# Tutorial 4

Gidon Rosalki

2025-12-03

**Notice:** If you find any mistakes, please open an issue at `https://github.com/robomarvin1501/notes_intro_to_crypto`

## 1 Question 1

Let there be a family of collision resistant hash functions $\{H_s\}_{s\in\{0,1\}^*}$, such that $H_{s'} : \{0,1\}^{2n} \to \{9,1\}^n$. Build a new family of CRHFs $\{H'_s\}_{s\in\{0,1\}^*}$ such that $H'_s : \{0,1\}^{ln\to\{0,1\}^n}$, such that $l \geq 3$.

A hash function is collision resistant if

$$\forall PPT\ \mathcal{A}\ \Pr\left[A\left(H_s\right) \to x_1, x_2\ :\ H_s\left(x_1\right) = H_s\left(x_2\right)\right] \leq neg\left(n\right)$$

### 1.1 Solution

To achieve this we will split the input into inputs of size $2n$, such that

$$H'_s\left(x\right) = H'_s\left(x_1 x_2 \dots x_l\right)$$

We may now input $x_1 \| x_2$ into $H_s$: $H_s\left(x_1 \| x_2\right) = y_1$, we now compute $H_s\left(y_1 \| x_3\right) = y_2$, and so on. The output will be the resultant hash.

Why is this CR? Let us begin with $l = 3$, which will extend to arbitrary length $l$. We will assume towards contradiction that there exists a collision, so therefore there exists an adversary $\mathcal{A}$ which can output $\begin{bmatrix} x \\ x' \end{bmatrix}$ where $H'_s\left(x\right) = H'_s\left(x'\right)$. We will use this to construct an algorithm to find the collision. Let there be the algorithm $\mathcal{B}$ as follows:

1. Run $\mathcal{A}$, which returns $\begin{bmatrix} x \\ x' \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & x_3 \\ x'_1 & x'_2 & x'_3 \end{bmatrix}$
2. If $H_s\left(x_1 \| x_2\right) = H_s\left(x_1 \| x_2\right)$, then it returns that pair, and we are done. If not, then it must hold that

$$H_s\left(H_s\left(x_1 \| x_2\right) \| x_3\right) = H_s\left(H_s\left(x'_1 \| x'_2\right) \| x'_3\right)$$

### 1.2 Solution II

An alternative is to instead build a tree, where we pad the input into the nearest power of 2, split it into blocks of size $n$, and then run $H_s$ on each pair of blocks. We then continue doing this recursively until we reach a single hash of length $n$. This may be proven similarly as to the previous solution.

How does this compare to the previous? The first solution only needs $O\left(1\right)$ of memory, where the second requires $O\left(\log\left(n\right)\right)$ of memory. However, a benefit of the second method is that we can split it trivially across many processing cores, where for the first solution each step is dependent on the previous, and so it cannot be split so easily.
An additional benefit of the first solution is that it is incredibly easy to implement, whereas the second is a bit more complicated. However, an additional benefit of this second method is as follows. If we consider this has to be a hash of your entire hard disk, then when we change a block, we do not need to recompute every hash further up the chain from this block, but rather only the hashes in the tree that are impacted by this singular block.

## 2 Question 2

Let there be a PRF $F_k : \{0,1\}^{2n} \to \{0,1\}^{2n}$. Let $\Pi = (KeyGen, MAC, Vrfy)$.

- $KeyGen\left(1^n\right) \to k$ for PRF
- $MAC\left(k, m\right) \to F_{F_k\left(0^{2n}\right)}\left(m \| 0^n\right) \| F_k\left(m \| 1^n\right) = t$
- $Vrfy\left(k, m, t\right) = 1$ **if and only if** $MAC\left(k, m\right) = t$

Is this a secure MAC algorithm?

## 2.1 Solution

It is so. Let us assume towards contradiction that there exists $\mathcal{A}$ that can win the MacForge game against $\Pi$. We want to build $\mathcal{B}$ that can win the PRF game. So, $\mathcal{B}$ has an oracle $\mathcal{B}^{\mathcal{O}}$. Remember, the MacForge game is that $\mathcal{A}$ has oracle access to MAC, and it spits out $(m^*, t^*)$ such that it has not asked the Mac of $m^*$, and its hash is $\sqcup^*$.

Let us define the new function $F_{\mathcal{O}(0^{2n})}(m \| 0^n) \| \mathcal{O}(m || 1^n)$.

1. $\mathcal{A}$ asks questions, and we use this function to ask them
2. Return $Vrfy(m^*, t^*)$

**Case I**: If $\mathcal{O}$ is random, then there is no way that $\mathcal{A}$ can guess the output, which is at least partially dependent on $\mathcal{O}$, so therefore $\Pr[Vrfy = 1] \leq \frac{1}{2^n}$

**Case II**: Here $\mathcal{O}$ is a PRF, then $\mathcal{A}$ is playing the regular MAC game, and so can guess the output, and we will return that this is a PRF.

## 2.2 Extension

Given a family of PRF functions $F_k : \{0,1\}^{2n} \to \{0,1\}^n$. Is $F_k$ CRH?

No it is not. We will build a family $F'_{k,a,b}$ such that it is still pseudorandom, but since we know $k$, we can show that it is not a CRH.

$$F_{k,a,b} = \begin{cases} 0, & \text{if } x = a \\ 0, & \text{if } x = b \\ F_k(x), & \text{else} \end{cases}$$

We will theorise firstly that this family is pseudorandom. Since it is highly unlikely that our adversary will find $a$ or $b$, we may state that $F'$ is a PRF.

Secondly, we will theorise that $F'$ is not collision resistant. This follows obviously. PRFs are only PRFs if we do not have access to the key in $F_k$. Since for CRFs we are given the key, we may trivially bring $a, b$ which are a collision, and so disprove the fact that $F'$ is collision resistant.