Introduction and Cryptographic Tools

50.037 Blockchain Technology

Paweł Szałachowski

About Me

- ISTD's faculty member since Aug 2017
- https://pszal.github.io ; pawel@sutd.edu.sg ; 1.402-35
- Research and systems building
  - Blockchain and Internet-level Consensus
  - Public-key and Trust Infrastructures, SSL/TLS, Internet security...
  - SCION Internet Architecture: https://www.scion-architecture.net/

Organization

- Teik Guan Tan (TA); teikguan_tan@mymail.sutd.edu.sg
- Schedule consultations via email
- Letter Grade, 2-3-7 (12 credits)
    - Projects: 30% + 30% (deadlines: 15 Mar, 26 Apr, both until 23.59:59 SGT)
    - Final: 40% (2 May 9:00-11:00 SGT, room TBC)
- Plagiarism will **not** be tolerated
- Textbook: http://bitcoinbook.cs.princeton.edu/
    - The library has it but a free preprint is available on the website
    - Recommended reading will be listed at the end on every lecture
    - Other related-courses: https://crypto.stanford.edu/cs251/ ; http://soc1024.ece.illinois.edu/teaching/ece598am/fall2016/ ; https://achievement.network/ ; …

High-level Picture

- Blockchains, Distributed Ledgers, Cryptocurrencies, …
- Beside all the hype what is actually new here?
- Great combination of
  - *cryptography*: how to protect data and computation?
  - *distributed systems*: how distributed components can communicate and coordinate their actions?
  - *economics and game theory*: how to design systems/protocols such that decisions of “rational” individuals help in achieving the system/protocol goals?
- If you are skeptical about cryptocurrencies (immature/risky technology, govs, …), the good news is that you will learn fundamental concepts from the fields above

Course

- Stated goals: you can read that in the course description
- Hidden goal: get you interested in cryptography and systems security
  - More research needed!
- Course structure
  - Introduction, Cryptography, Bitcoin, Altcoins, Ethereum, Scalability, Privacy, Permissioned ledgers, Ecosystem, misc.
  - Self-learning is appreciated
- … and when I say “crypto” almost certainly I mean “cryptography”

Cryptographic Tools

Cryptography

- Cryptography: art and science of encryption (ciphers)
- More than encryption (other primitives)
    - hash functions, MACs, (P)RNG, RSA, DH, ZK proofs….
- Higher-level constructions
    - secure channel, key server, PKI
- Real-world systems

Cryptography

- Threat model
  - understand what and against whom you are trying to protect
- Cryptography is very difficult
  - proofs but with many assumptions, implementation issues, side-channel attacks, security vs. performance
- Cryptography is the easy part
  - systems are very complex and without well-defined boundaries
- Cryptography is not the solution
  - e.g., generating secure keys vs. their management

Cryptographic Hash Functions

- H: $\{0,1\}^* \rightarrow \{0,1\}^n$
  - for an arbitrarily long string produces a fixed-size output
    - output is called **digest**, or **fingerprint**, or just **hash**
      - usually between 128 and 1024 bits
- Many applications
  - data integrity, checksums, key fingerprints, authentication codes, digital signatures, …

Requirements

- Collision resistance
  - it is hard to find $m_1 \neq m_2$ such that $H(m_1) = H(m_2)$
- Pre-image resistance (one-way property)
  - given a hash value x it should be difficult to find any message m such that $x = H(m)$
- 2^{nd} pre-image resistance
  - given an input m_1 it should be difficult to find different input m_2 such that $H(m_1) = H(m_2)$

Birthday Attack

- Generic attack against hash functions
  - *What is the minimum number of of people in a room, that the chance that two of them will have the same birthday exceeds 50%?*
    - 23
  - N different values, choose k elements, then there are k(k-1)/2 pairs of elements, each of which has 1/N chance of being a pair of equal values
    - chance of finding a collision is close to k(k-1)/2N, and when k~= sqrt(N) this is close to 50%
- For a hash function that outputs *n* bits it is possible to find a collision in about $2^{n/2}$ steps as sqrt(2^n) = $2^{n/2}$

Security

- The **ideal hash function** behaves like a random mapping from all possible input values to the set of all possible output values
- An attack on a hash function is a non-generic method of distinguishing the hash function from an ideal hash function
- Security
  - Collision attack: $2^{n/2}$ steps
  - Pre-image attacks: 2^n steps

Real Hash Functions

- Should be
  - deterministic, fast, secure, easy to analyze, …
- MD5 (insecure), SHA1 (insecure)
- SHA2 (still secure), SHA3 (still secure)

Proof-of-Work (PoW)

Alice

Bob

Hi Bob, I need to talk to you.

N = random()

H(N) = d98d1ce48

I’m pretty busy, prove it is important, find d98d1ce48.

H(1) = 8c887396a

H(2) = 1b3ebc206

H(3) = ae810a43f6

…

H(N) = **d98d1ce48**

It is N !

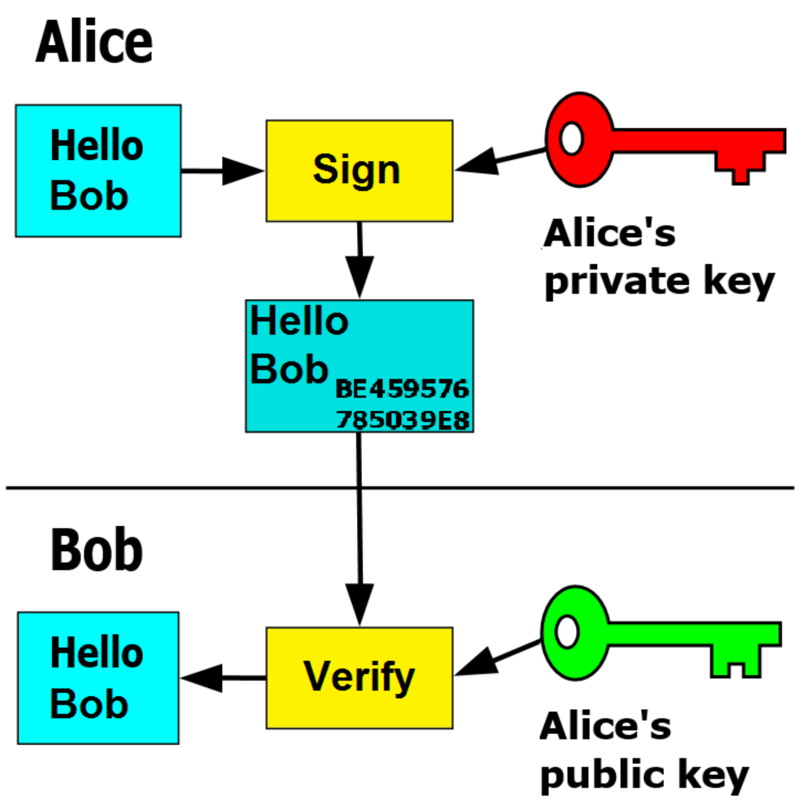
Ok, it must be important, let’s talk.

Non-interactive PoW

- Hashcash
  - SPAM prevention
  - To send email client has to prove some work
  - New header field introduced
    - X-Hashcash: 1:20:060408:adam@cypherspace.org::1QTjaYd7niiQA/sc:ePa
    - H("1:20:060408:adam@cypherspace.org::1QTjaYd7niiQA/sc:ePa") = **00000**a4a8bd07bddbdb0c4ea9ddb2d29b8d1cc5e

Digital Signatures

- Gen()
  - returns a key pair (i.e., public and private key)
- Sign(priv_key, msg)
  - Signs the message using the private/secret key. Returns the signature
- Verify(pub_key, msg, sign)
  - Verifies the signatures of the message, using the public key. Returns boolean (true/false).
- Examples: RSA, DSA, ECDSA, …



Digital Signatures Provided Properties

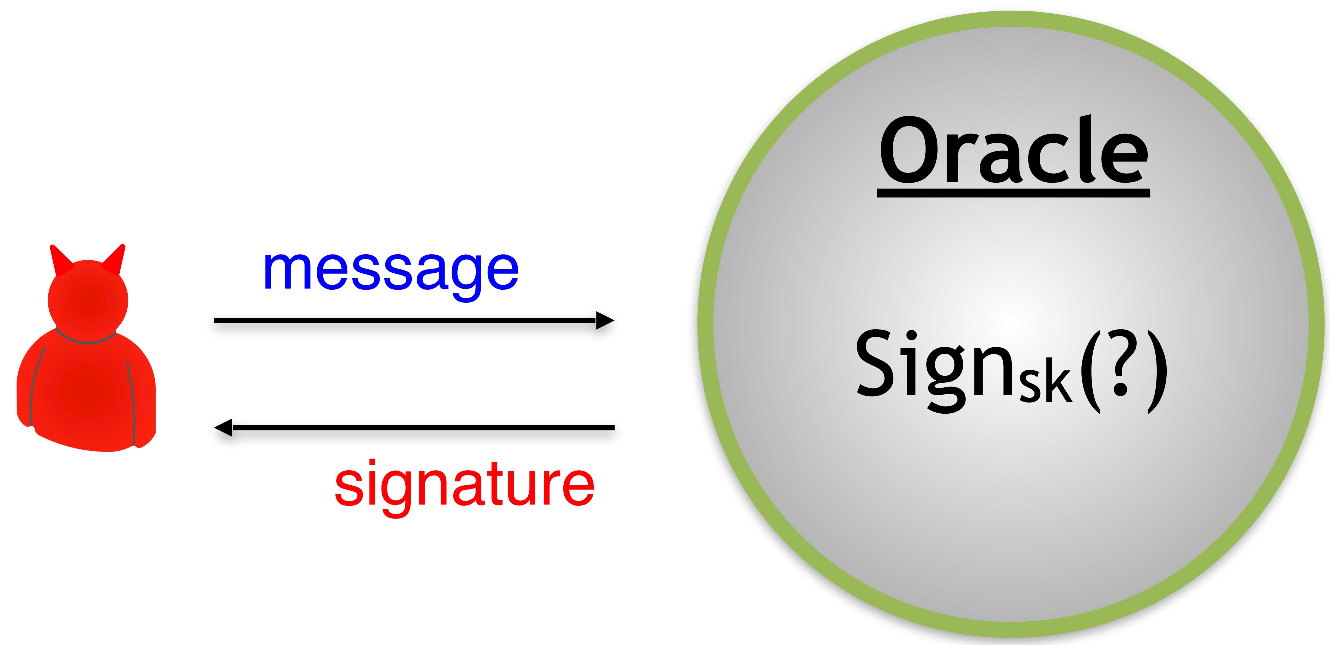
- Authentication
  - a party verifying the message knows that the owner of the corresponding private key has signed it. Sometimes it is stronger, e.g., when public keys are bound to identities.
- Non-repudiation
  - signer cannot deny having sign the message,
- Integrity
  - the message was not modified.

Security Property

Unforgeability

Given one or more msg-signature pairs **[x_i , Sign$_{sk}$(x_i)]**, *it is computationally infeasible to compute any msg-signature pair* **[x, Sign$_{sk}$(x)]** *for any new input* **$x \neq x_i$**

Queries:
{„Hello world” ➔ d80c9d…,
„Hello world2” ➔ 828c82…,
„I’m Alice” ➔ bdbb07…,
…}



Can adversary (after querying) generate a new message and its valid signature?

Commitments and Authenticated Data Structures

Commitments

- Commit phase: publish x = H(“I know that …”)
- Reveal phase: reveal the message

WikiLeaks @wikileaks Follow

pre-commitment 1: John Kerry
4bb96075acadc3d80b5ac872874c3037a386f
4f595fe99e687439aabd0219809

12:08 AM - 17 Oct 2016

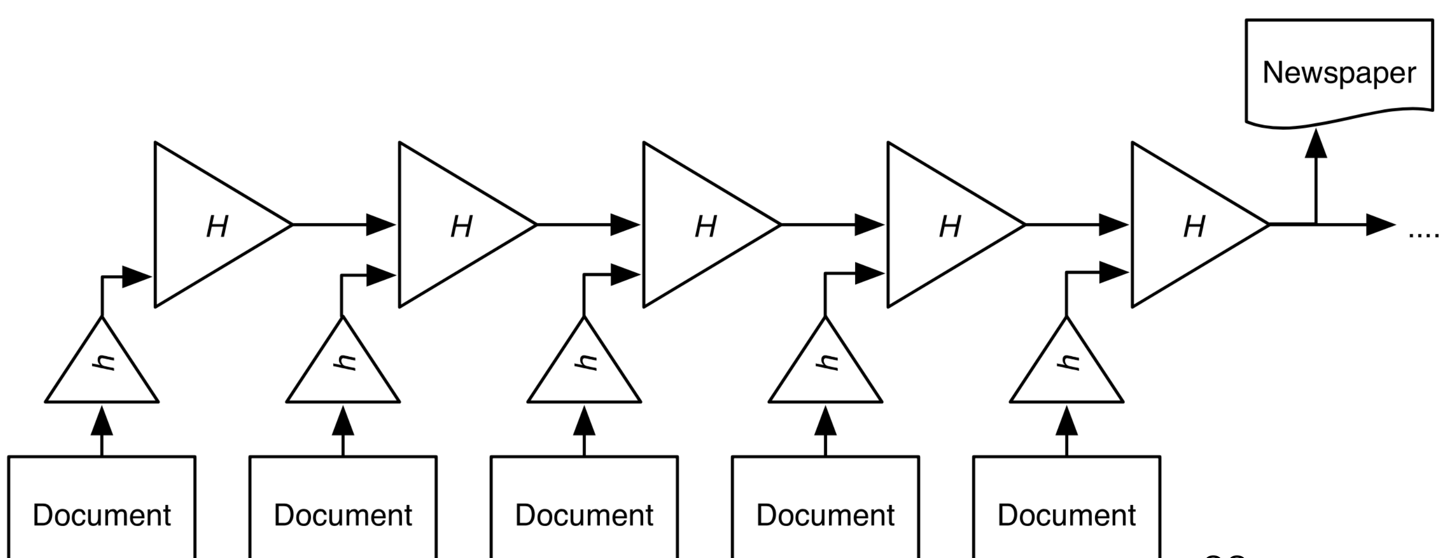
- How to realize coin flipping protocol?
  - Has to be fair

Hash Chains

- $H(x)$, $H(H(x))$, $H(H(H(x)))$, … $H^n(x)$
- Lamport's One-Time Passwords
  - For random s Alice computes $H^{1000}(s)$
  - Alice bootstraps server with $H^{1000}(s)$
  - To authenticate 1st session Alice reveals $H^{999}(s)$. For the 2nd $H^{998}(s)$…
  - Disadvantages?

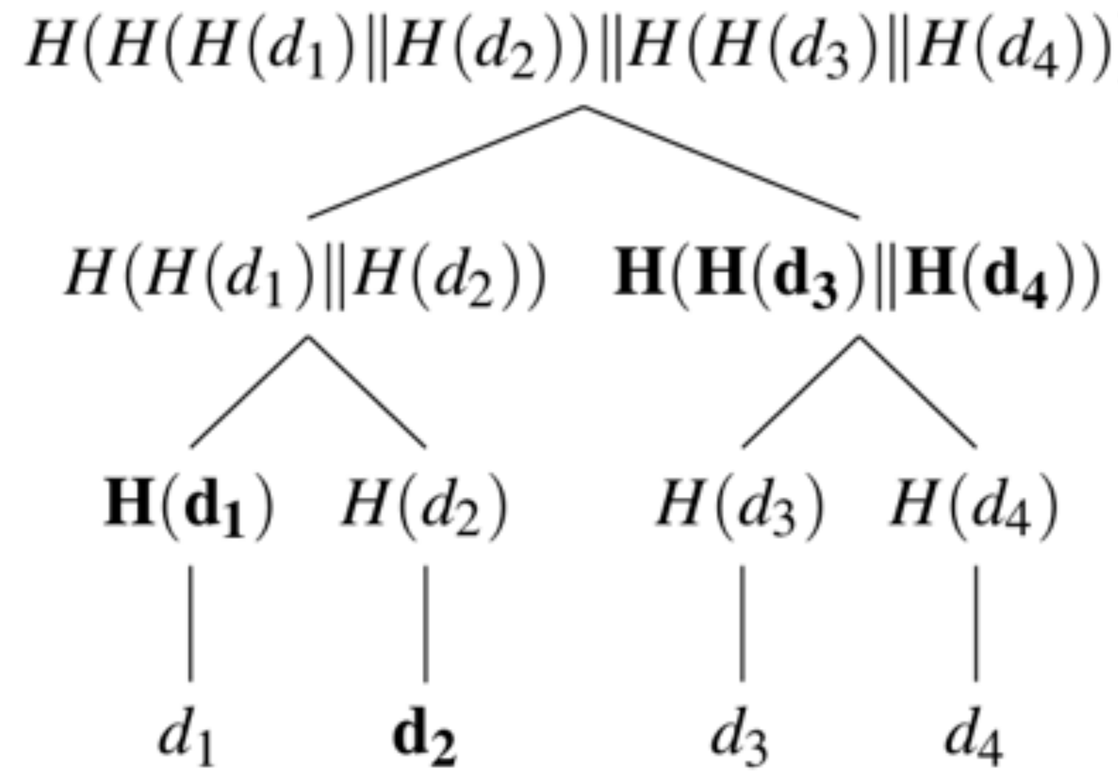
Hash Chains with Data

- Associate data with chain
  - Used for timestamping and integrity
- The oldest blockchain (since 1995 ;-)
- Does not scale with many documents to be logged
  - Hash chain grows 1:1 with the number of documents



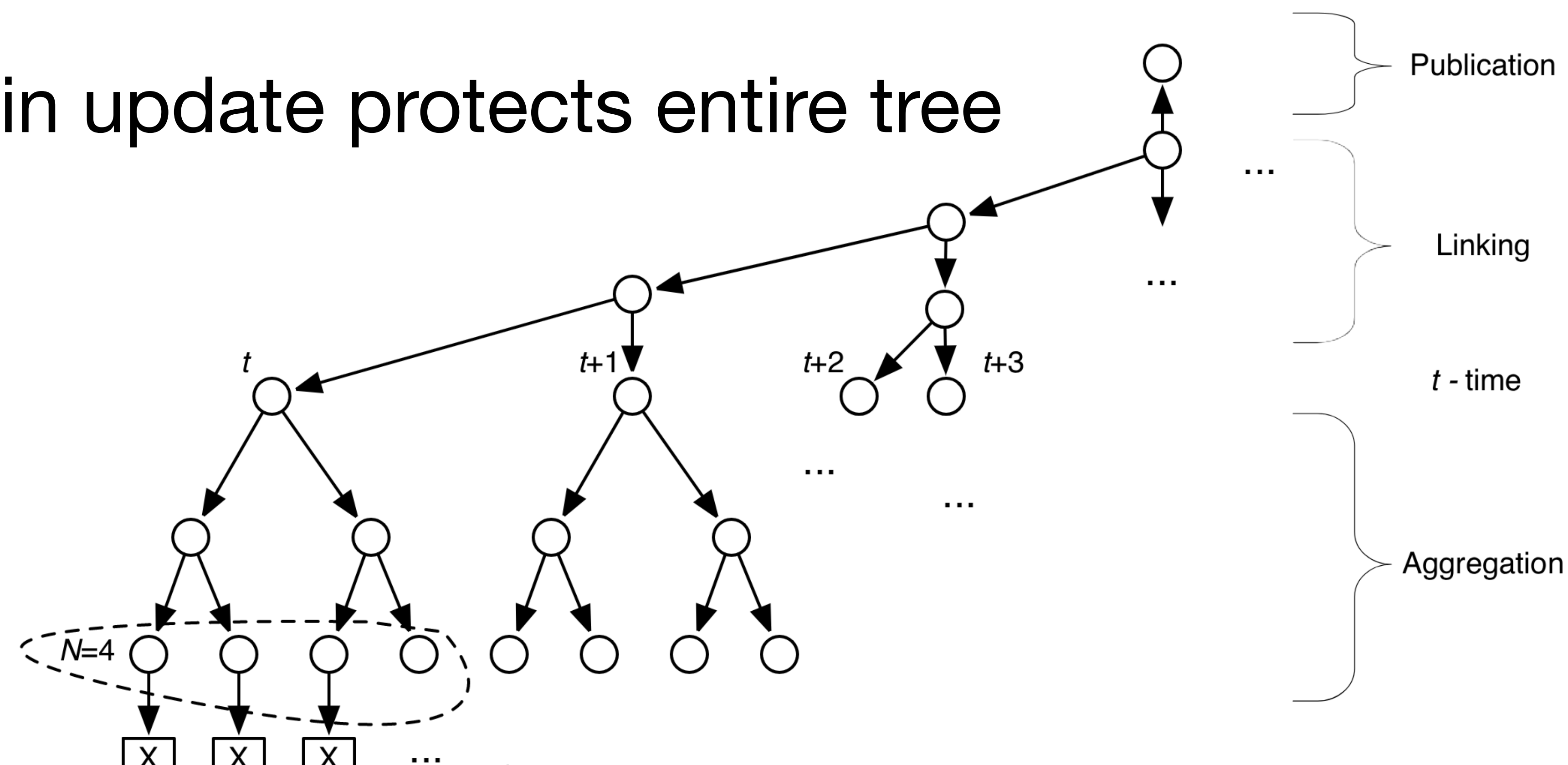
Merkle Hash Trees

- **Hash tree**: every non-leaf node is labelled with the hash of the labels of its child nodes, while leaf node is labelled with the hash of data
- Presence proofs
  - Smallest set of nodes that allow to rebuild root (need to encode sides)
  - Logarithmic (efficient)
- Nodes encoding
  - (for security) it makes sense to distinguish leaf nodes from other nodes



Combination

- Data aggregated in hash trees
- Trees are associated with a node of the hash chain
- Data added in batches
  - Single chain update protects entire tree



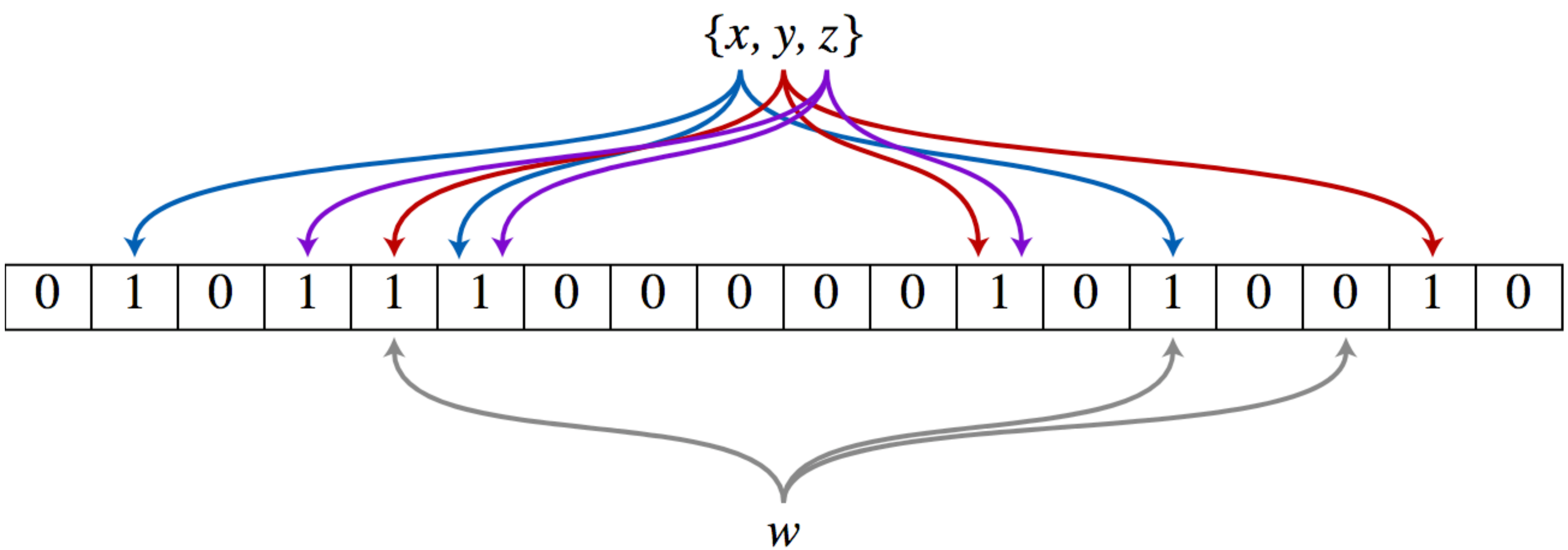
Bloom Filters

Bloom Filters (BFs)

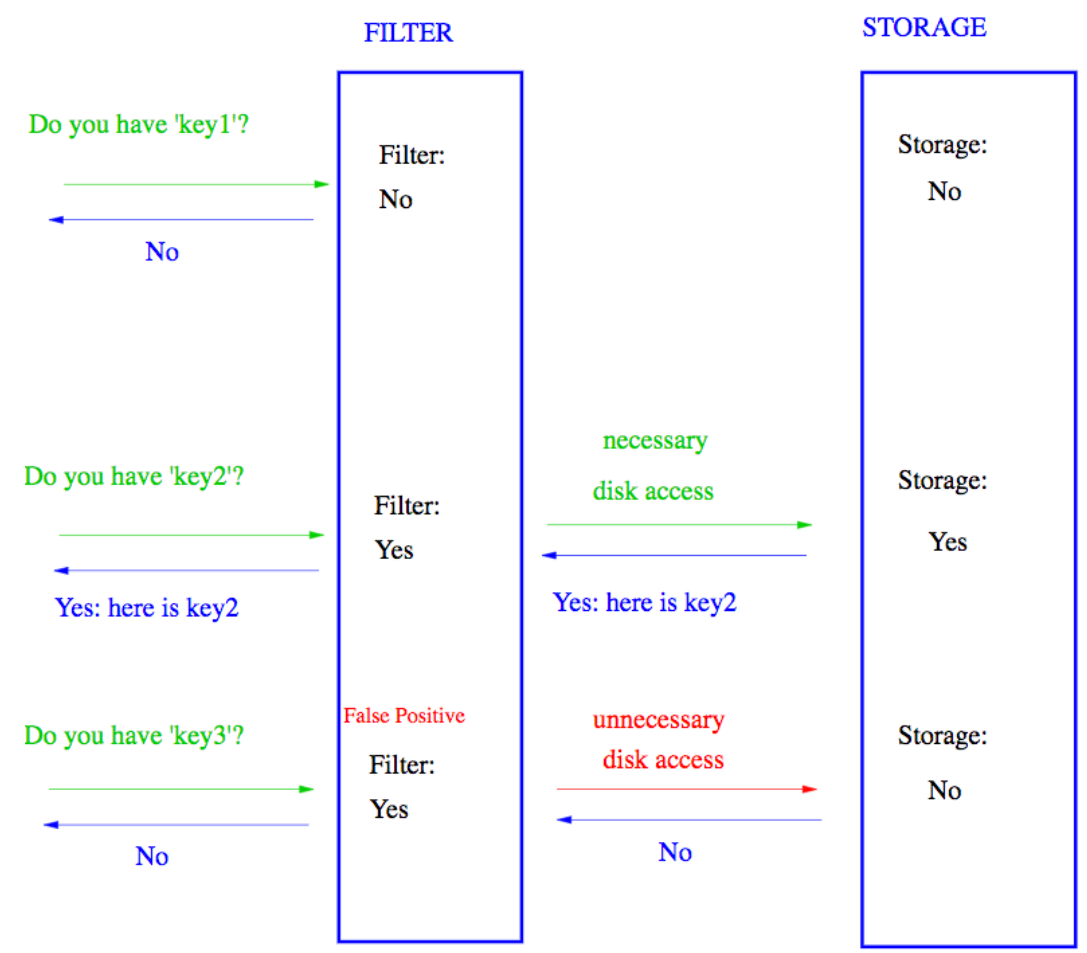
- Space-efficient probabilistic data structure
- Set membership (check quickly whether an element is in a set, without storing the element)
- *m*-bits long bit array (initially, all bits set to 0)
- *k* different hash functions, each maps set's element to one of *m* array positions (usually $k << m$)
    - They do not have to be cryptographically-strong hash functions
- Adding element
    - Hash element with *k* hash functions and set 1 on the obtained positions
- Querying element
    - Hash element with *k* hash functions, if bits on all positions equal 1 return TRUE, o/w FALSE
    - False positives possible, no false negatives

BF Example

- Represent set {*x*,*y*,*z*}
- Query for *w*

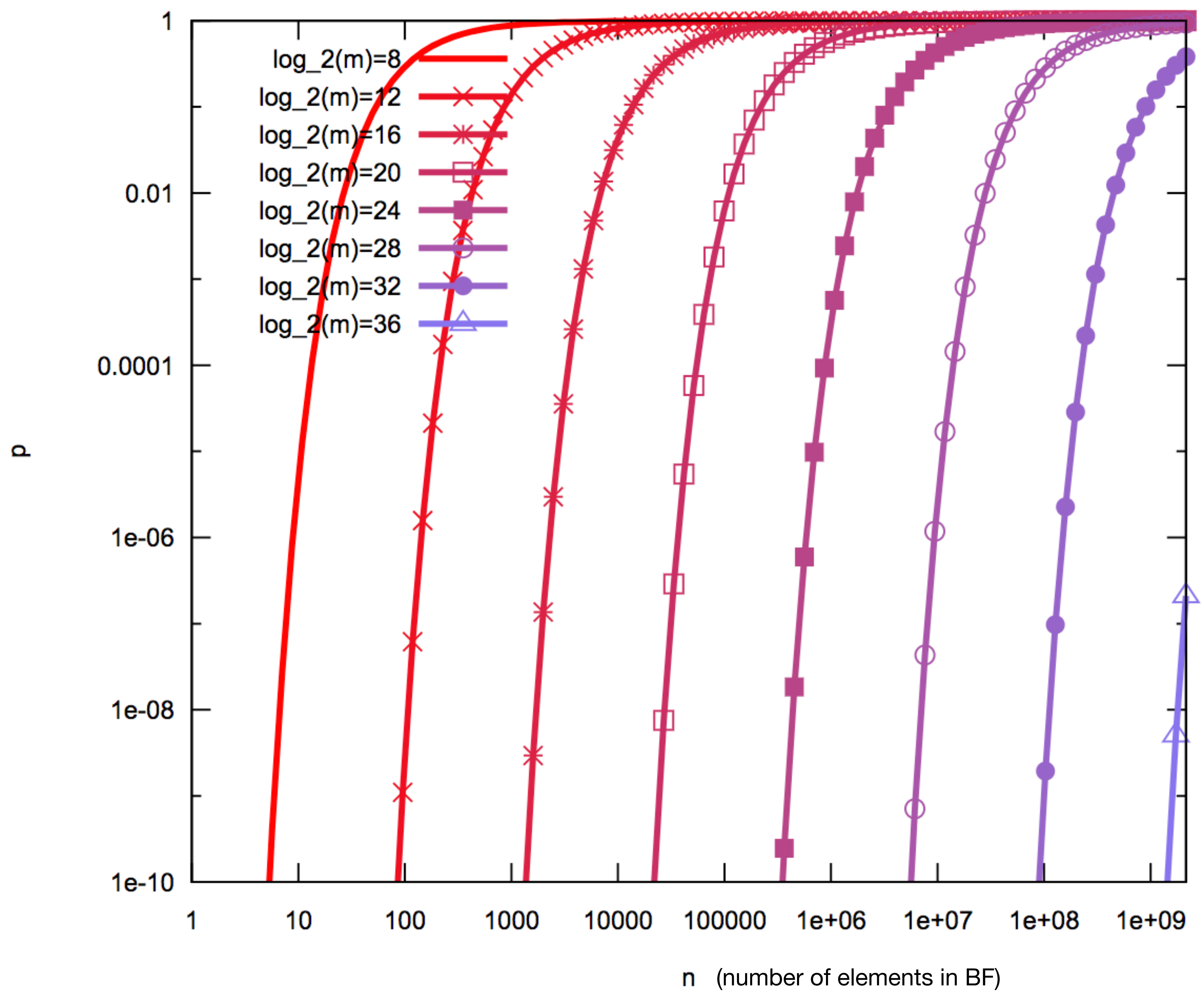


Applications



- BF-backed Databases
  - Storage lookup is expensive
    - Availability
  - Pre-filter queries
- **Private Queries**
  - Query for some objects without revealing criteria
  - Query with a BF that has inserted wanted objects
    - Thanks to false positives the responder will not be sure what was queried
    - Some unwanted objects will be returned
    - Can adjust anonymity by inserting fake criteria

False Positives Probability



k=(m/n)ln2 (optimal)

Reading

- Textbook 1.1, 1.2, 1.3
- *"Cryptography Engineering: Design Principles and Practical Applications"* http://ebookcentral.proquest.com.library.sutd.edu.sg:2048/lib/sutd/detail.action?docID=661548 Chapter 5
- https://www.anf.es/pdf/Haber_Stornetta.pdf
- http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.71.4891&rep=rep1&type=pdf
- https://people.eecs.berkeley.edu/~raluca/cs261-f15/readings/merkleodb.pdf