The Theory Behind Blockchains (Spring 19)
Recitation 1

Eliad Tsfadia

1 Collision Resistant Hash Functions and Merkle Trees

1.1 Collision Resistant Hash (CRH)

Informally, we say that a function h is a *collision resistant* hash function, if it is infeasible to find (efficiently) two different inputs $x \neq x'$ such that $h(x) = h(x')$. In practice, we use functions like SHA-2, SHA-3. However, note that this definition cannot be true in theory, since for any fixed function $h\colon \{0,1\}^* \to \{0,1\}$ there is an efficient algorithm A_h that outputs a collision: Just hardwire two such input x,x' with $h(x) = h(x')$ into A_h. Therefore, in theory we define collision resistant as a property of a family of functions, and not of a single function. For example, we might think of the family $\{\mathrm{SHA}(r,\cdot)\}_{r\in\{0,1\}^n}$, i.e. given a random r we obtain the function $\mathrm{SHA}_r(x) = \mathrm{SHA}(r,x)$. Now it is more reasonable to claim that it is infeasible to find a collision given SHA_r for a random r. But still, this is not enough if we want to achieve asymptotic result: Recall that in theory, in order to prove that something is "infeasible", we usually says that an efficient algorithm can only succeed with *negligible* probability.

Definition 1. *$f(n) \in \mathrm{negl}(n) \iff \forall$ polynomial p $\exists N \in \mathbb{N}$ s.t. $\forall n > N$ is holds that $f(n) < 1/p(n)$ (e.g. $f(n) = 2^{-n}$, $f(n) = n^{-\log\log n}$).*

In our example, SHA outputs a fixed length string (for example, SHA-256 outputs binary string of length 256). Therefore, regardless of the length of the random prefix r, it is possible to invert any SHA_r function within 2^{256}, and even 2^{128} steps (as we see next). Therefore, collision resistance is defined to be a property of sequence of families:

Definition 2 (Collision Resistant Hash Functions)**.** *A family of functions $\mathcal{H} = \{\mathcal{H}_n\colon \{0,1\}^* \to \{0,1\}^n\}$ is collision resistant, if*

$$\Pr_{\substack{h\leftarrow\mathcal{H}_n \\ (x,x')\leftarrow A(1^n,h)}}[x \neq x' \wedge h(x) = h(x')] \leq \mathrm{negl}(n),$$

for any PPT algorithm A.

For the sake of simplicity, we still sometimes say that a given function h is collision resistant, and one can think of it as a function that was taken from such collision resistant family.

Given a hash function $h\colon \{0,1\}^* \to \{0,1\}^n$, it is possible to find a collision just by evaluating h on $2^n + 1$ distinct elements (pigeonhole principle). But there is a more efficient way:

Fact 1 (Birthday Paradox)**.** *Let $h\colon \{0,1\}^m \to \{0,1\}^n$ and assume $m >> n$. Then there exists an algorithm that runs in $O(2^{n/2})$ steps and finds a collision.*

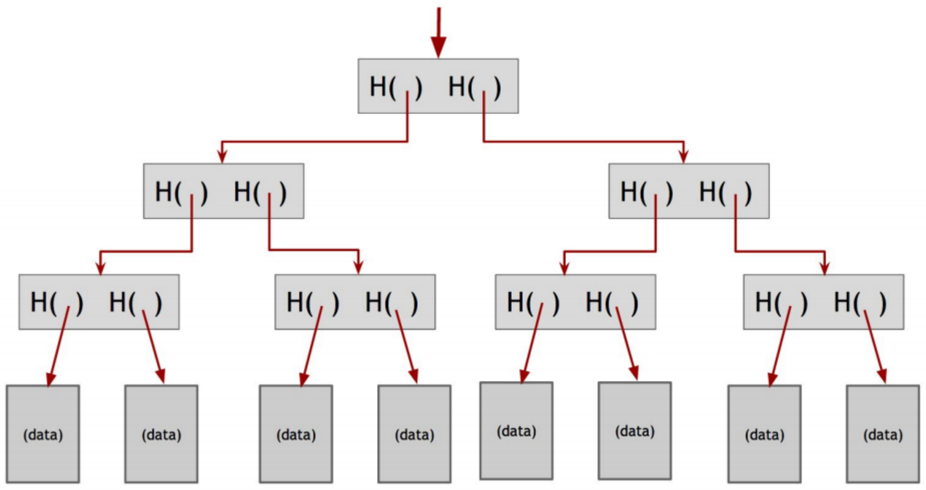
Proof sketch. The algorithm picks $10\cdot 2^{n/2}$ uniform and independent strings $x_1,\ldots,x_{2^{n/2}} \leftarrow \{0,1\}^m$ and search for a collision $x_i \neq x_j$ s.t. $h(x_i) = h(x_j)$. By the birthday paradox, the success probability is 90%. $\square$

Fact 2. *A quantum algorithm can find a collision in time $O(2^{n/3})$.*

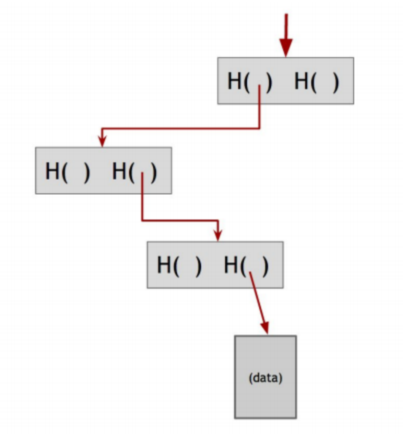
Still, in practice $O(2^{n/2})$ and even $O(2^{n/3})$ are very large if we consider $n \geq 256$.

1.2 Merkle Trees (MT)

Assume we are given a *collision resistant* hash function h. In Merkle tree, data blocks are grouped into pairs, and the hash of each of these pairs is stored in a parent node. The parent nodes are in turn grouped in pairs, and their hashes stored one level up the tree. This pattern continues up the tree until we reach the rood node.



Proof of Membership To prove that a data block is included in the tree, one only needs to show the blocks in the path from that data block to the root. Proof length and verification time and space: $\Theta(\log m)$ where m is the number of data blocks in the tree.



The following claim states that producing a cheating proof of membership is not feasible for an efficient algorithm.

Claim 3. *For any CRH family $\mathcal{H} = \{\mathcal{H}_n\}$ and any PPT algorithm A it holds that*

$$\Pr_{\substack{h \leftarrow \mathcal{H}_n \\ (T,\pi) \leftarrow A(1^n, h)}} [T \textit{ is an } h \textit{ consistant MT and } \pi \textit{ is a proof of membership for } x^* \notin T] \leq \operatorname{negl}(n)$$

Proof. Assume there exists PPT algorithm A such that the above probability equals to $\epsilon = \epsilon(n)$ and assume without loss of generality that it always outputs a valid tree T over $m = m(n) \in \operatorname{poly}(n)$ elements for $m = 2^\ell$ (denote them by $x_1, \ldots, x_m$). We show how to convert any cheating proof into a collision in $\mathcal{H}$, yielding an efficient algorithm that finds a collision in $\mathcal{H}$ with probability ϵ. Conditioned on a successful construction of a cheating proof, it holds that $\pi = x^*, y_1, y_1', y_2, y_2', \ldots, y_\ell, y_\ell', y_{\ell+1} = y_{root}$, where

1. $h(x^*) = y_1$.
2. for all $i \in [\ell]$, $h(y_i, y_i') = y_{i+1}$.
3. $x^* \notin \{x_1, \ldots, x_m\}$.

If $h(x^*) = h(x_i)$ for some i, then we found a collision in h. Otherwise $h(x^*) = y_1 \notin T_1$ (where we denote T_i as the nodes in the i'th level above the leafs). Since $y_\ell = y_{root}$, there must exists $i \in [\ell]$ such that $y_i \notin T_i$ and $y_{i+1} \in T_{i+1}$, and by assumption, $h(y_i, y_i') = y_{i+1}$. The above yields that there exists $\widetilde{y_i}, \widetilde{y_i}' \in T_i$ where $\widetilde{y_i} \neq y_i$ such that $h(\widetilde{y_i}, \widetilde{y_i}') = y_{i+1} = h(y_i, y_i')$, and we found a collision.

A sorted Merkle tree. This is just a Merkle tree where we take the blocks at the bottom, and we sort them using some ordering function. This can be alphabetical, lexicographical order, numerical order, or some other agreed upon ordering.

Proof of non-membership. With a sorted Merkle tree, it becomes possible to verify non-membership in a logarithmic time and space. That is, we can prove that a particular block is not in the Merkle tree. And the way we do that is simply by showing a path to the item that's just before where the item in question would be and showing the path to the item that is just after where it would be. If these two items are consecutive in the tree, then this serves as a proof that the item in question is not included. For if it was included, it would need to be between the two items shown, but there is no space between them as they are consecutive. We've discussed using hash pointers in linked lists and binary trees, but more generally, it turns out that we can use hash pointers in any pointer-based data structure as long as the data structure doesn't have cycles. If there are cycles in the data structure, then we won't be able to make all the hashes match up. If you think about it, in an acyclic data structure, we can start near the leaves, or near the things that don't have any pointers coming out of them, compute the hashes of those, and then work our way back toward the beginning. But in a structure with cycles, there's no end we can start with and compute back from.

□