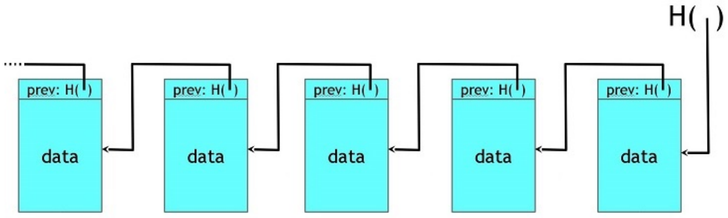
On the cryptography of Distributed Ledger Technology

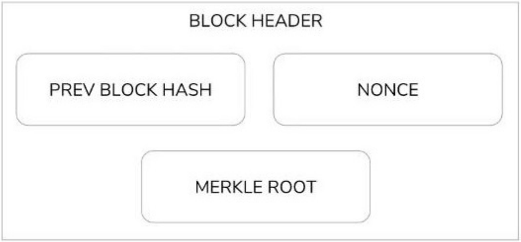
Andrea VISCONTI

Department of Computer Science
Università degli Studi di Milano

Overview

A chain of blocks...





1. Hash Functions

Hash Functions

What are hash functions?

Hash Functions are cryptographic functions. They

- can be **efficiently computed**;
- input **any string** of any size;
- provide a **fixed size output**.

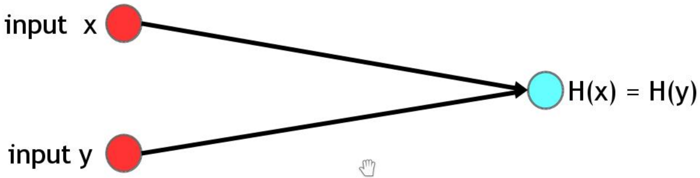
Some examples of hash functions:

- RIPEMD160 –> a message digest (MD) of 160 bits;
- SHA1 –> a message digest of 160 bits;
- SHA-256 –> a message digest of 256 bits;
- MD5 –> a message digest of 128 bits;
- SHA-3 –> . . .

Hash Functions

Properties...

1. **Collision-free**

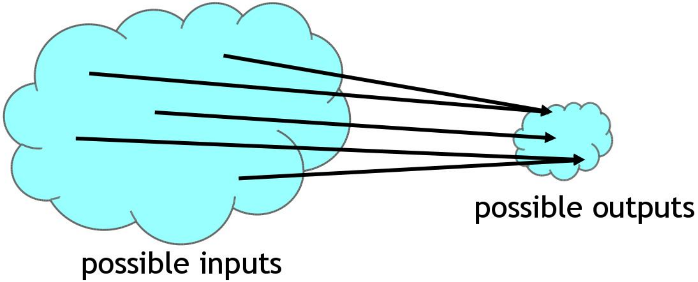


Nobody is able to find two strings x and y s.t. $x \neq y$ and $H(x) = H(y)$

Collision do exist!

Indeed,

- we input any string of any size — say n, this means 2^n;
- we provide a fixed size output — for example 256 bits, this means 2^{256};



But are you able to find them?

I can suggest you an algorithm...

If you compute the hash of 2^{130} strings, you have more than 99% chance that two inputs collide (at least two!!).

Unfortunately these **collisions are not findable by regular users** using regular computers because this process takes a very very long time.

This number is astronomical!

The **probability** to find a collision is **negligible**.

Hash Functions

We have understood that

- **no hash function** has been **proven to be collision free**;
- it is very hard to find a collision.

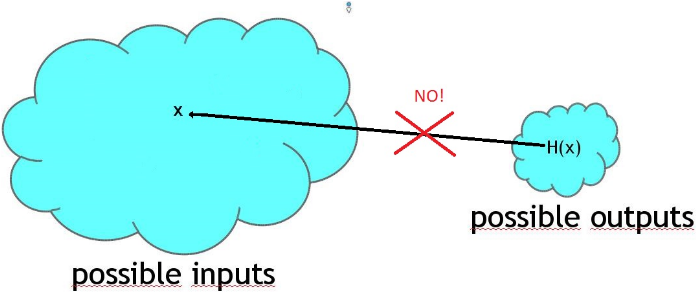
So, **we choose to believe** that hash functions are collision free.

Therefore we assume that if $H(x) = H(y)$ then $x = y$.
This means that

- we do not store x and y (that can be huge) but only their hashes (usually very small);
- we are able to compare data using only a bunch of bits.

Properties...

2. **Preimage resistant**



Given $H(x)$, it is computationally infeasible to find x.

Hash Functions

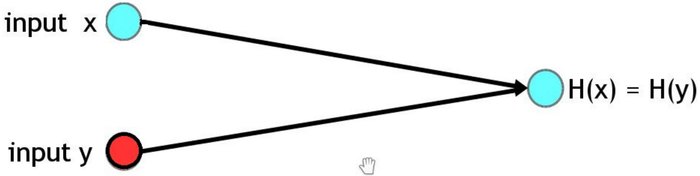
Preimage resistance refers to the hash function's ability to be non-reversible: **one-way function**.

In particular, this property suggest us that

- there is no value of x which is particularly likely;
- attackers have to try all possible input values to find x. Again, this number is astronomical!
- we can hide x using $H(x)$.

Properties…

3. **Second preimage resistant**



Given x, it is computationally infeasible to find y s.t. $H(x) = H(y)$.

These three properties ensure that it is hard to cheat.

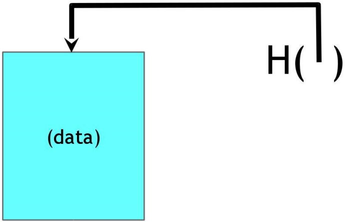
2. Hash Pointers

Hash Pointers

What they are...

A hash pointer

- is a data structure;
- points to where some information is stored;
- also points to a Hash of the information.



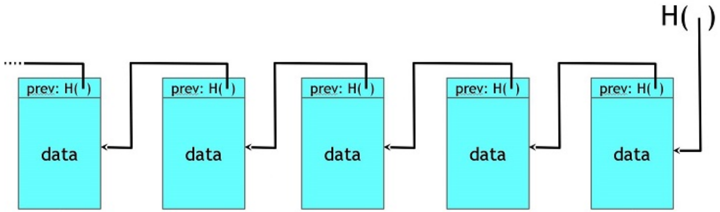
Hash Pointers

What their properties are...

A hash pointer

- allows to **retrieve information**;
- gives us the possibility to check that the information has **not been modified**.

Using hash pointers we can build a **linked list** — i.e. a chain of blocks

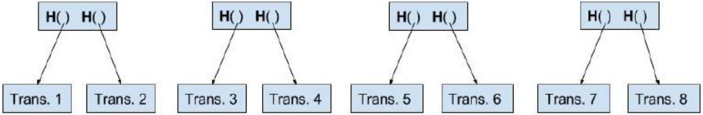


Hash Pointers

Using hash pointers we can build a **binary tree** — i.e. Merkle tree.

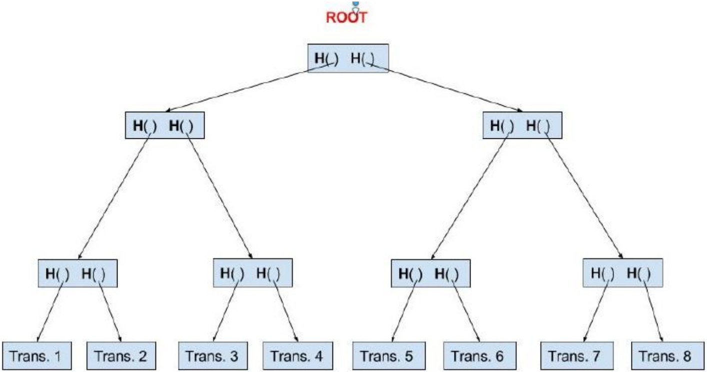
As an example, consider the set of eight transactions (leaves of the tree).

Transactions are grouped into pairs of two. Then, a data (composed of two hash pointers) is computed.



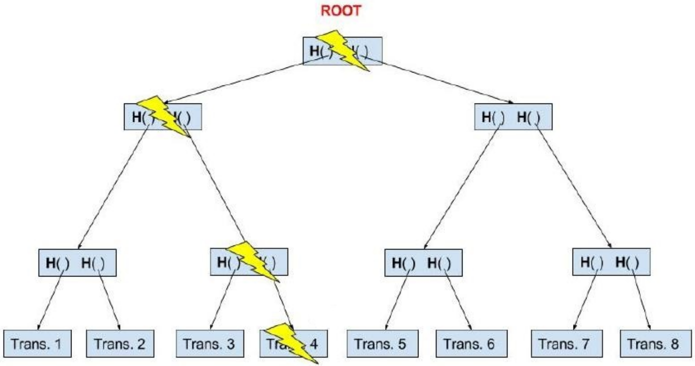
Hash Pointers

This process is iterated until a single data is reached. We call this data **root of the tree**.



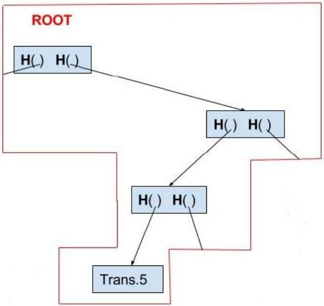
. . . and this is our Merkle tree.

With a Merkle tree we are able **to detect tampering**. Indeed, if an **attacker tampers a transaction**, or an intermediate node, the root immediately changes value.



We also use a Merkle tree to **provide membership**.

If we want to prove publicly that a certain transaction is contained in the block, and we know only the root of the tree, we need to show the nodes in the path from the transaction to the root.



Hash Pointers

To sum up, hash pointers provide us the possibility to

- build data structures;
- detect tampering;
- verify membership;

3. Digital signatures

Digital Signatures

What are digital signatures?

A **Digital Signature** is a cryptographic tool based on **secrete keys** s_u and **public keys** p_u which provides a solution for creating legally enforceable electronic records.

Every user **generates** and **takes responsibility** for its own pair of keys — e.g. Alice holds s_{Alice} and p_{Alice}, Bob holds s_{Bob} and p_{Bob}, Carl holds . . .

Digital Signatures

Digital signatures guarantee:

- **Authentication**: the receiver can verify the identity of the signer.
- **Non-repudiation**: the signer cannot deny to have signed a message.
- **Integrity**: attackers cannot modify a signed message without invalidating the signature.

Digital signatures

- provide **you** the possibility **to sign** a document with your secret key;
- provide **anyone** the possibility **to verify** the signature with your public key;
- are **tied to specific documents** — i.e. no cut&paste.

Digital Signatures

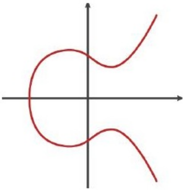
Alice wants to send a signed document to Bob

She needs a digital signature algorithm — e.g. Elliptic Curve Digital Signature Algorithm (ECDSA)

Alice and Bob will use the following curve over a prime field $\mathbb{Z}_p$. An EC is a set of solutions (x, y) of the Weierstrass equation:

Weierstrass Equation:

$$y^2 + a_1xy + a_3y - x^3 - a_2x^2 - a_4x - a_6 = 0$$
$$a_1, a_2, a_3, a_4, a_6 \in \mathbb{K}$$



Digital Signatures

Thus, Alice and Bob have to

- choose an EC over a prime field;
- choose the signature algorithm (ECDSA);
- generate their private/public keys: (s_{Alice}, p_{Alice}) and (s_{Bob}, p_{Bob});

Using s_{Alice}, Alice can sign the **hash** of the document and send it to Bob.

Bob can verify the digital signature with Alice's public key.

What about security?

The security of Elliptic Curve Digital Signature Algorithm (ECDSA) is based on the Elliptic Curve Discrete Logarithm Problem (ECDLP):

"Given an elliptic curve EC over $\mathbb{Z}_p$, a point P on it of order n and another point Q, it is very hard to find k such that $k \cdot P = Q$"

We are talking about **scalar multiplication**: $P + P + \cdots + P = Q$

What attackers can do...

Malicious users can try to mount an attack to

- ECDSA;
- the hash function used — e.g. birthday attack;
- the implementation — we need good source of randomness to prevent the leakage of private keys.

In 2010, an attack has been mount to recover Sony's private key (used to sign PlayStation 3's software).

4. An application of DLT and smart contracts

An application of DLT and smart contracts

A funded blockchain project and a joint work with:

- Accademia di Belle Arti di Brera;
- Department of Cultural Heritage and Environment, Università degli Studi di Milano;
- Department of Computer Science, Università degli Studi di Milano;

An application of DLT and smart contracts

Main idea

Art is a universal language. Why don't we share it?

We will **focus on young photographers** and their images.

We need to protect their artworks because

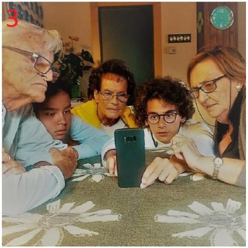
- photographers,
- auction houses,
- galleries,
- world's art collectors
- ...

may **have conflicting interests**!!

An application of DLT and smart contracts

A simple challenge...

Photography is one of the best deal in the art market . . . Who would buy one of these images for $1,000?



Which would you bet on?

. . . without expertise!

An application of DLT and smart contracts

These images are not randomly chosen. Indeed...

Vibeke Tandberg is a Norwegian artists. She is known for manipulating her images to contort human figures and the spaces they occupy.

Estimated about $\$1,000$.

An application of DLT and smart contracts

These images are not randomly chosen. Indeed...

Jeffrey Wall is a well-known Canadian artist. He is an influential photographer.

"*Card players*", estimated about $\$300,000 - \$400,000$.

An application of DLT and smart contracts

These images are not randomly chosen. Indeed...

Maddalena is a professor @UniMI. She is not a photographer.

“Relatives”, estimated about $0.

An application of DLT and smart contracts

This simple challenge show us why **expertise is so important**.

A blockchain can be used **not only to store** the fingerprint of images!

We also need to store their **history**, what the viewer **observers**, **thinks**, and **feels** about these images.

. . . and to do so, we need to design and implement a smart contract application.

Thanks for your attention!

andrea.visconti@unimi.it
www.di.unimi.it/visconti