CSBC2000

Week 2 | Class 2

Cybersecurity Fundamentals

Last Class

- IAM in blockchains
- HLF ecosystem (Zoom recording up)
- Public permissioned, private permissionless

This Class

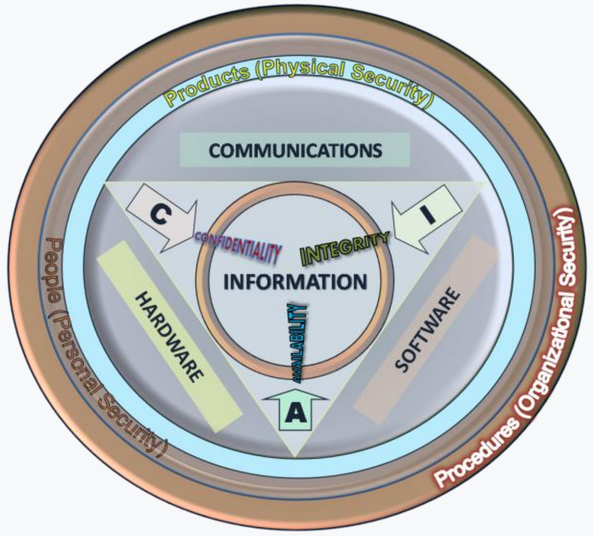
- The CIA Triad
- Hashing: merkle trees, bloom filters
- Applications of hashing in DLT

Information Security

- **Infosec**, is the practice of protecting information by mitigating information risks.
- Typically involves preventing or reducing the probability of unauthorized/inappropriate access to data, or the unlawful use, disclosure, disruption, deletion, corruption, modification, inspection, recording or devaluation of information.

The CIA Triad

- Confidentiality, Integrity, Availability
- Recall the Byzantine Generals problem
- Refer to data-in-motion and data-at-rest


Products (Physical Security)
COMMUNICATIONS
C
I
CONFIDENTIALITY
INTEGRITY
INFORMATION
AVAILABILITY
HARDWARE
SOFTWARE
A
People (Personal Security)
Procedures (Organizational Security)

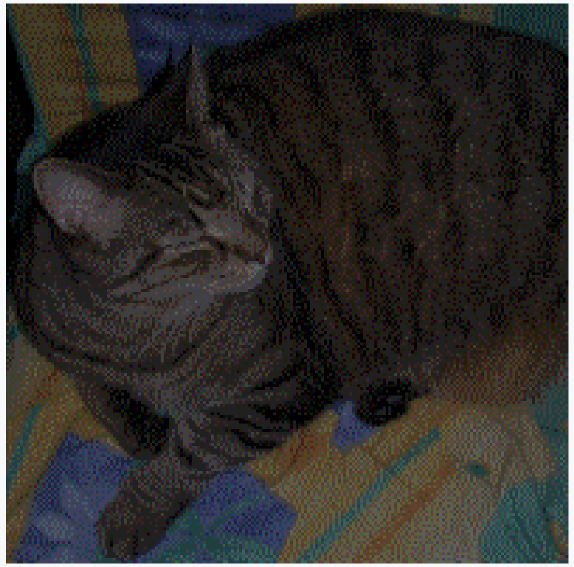
The CIA Triad: Confidentiality

- Controlling access to data to prevent unauthorized disclosure
- Confidential data must be accessible only to those parties that have been granted access to it
  - Those who are unauthorized are actively prevented from obtaining access

The CIA Triad: Confidentiality

- Can be compromised by direct attacks designed to gain unauthorized access in order to steal or tamper with data. E.g. MITM, keyloggers, ransomware
  - Can also be violated unintentionally through human error, carelessness, or inadequate security controls.
- Can be enforced by data classification and labeling; strong access controls and authentication mechanisms; encryption of data in process, in transit, and in storage; **steganography**; remote wipe capabilities; and adequate education and training for all individuals with access to data

Steganography



The CIA Triad: Integrity

- Integrity is about ensuring that data has not been tampered with and, therefore, can be trusted
- Receiver sees exactly what the sender sends them; nothing more, nothing less
- E.g. Banking customers need to be able to trust that their banking information and account balances have not been tampered with

The CIA Triad: Integrity

- Can be compromised by an *attack vector* (such as tampering with intrusion detection systems, modifying configuration files, or changing system logs to evade detection) or unintentionally, through human error, lack of care, coding errors, or inadequate policies, procedures, and protection mechanisms
- Can be secured by *checksums,* digital certificates, Trusted *CAs*, intrusion detection systems, auditing, version control

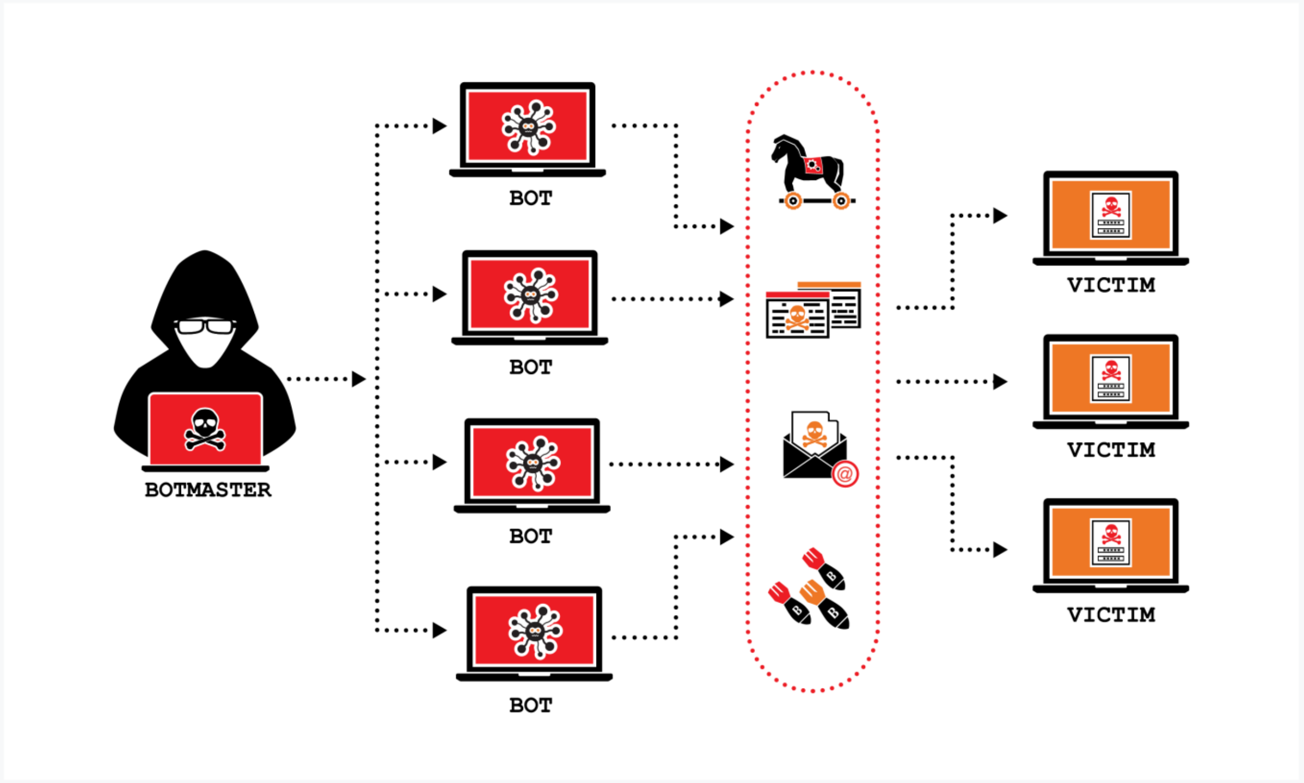
The CIA Triad: Availability

- Availability means that networks, systems, and applications are up and running
- Recall CAP
- Ensures that authorized users have timely, reliable access to resources when they are needed

The CIA Triad: Availability

- Can be compromised by many things, including hardware or software failure, power failure, natural disasters, and human error
- Can be protected by redundancy (recall horizontal vs vertical scaling), hardware fault tolerance (for servers and storage), regular software patching and system upgrades, backups, comprehensive disaster recovery plans

DDoS

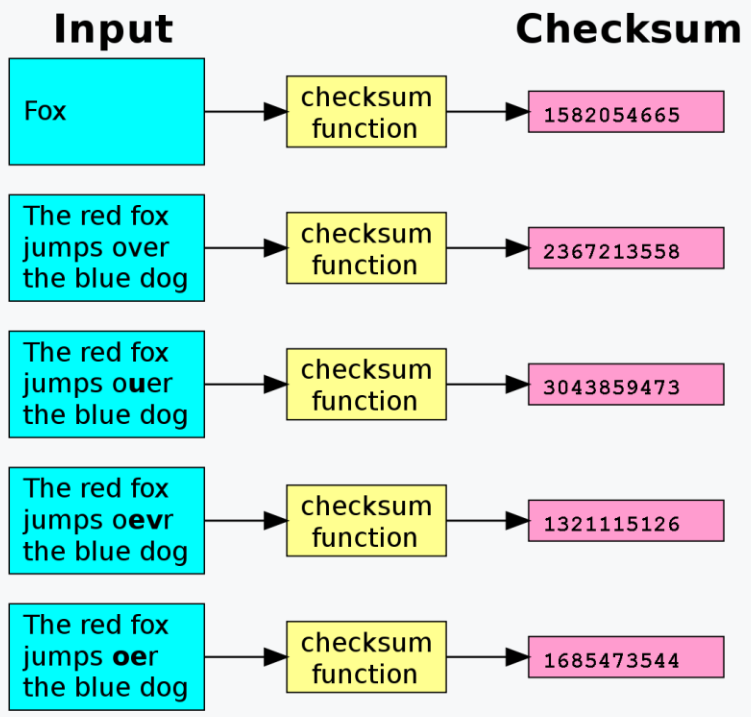


Hashing

- We have some idea of hashing at this point
- A function that takes any data and gives you unique fingerprint
  - This fingerprint is often called a *digest* (because it's compression)
- Crucial for integrity verification

Checksum

- Can help verify integrity of data with a hash
- A good hash is sensitive to extremely minute changes
- Need to make sure there's no *collision*

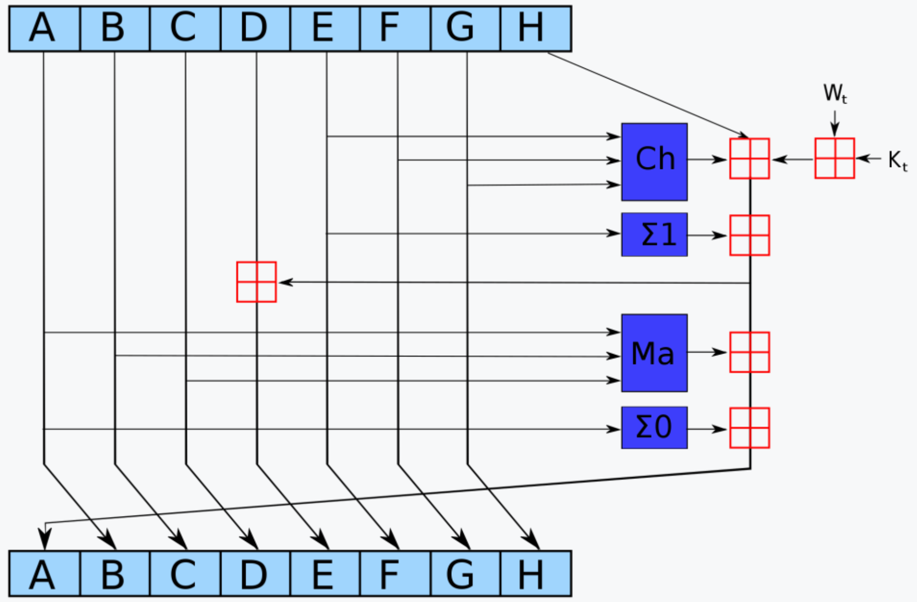

Input
Checksum
Fox
The red fox jumps over the blue dog
The red fox jumps **ou**er the blue dog
The red fox jumps o**ev**r the blue dog
The red fox jumps **oe**r the blue dog
checksum function
1582054665
2367213558
3043859473
1321115126
1685473544

Hash Functions

- How do we ensure collisions don't happen?
- We need a *Cryptographic Hash Function*
- These are extremely resistant to *brute-forcing* and hence collisions
  - This is why you should have long, complex passwords!
- CHFs have the property of producing seemingly random output for any input, no matter how "close" they are

Hash Functions

- Some popular CHFs:
  - MD5 (broken)
  - SHA1 (broken)
  - SHA2
  - SHA3 (Keccak)

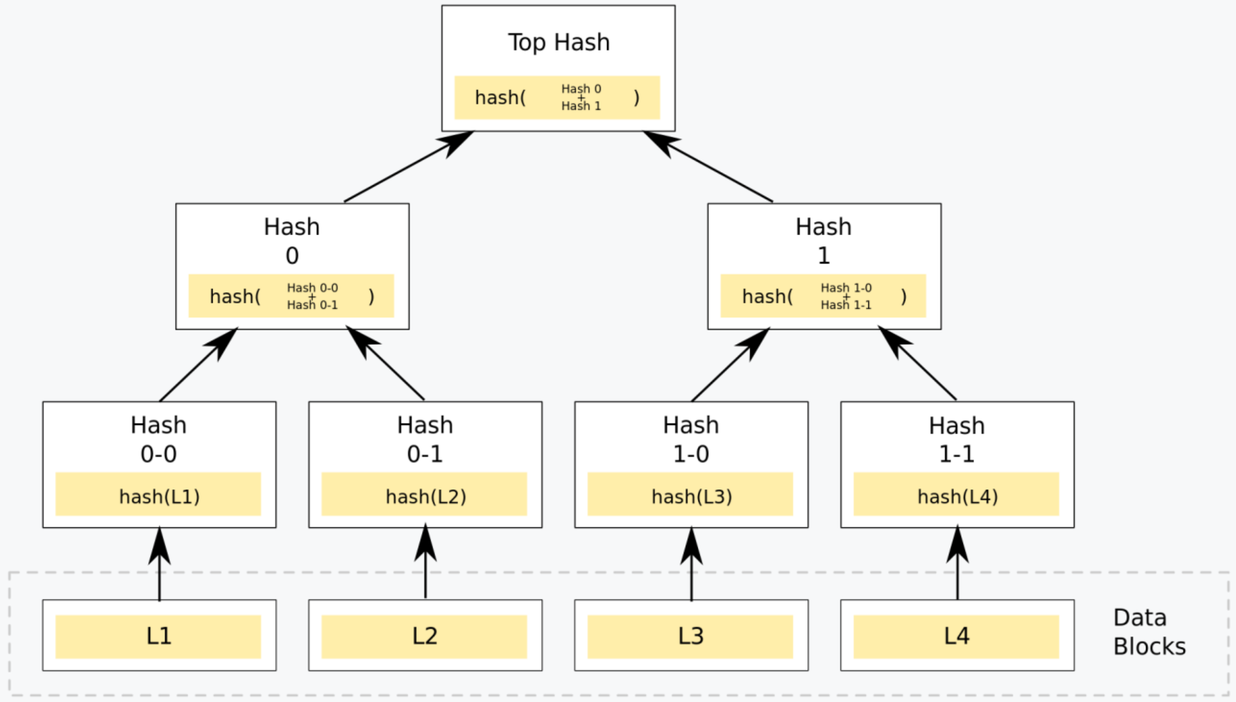


Merkle Trees

- A *tree* in which every leaf node is labelled with the cryptographic hash of a data block, and every non-leaf node is labelled with the cryptographic hash of the labels of its child nodes

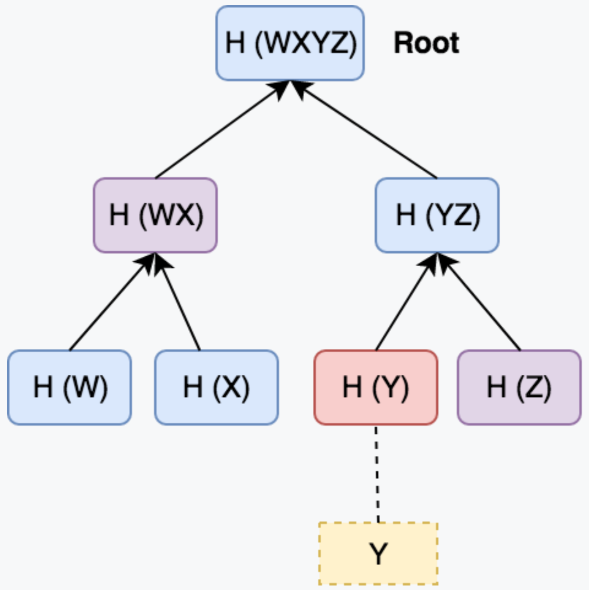
- Hash trees allow efficient and secure verification of the contents of large data structures

Merkle Trees



Merkle Tree Verification

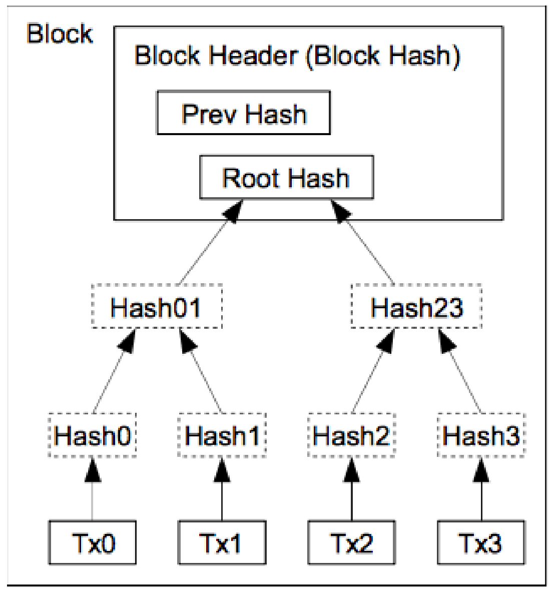
- Verification happens using a given merkle root from a trusted source (e.g. the block header)
  - Get all the data from the untrusted network
  - Compute the Merkle root from them
  - If they differ, data lacks integrity
- It's very efficient!



Merkle Tree In Blockchains

- Merkle root is stored in block header

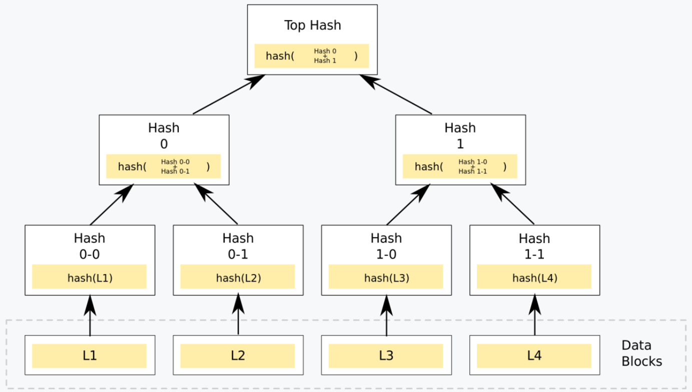
- It is calculated from all txs at the time the block is mined



Transactions Hashed in a Merkle Tree

Merkle Tree Second Preimage attack

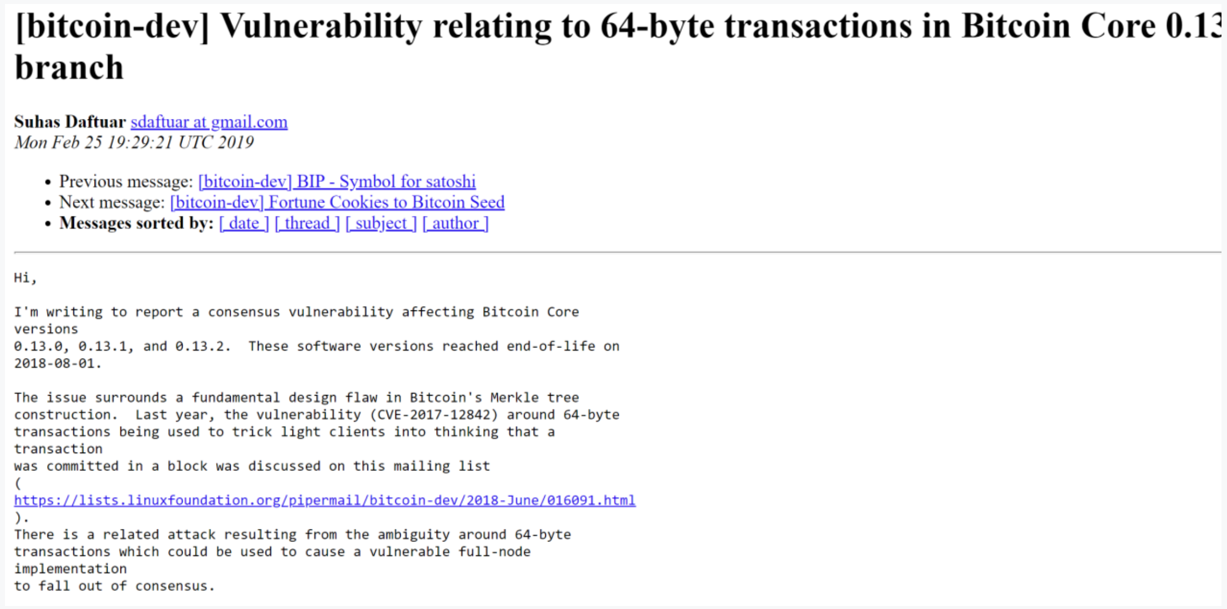
- The Merkle hash root does not indicate the tree depth, enabling a second-preimage attack in which an attacker creates a document other than the original that has the same Merkle hash root
- For example, an attacker can create a new document containing two data blocks, where the first is h(h(0-0)+h(0-1)), and the second is hash h(h(1-0)+h(1-1))



Merkle Tree Second Preimage attack

- Bitcoin is still vulnerable to this attack!

- Although it would take significant work to produce a block where the transactions produce hashes which can then be deserialized as valid transactions



[bitcoin-dev] Vulnerability relating to 64-byte transactions in Bitcoin Core 0.13 branch

Suhas Daftuar sdaftuar at gmail.com
Mon Feb 25 19:29:21 UTC 2019

- Previous message: [bitcoin-dev] BIP - Symbol for satoshi
- Next message: [bitcoin-dev] Fortune Cookies to Bitcoin Seed
- **Messages sorted by:** [date] [thread] [subject] [author]

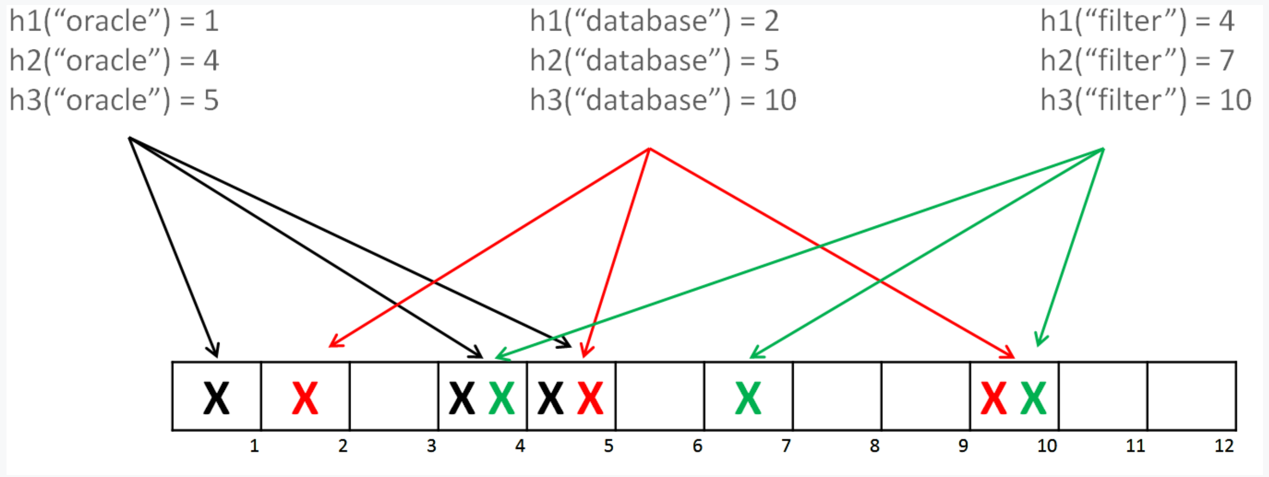
```
Hi,

I'm writing to report a consensus vulnerability affecting Bitcoin Core
versions
0.13.0, 0.13.1, and 0.13.2.  These software versions reached end-of-life on
2018-08-01.

The issue surrounds a fundamental design flaw in Bitcoin's Merkle tree
construction.  Last year, the vulnerability (CVE-2017-12842) around 64-byte
transactions being used to trick light clients into thinking that a
transaction
was committed in a block was discussed on this mailing list
(
https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2018-June/016091.html
).
There is a related attack resulting from the ambiguity around 64-byte
transactions which could be used to cause a vulnerable full-node
implementation
to fall out of consensus.
```

Bloom Filter

- A Bloom filter is a space-efficient probabilistic data structure, conceived by Burton Howard Bloom in 1970, that is used to test whether an element is a member of a set
- False positives are possible, but false negatives are not – in other words, a query returns either "possibly in set" or "definitely not in set"
- Elements can be added to the set, but not removed; the more items added, the larger the probability of false positives

Bloom Filter

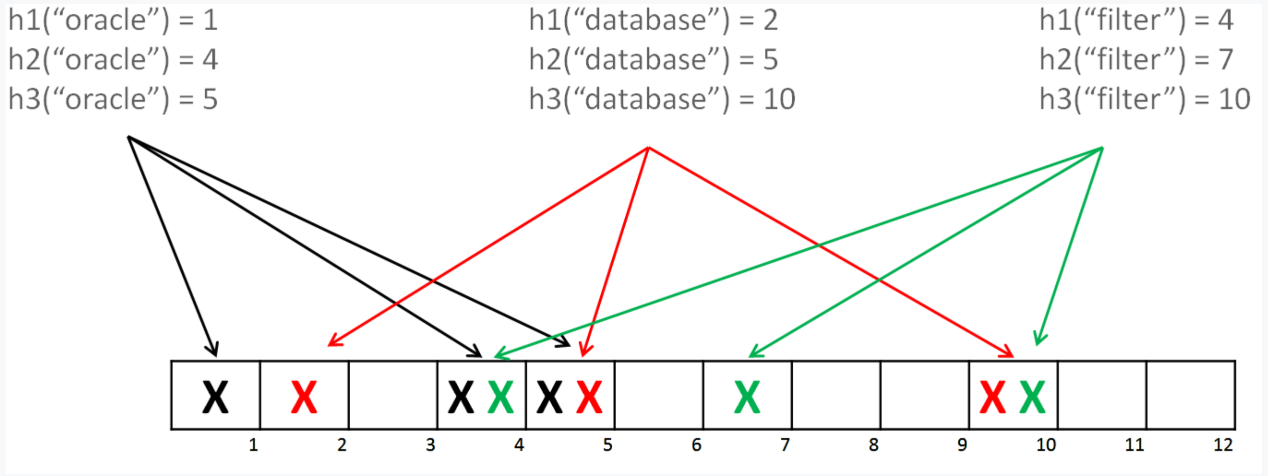
- Have a bunch of hash functions, a number of buckets
- When data is inputted, each hash function makes a mark on one of the buckets (uniformly distributed
- The buckets after the operation become the signature of the data



Bloom Filter

- Can pass this bucket around and get the verification that data belongs in the filter
- False positive happens when h1 for data1 is set and h2 for data2 is not set but they happen to be the same spot
- In this case, data2 is given

a false positive



Bloom Filter: SPV

- Simplified Payment Verification (SPV) is a method employed by some thin clients within the bitcoin network in order to verify transactions without the requirement to keep an entire copy of the blockchain. As such these thin nodes use bloom filters to specify only the transactions they are interested in receiving updates for

Questions/Comments?

Let's code!