Constructing CRHF's:

Many cryptographic hash functions (e.g., MD5, SHA-1, SHA-256) follow the Merkle-Damgard paradigm: start from hash function on short messages and use it to build a collision-resistant hash function on a long message:

1. Split message into blocks
2. Iteratively apply compression function (hash function on short inputs) to message blocks

Suppose \( h \) is a compression function, and \( t_0, t_1, \ldots, t_n \) are chaining variables.

Hash functions are deterministic, so IV is a fixed string (defined in the specification) — can be taken to be all-zeros string, but usually set to a custom value in constructions.

For SHA-256:
\[
x = 0 \cdot 1^{32} = y
\]

Theorem: Suppose \( h : X \times Y \rightarrow X \) is a compression function. Let \( H : \{0, 1\}^* \rightarrow X \) be the Merkle-Damgard hash function constructed from \( h \). Then, if \( h \) is collision-resistant, \( H \) is also collision resistant.

Proof: Suppose we have a collision-finding algorithm \( A \) for \( h \). We use \( A \) to build a collision-finding algorithm for \( H \):

1. Run \( A \) to obtain a collision \( M \) and \( M' \) (\( H(M) = H(M') \).)
2. Let \( M = m_1, m_2, \ldots, m_n \) and \( M' = m'_1, m'_2, \ldots, m'_n \) be the blocks of \( M \) and \( M' \), respectively. Let \( t_0, t_1, \ldots, t_n \) and \( t'_0, t'_1, \ldots, t'_n \) be the corresponding chaining variables.
3. Since \( H(M) = H(M') \), it must be the case that
   \[
   H(M) = h(t_{n-1}, m_n) = h(t'_{n-1}, m'_n) = H(M')
   \]
   If either \( t_{n-1} \neq t'_{n-1} \) or \( m_n \neq m'_n \), then we have a collision for \( h \).

Otherwise, \( m_n = m'_n \) and \( t_{n-1} = t'_{n-1} \). Since \( m_n \) and \( m'_n \) include an encoding of the length of \( M \) and \( M' \), it must be the case that \( n = n' \). Now, consider the second-to-last block in the construction (with output \( t_{n-2} = t'_{n-2} \)):
   \[
   t_{n-2} = h(t_{n-3}, m_{n-1}) = h(t'_{n-3}, m'_{n-1}) = t'_{n-2}
   \]

Either we have a collision or \( t_{n-2} = t'_{n-2} \) and \( m_{n-1} = m'_{n-1} \). Repeat down the chain until we have collision or we have concluded that \( m_i = m'_i \) for all \( i \), and so \( M = M' \), which is a contradiction.

Note: Above construction is sequential. Easy to adapt construction (using a tree) to obtain a parallelizable construction.
Sufficient now to construct a compression function.

Typical approach is to use a block cipher.

Divides-Meyer: Let \( F : \{0,1\}^n \times \{0,1\}^* \rightarrow \{0,1\}^* \) be a block cipher. The Davies-Meyer compression function \( h : \{0,1\}^n \times \{0,1\}^* \rightarrow \{0,1\}^* \) is then

\[
h(k, x) = F(k, x) \oplus x
\]

Many other variants also possible: \( h(k, x) = F(k, x) \oplus k \oplus x \) [used in Whirlpool hash family]

Need to be careful with design:
- \( h(k, x) = F(k, x) \) is not collision-resistant: \( h(k, x) = h(k, F^{-1}(k', F(k, x))) \)
- \( h(k, x) = F(k, x) \oplus k \) is not collision-resistant: \( h(k, x) = h(k', F^{-1}(k', F(k, x) \oplus k \oplus k')) \)

Theorem: If we modulate \( F \) as an ideal block cipher (i.e., a truly random permutation for every choice of key), then Davies-Meyer is collision-resistant.

Conclusion: Block cipher + Davies-Meyer + Merkle-Damgard \( \Rightarrow \) CRHFs

Examples: SHA-1: SHACAL-1 block cipher with Davies-Meyer + Merkle-Damgard
SHA-256: SHACAL-2 block cipher with Davies-Meyer + Merkle-Damgard

Why not use AES?
- Block size too small! AES outputs are 128-bits, not 256 bits (as birthday attack finds collision in \( 2^{64} \) time).
- Short keys means small number of message bits processed per iteration.
- Typically, block cipher designed to be fast when using same key to encrypt many messages.
  \( \Rightarrow \) In Merkle-Damgard, different keys are used, so alternate design preferred. (AES key schedule is expensive)

Recently: SHA-3 family of hash functions standardized (2015)
  \( \Rightarrow \) Relies on different underlying structure ("sponge" function)
  \( \Rightarrow \) Both SHA-2 and SHA-3 are believed to be secure (most systems use SHA-2 — typically much faster)

\( \rightarrow \) or even better, a large-domain PRF

Back to building a secure MAC from a CRHF — can we do it more directly than using CRHF + small-domain MAC?

Main difficulty seems to be that CRHFs are keyless but MACs are keyed

Idea: include the key as part of the hashed input

By itself, collision-resistance does not provide any "randomness" guarantees on the output.

- For instance, if \( H \) is collision-resistant, then \( H'(m) = m \| 1 \| m \| 1 \| H(m) \) is also collision-resistant even though \( H' \) also leaks the first 10 bits/blocks of \( m \).
- Constructing a PRF/MAC from a hash function will require more than just collision resistance.
  
  - Option 1: Model hash function as an "ideal hash function" that behaves like a fixed truly random function (modeling heuristic called the random oracle model — will encounter later in this course)
  
  - Option 2: Start with a concrete construction of a CRHF (e.g., Merkle-Damgard or the sponge construction)
  
  and reason about its properties.

\( \Rightarrow \) We will take this approach
How long does the output of a CRHF have to be?

Birthday attack on CRHFs. Suppose we have a hash function $H : \{0,1\}^* \rightarrow \{0,1\}^k$. How might we find a collision in $H$ (without knowing anything more about $H$)?

1. Approach 1: Compute $H(1), H(2), \ldots, H(2^k-1)$. By Pigeonhole Principle, there must be at least one collision — runs in time $O(2^k)$

2. Approach 2: Sample $m_i \in \{0,1\}^*$ and compute $H(m_i)$. Repeat until collision is found. How many samples needed to find a collision?

Theorem (Birthday Paradox). Take any set $S$ where $|S| = n$. Suppose $r_1, \ldots, r_n \in S$. Then,

$$\Pr[\exists i \neq j : r_i = r_j] \geq 1 - e^{-\frac{n(n-1)}{2n}}$$

Proof. $\Pr[\exists i : r_i = r_j] = 1 - \Pr[\forall i : r_i \neq r_j] = 1 - \prod_{i=1}^{n-1} \frac{n-2i}{n}$

$$\geq 1 - \left(1 - \frac{1}{n}\right)^{n-1} = 1 - e^{-\frac{n(n-1)}{2n}}$$

$\Pr[\text{collision}] = \frac{1}{2}$. [For birthdays, $\sqrt{\frac{1}{2} \times 256} \approx 23$]

For hash functions with range $\{0,1\}^k$, we can use a birthday attack to find collisions in time $\sqrt{2^k} = 2^{k/2}$. Can even do it with constant space!

For 128-bit security (e.g., SHA-256), we need the output to be 256-bits (hence SHA-256)

Quantum collision-finding can be done in $\sqrt[4]{2}$ (cube root attack), though requires more space (via Floyd's cycle finding algorithm).