Bitcoin, the Gold Mine for GIS

How the fundamental computer science concepts behind emerging technologies like cryptocurrencies can be applied to improved geospatial data security, integrity, and sharing.

Joshua Tanner
State of Oregon
Geospatial Enterprise Office

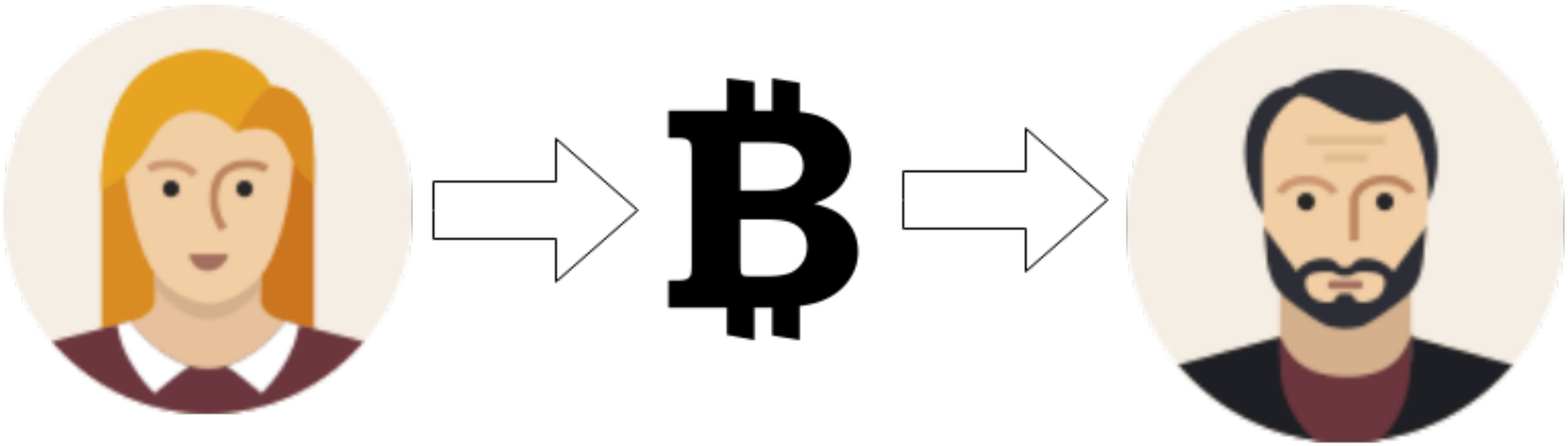
Goals

- Understand key terms
- How it relates to GIS
- Get people talking

A Very Very Basic Explanation of Cryptocurrencies

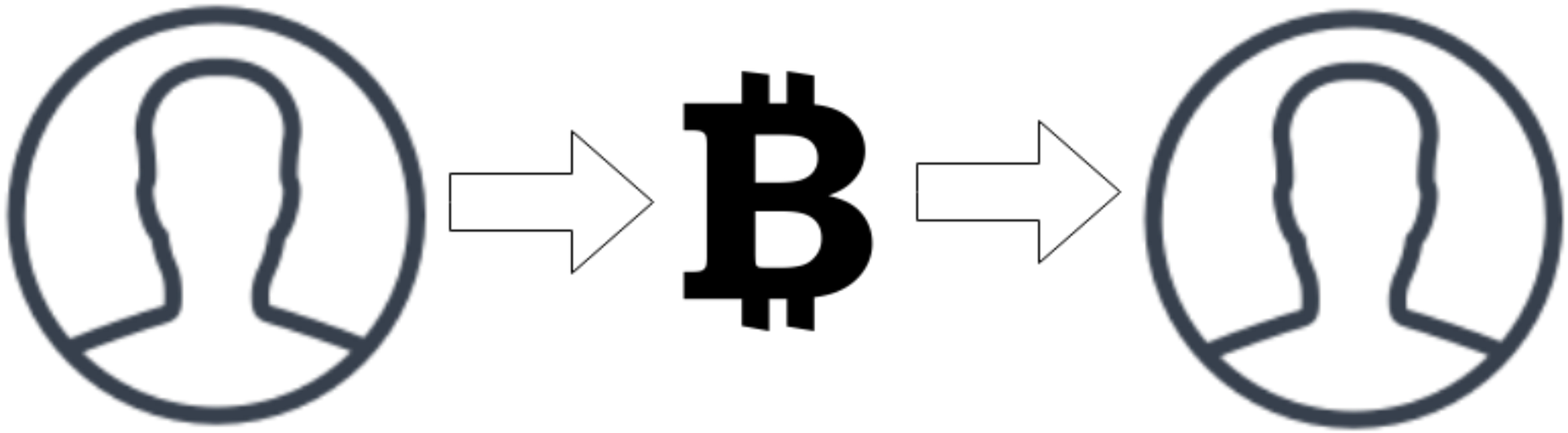
Blockchain

- Public append only ledger
- Add data, can’t change previous data
- Create consensus among others (distributed)
- Relies on challenge to add ‘block’, hard to solve, easy to verify



Alice

Bob



E2354

G1249

Alice

Public Key E2354

Private Key 87Hg7


```
{
  "from": "E2354",
  "to": "G1249",
  "amount": "0.025btc",
  "timestamp": "152287327726l"
}
```


Block 254

Transactions:

[{from: ..., to: ..., amount, timestamp}, ...]

Hash: 398JjnjfLkd

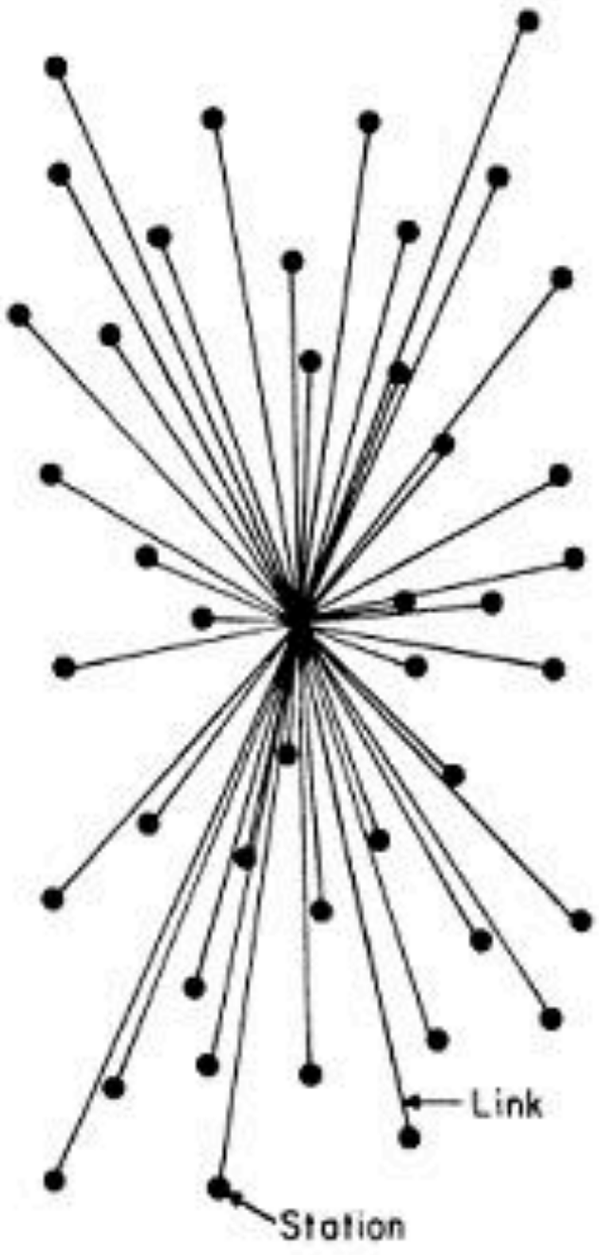
previousHash:

knsajk89N

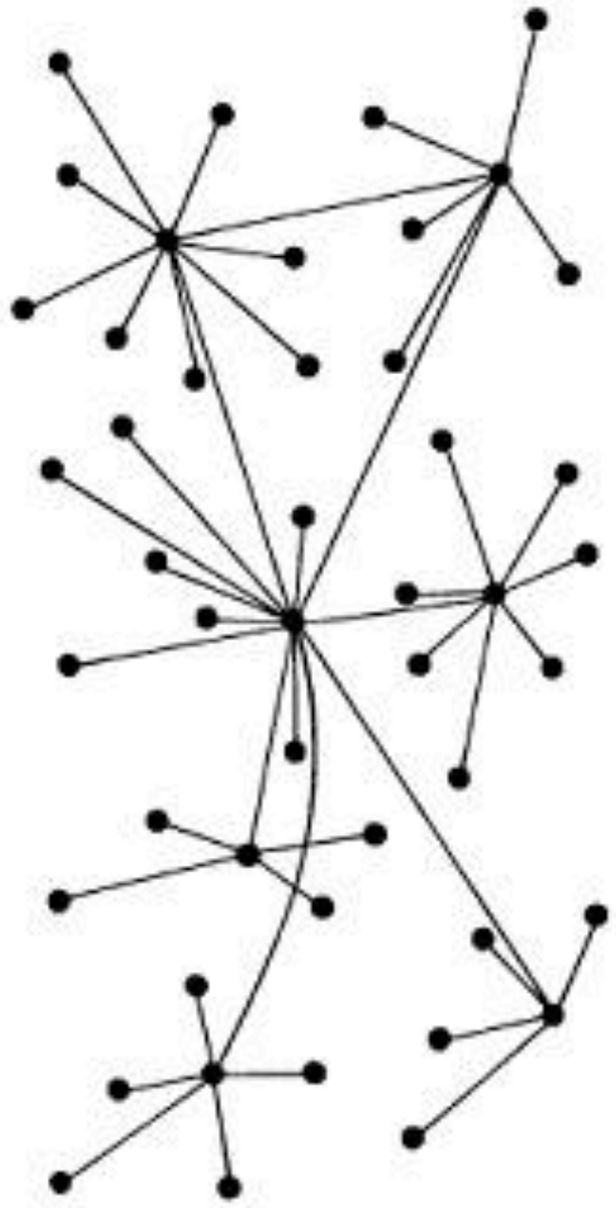
Proof of Work:

43871

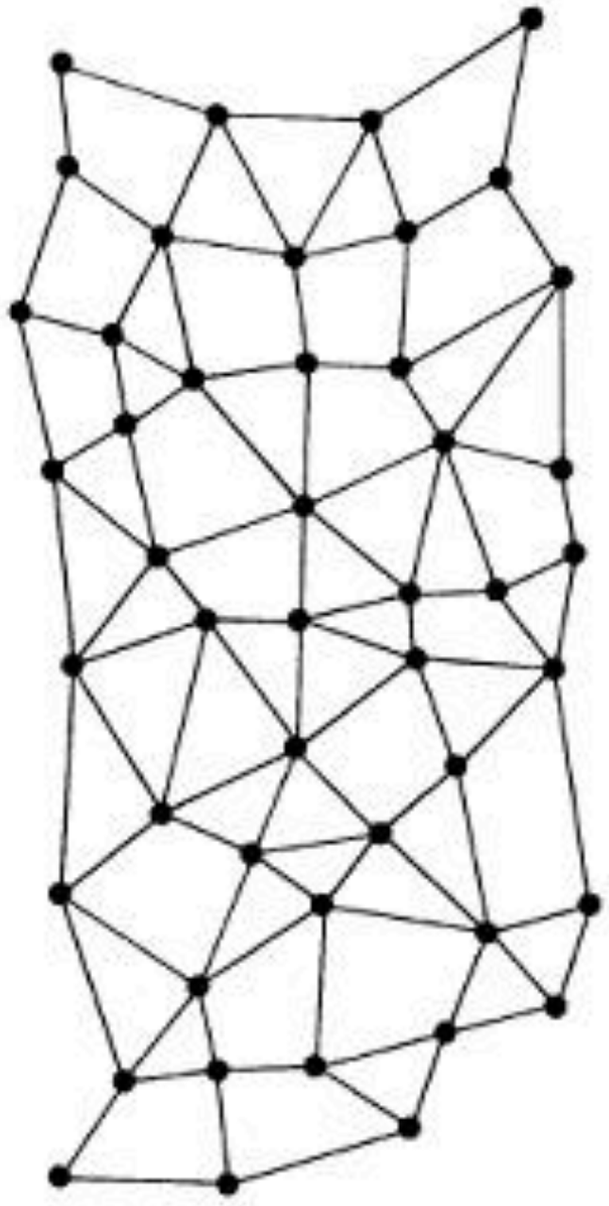
Solve New Block



CENTRALIZED
(A)



DECENTRALIZED
(B)



DISTRIBUTED
(C)

Terms

- Cryptographic Hash
- Merkle Tree
- Root Hash
- Distributed Network
- Public / Private Key Encryption
- Proof of Work

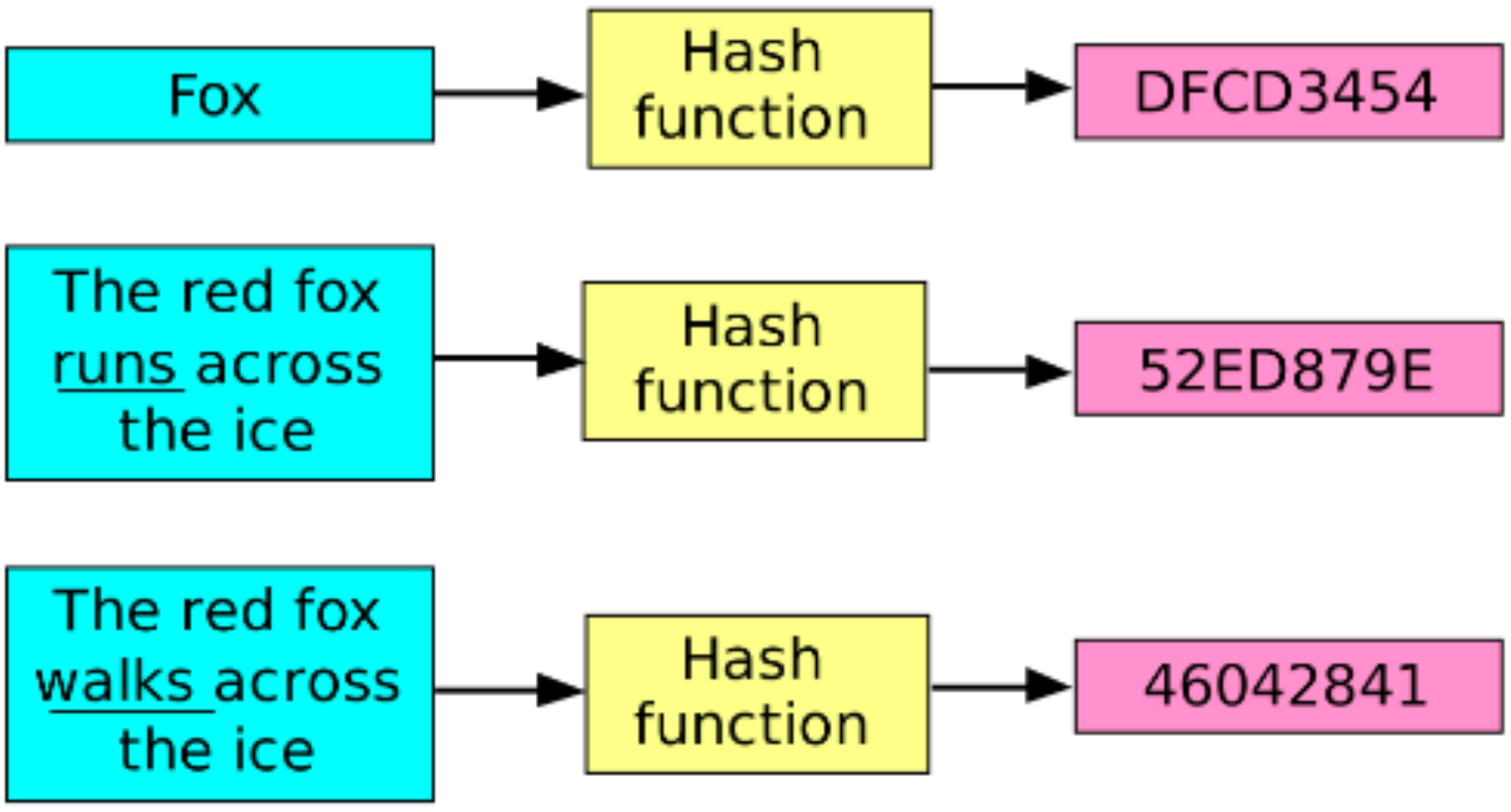
Cryptographic Hash

hash()

ef61a579c907bbed674c0dbcbcf7f7
af8f851538eef7b8e58c5bee0b8cfdac4a

Infinite

Fixed



Hashing Algorithms
Blake, MD5, SHA-256

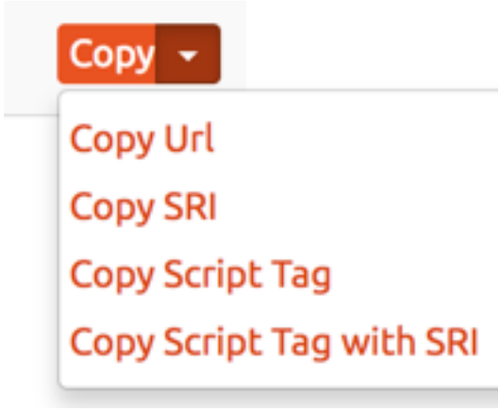
ef61a579c907bbed674c0dbcbcf7f7
af8f851538eef7b8e58c5bee0b8cfdac4a

ef61a579c907bbed674c0dbcbcf7f7
af8f851538eef7b8e58c5bee0b8cfdac4a

GIS 1: Checksum

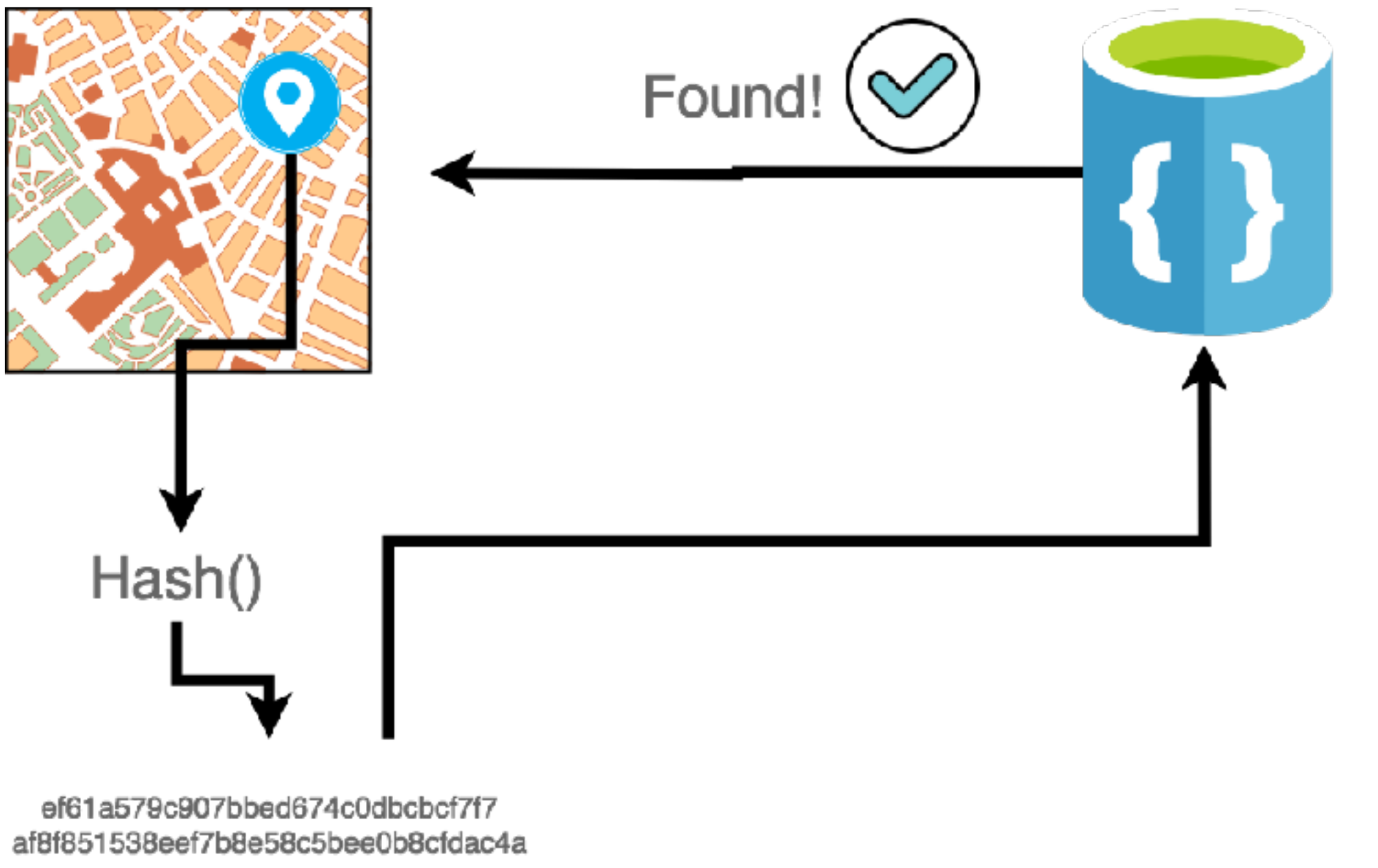
Filename	Checksum
jdk-9.0.4_linux-x64_bin.rpm	sha256: fd1da16430321827c7f4a0ece4e74d042a6632381d1d8e2c679f9de0ba0355cf
jdk-9.0.4_linux-x64_bin.tar.gz	sha256: 90c4ca877c816c3440862cfa36341bc87d05373d53389cc0f2d54d4c8c95daa2
jdk-9.0.4_osx-x64_bin.dmg	sha256: f5c827ab4c3cf380827199005a3dfe8077a38c4d6e8b3fa37ec19ce6ca9aa658
jdk-9.0.4_windows-x64_bin.exe	sha256: 56c67197a8f2f7723ffb03241911510 75cdec0ff0891861e36f3fadda28d556c3
jdk-9.0.4_solaris-sparcv9_bin.tar.gz	sha256: 9f424553d80b8b7337d8c5014dbb8f09dc6d242291d1e73a30e00aaefe47ba89
serverjre-9.0.4_linux-x64_bin.tar.gz	sha256: d29b6b3008c814abd8ab5e4bde9278d6ee7699898333992ee8d080612b5197ca
serverjre-9.0.4_windows-x64_bin.tar.gz	sha256: 6126cffa9f4a937d1435a21815c714654939e1054c9a8539156e40f4c5e54b95
serverjre-9.0.4_solaris-sparcv9_bin.tar.gz	sha256: 88bc237eed5cc49cca47fd8d8e3d1fab125e44c36f83fa3e2f4684f696768c7e
jre-9.0.4_linux-x64_bin.rpm	sha256: 7c7955a9eb4247d8880c696d1328c70b87a0735c3af0fc3f9f60b8827354990c
jre-9.0.4_linux-x64_bin.tar.gz	sha256: 331d6560ba0eadd6266e082e1a3ccd26777c48db881be07cb496805cd301d705
jre-9.0.4_osx-x64_bin.dmg	sha256: 6026fe4463da825d34c9f012a1aa99a77fdf4acb7731811011c6afed7e32a14e
jre-9.0.4_osx-x64_bin.tar.gz	sha256: 61145430ffc932ae0119500603e560df0589dcfb96583014a715b52d376e3cch
jre-9.0.4_windows-x64_bin.exe	sha256: 874b71eeb072163d7a07cf03c3c0f7061e24cf739dc926e7f058a8b6b6dc7edf
jre-9.0.4_windows-x64_bin.tar.gz	sha256: ffcd6d774cfba78d88a1af253eecad0ec3639bdeabdfb3345e61d1c2355267a4
jre-9.0.4_solaris-sparcv9_bin.tar.gz	sha256: ad7adccbebd91d5062e5890fdf0be6e6e231efe19d9eef3e92053a9c07758abc

GIS 2: Subresource Integrity

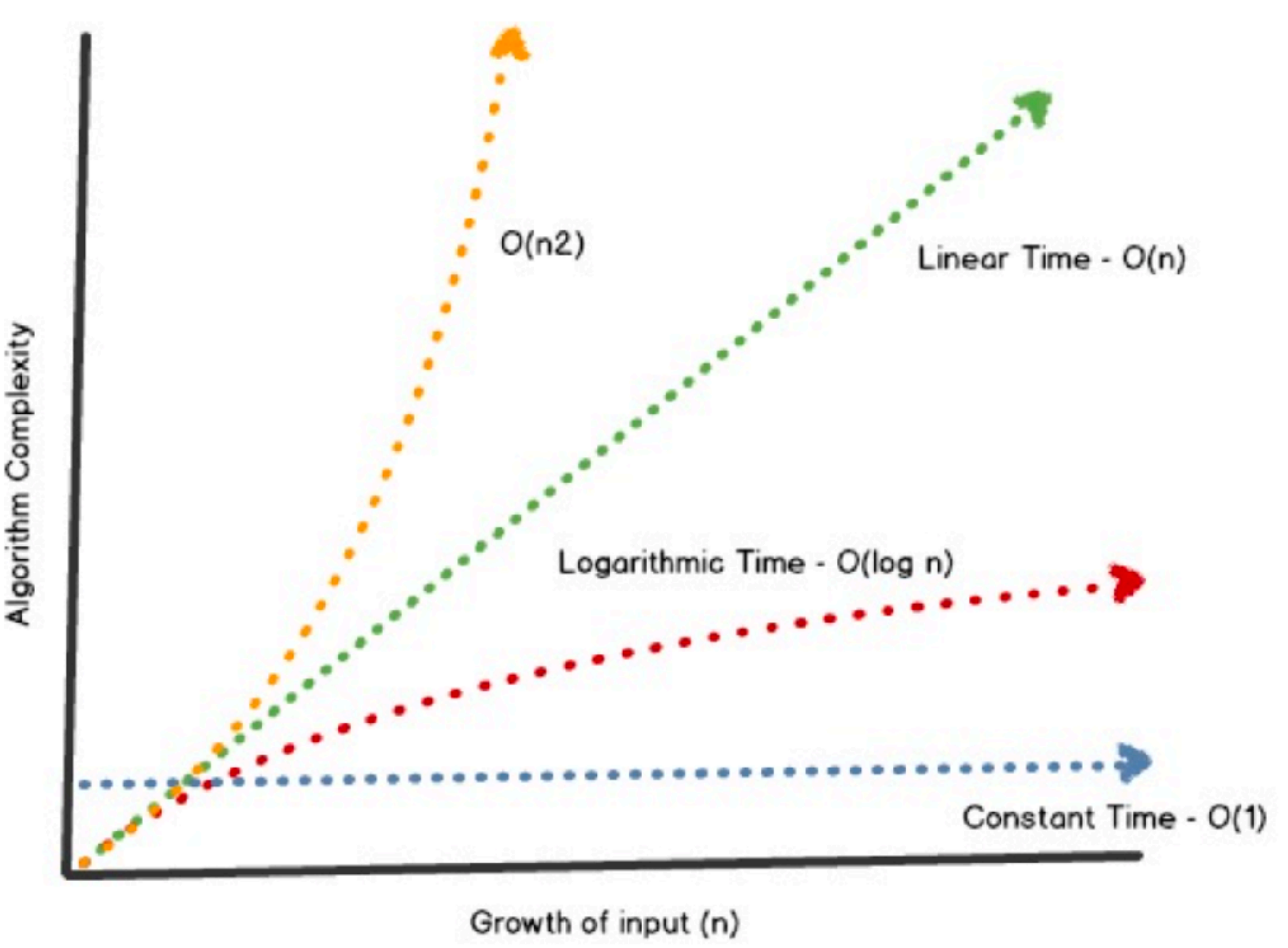



```
<script src="https://example.com/example-framework.js"
        integrity="sha384-oqVuAfXRKap7fdgcCY5uykM6+R9GqQ8K/uxy9rx7HNQ
        crossorigin="anonymous"></script>
```

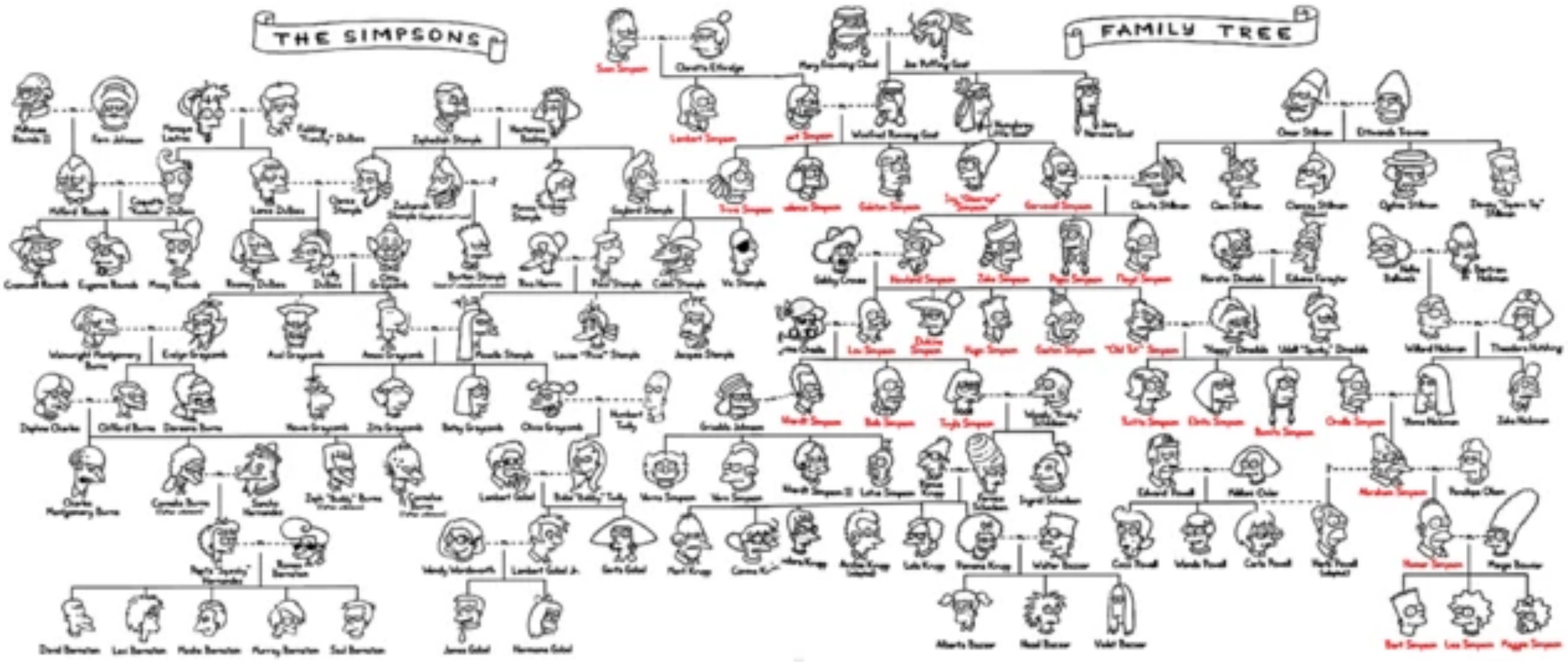
GIS 3: Unique Feature Signatures

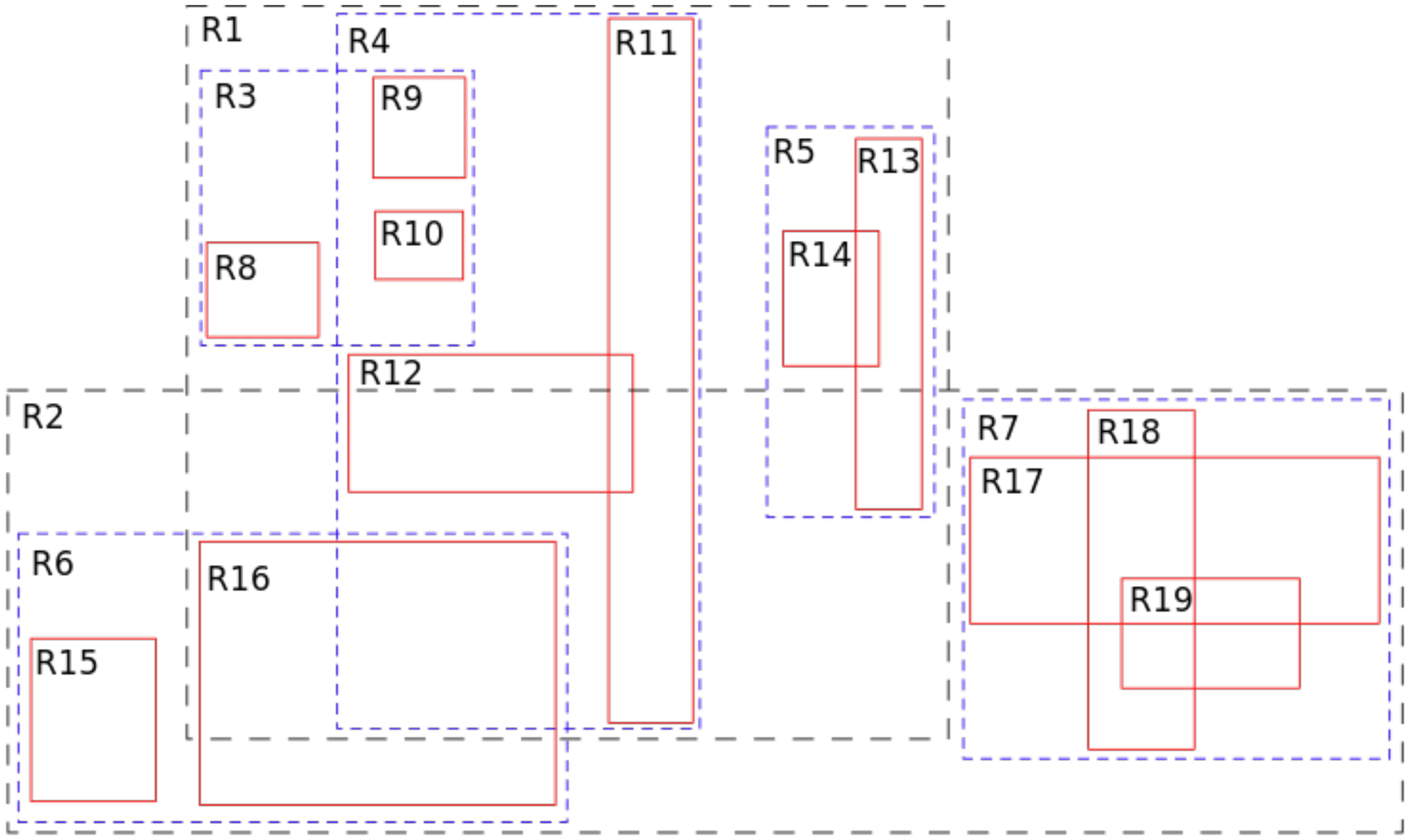


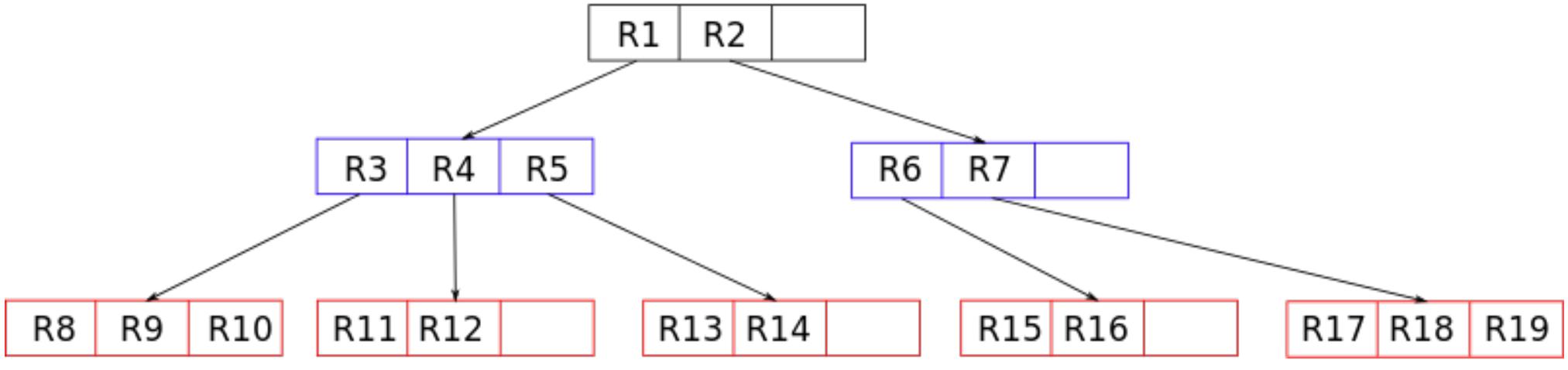
EXAMPLE


Algorithm Complexity
O(n2)
Linear Time - O(n)
Logarithmic Time - O(log n)
Constant Time - O(1)
Growth of input (n)

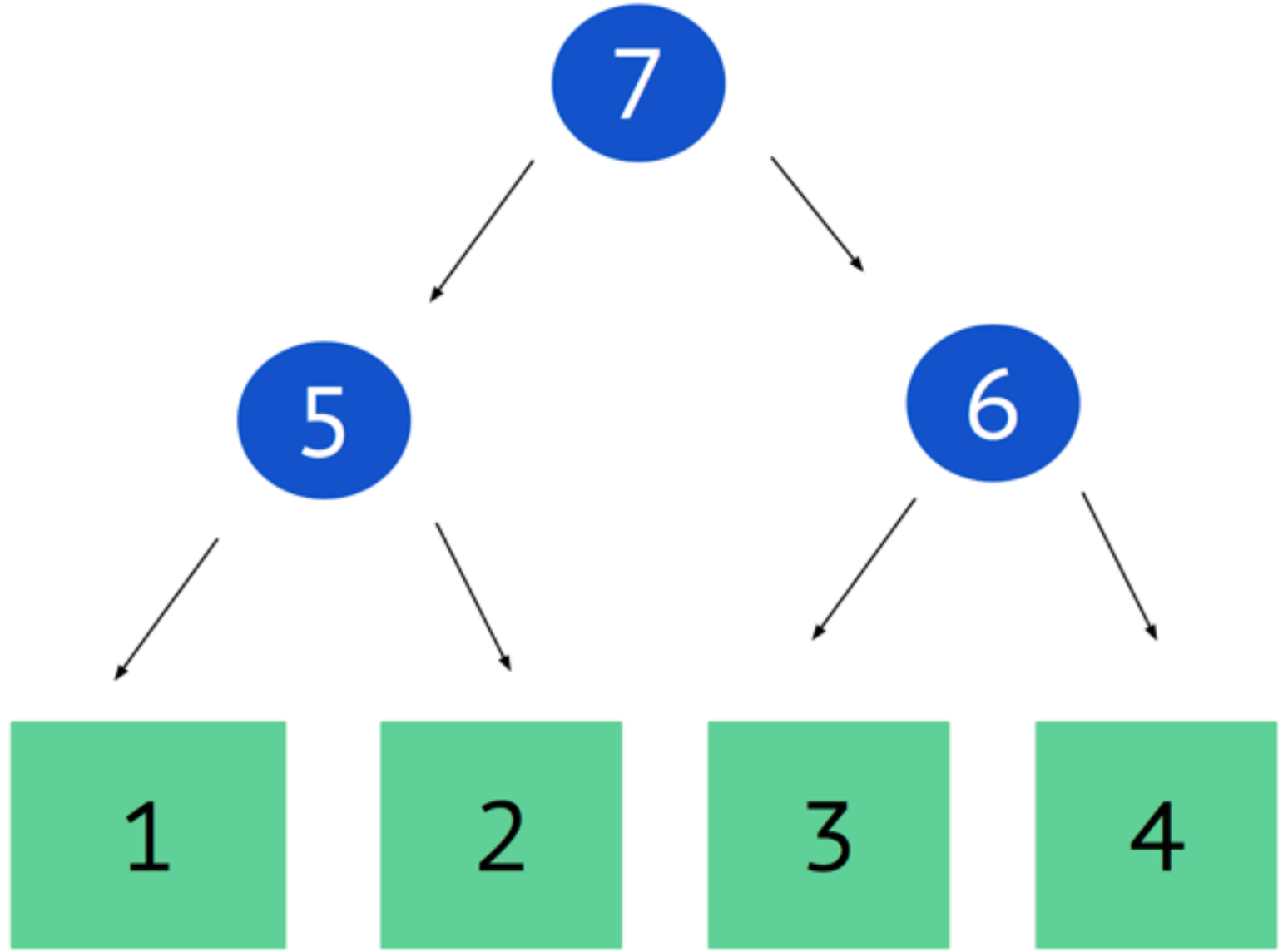
Merkle Tree & Root Hash

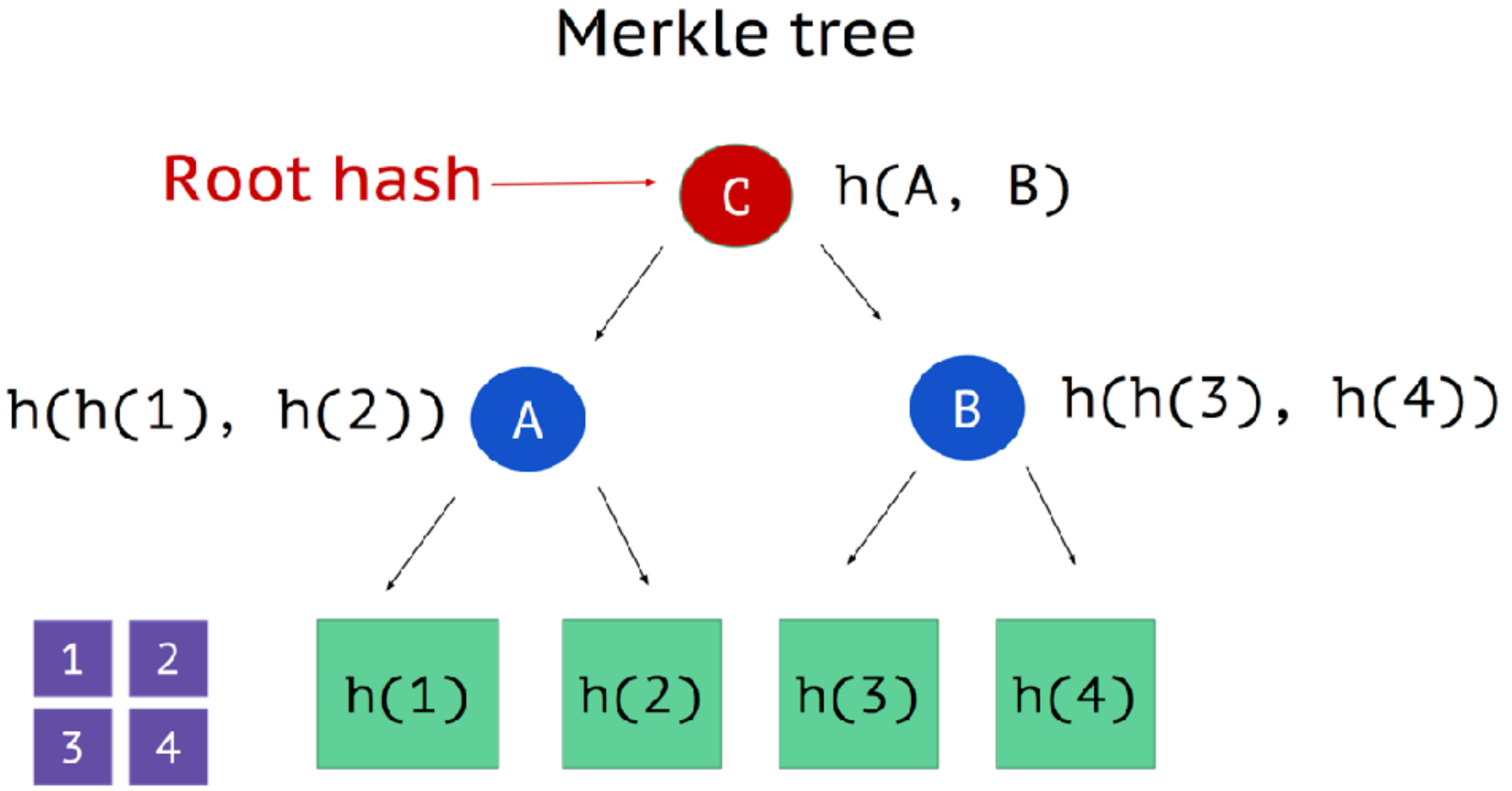

THE SIMPSONS
FAMILY TREE


R1
R4
R11
R3
R9
R5
R13
R10
R14
R8
R12
R2
R7
R18
R17
R6
R16
R19
R15

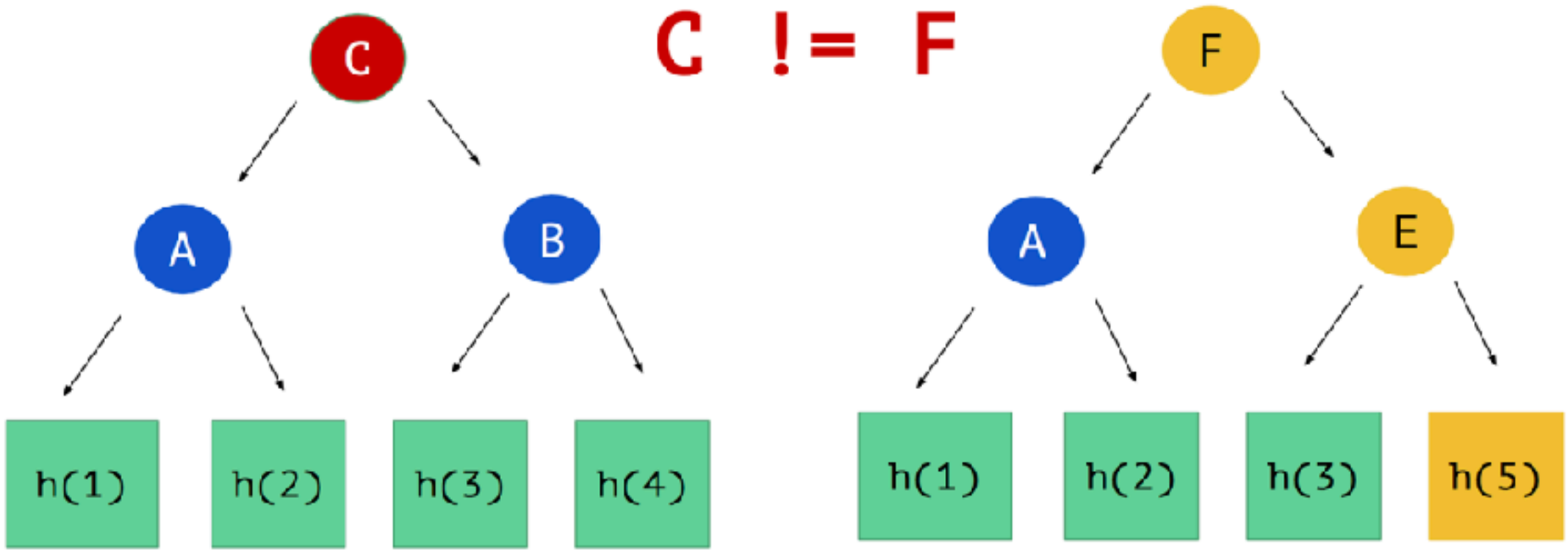

R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15
R16
R17
R18
R19

Regular binary tree




Merkle tree
Root hash
C
h(A, B)
h(h(1), h(2))
A
B
h(h(3), h(4))
1
2
3
4
h(1)
h(2)
h(3)
h(4)

Checking for equality



tree	
.	9c435a86e664be00db0d973e981425e4a3ef3f8d

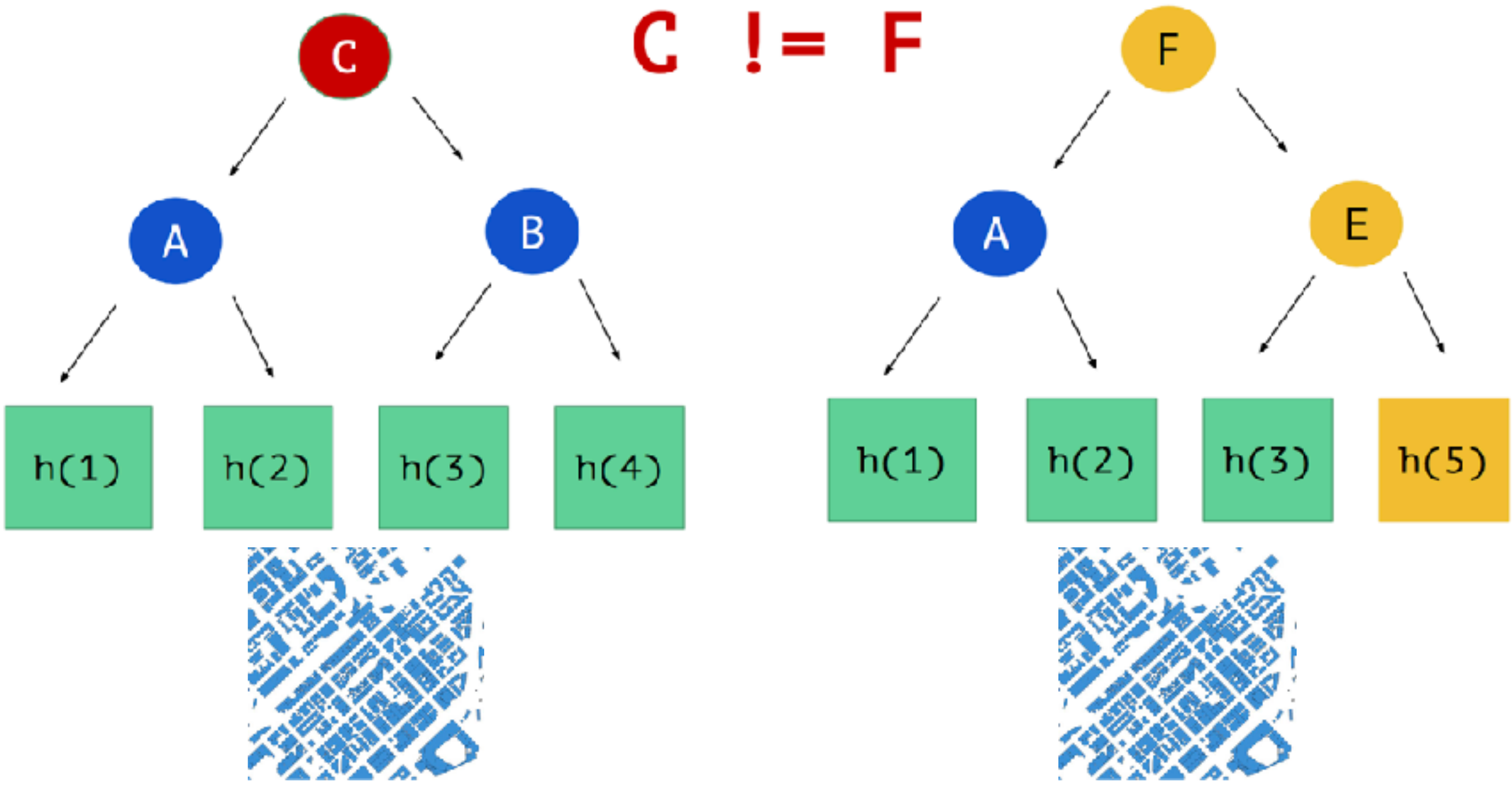
tree	
assets	7cf2a17f3345635d59e063cffddd23573b6e4a75
app.js	29bfcf9fa5824331081b31f0c307806c6f6b6f06

blob

tree	
logo.png	aa1b2fb696a831c89c53f787e03d8
app.css	4c511f16ef2644854d04cabebfced

GIS 1: Syncing Updates

Checking for equality



GIS 2: Rabin Fingerprinting

Content Defined Chunking

CDC algorithms such as Rabin Fingerprints let you chunk a file based on the content of the file itself.

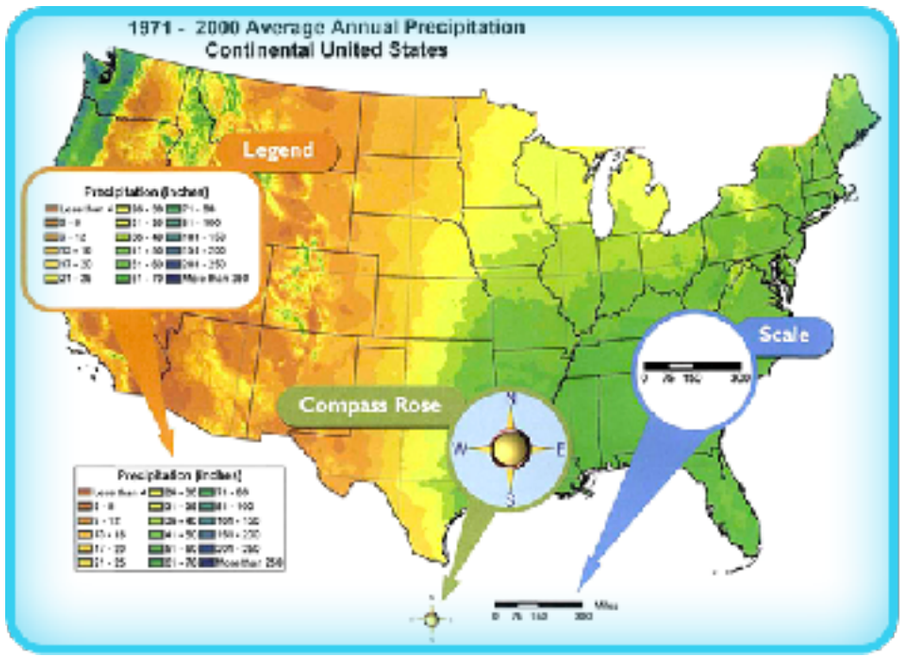
They are an alternative to a simpler approach called Fixed Sized Chunking which uses a hard coded chunk length, but has the downside of not being "shift resistant", meaning when new data is inserted into the file, all chunk boundaries to the right of the insert are changed.

Here is an example. First a file is split up into chunks of roughly even size based on content using a CDC algorithm.

New data is inserted into the middle of the file, causing everything to the right of the insert to get shifted.

File is split up again using the CDC algorithm from before. 4 out of 5 chunks match.

This new chunk is the difference between version 1 and version 2 of the file.



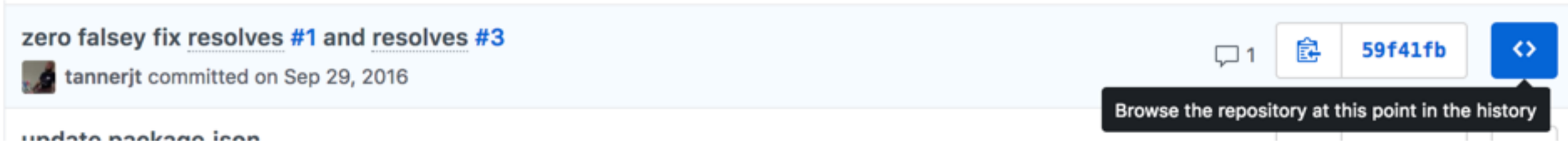
GIS 3: Referencing Specific Versions of Data

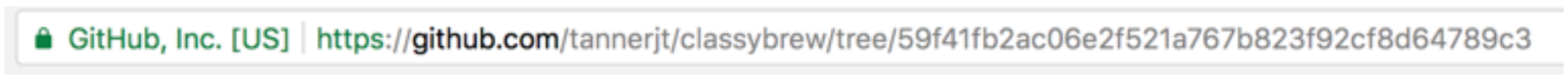
Type the following in the **virtual terminal** in the browser window:

```
$ dat share
```

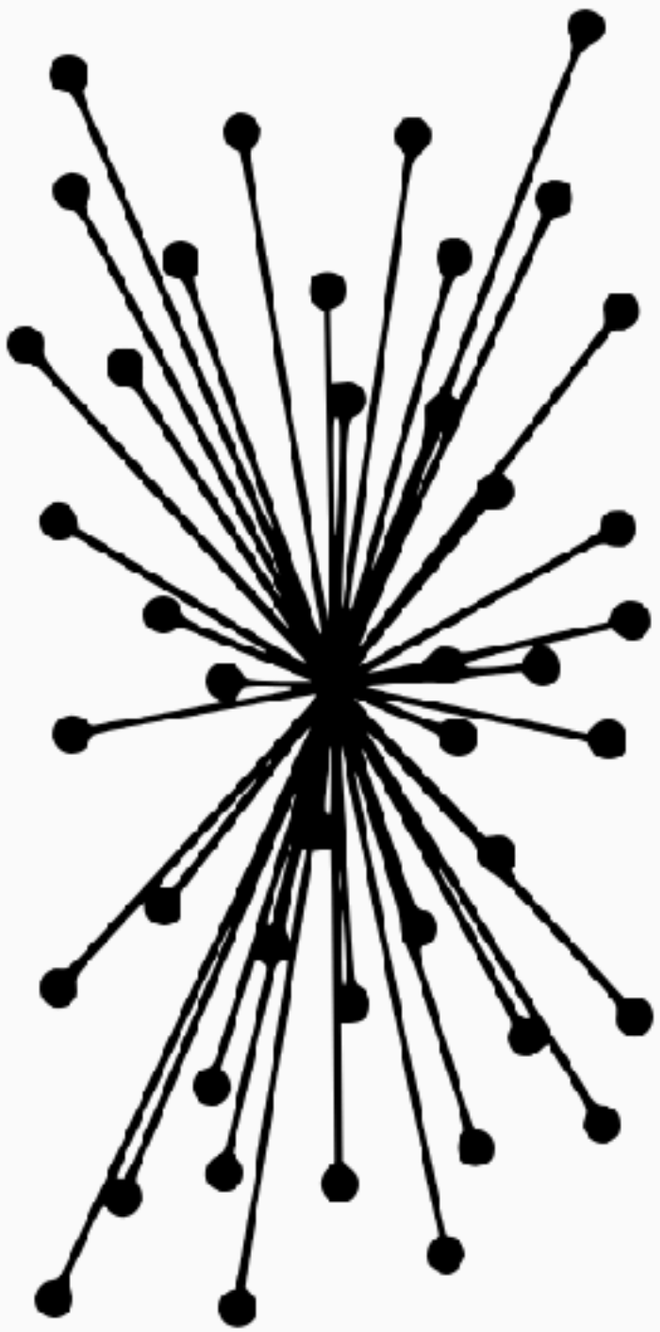
This will create a link, that looks like `dat://...`. Your output in the **virtual terminal** will look something like this:

```
$ dat share
Created new dat
dat://5a4575c632d1a573...
```

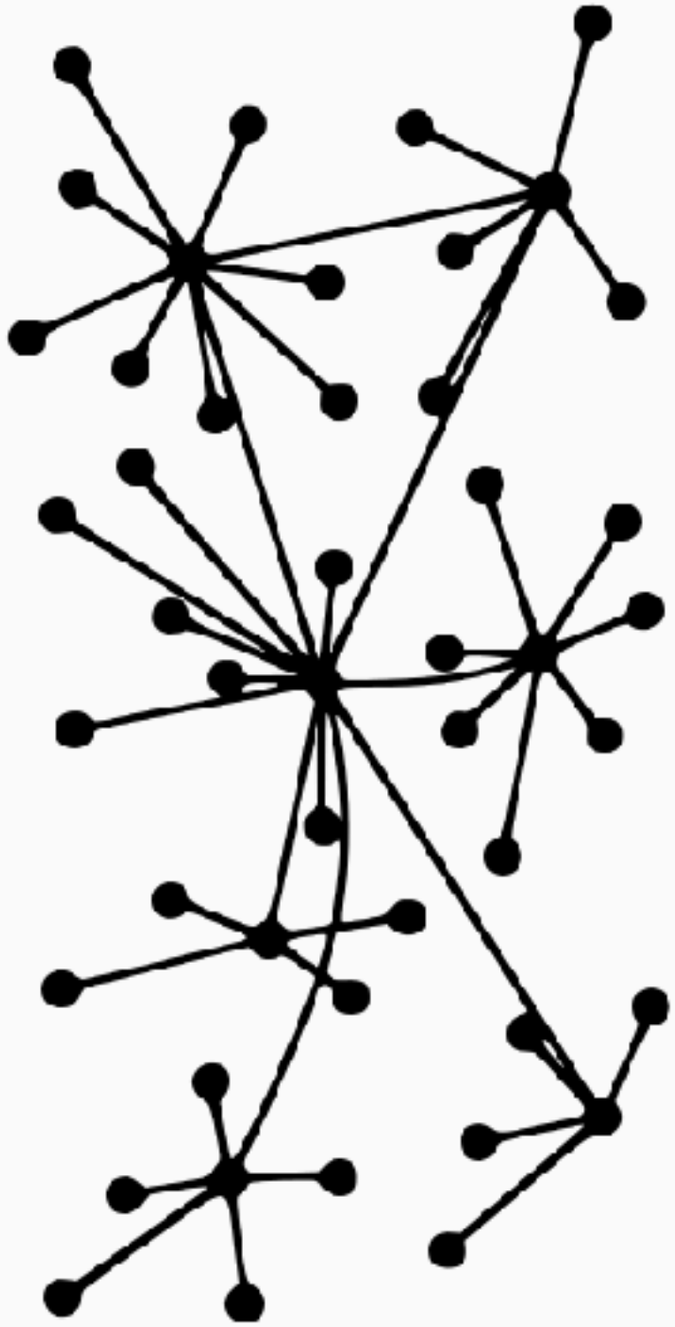




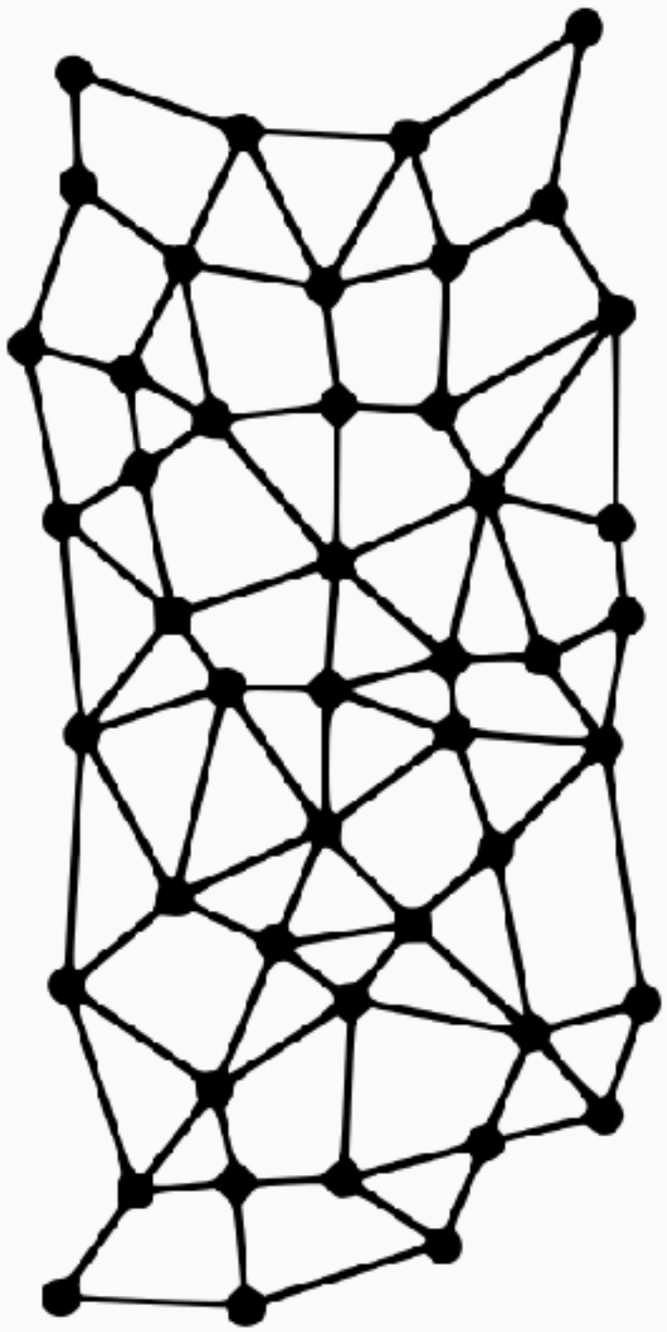
Distributed System



Centralized

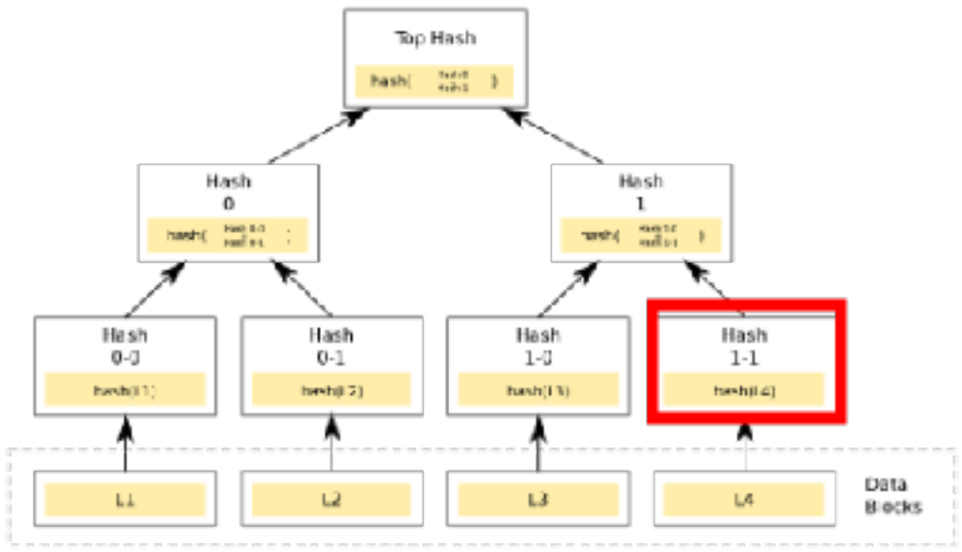


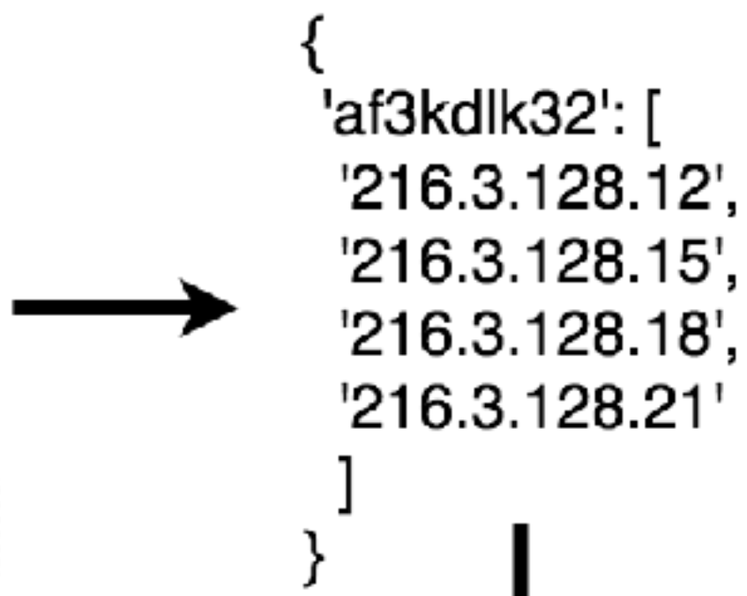
Decentralized

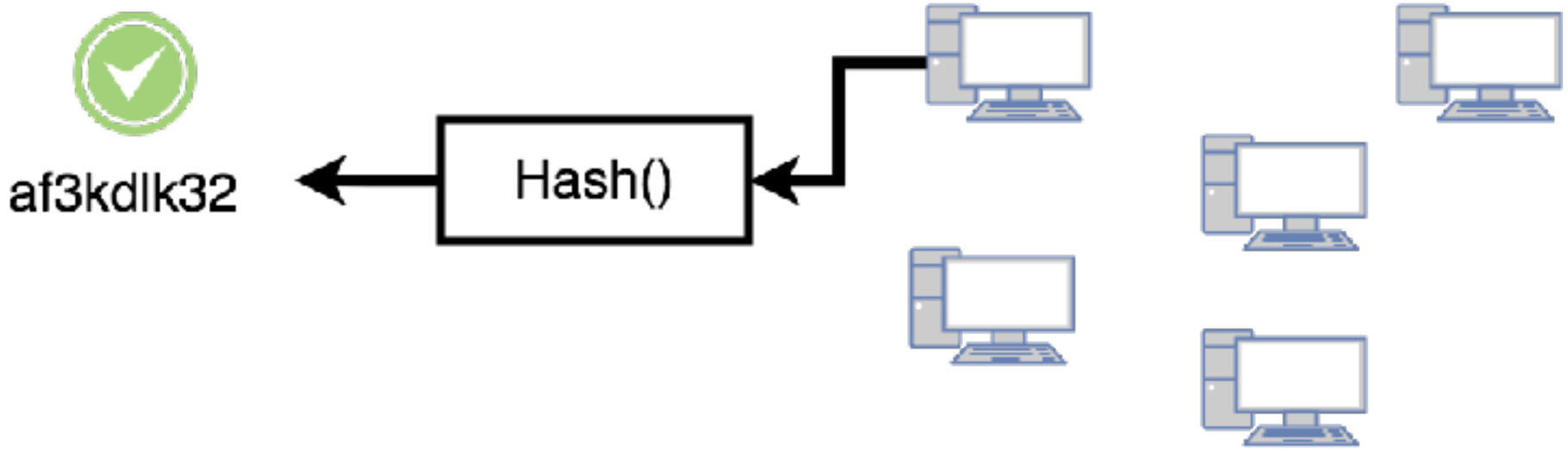


Distributed

Untrusted Peers





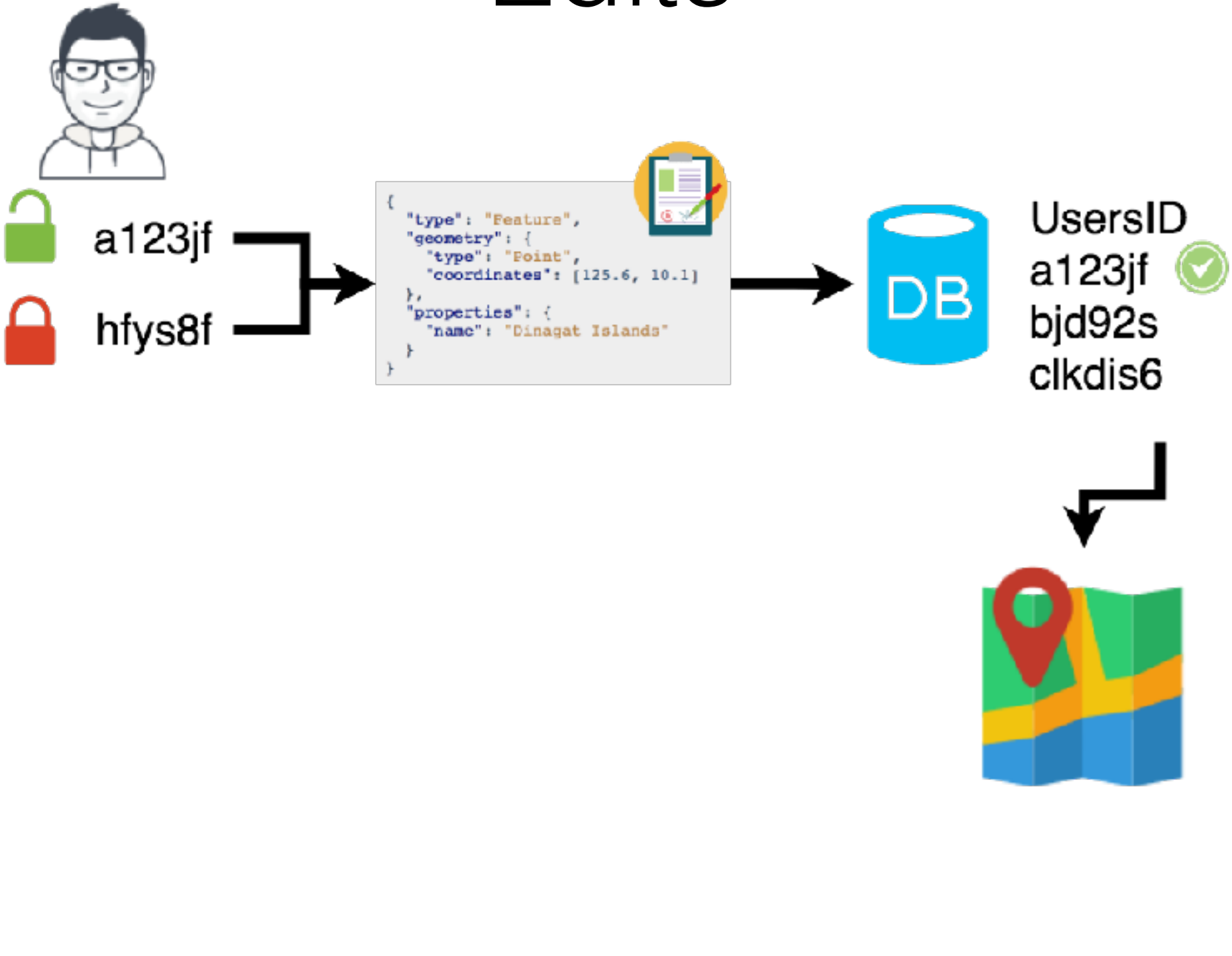


Public / Private Key Encryption

- Bob wants to send a message to Alice
- He wants Alice to know it came from him
- The message contents can be read by others
- Bob gives away copies of his seal and signature
- Bob signs and seals messages before sending to Alice

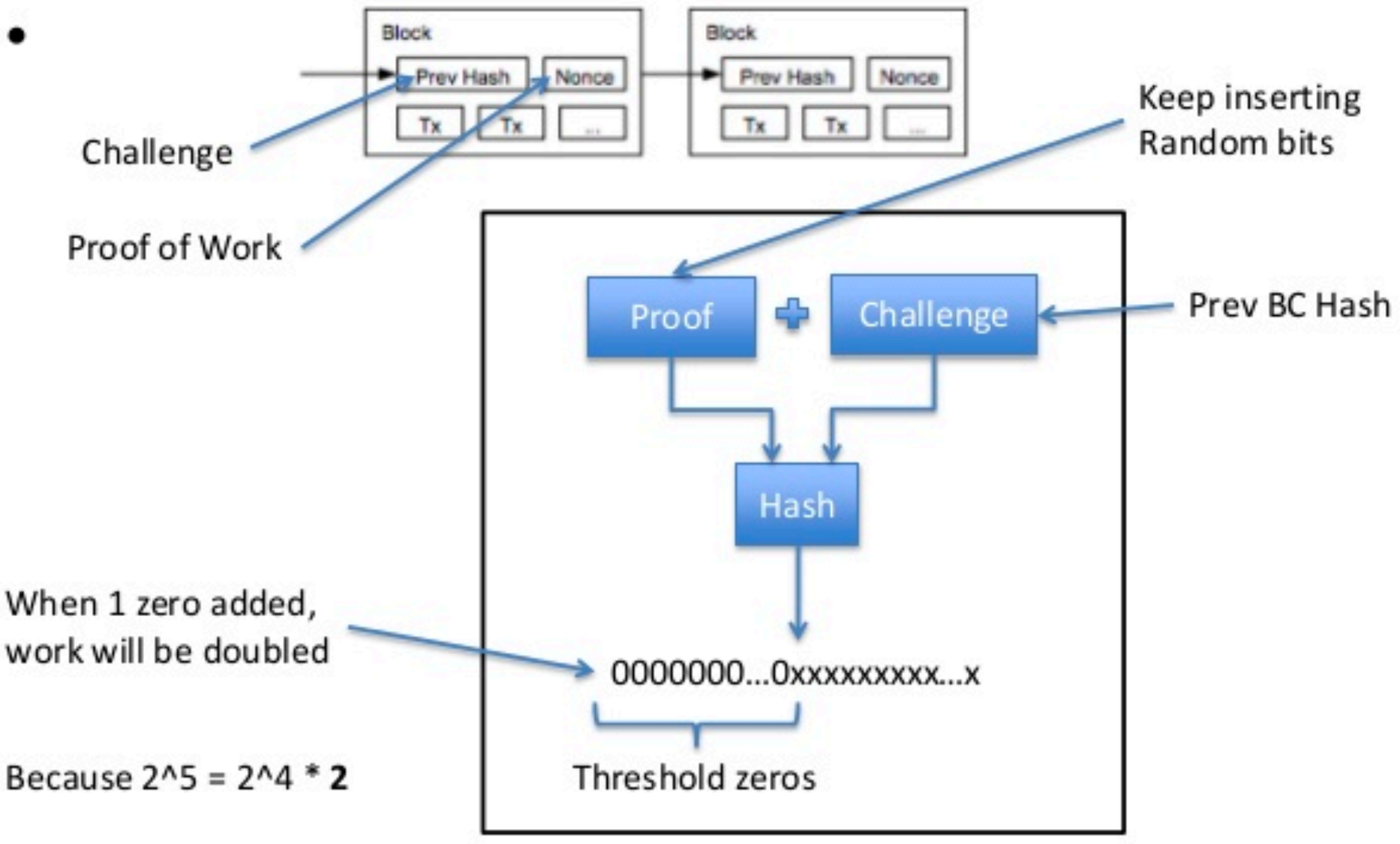
```
// Bob generates a key pair and publishes his public key
let { publicKey, privateKey } = crypto_sign_keypair()

// Bob composes an important message
let message = 'Tonight is the meetup at 7pm'

// And signs it using his private key
let signature = crypto_sign_detached(message, privateKey)

// Alice gets the signed message and wants to verify it
let isVerified = crypto_sign_verify_detached(signature, message, publicKey)

console.log({message: message, isVerified: isVerified})
```

GIS 1: Secure Updates / Edits



Proof of Work

Proof of Work

- 

Hashcash.IO

Protects Against Web Spam

It is typical for "SEO Marketing Companies" to take advantage of forums and blogs by creating worthless posts and comments with links to a website they are promoting.

If your web-property allows user-generated content, you are likely to use CAPTCHA. CAPTCHAs are not perfect:

- They annoy your visitors.
- They provide an illusory sense of security.

Today it is possible to buy access to an API which solves any CAPTCHA for just $0.70 per 1000 CAPTCHA images. And do you think your customer will be happy to try to solve one of these ridiculous CAPTCHAs?

Secures Against Brute Force Attacks

Many modern web-applications are susceptible to brute force attacks.

Take a typical login form, for example. Hackers can compromise account security by trying every possible password combination. They can also leverage a large network of proxy servers to parallelize this attack.

Forcing their browser to work hard makes it too expensive and slow for hackers to perform a brute force attack.

Applications in GIS

- Throttle server requests (DDoS attacks)
- Comment forms for user input
- Brute force password requests
- Reducing SPAM email!!

- DAT Whitepaper
- Rabin Fingerprinting
- Merkle Trees Explained
- Blockchain Explained in 5 Levels of Difficulty
- Khan Academy - Bitcoin
- A Dive Into Spatial Search Algorithms
- An Intro to DAT's Cryptography
- GitHub Example - Hashing Spatial Features
- Beaker Browser (Distributed Web Browser)

Questions?