CMPSC 390
Crypto Fundamentals

Janyl Jumadinova

Credit: Authors of “Bitcoin and Cryptocurrency Technologies”

January 21-22, 2021
Identity: an account (node) in the system.
Transactions: sending and receiving units of cryptocurrency.
Distributed Ledger: a public record of transaction history (blockchain).
Trustless Consensus: agreement on changes to the ledger.
Cryptographic Hash Functions: ensure trust in communication in a trustless system.
Cryptographic Hash Functions: ensure trust in communication in a trustless system.

- Input of any size, fixed-size output (e.g., 256 bits).
Security Properties

Hash property 1:

**Collision resistance**

Nobody can find \( x \) and \( y \) such that \( x \neq y \) and \( H(x) = H(y) \).

Nobody can find this!
Collisions

Collisions exist ...
Collisions exist ... but can anyone find them?

- Try $2^{130}$ randomly chosen inputs.
- There is 99.8% chance that two of them will collide.
Collisions exist ... but can anyone find them?

- Try $2^{130}$ randomly chosen inputs.
- There is 99.8% chance that two of them will collide.
- This works no matter what $H$ is ...
Collisions exist ... but can anyone find them?

- Try $2^{130}$ randomly chosen inputs.
- There is 99.8% chance that two of them will collide.
- This works no matter what $H$ is ... but it takes too long to matter.
Collisions exist ... but can anyone find them?

- Try $2^{130}$ randomly chosen inputs.
- There is 99.8% chance that two of them will collide.
- This works no matter what $H$ is ... but it takes too long to matter.
- No $H$ has been **proven** collision-free.
Hash property 2:

Hide information

Given $H(x)$, it is infeasible to find $x$. 
(com, key) := commit(msg)  
match := verify(com, key, msg)  

To seal msg in envelope:  
   (com, key) := commit(msg) -- then publish com  
To open envelope:  
   publish key, msg  
   anyone can use verify() to check validity
Security Properties

Hash property 3:

**Computational efficiency**

Set of computation to get a hash should not take a long time.
SHA-256 hash function

Theorem: If c is collision-free, then SHA-256 is collision-free.

- **SHA-256**: A cryptographic hash function designed by the NSA.
- Bitcoin uses SHA – 256² ("SHA-256 squared"), meaning that $H(x)$ actually means $SHA256(SHA256(x))$. 
Generating hash function in Python

```python
import hashlib

hashlib.sha256(string.encode()).hexdigest()
```
- A pointer to where some info is stored, and
- (cryptographic) hash of the info.
Hash Pointer

- A pointer to where some info is stored, and
- (cryptographic) hash of the info.

*With a hash pointer, we can:*

- ask to get the info back, and
- verify that it hasn’t changed.
Key Idea: build data structures with hash pointers

will draw hash pointers like this
Blockchain

Linked list with hash pointers = “block chain”

prev: $H(\cdot)$

prev: $H(\cdot)$

prev: $H(\cdot)$

data

data

data
**Binary tree:** a tree data structure with each node having at most two children.
Merkle Tree

A binary tree with hash pointers

```
  H( )  H( )
 /       /
H( )  H( )
 |       |
H( )  H( )
 |       |
H( )  H( )
 |
(data) (data) (data) (data) (data) (data) (data) (data)
```
Advantages of Merkle Trees

- Just need to remember the root hash.
- Can verify membership in $O(\log n)$ time/space.
Blockchain

BLOCK HEADER

PREV BLOCK HASH

NONCE

MERKLE ROOT
Digital Signatures

Want:

- Only you can sign, buy anyone can verify.
- Signature is tied to a particular document (can’t be cut-and-pasted to another doc).
Digital Signatures

Want:

- Only you can sign, buy anyone can verify.
- Signature is tied to a particular document (can’t be cut-and-pasted to another doc).

Public Key Cryptography:

a cryptographic system that allows for secure dissemination of identity and authentication of valid messages.
API for Digital Signatures

\[(sk, \ pk) := \text{generateKeys(keysize)}\]
\[
\text{sk: secret signing key}
\]
\[
\text{pk: public verification key}
\]

\[\text{sig} := \text{sign}(sk, \ message)\]

\[\text{isValid} := \text{verify}(pk, \ message, \ sig)\]
The algorithm used by Bitcoin to generate public keys and verify transactions.

A variant of standard DSA but with elliptic curves.

Good randomness is essential.
ECDSA Example

Instructor’s chicken scratches ...
Public Key Generation

Input: public key
Output: corresponding private key

256 bit private key, takes $O(\sqrt{n})$ operations to crack
15 * $\text{pow}(2,40)$ hashes per second on the ENTIRE Bitcoin network

$$\text{pow}(2,128) / (15 * \text{pow}(2,40)) / 3600 / 24 / 365.25$$
$$= 0.6537992112229596e18$$

650 million billion years
Public Key as Identity

Make new identity:

- Create a new, random key-pair \((sk, pk)\).
- You control the identity, because only you know the secret key, \(sk\).
Decentralized Identity Management

- Anyone can make a new identity at any time, as many as want.
- There is no central point of coordination.
- Identities are called addresses in Bitcoin.
Privacy

- Addresses not directly connected to real-world identity.
- But observer can link together an address’s activity over time, make inferences.
- Will discuss privacy in Bitcoin in more detail later.