Security Properties of Hash Functions

Let $h : \{0,1\}^* \to \{0,1\}^n$ be a hash function that we suppose collision resistant. Let h' be the following function :

$$h' : \begin{cases} \{0,1\}^* & \to & \{0,1\}^{n+1} \\ x & \mapsto & \begin{cases} 0\|x \text{ if } |x| = n \\ 1\|h(x) \text{ otherwise} \end{cases} \end{cases}$$

Question 1. *Show that h' is collision resistant.*

Question 2. *Show that h' is not preimage resistant.*

Collisions

During this exercise, if you need to generate random bits, you can use the function `random.getrandbits` from the `random` library. If you need a random integer from $\{0, 1, \ldots, N-1\}$, you can use the function `randrange(N)`.

We use the library `hashlib` from Python's standard library, which implements many hash functions (MD5, SHA-1, SHA-2, SHA-3). These constructions take as inputs objects of type `bytes` and not `str`. In particular, you need to use the `.encode()` function or directly construct a byte string with the prefix "b". Read the documentation [1] for more information. Here is an example :

```python
import hashlib

sha2 = hashlib.sha256
print(sha2(b"Hello world").hexdigest())
print(sha2("Hello world".encode()).hexdigest())
```

Collision on Truncated Hash Function

Choose a prefix (for example your first name) and implement the generic collision search algorithm using the birthday paradox, to find two strings s_1 and s_2 both starting with this prefix, so that the 32 first bits (i.e., 8 first characters in the hex string) of SHA2(s_1) and SHA2(s_2) are equal. For example :

```
Collision found !
Input 1: maxime1585757390515751l205
Input 2: maxime1387137317230990062 6
sha2 ( input1 ) =
a4de129026e4f1b46270dc73772a14c26d90c3df19d2a040d347cc154d38c4f8
sha2 ( input2 ) =
a4de1290d4324581554e4804b53f01f95211371a4241386372502d571fc1e06c
SHA256 prefix ( first 32 bits ) : a4de1290
```

Question 3. *How many evaluations of the hash function do you need on average ?*

Question 4. *What is the memory complexity ?*

1. https://docs.python.org/3/library/hashlib.html

Collisions with Small Memory

In the following, we identify an n-bit truncated hexadecimal hash to an integer between 0 and $2^n - 1$.

```python
import hashlib
sha2 = hashlib.sha256

def sha2Trunc(x, N=32):
    fullHash = sha2(str(x).encode()).hexdigest()
    return int(fullHash[:N//4],16)

print( sha2Trunc(10) )
```

Let $H : \{0, 1 \ldots, 2^n - 1\} \to \{0, 1, \ldots, 2^n - 1\}$ be a function (for example the sha2Trunc function). Starting from a message X_0, we define the sequence $X_{i+1} := H(X_i)$. Since it takes values in a finite set, it's necessarily periodic after some point.

We denote by c the length of the pre-period $X_0, \ldots, X_{c-1}$ and ℓ the cycle length, so that $X_0, \ldots, X_{c+\ell-1}$ are all distinct.

Floyd's Cycle-finding Algorithm

The algorithm defines another sequence $Y_i = X_{2i}$, i.e., $Y_0 = X_0$ and $Y_{i+1} = H(H(Y_i))$.

Algorithm 1 Floyd's cycle-finding algorithm.

1: $x \leftarrow H(X_0)$
2: $y \leftarrow H(H(X_0))$
3: **while** $x \neq y$ **do**
4: $\quad x \leftarrow H(x)$
5: $\quad y \leftarrow H(H(y))$
6: **end while**

Question 5. *Implement Floyd's algorithm to find an element $x = X_i$ in the cycle (start with a uniformly random value for X_0).*

Question 6. *Write an algorithm to find the length ℓ of the cycle that contains x.*

Question 7. *Implement the computation of collisions : if $c > 0$ and $\ell > 1$ one has :*

$$H(X_{c-1}) = X_c = X_{c+\ell} = H(X_{c+\ell-1})$$

and by definition $X_{c-1} \neq X_{c+\ell-1}$. For this, start from X_0 and X_ℓ and compute X_i and $X_{i+\ell}$ until equality.

Question 8. *Deduce examples of collisions on the n first bits of SHA256 for $n = 16, 32, 40, 48$.*