Crypto for Cloud and Blockchain

Sushmita Ruj
Indian Statistical Institute, Kolkata
http://www.isical.ac.in/~sush
Email: sush@isical.ac.in
Sushmita.ruj@gmail.com

Outline of today's talk

- Cloud security and motivations
- Crash course on Cryptography
- Access control of cloud data
- Auditing for ensuring integrity of data
- Blockchain Technology
- Smart Contracts

Clouds: the buzzword

Clouds

Why buy when we can rent?

Ref: Wikipedia

Security issues in Cloud Computing

- A user's data should be protected against adversaries or other users
- Cloud should be oblivious to the data stored
- Cloud should be oblivious to data it is computing
- Cloud should be accountable for its services

Cloud service provider as adversary

- Read/modify data
- CSP might not provide the desired amount of redundancy
- Might not provide the amount of storage as specified in the SLA
- Might not provide enough computational resources as specified in the SLA

Privacy issues in Cloud Computing

- Cloud service providers should not be able to track the position of a user/mobile device
- Legal issues in privacy protection
  - Data might be stored in different servers across different countries
  - Different privacy laws across different nations

Different faces of cloud security

- Cryptographic security
  - Authenticating users
  - Hiding data from cloud: computing and searching on encrypted data
  - Access control
  - Data auditing for integrity verification

- Network Security
  - Ensure that all communication channels are secure
- Operating system security
  - Virtualization security

Cryptographic techniques for Cloud computing

- Data auditing: Verify data integrity
- Fine-grained access control: Grants authorized access to user who have paid for service and denies access to unauthorized users
- Homomorphic encryption: Cloud does not know what data it is operating on, just gives back the result
- Searchable encryption: Cloud returns result of a query without knowing what the query is
- Verifiable computation:

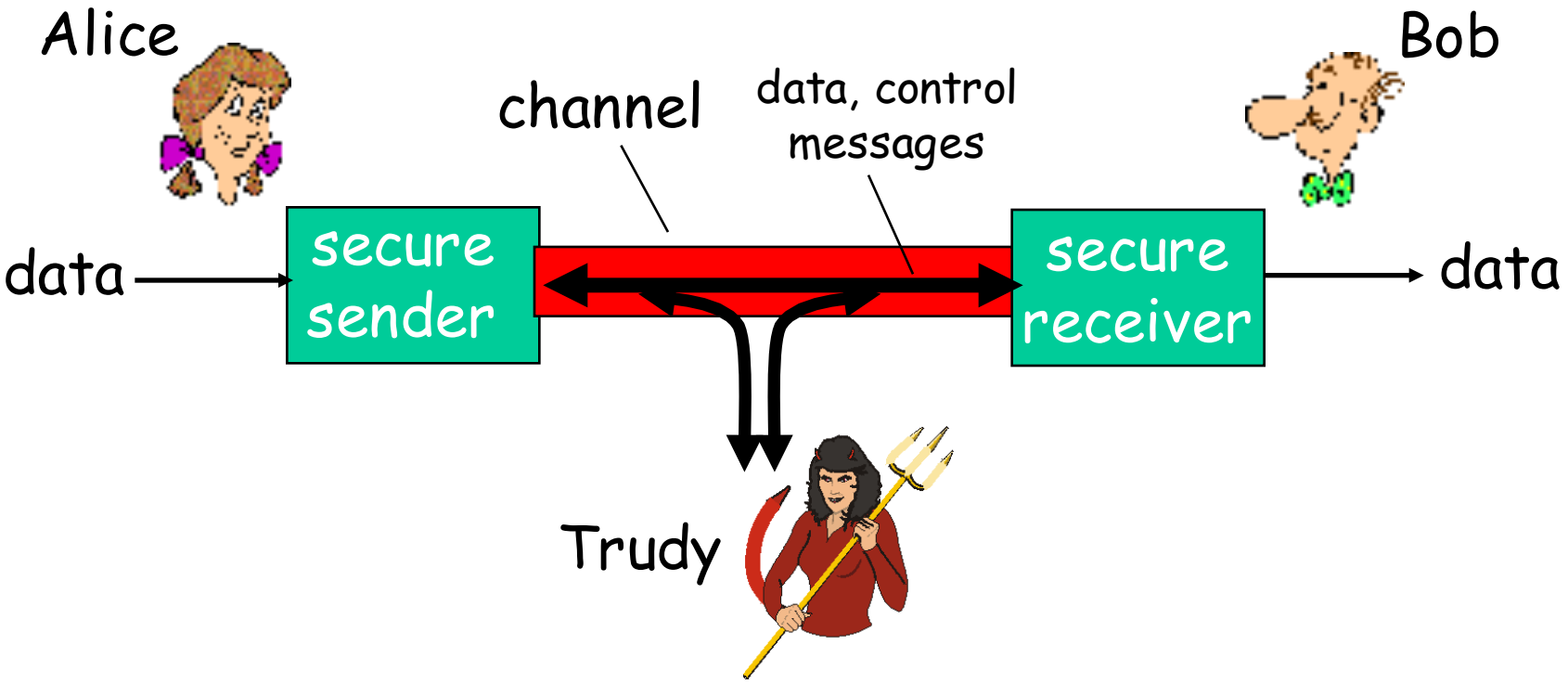
Crash Course on Cryptography

Basic Requirements of a secure system

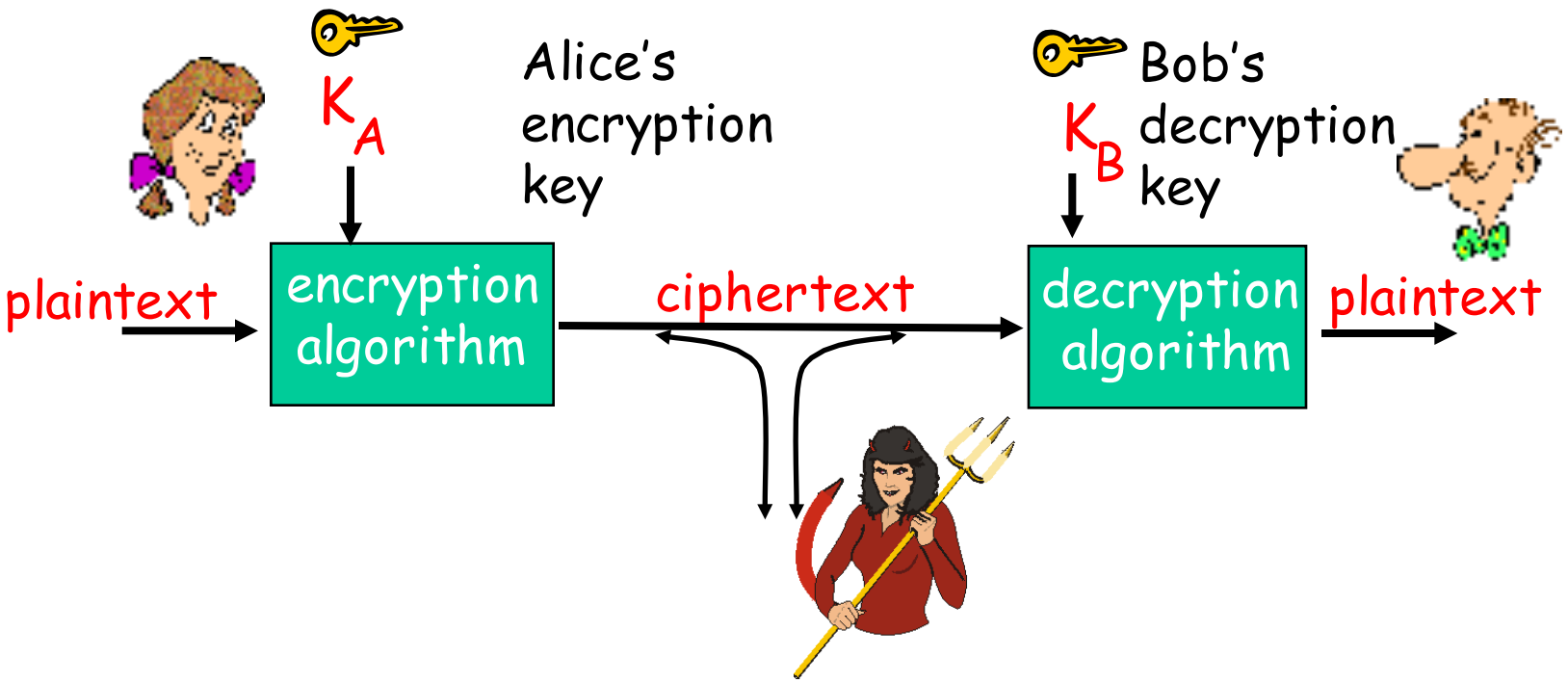
- Confidentiality: only sender, intended receiver should "understand" message contents

  Achieved using encryption
- Authentication: sender, receiver want to confirm identity of each other
- Message Integrity: sender, receiver want to ensure message not altered (in transit, or afterwards) without detection
- Accessibility and Availability: services must be accessible and available to users

Friends and enemies: Alice, Bob, Trudy



The language of cryptography



- symmetric key crypto: sender, receiver keys *identical*
- public-key crypto: encryption key *public*, decryption key *secret* (private)

Encryption/ Decryption

Encryption: a process of transformation

$$C = E_K(M)$$

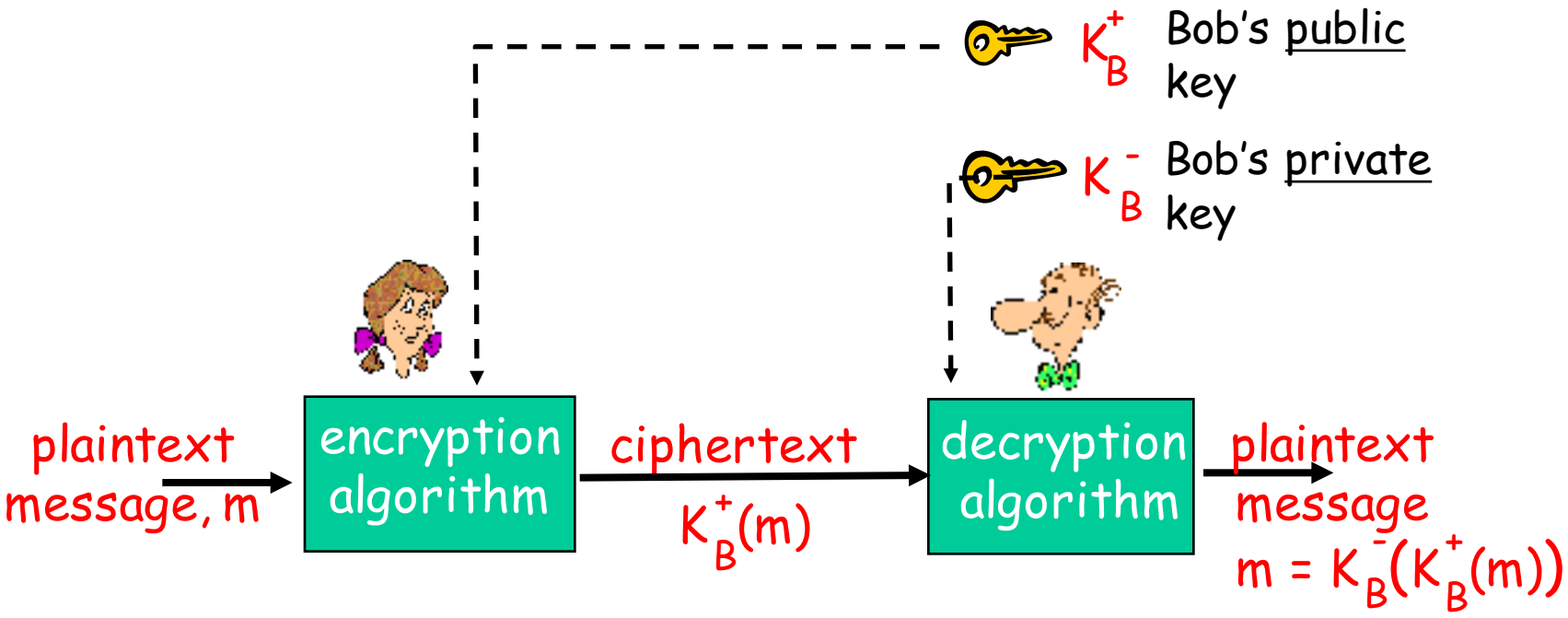
Decryption: recovering the original message

$$M = D_{K'}(C)$$

Public Key Cryptosystem

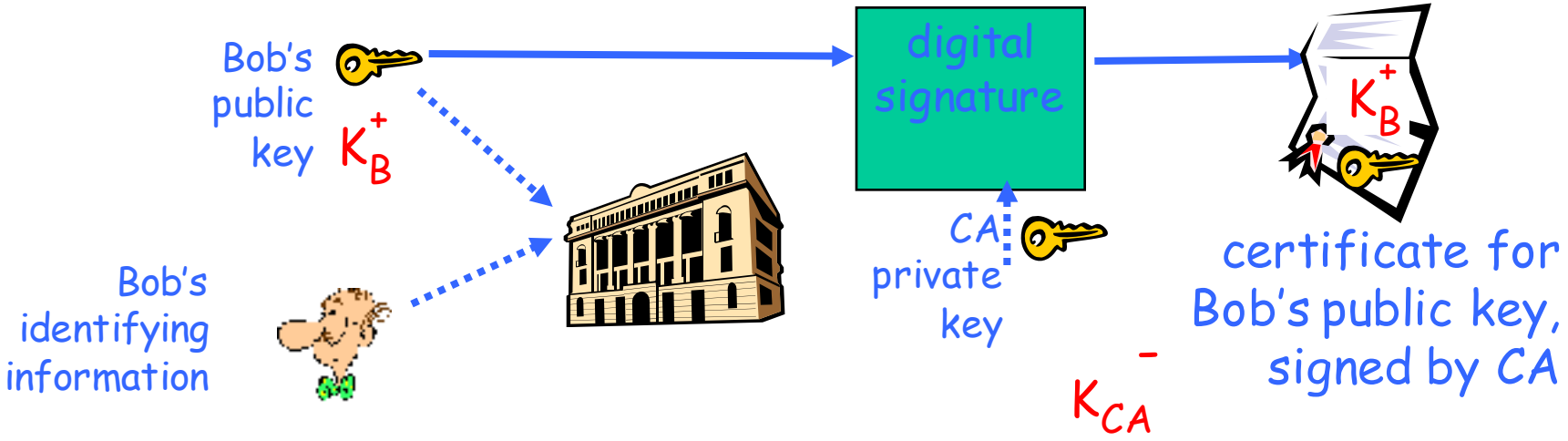
- Setup: Generate system parameters, public key pk and secret key sk
- Encrypt: Given message M and public key pk of receiver, generates ciphertext C
- Decrypt: Given ciphertext C and secret key sk, generates M

Public key cryptography



Public Key Infrastructure (PKI)

- How to bind the public key to a user?
- Certification authority (CA):



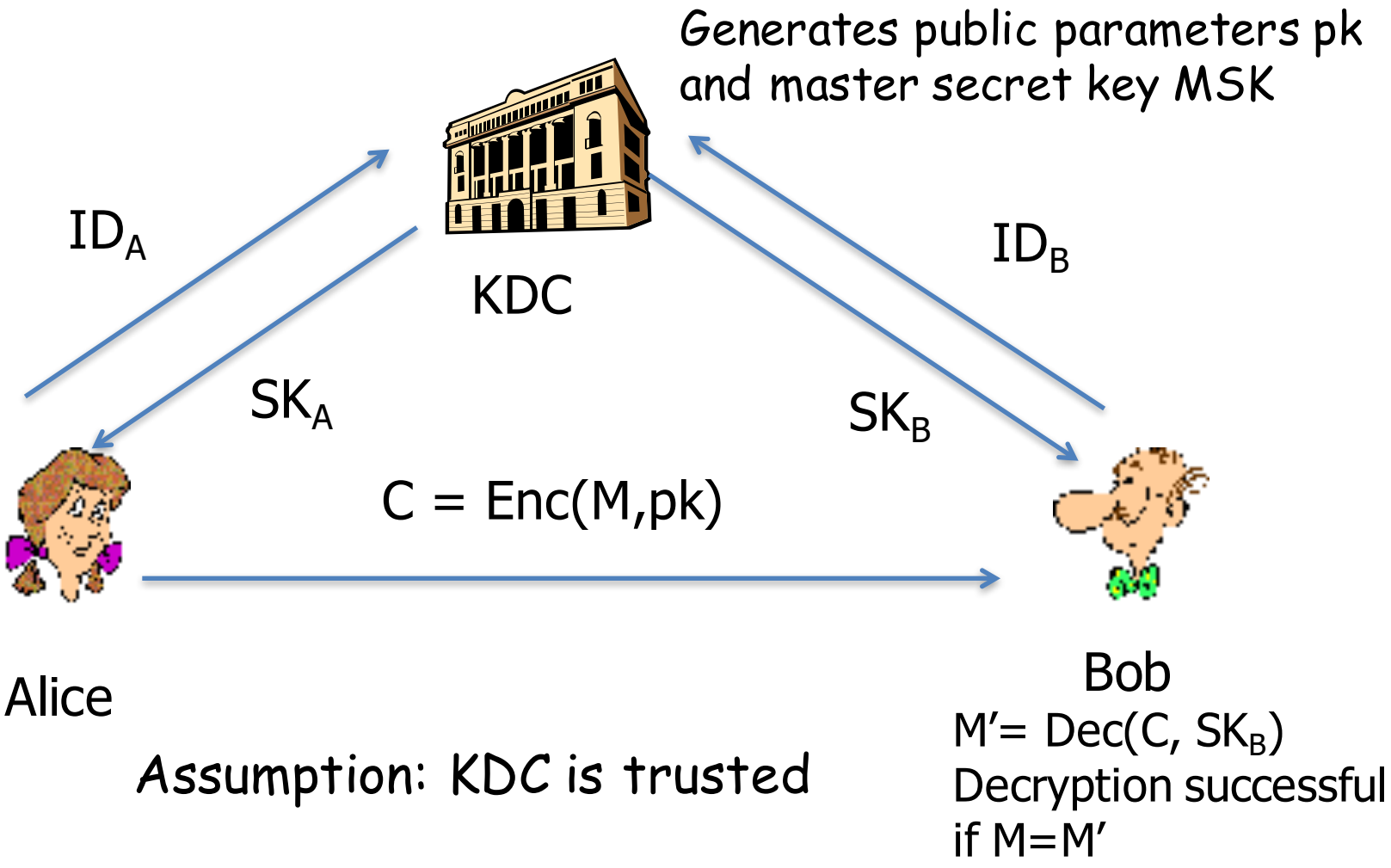
PKI: Problems and solutions

- Key management
- Assumption: CA is trusted
- Not a valid assumption: DigiNotar closed in Sept. '11
- Alternatives: Decentralized PKI
  - ✓Certificate transparency
  - ✓Using smart contracts

PROBLEM 1

- Designing efficient decentralized certificate management schemes

Identity Based Encryption (IBE)



IBE: Algorithms

- Setup: Generate system parameters, public key pk and master secret key MSK
- KeyGen: Using MSK and identity of user generates secret key sk
- Encrypt: Given message M and public key pk of receiver, generates ciphertext C
- Decrypt: Given ciphertext C and secret key sk, generates M

- Proposed by Shamir in 1984.
- Solved by Boneh-Franklin (using pairing based crypto) and Cock in 2001

Cryptographic techniques for Cloud computing

- Data auditing: Verify data integrity
- Attribute based access control: Grants authorized access to user who have paid for service and denies access to unauthorized users
- Homomorphic encryption: Cloud does not know what data it is operating on, just gives back the result
- Searchable encryption: Cloud returns result of a query without knowing what the query is
- Verifiable computation: Verify that the computation is done correctly

Data Auditing

Roadmap

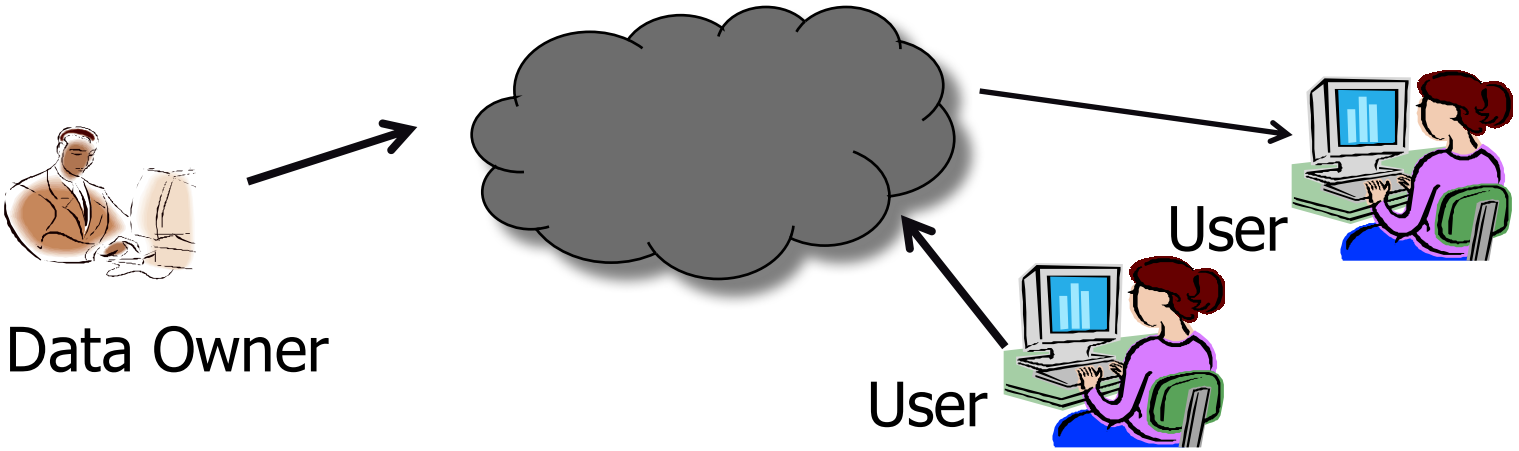
- Secure Cloud Storage
- Auditing Protocols
  - Simple examples
  - Desirable properties
- Building Blocks
- Concrete Construction

Data Storage

- How to provide long-term reliable storage
- Servers can behave unfaithfully

  -discard old data

  -hide data loss
- If data is deleted no way to recover it
- Client needs a guarantee that data is stored correctly.



Data Auditing

- Data auditing is a periodic event to assess quality or utility of data to evaluate

  -security,

  -data integrity

  -privacy preservation

  -computational accuracy

Data Auditing (First Attempt)

- Download entire data and verify all the blocks of data
- Problems:
- Data has to be maintained at the source to compare with downloaded data
- Contradicts the purpose of storing in clouds
- Large communication overheads
- High computation overhead

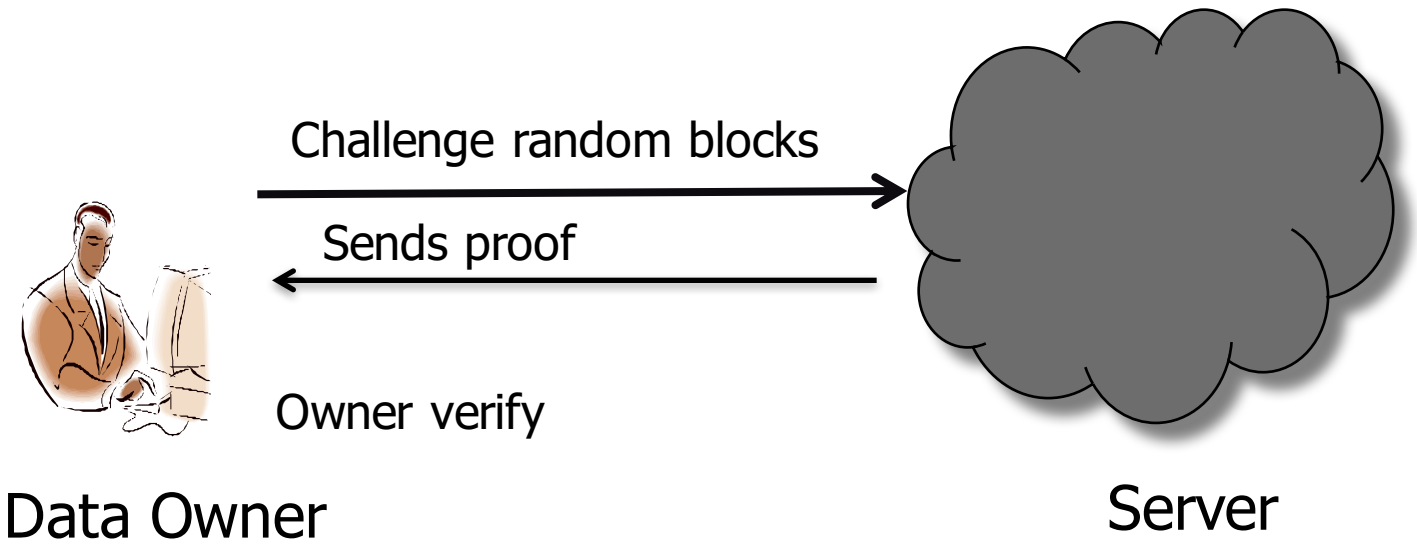
Data Auditing (Second Attempt)

- Store some of the data blocks at the client side
- Download those blocks and verify with that stored
- Problems:
- Still needs large amount of storage
- Does not guarantee that the other blocks cannot be modified and still go undetected

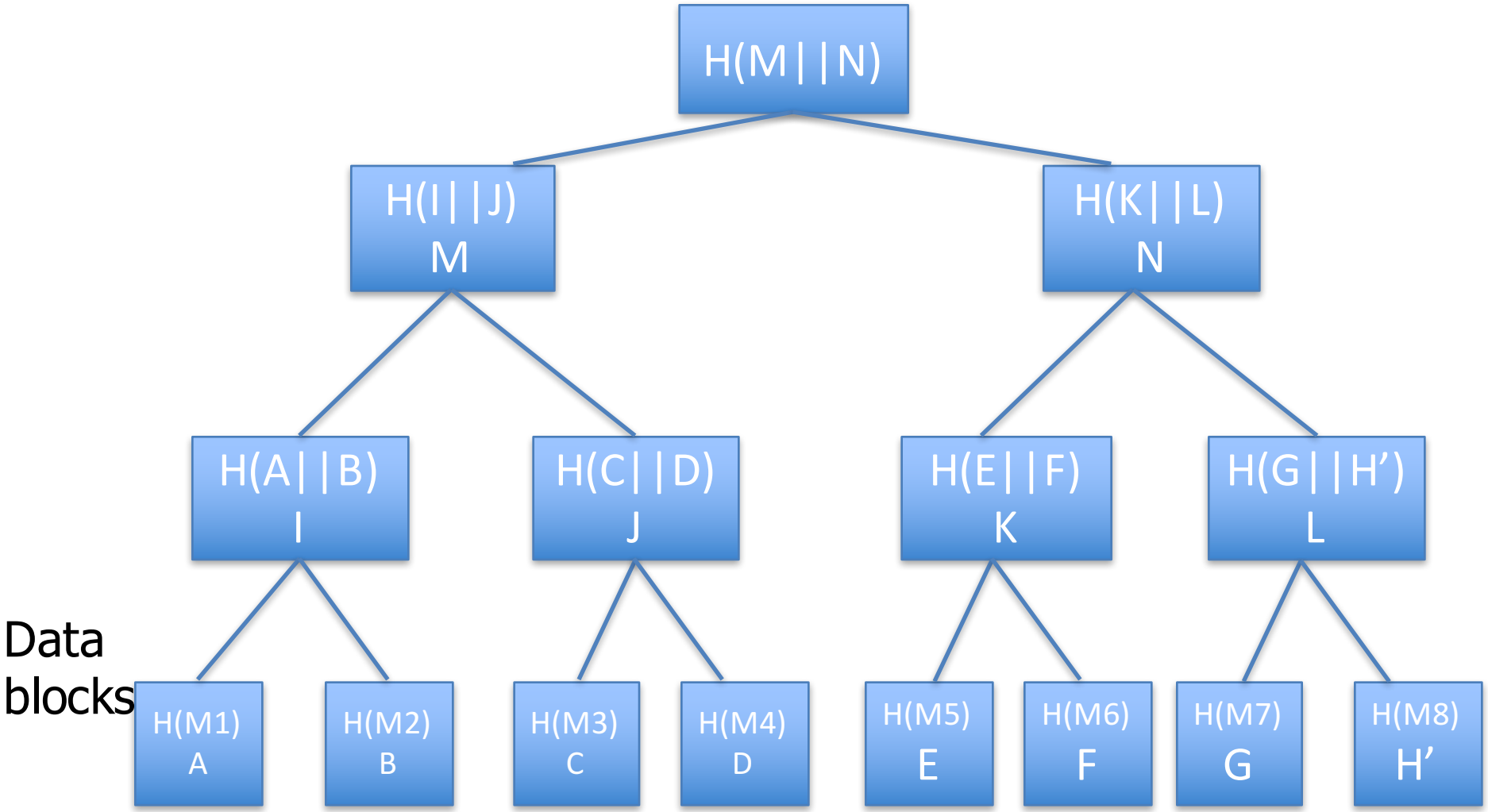
Data Auditing (Third Attempt)

- Store an aggregated information at the user end
- Randomly request for blocks
- Download them and some auxiliary information
- Calculate the aggregated value and match with that at the client
- How to organize the data to make this possible?

Data Storage

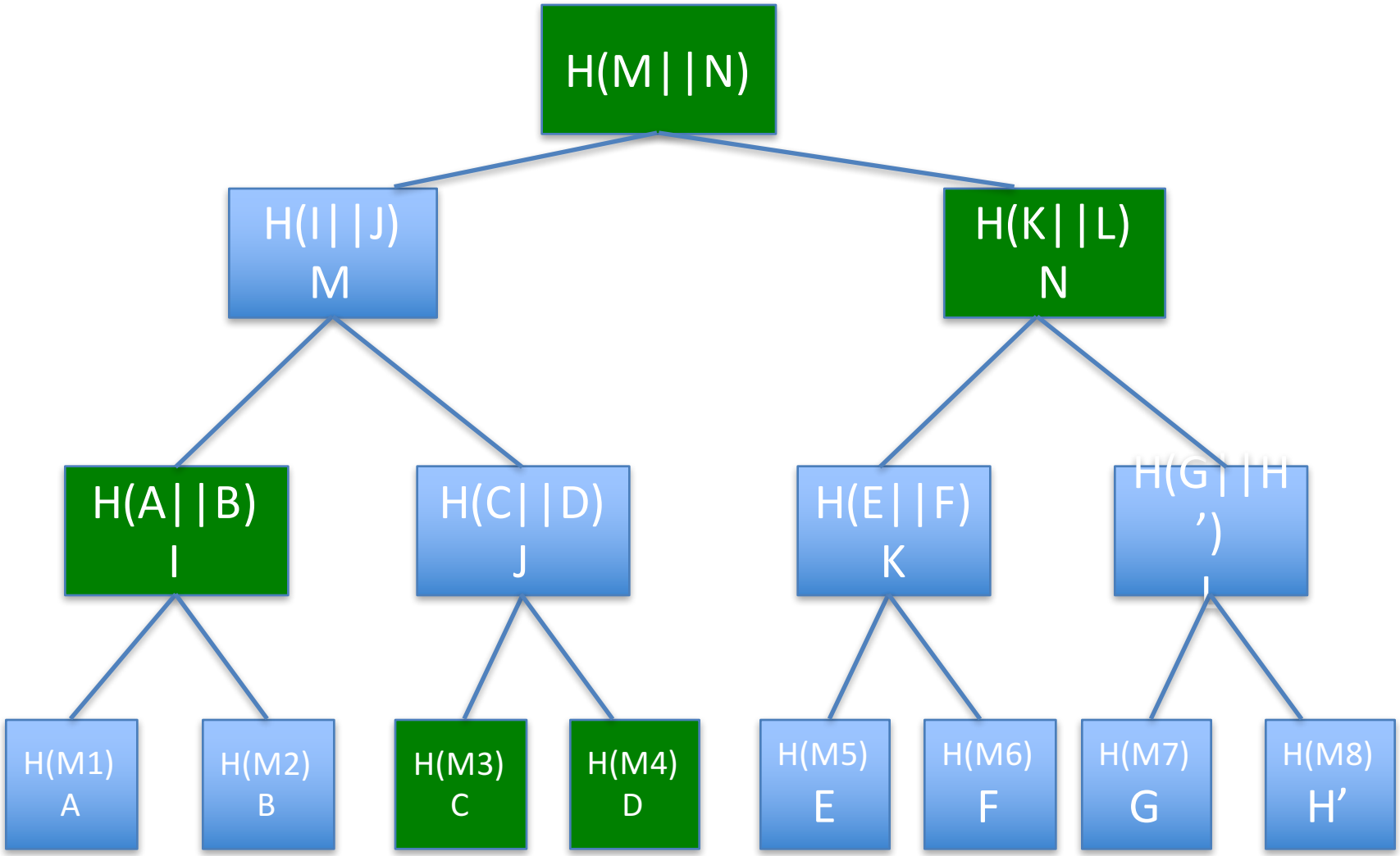


Merkle-tree



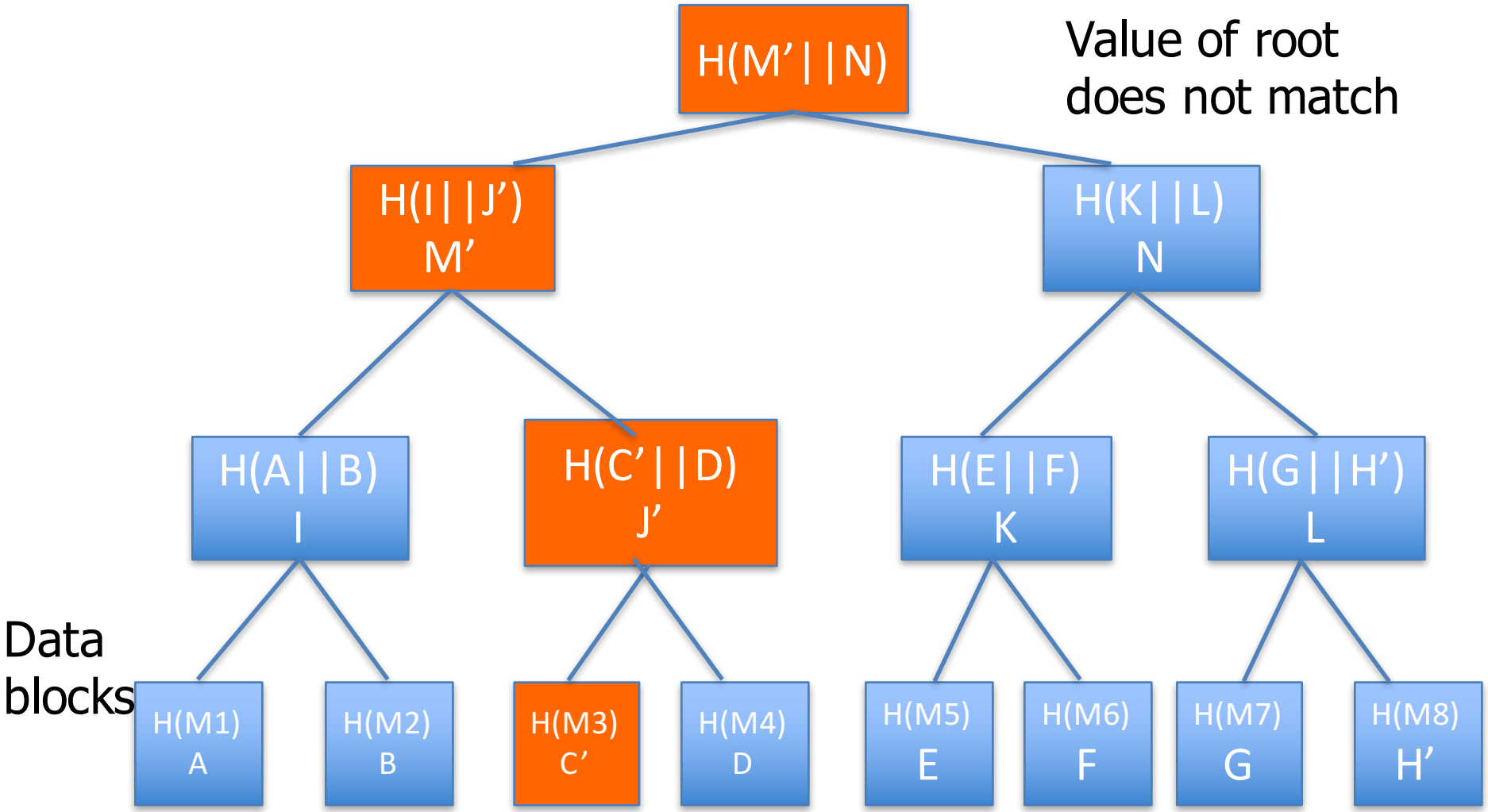
To check D, Proof = <H(M4), H(M3), H(A||B), H(K||L), H(M||N)> should match with root

Merkle-tree

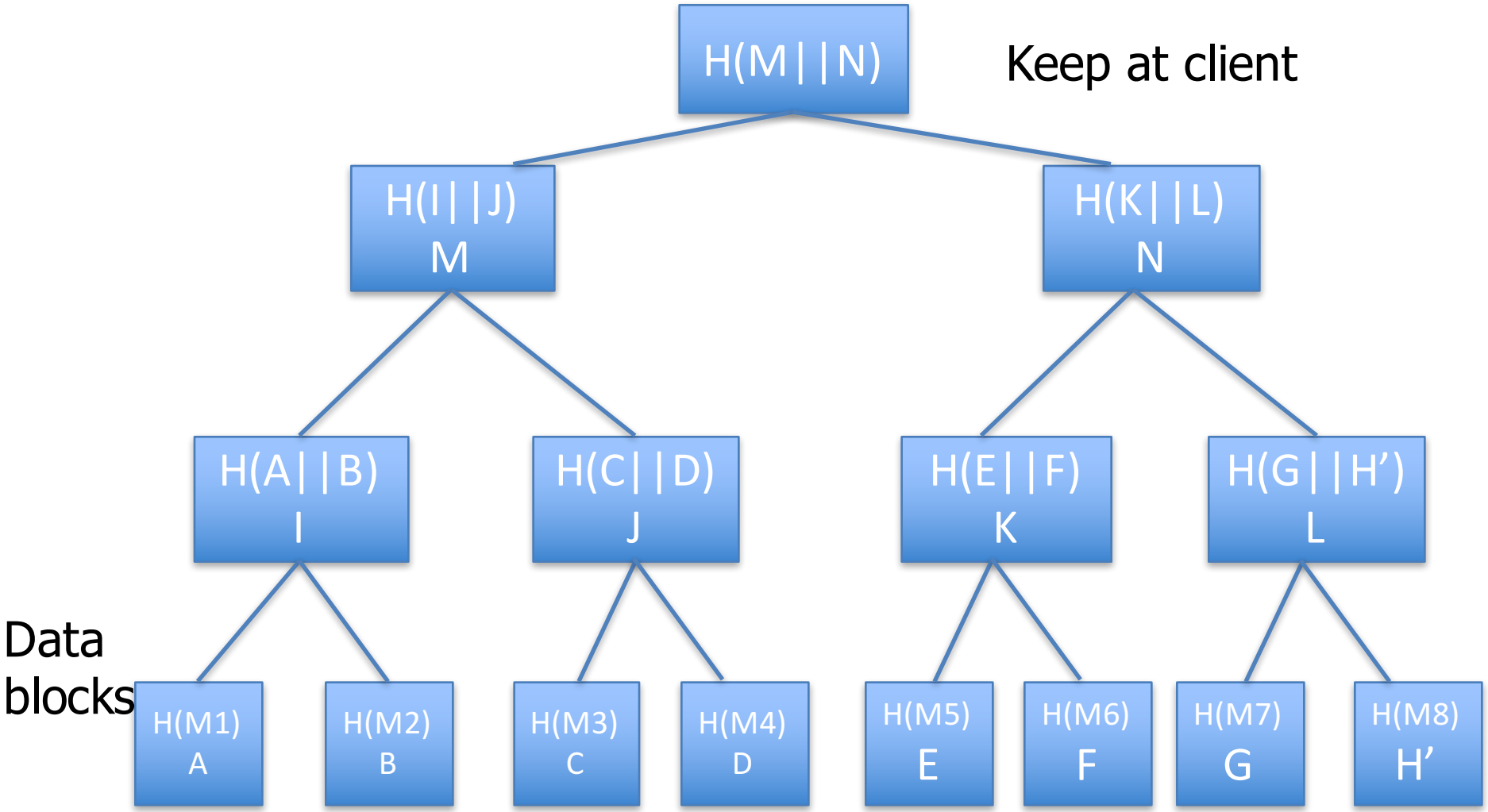


To check D, Proof = <H(M4), H(M3), H(A||B), H(K||L), H(M||N)> should match with root. Proof size log(n), n is the number of blocks

Merkle-tree


H(M’||N)
Value of root does not match
H(I||J’)
M’
H(K||L)
N
H(A||B)
I
H(C’||D)
J’
H(E||F)
K
H(G||H’)
L
Data blocks
H(M1)
A
H(M2)
B
H(M3)
C’
H(M4)
D
H(M5)
E
H(M6)
F
H(M7)
G
H(M8)
H’

Merkle-tree



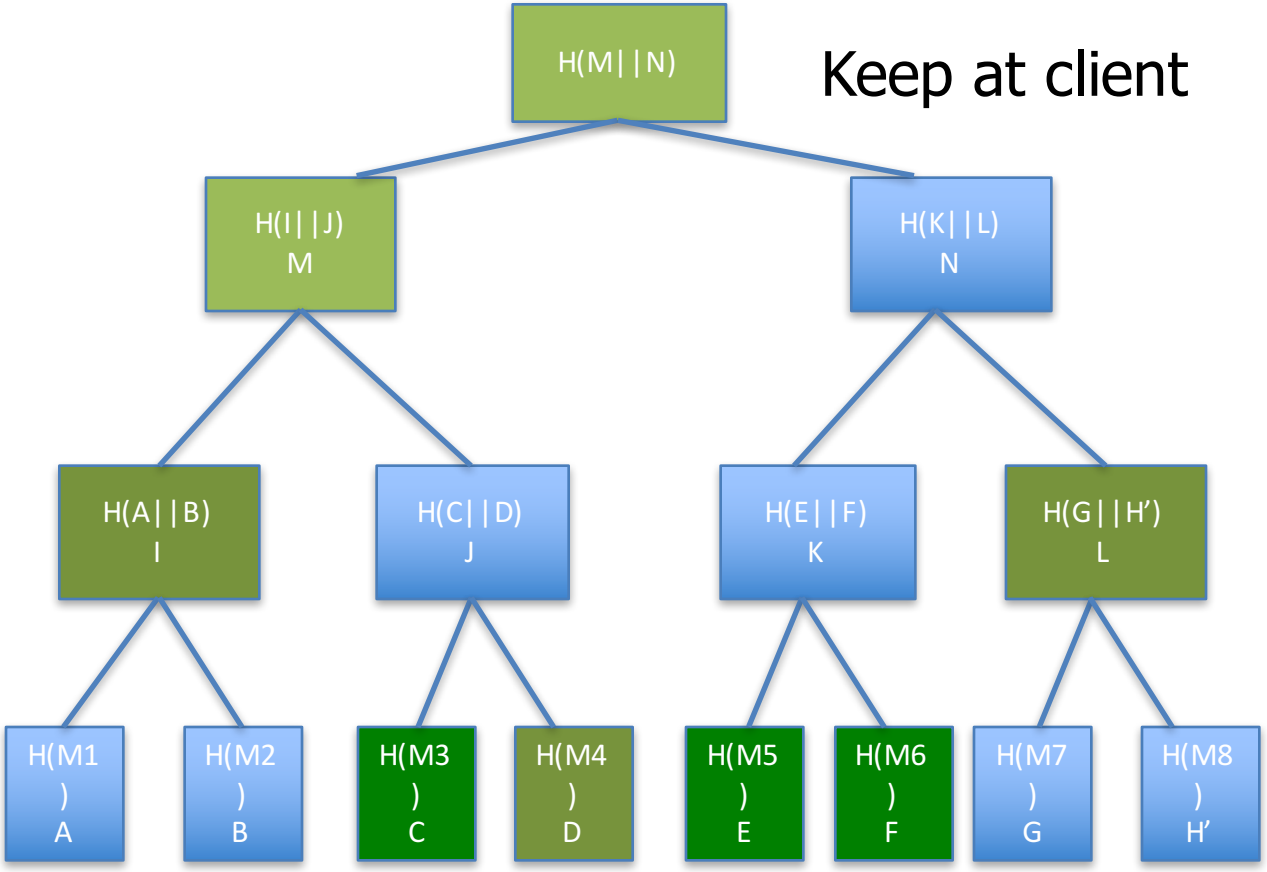
To check D, Proof = <H(M4), H(M3), H(A||B), H(K||L), H(M||N)> should match with root

Merkle Tree for Data Auditing

Challenge blocks: C, E, F

Proof: D,I, L, M, Root

Verify with stored root value



The size of the proof is O(l*log n), where n = number of data blocks and l is the number of challenge queries

Types of Proofs

- Provable Data Possession (PDP):

Without retrieving data, Client (verifier) allows to verify that the CPS still possesses the client's original data.

Proofs of Retrievability (PoR):

The client (verifier) runs an efficient data audit proof in which the data storage server (prover) proves that it still possesses the client's data and client can recover entire file

Related Work

- Provable Data Possession (PDP):
- Giuseppe Ateniese, Randal C. Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary N. J. Peterson, Dawn Song, CCS '07,
- Proofs of Retrievability (PoR):
- Ari Juels, Burton S. Kaliski Jr., CCS '07,

Preprocessing for auditing: Erasure Codes

- An $(n; f; d)_{\Sigma}$ erasure code over finite alphabet Σ is an error-correcting code that consists of
- Enc: $\Sigma^{f} \rightarrow \Sigma^{n}$ An encoding algorithm
- Dec: $\Sigma^{n} \rightarrow \Sigma^{f}$ - decoding algorithm

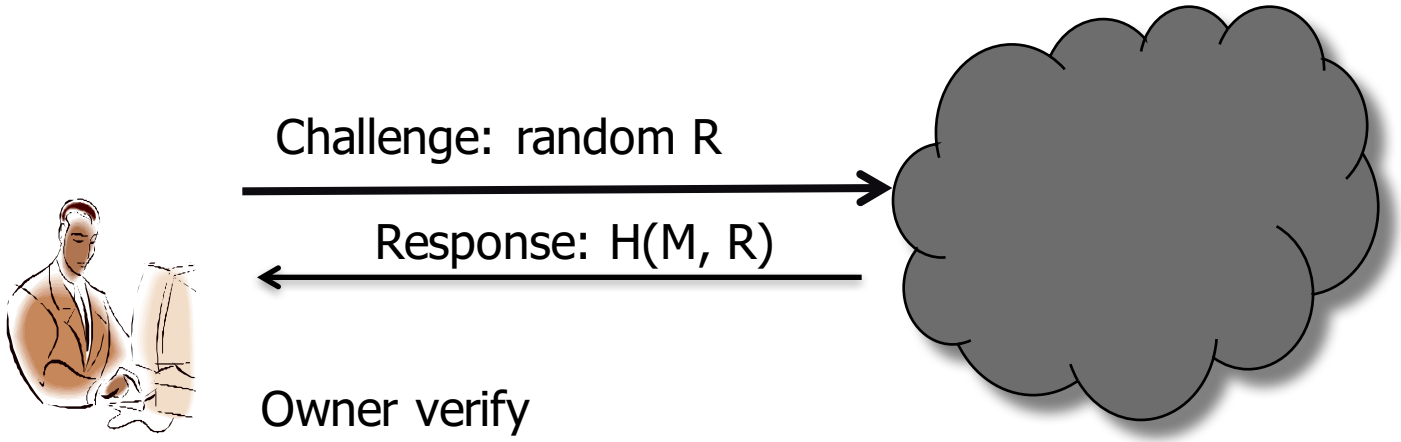
d is the minimum distance (Hamming distance between any two codewords is at least d) of the code.

- An (n; f; d) erasure code can tolerate up to d -1 erasures.
- If d = n - f + 1 , we call the code a maximum distance separable (MDS) code.
- For an MDS code, the original message can be reconstructed from any f out of n symbols of the codeword.
- Examples: Reed-Solomon codes

Basic steps for auditing

- Given a file F_0 , it is erasure coded to F
- An authenticator is attached to each block in F
- All blocks and authenticators are uploaded to the server
- Audit consist of two algorithms
  -proof generation (by server)
  -proof verification (by auditor)
- Similar to challenge, response

How to Audit



Data Owner

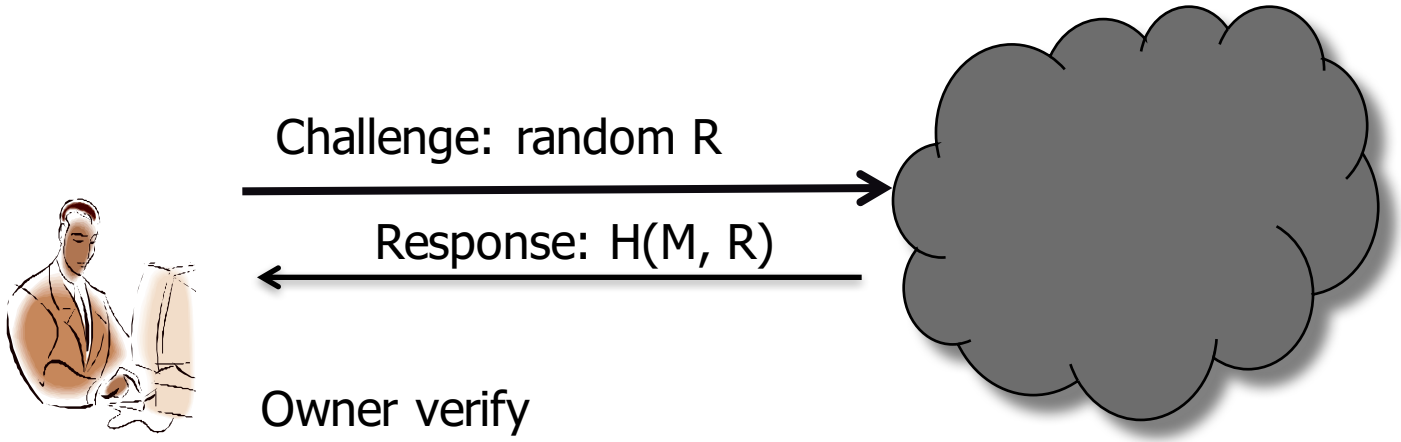
Requirements of an Auditing Scheme

- Verification should be fast
- Proof should be short (low communication cost)
- Anyone can verify (public verifiability)
- A third party performing the audit should have no knowledge of the data (Privacy preserving)
- Unlimited verification

Discussion about Merkle-tree based auditing

- Verification should be fast ✔
- Proof should be short (low communication cost) $O(\log n)$
- Anyone can verify (public verifiability) ✗
- A third party performing the audit should have no knowledge of the data (Privacy preserving) ✗
- Unlimited verification ✔

How to Audit



Data Owner

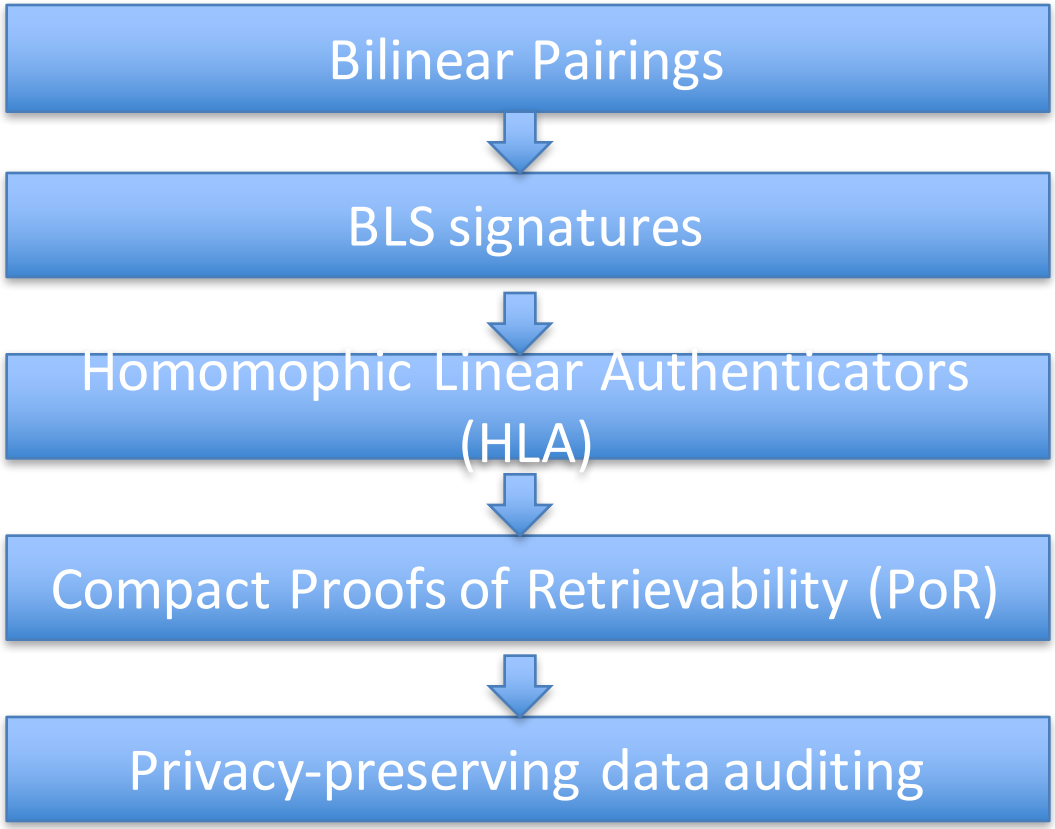
Data Auditing

- KeyGen: Choose N = pq (p,q are primes).
- PK= (N,g), g is an element of Z_N^*
- Tag of a block b, T(b) = g^b mod N
- Merkle tree maintains the tags
- Challenge is the set of indices

  $\{(i_1,v_1), (i_2 v_2), .. (i_c v_c)\}$
- Response from server M = $\Sigma^c_{j=1}\ v_i m_{ij}$
- Verification: If g^M mod N is $\Pi^c_{j=1}\ T(m_{ij})^{vj}$?

Ref: DPDP: Erway et al , TISSEC 2015

Constructing a desirable auditing scheme



Bilinear Pairings

- G, G_T are groups of order p (prime)
- $e : G \times G \rightarrow G_T$ is an a bilinear map if:

  -Non degenerate
  $e(g,g) \neq 1$

  -Bilinear: $e(g^a,g^b) = e(g,g)^{ab}$, $a,b \in Z_p^*$, $g \in G$

  -e can be computed efficiently

Boneh-Lynn-Shacham Signature (BLS)

- H: {0,1}* -> G
- Private signing key sk = $x \in Z_p$
- Public verification key pk = g^x
- Sign(M,sk): $\sigma = H(M)^x$
- Verify(M, σ,pk) : Valid iff $e(\sigma,g) = e(H(M),pk)$
- Correctness: $e(\sigma, g) = e(H(M)^x, g) = e(H(M),g^x)$

Data Auditing

Randomly select blocks 2,3, 5

m_1 σ_1	m_2 σ_2	m_3 σ_3	m_4 σ_4	m_5 σ_5	..	...	m_n σ_n

Send μand σ

Prover verifies using only μand σ

μ
σ

Prover computes a combined value of blocks and authenticators

Homomorphic Linear Authenticator

- Let σ_1, σ_2 be 2 authenticators on m_1, m_2 resp.
- $(\sigma_1)^a(\sigma_2)^b$ is an "authenticator" on $(m_1)^a(m_2)^b$
- Easily forgeable?
- "Linear combination"
- BLS signature: $[H(m)]^x$

Compact Proofs of Retrievability

- sk = x $\in$ Zp, pk = g^x, u $\in$ G, H: {0, 1}* -> G
- Auth(sk, m_i, i): σ_i = [H(name || i) u^{m_i}]^x
- Name is randomly chosen from a large domain
- Ver(pk, σ_i, m_i, i, name): (let W_i = name || i)
- Output `1' iff $e(\sigma_i, g) = e(H(W_i), pk)\ e(u^{m_i}, pk)$
- Shacham and Waters, Asiacrypt 08, JoC 2013

Homomorphic Property

- $\sigma_i = [H(W_i)\, u^{mi}]^x$, $e(\sigma_i, g) = e(H(W_i), pk)\, e(u^{mi}, pk)$
- $\sigma_i = [H(W_i)\, u^{mi}]^x$, $\sigma_j = [H(W_j)\, u^{mj}]x$
- Suppose $\sigma = (\sigma_i)^a(\sigma_j)^b$
- $e(\sigma, g) = (e(\sigma_i, g))^a(e(\sigma_j, g))^b$
  $= e(H(W_i)^a H(W_j)^b, pk)\, e(u^{a(mi) + b(mj)}, pk)$
- Linear combination in the exponent: $a(m_i) + b(m_j)$

(Public-Verifiable) PoR from HLA

$I = \{i_1, i_2, i_4\}$

$(i_1, v_1)\ (i_2, v_2)\ (i_4, v_4)$

$\sigma\ ,\ \mu$

Check if $e(\sigma, g) =$
$e((\Pi_{i=1,2,4}\ H(W_i)^{v_i})u^{\mu},\ pk)$?

$\sigma = \Pi_{i \in I}(\sigma_i)^{v_i}$

$\mu = \Sigma_{i \in I}(v_i m_i)$

Privacy is leaked!

TPA

$(i_1,v_1)\ (i_2,v_2)\ \ldots(i_4,v_4)$

$\mu=v_1m_1 + v_2m_2+ v_4m_4$

$(i_1,v_1)\ (i_2,-v_2)\ \ldots(i_4,-v_4)$

$\mu'=v_1m_1 - v_2m_2- v_4m_4$

The TPA knows the value of data block m_1

Privacy – preserving auditing

$(i_1,v_1)\ (i_2,v_2)\ (i_4,v_4)$

$\mu = v_1 m_1 + v_2 m_2 + v_4 m_4$

$\sigma = \sigma_1^{v1} + \sigma_2^{v2} + \sigma_4^{v4}$

Send R, μ'

1. Server chooses random r
2. Set $R = e(u,g^x)^r$, $y = H(R)$
3. Set $\mu' = r + \mu y$

Check if $R.e(\sigma,g)^y = e((\Pi_{i=1,2,4} H(W_i)^{v_i})^y u^{\mu'}, pk)$?

Wang et al, Trans. Of Computers 2013

Probability of Detection

- How to choose c?
- $P = 1 - (1 - t)^c$ when t fraction of data is corrupted
- When t = 1%, c = 300 for P = 95% (99%)
- Ref: Ateniese et al. CCS 2007

Ongoing Research and Future Directions

- Alternate data structures like skip lists instead of Merkle trees
- Dynamic data auditing and authenticated data structures
- When modifications are made the cloud knows the locations
- How to hide access patterns?
- Concepts of Oblivious RAM
- ORAM based auditing schemes:
  -Cash et al, TCC 2014
  - Shi et al, CCS 2013

PROBLEM 2

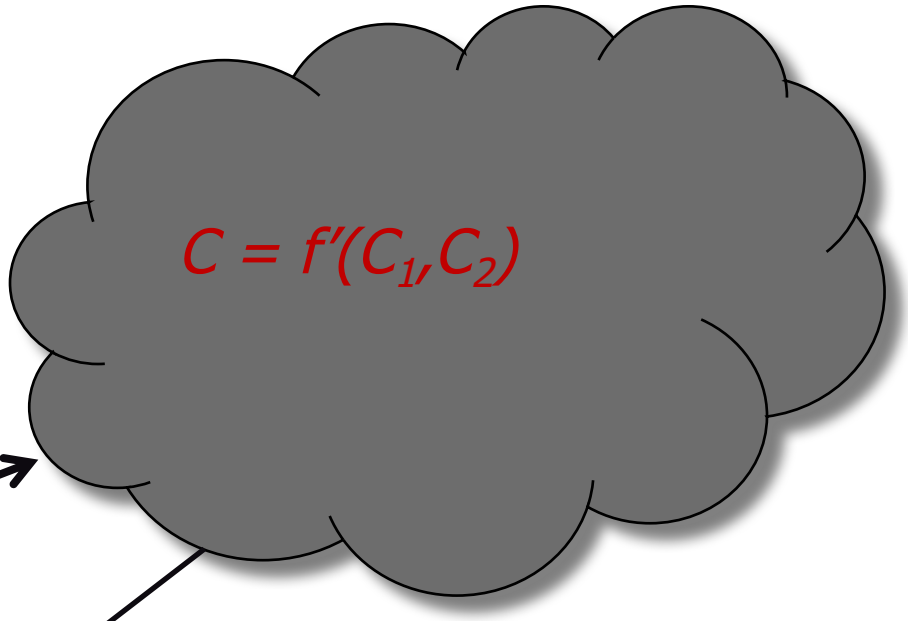
- Data structures for Secure Data Storage

PROBLEM 3

- Efficient Access Control with Revocation

Homomorphic encryption

Given M_1 and M_2, Calculate $f(M_1,M_2)$



$C_1 = E(M_1)$

$C_2 = E(M_2)$

C

Decrypt C and get $f(M_1,M_2)$

Homomorphic encryption

- Choose group G of order q with generator g
- *Public key* = (G,q,g,h), $h = g^x$
- *Secret key* = x
- $E(M) = (g^r, Mh^r)$, r is randomly chosen in Z_q
- Cloud cannot calculate r, and hence M (does not know x)
- $E(M_1)E(M_2) = (g^{r1+r2}, M_1M_2h^{r1+r2}) = E(M_1.M_2)$
- *Data owner knows r1 and r2, and x, can calculate* $M_1M_2 = (M_1M_2h^{r1+r2})/(g^{r1+r2})^x$

Fully Homomorphic Encryption

- Proposed by Gentry in STOC'09
- Complex functions instead simple addition, multiplications
- Very expensive for mobile devices
- A simple decryption operation would take 30 sec on a mobile phone??
- Why do we need full functionality: Have operations which are important. Lauter et al (2011)
- Addition, multiplication, inner products etc can be done

Searching on Encrypted Data

- Searching large data bases is a difficult problem
- Searching on encrypted databases is even a bigger challenge
- Known techniques include:

  -property-preserving encryption

  -functional encryption

  -fully-homomorphic encryption

  -searchable symmetric encryption

  -oblivious RAMs

  -secure two-party computation

Important problems not discussed

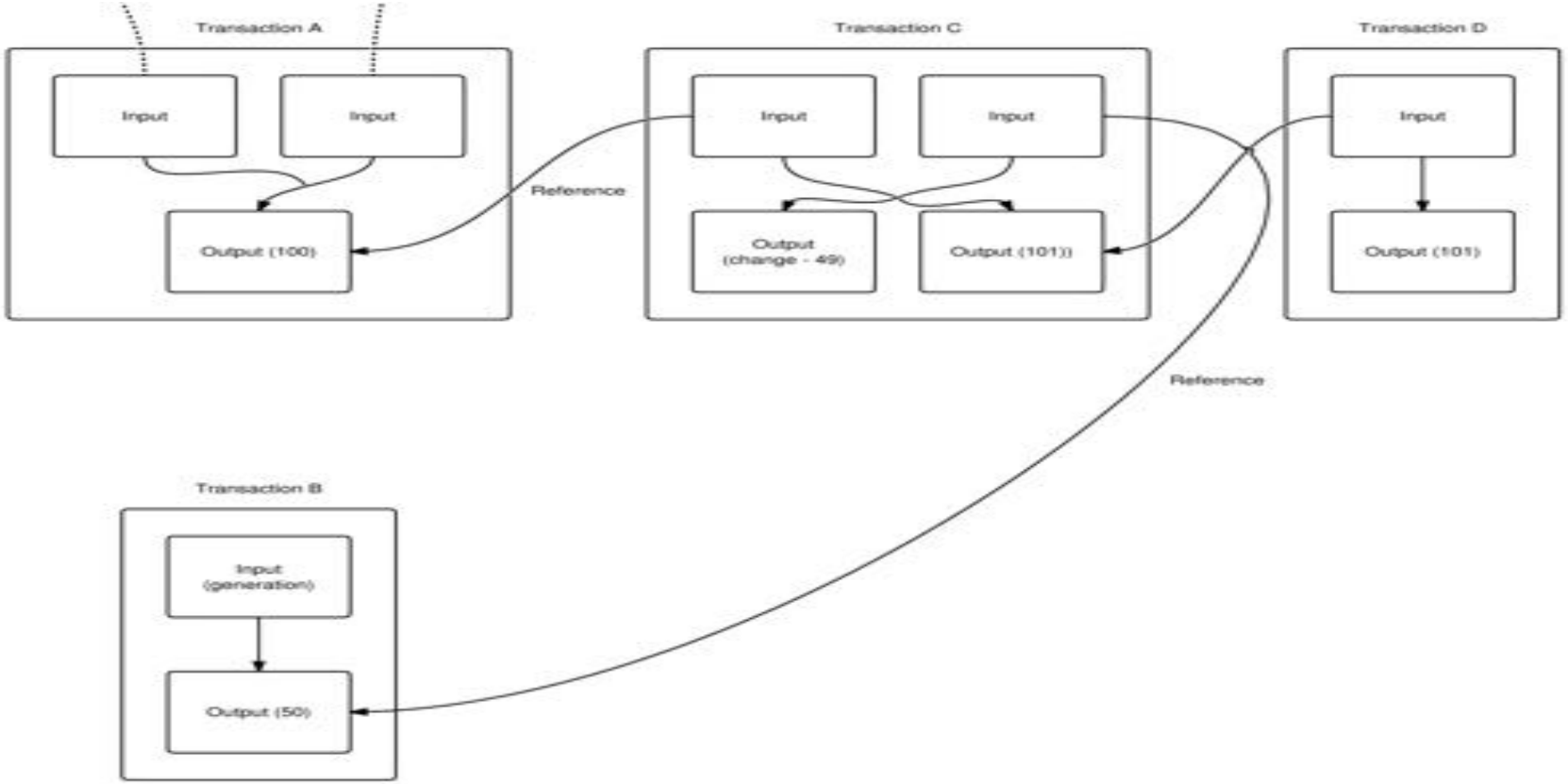
- Searchable Encryption techniques
- Fully Homomorphic encryption
- Verifiable computation

Blockchains

Activities of our team

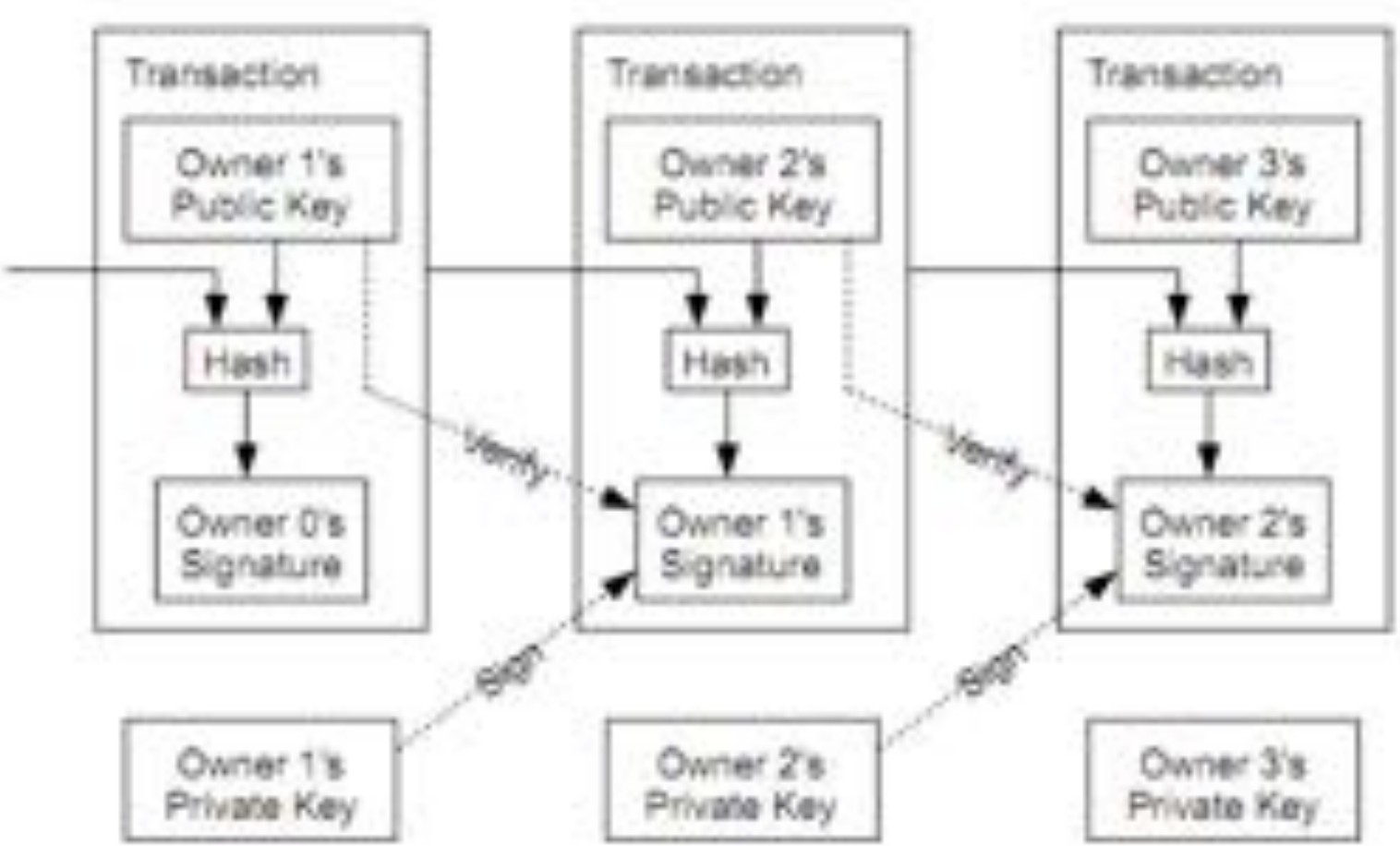
- RETRICOIN: Altcoin which uses proof of space, instead of proof of work
- Clique based proofs of work
- Countering collusion attacks of mining pools
- Analyzing Bitcoin transaction graphs
- Smart contracts for decentralized applications

Bitcoin Transaction



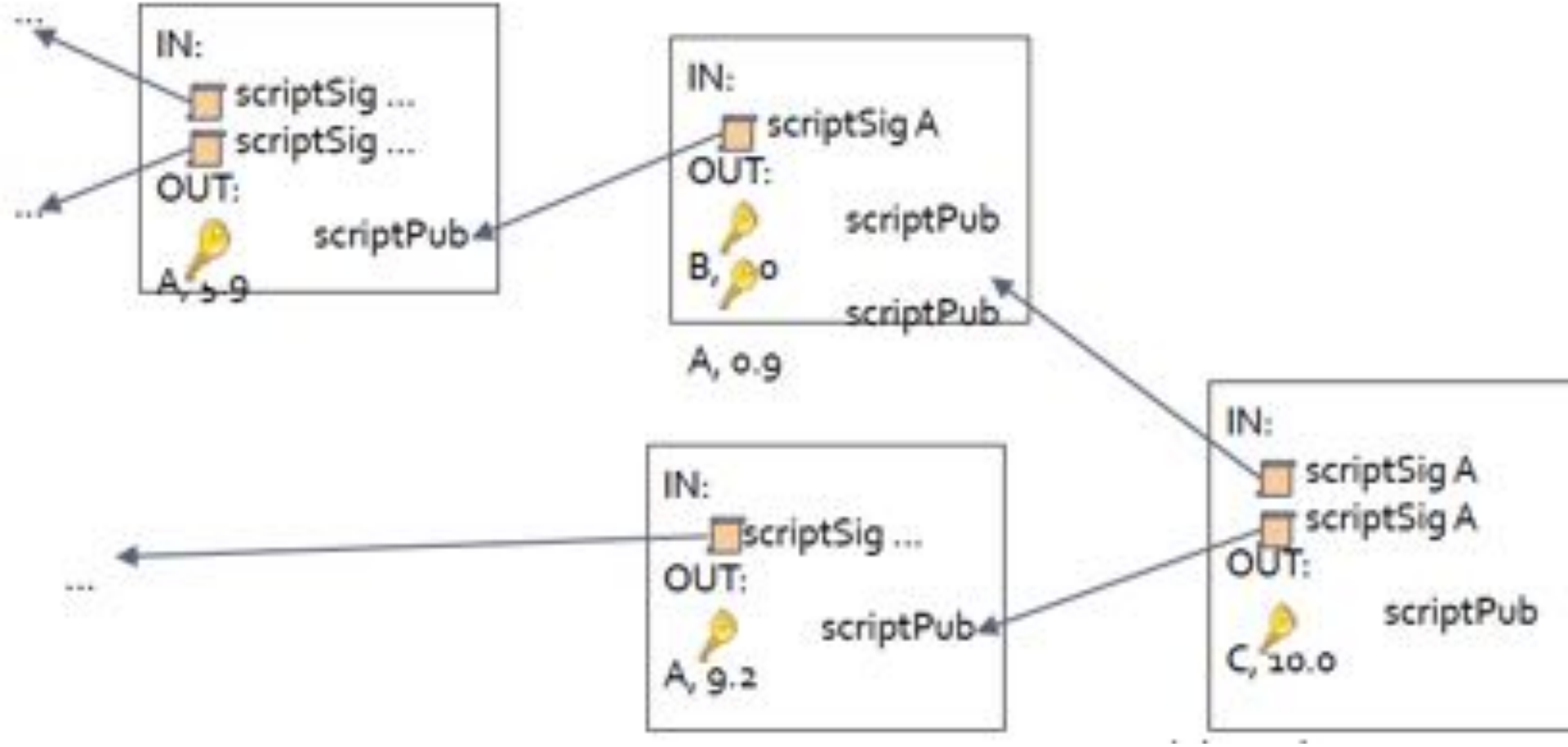
Ref: Bitcoin wiki

Bitcoin Transactions



Bitcoin by Satoshi Nakamoto

Bitcoin Transactions

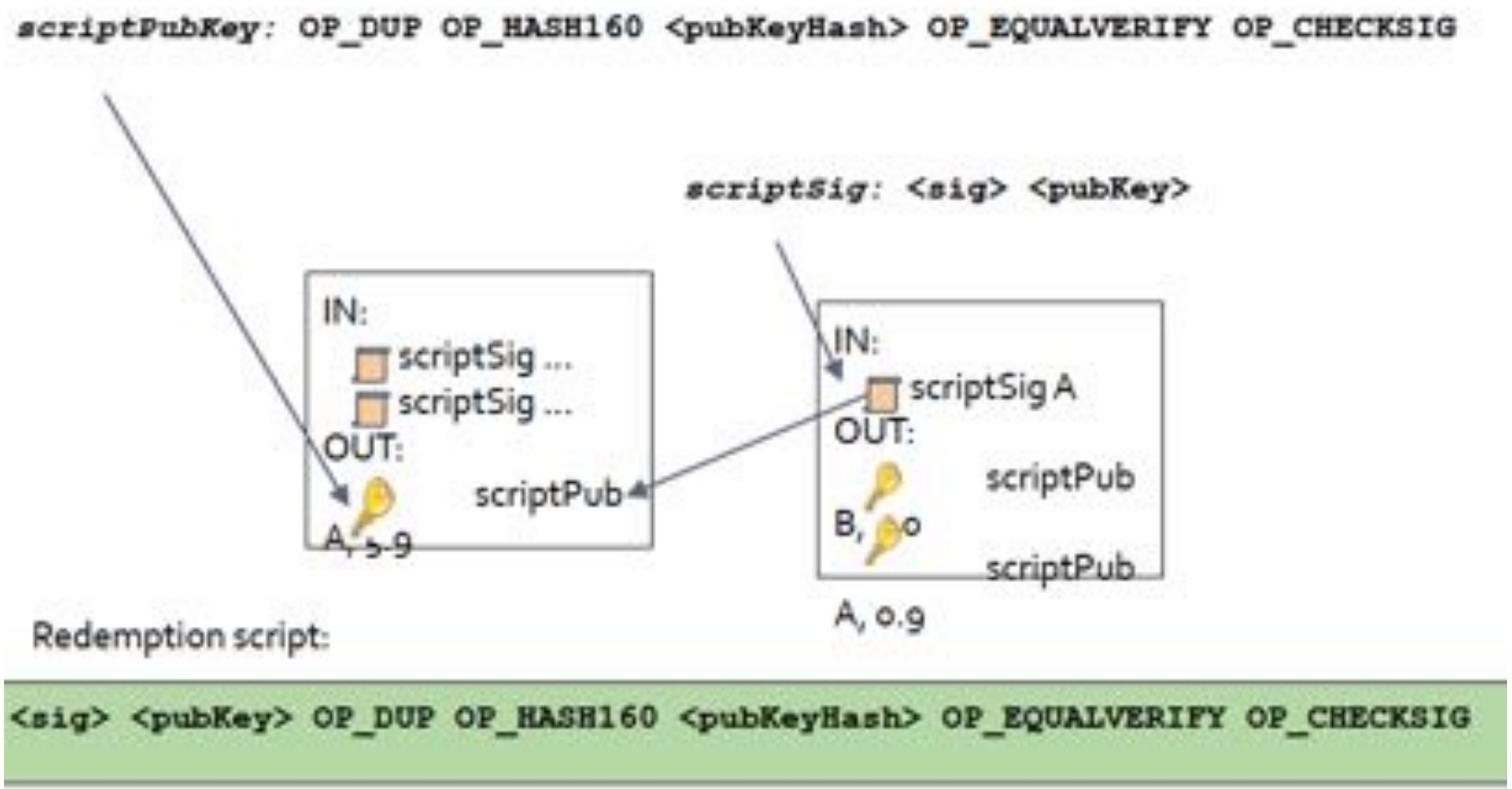


Ref:
https://www.ece.cmu.edu/~ece734/fall2014/lectures/21.Bitcoin.pdf

How a bitcoin transaction looks like

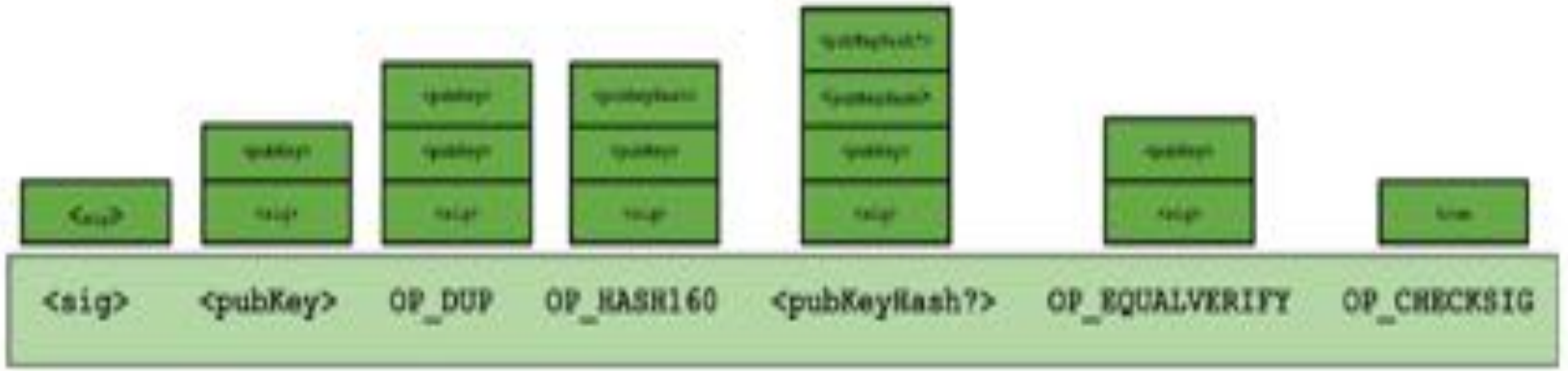
https://blockchain.info/rawtx/4cc38b124e7c98ad1d8134cba0f00ad3a28f429015ff83007cc496154791c51b

Bitcoin transaction scripts



Ref:
https://www.ece.cmu.edu/~ece734/fall2014/lectures/21.Bitcoin.pdf

Execution of a script



Blockchain protocols

- Certificate management
- Decentralized KYC
- Reputation and recommender systems
- IoT
- Many more
- Based on smart contracts

Smart Contracts

- Proposed by Nick Szabo around 1993
- Automatic contracts, triggered when certain conditions are met
- Stored in blockchain and publicly verifiable
- Many available platforms:
- Ethereum • Ripple • Stellar
- Tendermint • Factom • Hyperledger

Ethereum

- Developed by Vitalik Buterin, in 2013 for building decentralized applications
- Initially developed by Ethereum Switzerland GmbH (*EthSuisse*) and the Ethereum Foundation
- Smart contracts on Ethereum are written Solidity language

PROBLEM 3

- Smart contracts, blockchain applications


The ROYAL TREAT

Curiouser and curiouser!