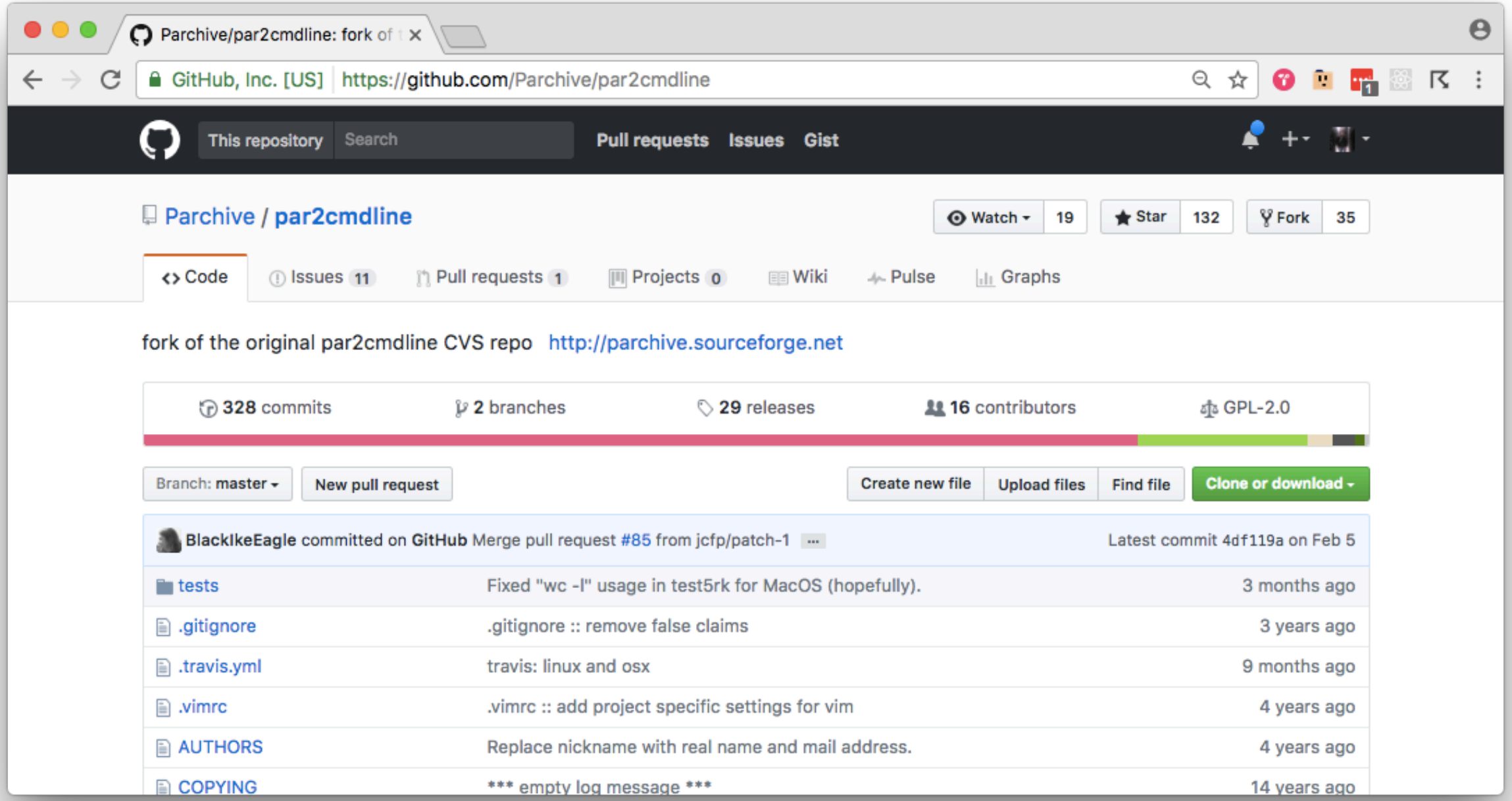



1 1 0 1 0 1 0 0 1 0 1 0

XOR (110,
101,
010) = 0 0 1




Corrupted
Missing

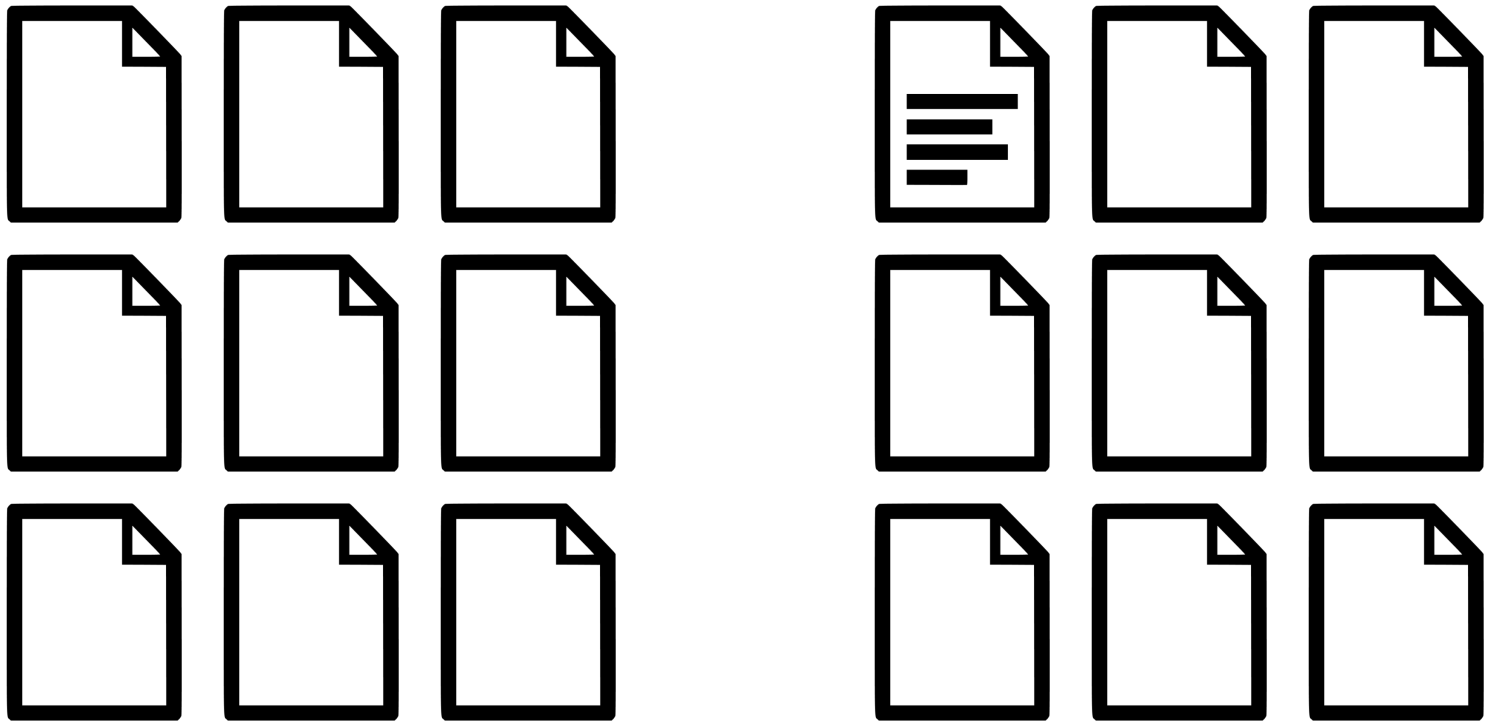

Parchive/par2cmdline: fork of
GitHub, Inc. [US] https://github.com/Parchive/par2cmdline
This repository Search
Pull requests Issues Gist
Parchive / par2cmdline
Watch 19 Star 132 Fork 35
Code Issues 11 Pull requests 1 Projects 0 Wiki Pulse Graphs
fork of the original par2cmdline CVS repo http://parchive.sourceforge.net
328 commits 2 branches 29 releases 16 contributors GPL-2.0
Branch: master New pull request
Create new file Upload files Find file Clone or download
BlackIkeEagle committed on GitHub Merge pull request #85 from jcfp/patch-1
Latest commit 4df119a on Feb 5
tests Fixed "wc -l" usage in test5rk for MacOS (hopefully). 3 months ago
.gitignore .gitignore :: remove false claims 3 years ago
.travis.yml travis: linux and osx 9 months ago
.vimrc .vimrc :: add project specific settings for vim 4 years ago
AUTHORS Replace nickname with real name and mail address. 4 years ago
COPYING *** empty log message *** 14 years ago

```
par2 create -r10 ubuntu.iso.par2 ubuntu.iso
```

```
ubuntu.iso.par2 - This is an index file for verification only
ubuntu.iso.vol00+01.par2 - Recovery file with 1 recovery block
ubuntu.iso.vol01+02.par2 - Recovery file with 2 recovery blocks
ubuntu.iso.vol03+04.par2 - Recovery file with 4 recovery blocks
ubuntu.iso.vol07+08.par2 - Recovery file with 8 recovery blocks
ubuntu.iso.vol15+16.par2 - Recovery file with 16 recovery blocks
ubuntu.iso.vol31+32.par2 - Recovery file with 32 recovery blocks
ubuntu.iso.vol63+37.par2 - Recovery file with 37 recovery blocks
```

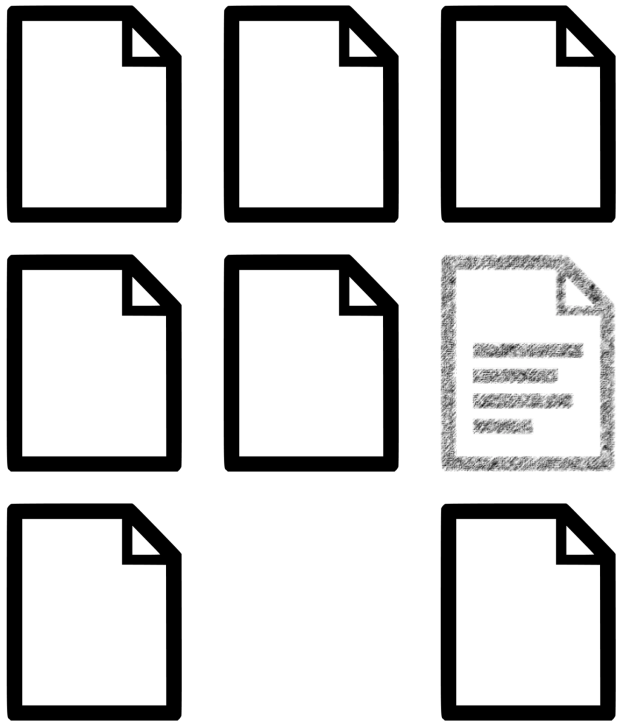
Files

Parchive

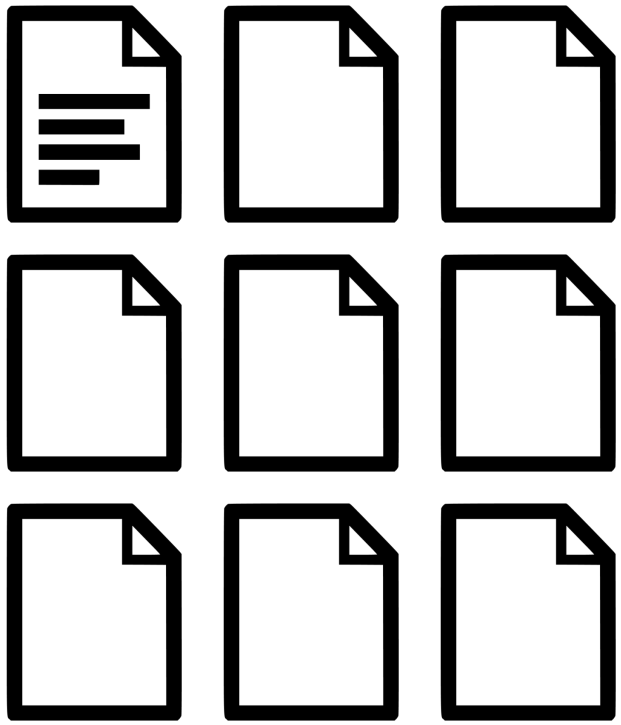


Parchive: Index file of hashes & parity volumes

Files



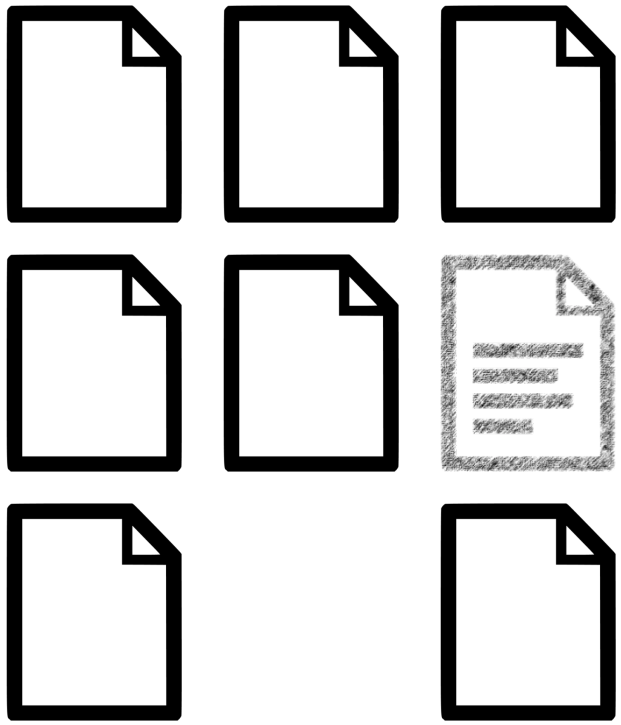
Parchive



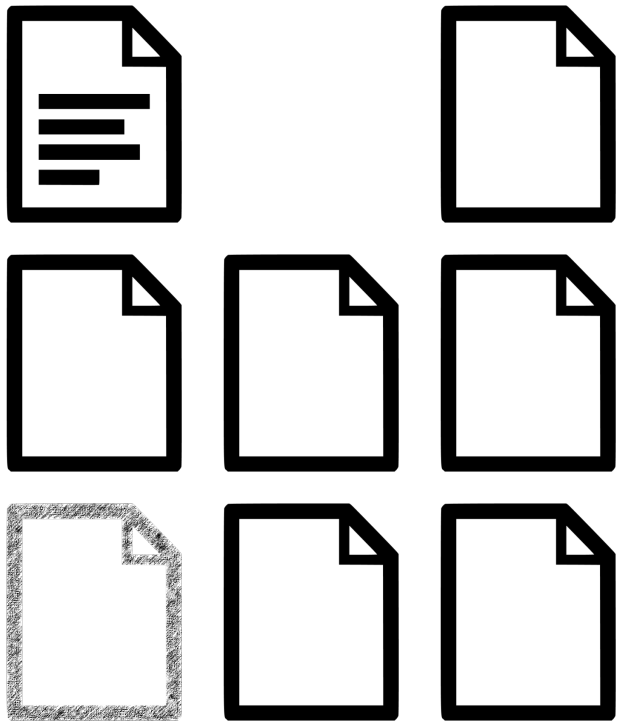
```
par2 verify ubuntu.iso.par2

par2 repair ubuntu.iso.par2
```

And you don’t even need a complete volume set—you can still recover data.

Files



Parchive

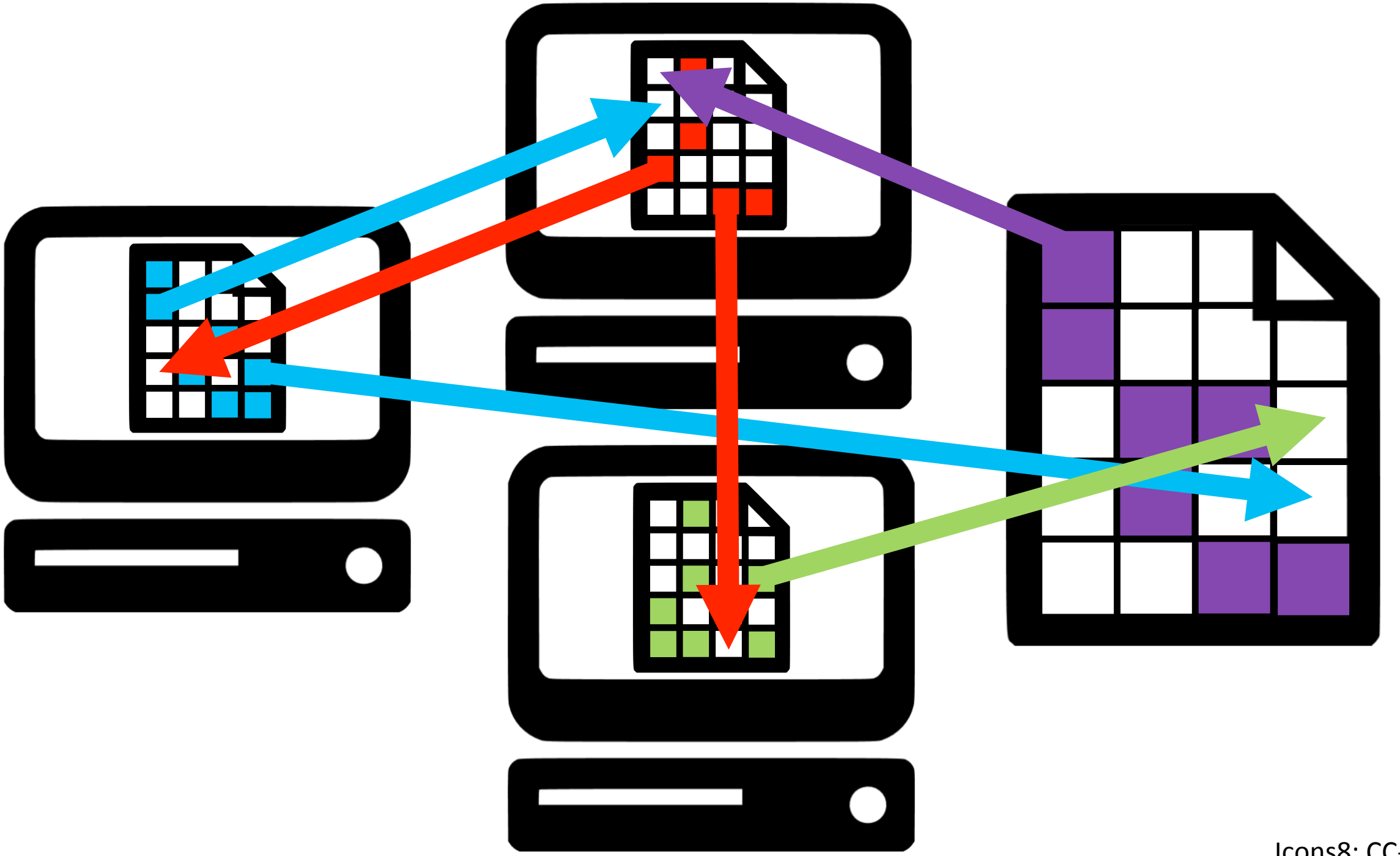


Takeaways

- Parity archives for those cases when other systems don’t catch data corruption

BitTorrent

- Peer-to-peer file sharing protocol
- Designed and released in 2001
- In 2013, responsible for 3.35% of internet traffic



A torrent file is metadata & a list of hashes

```
{
  'announce': http://trackerurl,
  'created by': 'Jeff',
  'date': 1476352188,
  'locale': 'en',
  'title': 'Ubuntu 16.04'
  'info': {
    'piece length': 388433,
    'pieces': SHA1a, SHA1b…
    'length': 1593835520,
    'files': [
      {
       'path': ['ubuntu.iso'],
       'length': 34377
      },...
    ],...
    'name': ’Example Torrent',
  }
}
```

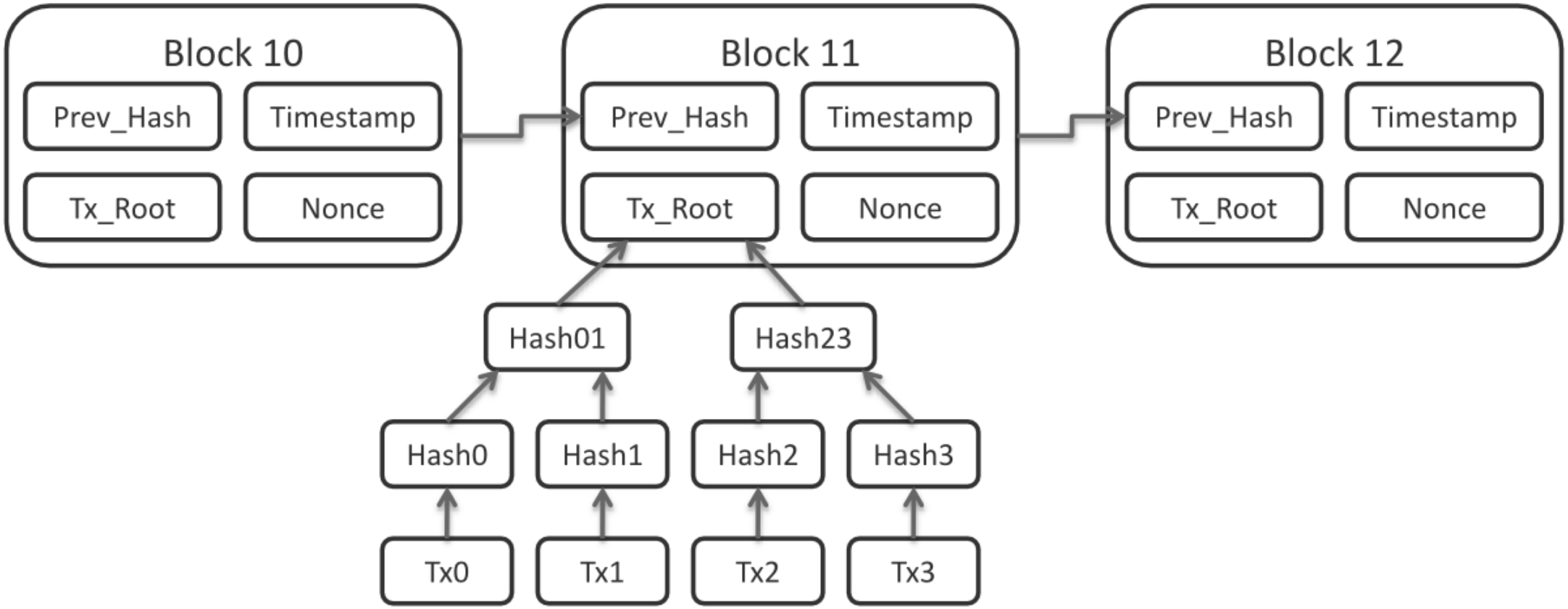
Distributed Hash Table

- A decentralized and distributed lookup service
- For torrents, this allows you to lookup a torrent via hash and get a list of peer IP addresses that are sharing the torrent’s files
- Fault-tolerant and scalable

Takeaways

- The torrent protocol for data storage
- Distributed hash tables to provide lookup for both content-addressable storage and peer discovery

- Cryptocurrency & electronic payment system
- Introduced in 2008
- The block chain is a **public**, **distributed**, **immutable** ledger or database
- Bitcoins are mined by people that create and verify hashes
- As of today, 1 BTC = 1,153 USD



- Block 12's Prev_Hash is the hash of Block 11's header, which has Block 10's hash built into it.
- The body of Block 11 (i.e., actual transactions) is represented by a hash in the header (Tx_Root)
- These headers get hashed by miners

Mining is hashing

- Mining requires proof of work: the work is difficult, but easily verifiable
- If mining is too easy, there would be too much BTC, and it would lose its value
- Hashing has a known difficulty and is easily verified
- Miner task: Generate a hash of transaction metadata by adding some random data at the end (via the nonce) such that the resulting hash, e.g., starts with 18 zeros
- Example:

SHA256(metadata + 1)
SHA256(metadata + 2)
…
SHA256(metadata + 4,294,967,296)

Last night, miners generated a hash with all those leading zeros that contained the transaction information for that block:

00000000000000000ebac8b94c2720e794e9a9c4d6f31a56c03745520d5c2d3

Number of transactions: 1938
Transaction volume: ~3,853 BTC
Reward: 12.5 BTC

Challenges

- Block chain size: BTC is ~110 GB
- Immutability
- Anything can be encoded and stored in the block chain

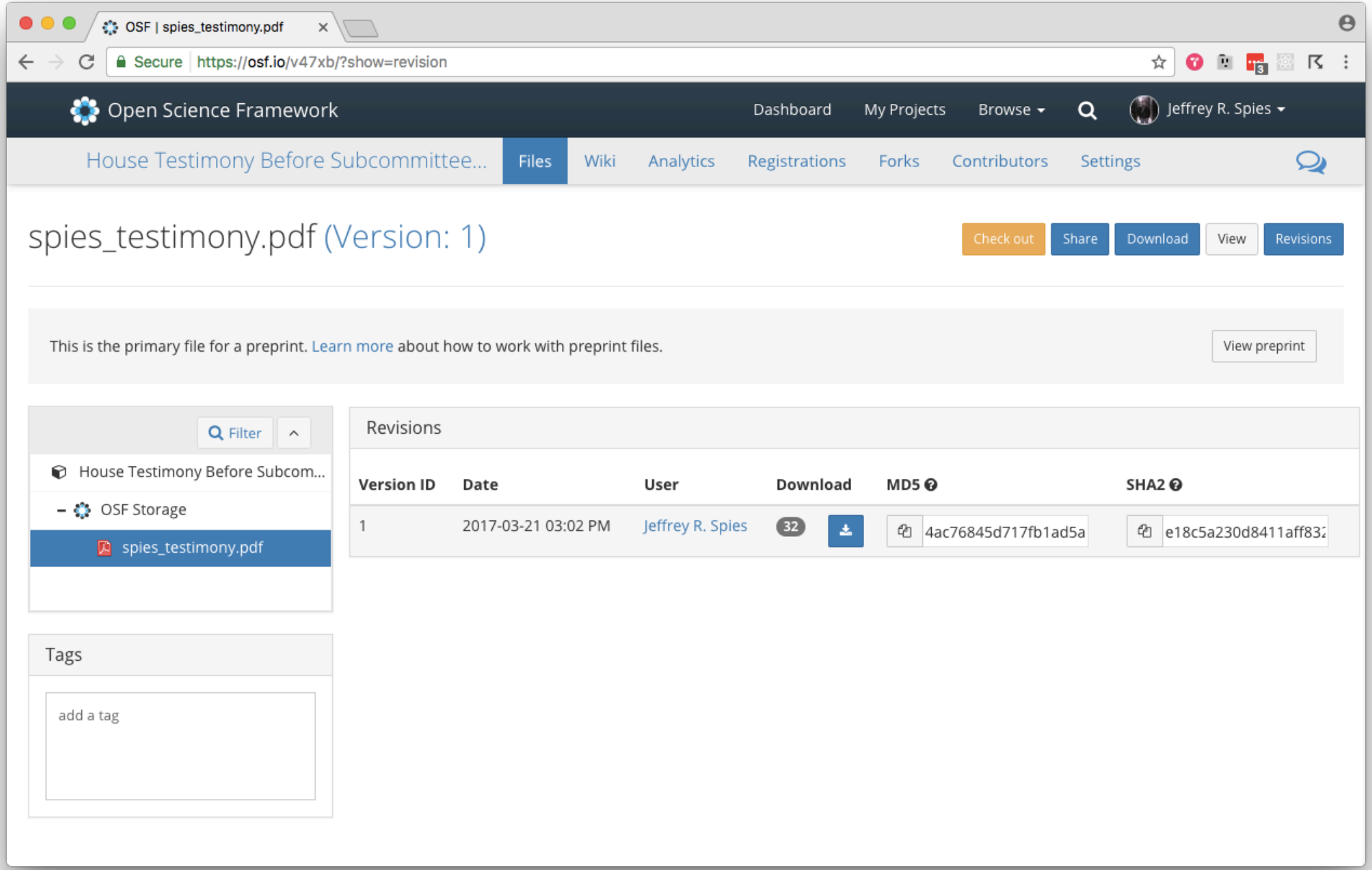
Takeaways

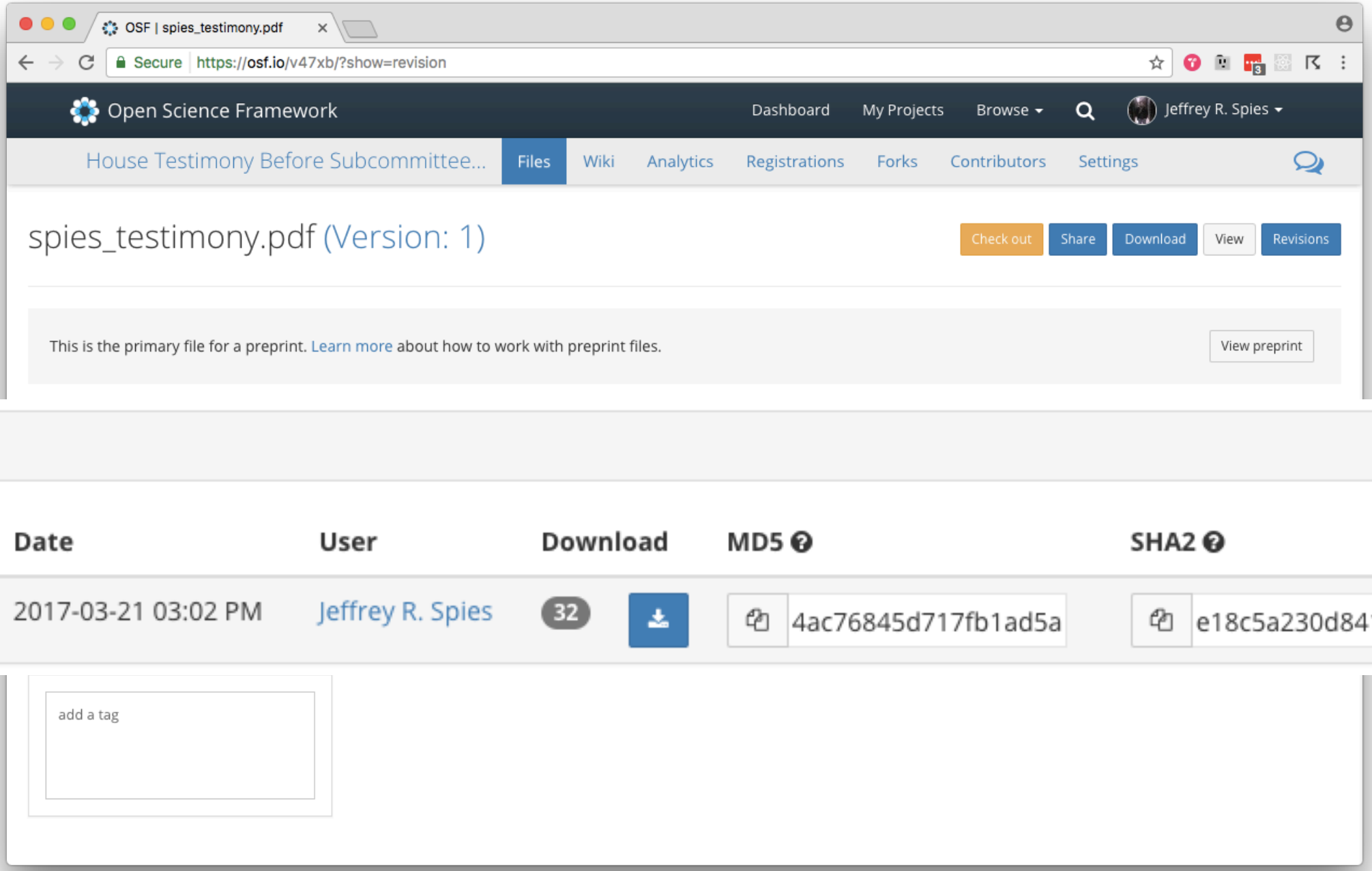
- Block chains for distributing immutable records
  - Data repositories and journals on the block chain?
  - True immutability may not be desirable for our use-case (e.g., accidents happen and private information is made public).
  - Rather than content (e.g., data, journals), I recommend storing hashes (and perhaps persistent identifiers and/or metadata)
- Compensation via transactions fees (e.g., for copy-editing, peer-review)

Summary: things I didn't cover

- Lots of details
- Hashes in data structures
- Hashes in web site security
- Other forms of parity
- Issues with RAID5
- Other types of RAID (e.g., RAID 10)
- Filesystems for better data integrity (e.g., btrfs, ZFS)
- Data corruption (e.g., using non-ECC memory)
- Vulnerabilities of decentralized and distributed systems

What have I stolen?
(No copyrighted material!)


OSF | spies_testimony.pdf
Secure https://osf.io/v47xb/?show=revision
Open Science Framework
Dashboard
My Projects
Browse
Jeffrey R. Spies
House Testimony Before Subcommittee...
Files
Wiki
Analytics
Registrations
Forks
Contributors
Settings
spies_testimony.pdf (Version: 1)
Check out
Share
Download
View
Revisions
This is the primary file for a preprint. Learn more about how to work with preprint files.
View preprint
Filter
House Testimony Before Subcom...
OSF Storage
spies_testimony.pdf
Tags
add a tag
Revisions
Version ID
Date
User
Download
MD5
SHA2
1
2017-03-21 03:02 PM
Jeffrey R. Spies
32
4ac76845d717fb1ad5a
e18c5a230d8411aff83


OSF | spies_testimony.pdf
Secure https://osf.io/v47xb/?show=revision
Open Science Framework
Dashboard
My Projects
Browse
Jeffrey R. Spies
House Testimony Before Subcommittee...
Files
Wiki
Analytics
Registrations
Forks
Contributors
Settings
spies_testimony.pdf (Version: 1)
Check out
Share
Download
View
Revisions
This is the primary file for a preprint. Learn more about how to work with preprint files.
View preprint
Revisions
Version ID
Date
User
Download
MD5
SHA2
1
2017-03-21 03:02 PM
Jeffrey R. Spies
32
4ac76845d717fb1ad5a
e18c5a230d8411aff83
add a tag

The OSF (http://osf.io) uses content-addressable storage for the data we host.

When projects are **forked** (copied and referenced) or **registered** (copied and archived), it doesn't change the amount of data the OSF stores.

OSF Storage stores three hashes of varying complexity and speed for auditing purposes.

OSF Storage stores parity archives for every file.

COS Labs is exploring uses of the block chain and torrent protocols for

- distributing and storing (meta)data,
- fostering sustainability,
- and facilitating collaboration and inclusivity by increasing the number of individuals that can contribute to scholarship.

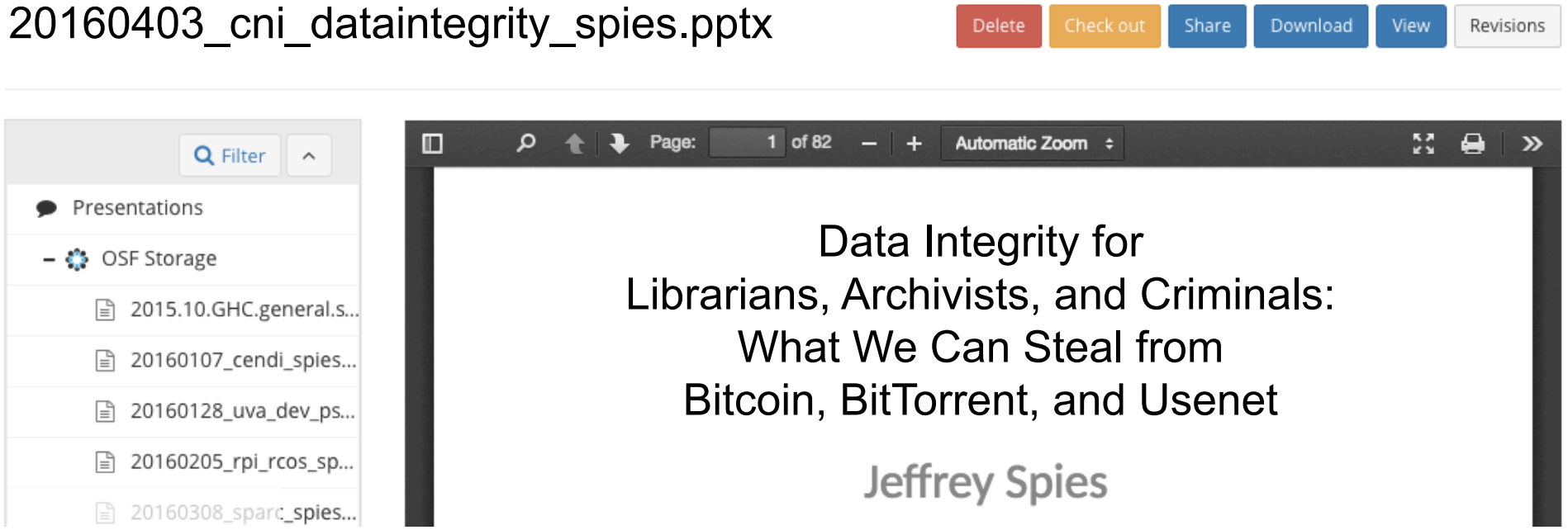
What can you steal?

- Pairing hashes with downloads
- Teach people to check hashes
- Content-addressable storage
- Parity archives
- Block chains for distributing immutable records
- Compensation via transactions fees
- The torrent protocol for inclusive contribution to data storage
- Distributed hash tables to provide lookup for both content-addressable storage and peer discovery

Find this presentation at

https://osf.io/view/CNI2017Spring/

20160403_cni_dataintegrity_spies.pptx



@jeffspies

jeff@cos.io

md5 is a 128-bit hash.

23c1d6085d85ae07378da9861e792c34 is a hex digest.

It uses 16 characters (0-9 and a-f) or 4 bits and therefore results in a 32 character string.

Math: $\log_2 16 = 4$; $128/4 = 32$

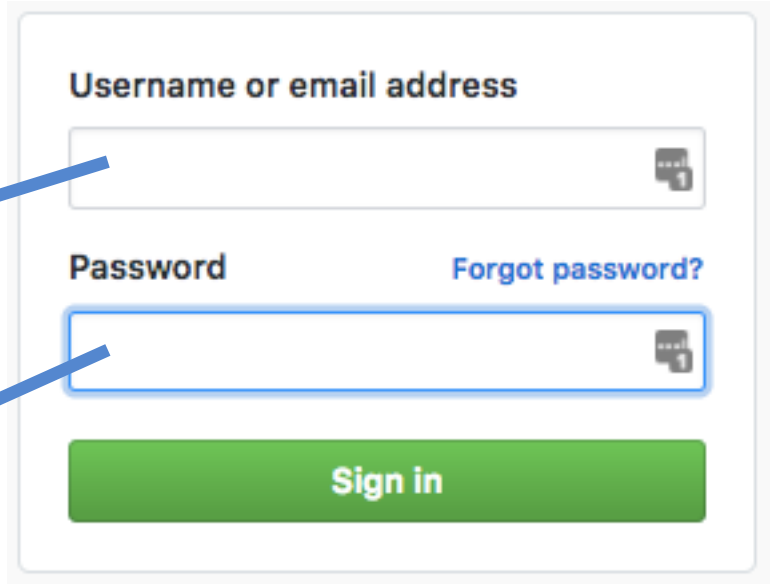
Have you ever clicked “Forgot your password?”, and it emailed you your password?

The site stored your password in plaintext.

There is no reason for this.

Ideal: store the hash in the database and then compare the computed hash of the password submitted with what is stored.

```
def can_login( request ):
  user = request['username']
  password = request['password']
  computed_hash = hash(password)
  hash_from_db = db.users[user].hash
  return is_same(computed_hash, hash_from_db)
```



Note: Popular modern hash functions for passwords are bcrypt, scrypt, and PBKDF2.

If hashes are non-invertable, why does it matter if they are leaked/sold in database hacks?

Dump:

jeff@cos.io, **5f4dcc3b5aa765d61d8327deb882cf99**
mary@yahoo.com, 25d55ad283aa400af464c76d713c07ad
…
chris@gmail.com, **5f4dcc3b5aa765d61d8327deb882cf99**

Frequently used password dictionary:

password, **5f4dcc3b5aa765d61d8327deb882cf99**
12345678, 25d55ad283aa400af464c76d713c07ad
…
abc123, e99a18c428cb38d5f260853678922e03

The block chain is a **public**, **distributed**, **immutable** ledger or database.

Mining adds transactions to the block chain—who sent how much to whom and when.

Blocks must contain a proof of work that is difficult to produce (costly, time-consuming) but easy to verify: that work is hashing using SHA256.

Miners try to generate a hash with a run of leading zeros (i.e., the value is lower than a target difficulty).

The input to the hash function is a block header containing the hash of a set of new transactions, some other metadata (including timestamp), and a nonce.

A nonce is 32 bits that is changed until the hash of the header and nonce is contains a run of leading zeros.

```
SHA256(header + 0)
SHA256(header + 1)
…
SHA256(header + 4,294,967,295)
```

It’s a chain because the block header also includes the hash of the previous block header.

At anytime, you can verify the integrity of the chain by calculating hashes forward.

If any part of the chain changes, the hashes no longer match.

Other uses of hashes in bitcoin

- Addresses are derived by hashing public keys—you don’t want two people to have the same address.
- Signatures validate transactions; they are generated from a private key and the hash of data to be signed.
- Transactions are addressed by their hash.
- Blocks are identified and verified by their hash.