


Some Cryptographic Tools (in Blockchains)

Manoj Prabhakaran

Outline

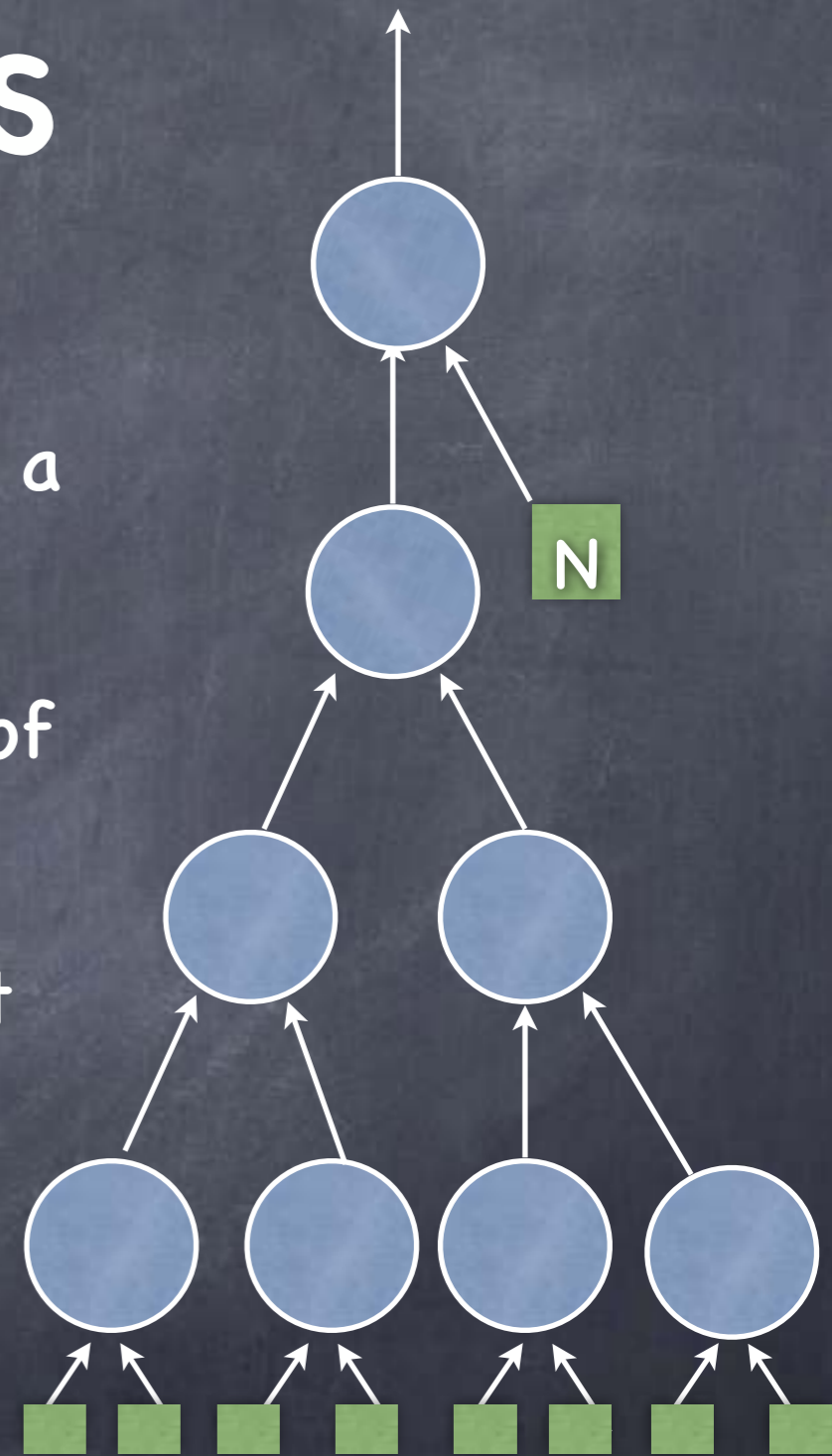
- Cryptographic tools for secure communication:
 - Hash Functions, Digital Signatures and Block Ciphers
- Other cryptographic tools (potentially) useful in blockchains
 - Zero-Knowledge (ZK) Proofs, SNARKs
 - Ring Signatures
 - VRFs
 - ZK Sets
 - Secure Multi-Party Computation & Computation on encrypted data

Hash Functions

- An efficiently computable function H that maps arbitrarily long strings to short fixed-length strings
- Main property sought: **Collision Resistance**
 - It should be computationally infeasible to find two different strings x, y s.t. $H(x) = H(y)$
 - Output cannot be too short: will be able to find collisions by random search ("birthday attack")
- Several other properties assumed in many applications
 - For any input, the output should be "random"; cannot find (x,y) s.t. x is short and $y=H(x)$, except by picking x first and evaluating it; ...
- A formal model: **Random Oracle Model**
 -  ROM is a heuristic model: Can do provably impossible tasks in this model!

Merkle Trees

- Originally devised (and patented) as part of a signature scheme, by Merkle [1979]
- A general technique for “domain extension” of hash functions
 - Given H that maps $2n$ -bit strings to n -bit strings, design a hash function H^* that can handle arbitrarily long strings
- Preserves collision resistance
- Internally used in many standard hash functions (e.g. SHA256)



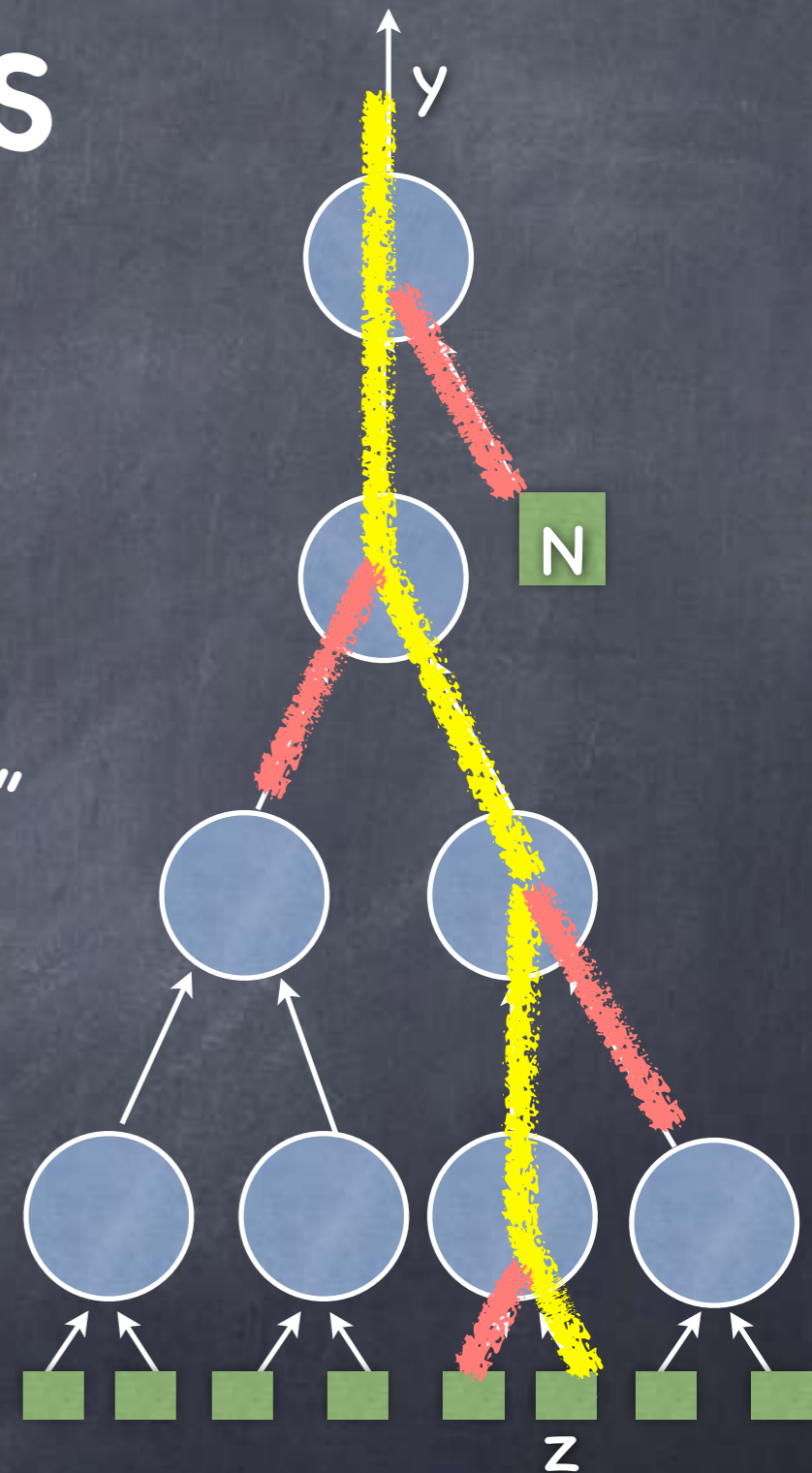
Merkle Trees

- A blockchain has a Merkle tree in it
- The "Genesis" block is a leaf
 - (Length not needed, since the genesis block is known)
- New block added at a new root



Merkle Trees

- Supports succinctly proving that “ $y=H(\text{msg})$ for some msg whose i^{th} block is z ”
- Collision resistance ensures that one cannot give a proof for two different values z
- In a balanced tree, the proof length is logarithmic in the total number of blocks



Digital Signatures

- Message authentication is a basic goal of cryptography
- Message Authentication Codes: Sender authenticates a message to a receiver by encoding it using a secret key; receiver uses the key to verify
- Digital Signature: Key used to encode (sign) is different from the key used to verify. Sender's "public key" is used to verify her signature (and it serves as her "virtual identity") e.g., RSA function
- One approach: Uses a "trapdoor function" (f, f^{-1}) , where f^{-1} is the secret trapdoor key, and f is the public key. H is a public hash function (modelled as a Random Oracle)
 - **$\text{Sign}(M) = f^{-1}(H(M))$. $\text{Verify}(M,s)$: Check $H(M) = f(s)$**

PRF & Block Cipher

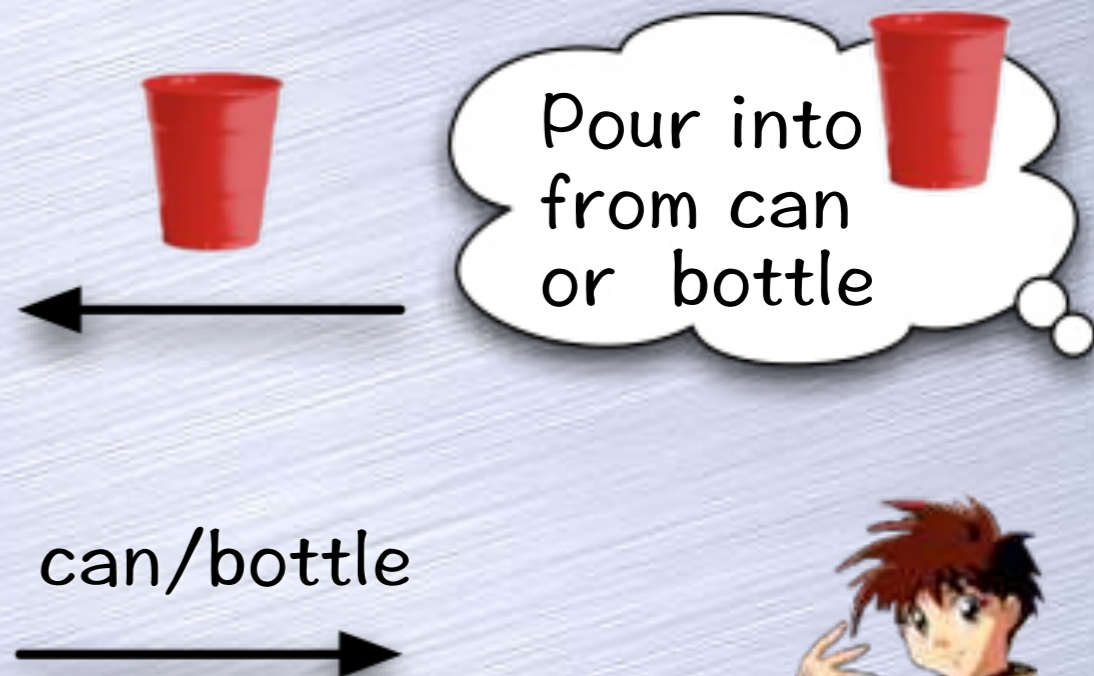
- A Pseudorandom Function is an important primitive in symmetric key cryptography
 - A PRF F takes a key K and an input x and outputs $y = F_K(x)$
 - Suppose Alice picks K at random, and then Bob queries her with x 's of his choice to get back $F_K(x)$. Bob can't tell if Alice is simply sending back random values instead!
- Used to build symmetric key encryption and Message Authentication Codes [e.g., if m short, $\text{Enc}_K(m;r) = (r, F_K(r) \oplus m)$ and $\text{MAC}_K(m) = F_K(m)$]
- A Block-Cipher is a practical version of a PRF, with more features
 - In particular, a Pseudorandom Permutation (for each K a one-to-one function) that is invertible given K
- Several modes of operation, to handle long messages

Zero-Knowledge Proof

- In cryptographic settings, often need to be able to verify various claims
 - e.g., 3 encryptions A, B, C are of values a, b, c s.t. $a = b + c$
 - Proof 1: Reveal a, b, c and how they get encrypted into A, B, C
 - Proof 2: Without revealing anything at all about a, b, c except the fact that $a = b + c$?
 - Zero-Knowledge Proof!

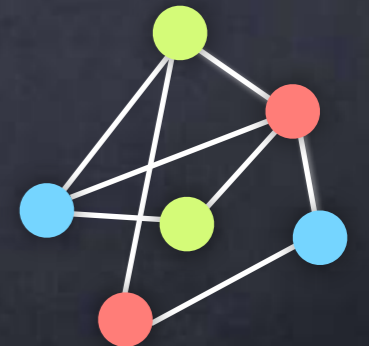
An Example

- Coke in bottle or can
- Prover claims: coke in bottle and coke in can are different
- ZK proof:
 - prover tells whether cup was filled from can or bottle
 - repeat till verifier is convinced



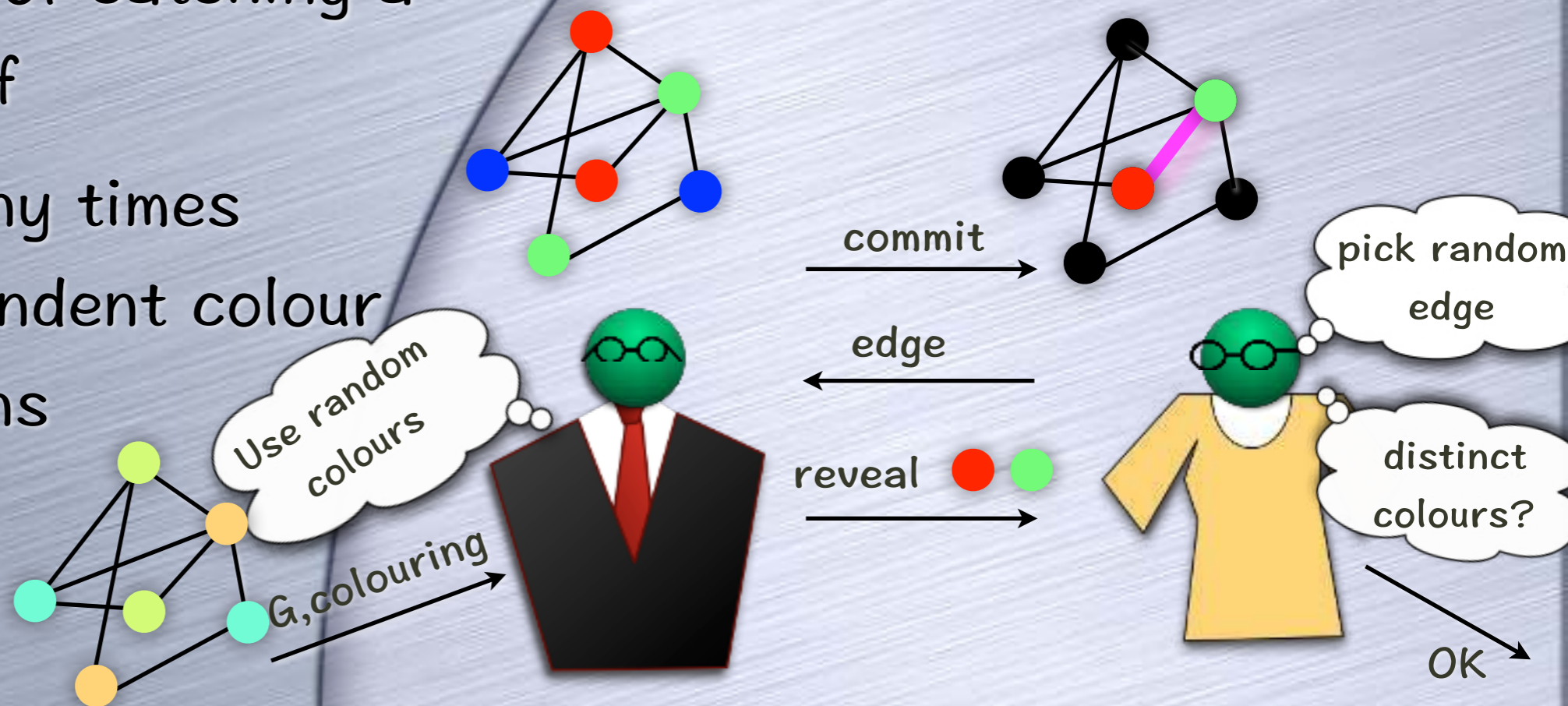
Commitment

- The functionality of **Commitment**:
 - Committing to a value: Alice puts the message in a box, locks it, and sends the locked box to Bob, who learns nothing about the message
 - Revealing a value: Alice sends the key to Bob. At this point she can't influence the message that Bob will get on opening the box.
- Implementation in the Random Oracle Model: $\text{Commit}(x) = H(x,r)$ where r is a long enough random string. To reveal, send (x,r) .
- An Example: To prove that the nodes of a graph can be coloured with at most 3 colours, so that adjacent nodes have different colours



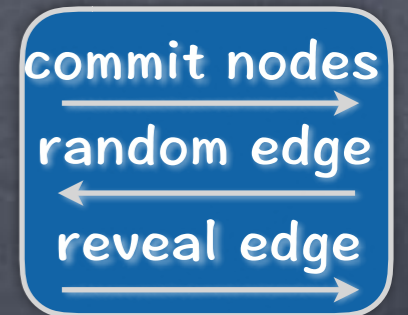
A ZK Proof for Graph Colourability

- Uses a commitment protocol as a subroutine
- At least $1/\#\text{edges}$ probability of catching a wrong proof
- Repeat many times with independent colour permutations



Zero-Knowledge Proof

- ZK Proofs are typically interactive, with the verifier sending random challenges to the prover
 - e.g., in the graph colourability protocol
- Non-interactive variants possible
- One approach: In the Random Oracle Model
 - The random challenge by the verifier is replaced by $H(m_1)$, where m_1 is the first message (commitment) sent by the prover
 - Non-interactive Proof: (m_1, m_3) , where m_3 is the response for the challenge $H(m_1)$



ZK-SNARK

- **Succinct Non-Interactive Argument of Knowledge (SNARK)**
 - Succinct means that the proofs are short and can be verified much faster than verifying the claim from the original witness (e.g., the colouring)
 - Argument is just a technical variant of a proof
 - ... of Knowledge: Not just proving that there is a witness (e.g., colouring) but that the prover “knows” such a colouring
- ZK-SNARK: also zero-knowledge
- E.g., Construction from Probabilistically Checkable Proofs (not ZK, not short) combined with a Merkle Tree and Random Oracles
- More efficient/practical schemes known, especially for proving special classes of statements

Ring Signatures

- Signature: A proof that a message has been endorsed by a party
 - Unforgeability: Even after seeing many proofs, can't create a new proof (for a new message). c.f. "Simulation Soundness"
- Ring Signature: Proof that a message has been endorsed by one among a set of parties (w/o revealing who)
 - Recall trapdoor permutation based signature:
Verify_f(M,s): Check $H(M) = f(s)$
 - Suppose we require $H(M) = f_1(s_1) \oplus \dots \oplus f_N(s_N)$ where f_1, \dots, f_N are N permutations
 - Can sign using $f_k^{-1} : s_i$ random for $i \neq k$, $s_k = f_k^{-1}(H(M) \oplus \bigoplus_{i \neq k} f_i(s_i))$
 - Security? Replace \oplus with a combining function

Ring Signatures

- Signature: A proof that a message has been endorsed by a party
 - Unforgeability: Even after seeing many proofs, can't create a new proof (for a new message). c.f. "Simulation Soundness"
- Ring Signature: Proof that a message has been endorsed by one among a set of parties (w/o revealing who)
- cf. Group Signatures: The group is pre-determined and involves a group manager who can trace the actual signer
- More generally, "Attribute-Based Signatures"
 - Endorse a message with ones credentials, but claim/prove only some property about the credentials held

Verifiable Random Functions

- VRFs are Pseudo-Random Function (PRFs) with an additional verifiability feature
- Recall PRF:
 - If Alice picks a random key K , and answers Bob's queries x with $F_K(x)$, to Bob it would look like Alice is answering with random values
- What if Alice is actually sending random values?!
 - Alice publishes a "public key" PK related to K
 - With $F_K(x)$ Alice attaches a "ZK proof" that Bob can verify using PK

Verifiable Random Functions

- Used in Algorand for selecting various committees (“sortition”)
- Each party has a secret-key K
- It computes $F_K(\text{public info})$ to obtain a value that determines whether it is selected or not (also based on the “stake” of the party)
- Nobody else knows the output of the VRF, until it is disclosed by the party
- At that point, the party can prove to everyone that the purported output of the VRF is correct


ZK Sets

- A commitment to a set of values (e.g., 256-bit strings)
 - Can prove in ZK, membership or non-membership of any element
 - Without revealing even the size of the set!
- Set can contain say, crypto coins, (hash of) transactions, etc.
- Construct from Merkle Trees, using hash functions which are also “mercurial commitments”
 - K can be sampled so that H_K is collision resistant or to allow easily solving for r s.t. $H_K(x;r)=y$, for any x,y . Can prove K sampled in one way or the other.
 - Can be based on the “hardness of discrete logarithm” assumption

ZK Sets

- A commitment to a set of values (e.g., 256-bit strings)
 - Can prove in ZK, membership or non-membership of any element
 - Without revealing even the size of the set!
- Set can contain say, crypto coins, (hash of) transactions, etc.
- Alternate formulation: Statistically Hiding Sets
 - Hiding is information-theoretic (not necessarily admitting efficient simulation of proofs)
 - Only soundness relies on computational assumptions (while the proofs are given)
 - Everlasting security!

MPC

- Secure Multi-Party Computation (a.k.a. MPC, a.k.a. SMC)
- Mutually distrusting parties facilitate collaboration among themselves by collectively emulating a trusted party 
- If a trusted party were available, every party can send its inputs to it. The trusted party can carry out a computation on the collective data and send outputs to various parties
 - E.g., an auction without a trusted auctioneer
- Most cryptographic tasks can be seen as special cases of MPC (with application-specific interaction pattern, security requirements, etc.)
 - e.g., ZK proofs are MPC for the “proof functionality.” VRF used to implement the “sortition functionality” non-interactively.
- Blockchains for MPC: e.g., implementing “fines” for aborting, to obtain “fair MPC”

Summary

- Cryptographic tools for secure communication:
 - Hash Functions, Digital Signatures and Block Ciphers
- Other cryptographic tools (potentially) useful in blockchains
 - Zero-Knowledge (ZK) Proofs, SNARKs
 - Ring Signatures
 - VRFs
 - ZK Sets
 - Secure Multi-Party Computation & Computation on encrypted data

Later