Collision Attack on the Concatenation Combiner

In this exercise, we consider Merkle-Damgård hash functions. For simplicity we do not use any padding (but all of this exercise would apply as well if we used a secure padding scheme).

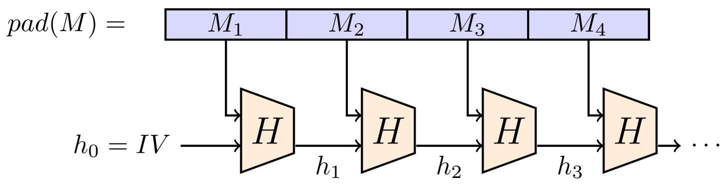

Figure 1: Merkle-Damgård construction.

The *concatenation combiner* combines the output of two Merkle-Damgård hashes applied to the same message. These two hash functions F_1, F_2 use different IVs and different compression functions f_1 and f_2. The output of the concatenation combiner is:

$$F(m_0, \ldots, m_{t-1}) = F_1(m_0, \ldots, m_{t-1}) \| F_2(m_0, \ldots, m_{t-1}) \ .$$

We suppose that the chaining values and the outputs of both functions have n bits each. Thus the combiner outputs $2n$ bits. The message blocks will have $2n$ bits, in order to facilitate our attacks.

Question 1. *Show that by using messages of length $n/2$ blocks, one can build a $2^{n/2}$-multicollision of F_1 in time $\mathcal{O}\left(n2^{n/2}\right)$, i.e., a set of messages x_i having all the same length and the same output chaining value. How much space do you need to store this multicollision?*

Question 2. *Show that we can find a collision of F in time $\mathcal{O}\left(n2^{n/2}\right)$.*

We use the functions `md_hash_1` and `md_hash_2` from the file `tp1_code.py`. Both of them take as input a list of 64-bit integers (the message blocks) and return a 32-bit integer. One uses SHA-2 and the other MD5, so they return completely unrelated outputs. Recall that you can use the function `randrange(1 << 64)` to output a random 64-bit integer.

It is possible to also access the compression function of, say, `md_hash_1`, as follows: `h' = md_hash_1([m], h)` where h is the current chaining value, h' the next one, and m the message block.

Question 3. *Implement the computation of the multicollision of* `md_hash_1`.

Question 4. *Implement the previous attack on the concatenation combiner of the two hash functions:* `md_hash_1`$\|$`md_hash_2`. *That is, find a pair of messages m_1, m_2 such that both hash functions have the same output.*

Preimage Attack on the Concatenation Combiner

Question 5. *Show that given a target t, we can find a set of $\mathcal{O}(2^n)$ preimages of t by F_1 in time $\mathcal{O}(2^n)$.*

Question 6. *Deduce that we can find preimages of the concatenation combiner in time $\mathcal{O}(2^n)$.*

We now use two different MD functions with 20-bit output, `md_hash_3` and `md_hash_4` (also defined in `tp1_code.py`).

Question 7. *Find a preimage of (0,0).*

Question 8 (Bonus question)**.** *Go back to the two first MD hashes. Append a third MD hash, this time using sha1 and 32 bits of output. Find a collision of the triple concatenation combiner. What is the number of blocks of the colliding messages?*