Sudipta Halder (2021202011)

Use Merkle-Damgard transform to obtain a provably secure collision resistant hash function

Below is the merkle-damgard transform from a fixed-length hash function with input length 2l(n) and output length l(n). Basically, merkle-damgard construction enables us to create a variable length hash function.

CONSTRUCTION 4.11 The Merkle-Damgård Transform.

Let (Gen_h, h) be a fixed-length hash function with input length $2\ell(n)$ and output length $\ell(n)$. Construct a variable-length hash function (Gen, H) as follows:

- $\mathsf{Gen}(1^n)$: upon input 1^n, run the key-generation algorithm Gen_h of the fixed-length hash function and output the key. That is, output $s \leftarrow \mathsf{Gen}_h$.
- $H^s(x)$: Upon input key s and message $x \in \{0,1\}^*$ of length at most $2^{\ell(n)} - 1$, compute as follows:
  1. Let $L = |x|$ (the length of x) and let $B = \lceil \frac{L}{\ell} \rceil$ (i.e., the number of blocks in x). Pad x with zeroes so that its length is an exact multiple of ℓ.
  2. Define $z_0 := 0^\ell$ and then for every $i = 1, \ldots, B$, compute $z_i := h^s(z_{i-1} \| x_i)$, where h^s is the given fixed-length hash function.
  3. Output $z = H^s(z_B \| L)$

[1]

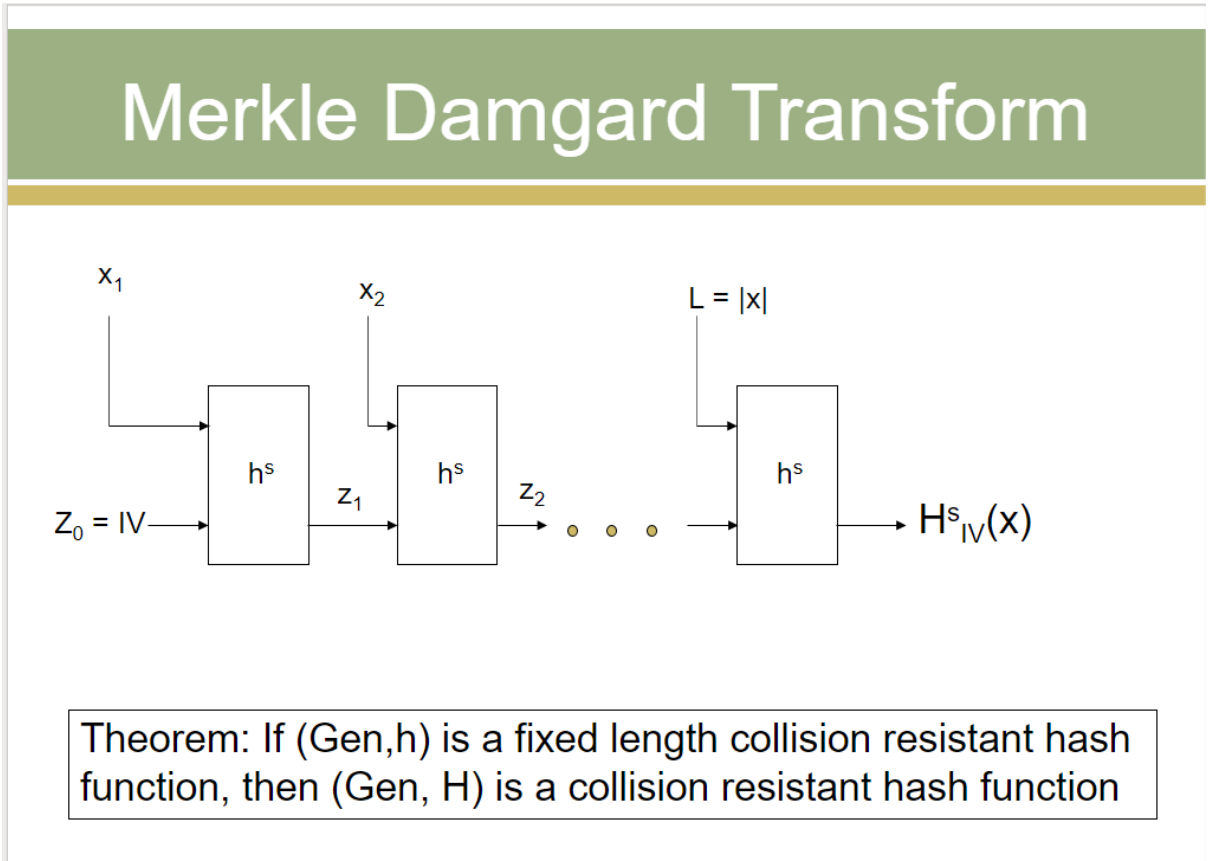
Now let's discuss the entire process how we implemented in the code.

Here in the below diagram h_s refers to our previously built DLP based hash function.

- First, we'll take inputs for prime(p), generator(g), seed for generating h through PRG. Then, it will take input of data whose max length can be 2l(n)-1, where l(n) is length of the prime in bits.
- Then, if the data length is not multiple of the length of the prime, then zero is padded at the end to make it so.
- Next, the data is divided into blocks of block-size = prime length.
- Then, an initial vector is chosen whose length = prime length and all 0's (00000..).
- Then a for loop runs d times, where d = no of data blocks.

- In first iteration, x1 = initial vector with all 0's, x2 = first block of data. These two go as input to DLP based fixed length hash function(h_s). The output goes as x1 for next iteration.
- For next iteration onwards, x1 = previous iteration hash function output, x2 = corresponding data block.
- For last block, x1 = output of the hash function in previous iteration, x2 = length of the data.

The output of the last block is the merkle damgard transformed hash data.



References

[1] J. K. a. Y. Lindell, Introduction to Modern Cryptography.