COMP 8920: Cryptography Winter 2025

Lecture 14 — Feb 27, 2025

Prof. Curtis Bright *Scribe: Aidan Bennett*

Overview

In the last lecture we introduced the Merkle-Damgård construction.

In this lecture we finish our discussion of the Merkle-Damgård construction, and introduce the sponge construction.

Merkle-Damgård construction (continued)

Suppose $c : \mathbb{Z}_2^{m+1} \to \mathbb{Z}_2^m$ is a compression function and

$$f(x_i) = \begin{cases} 0 & \text{if } x_i = 0 \\ 01 & \text{if } x_i = 1 \end{cases}$$

and $y(x_1 \dots x_n) = 11||f(x_1)|| \dots ||f(x_n) = y_1 \dots y_k$, where $y_i \in \mathbb{Z}_2$.

$$\begin{aligned} g_1 &= c(0^m||y) \in \mathbb{Z}_2^m \\ g_2 &= c(g_1||y_2) \\ &\vdots \\ g_k &= c(g_{k-1}||y_k) = h(x) \end{aligned}$$

Theorem 1. *If $c : \mathbb{Z}_2^{m+1} \to \mathbb{Z}_2^m$ is collision resistant, then $h : \bigcup_{i=m+1}^{\infty} \mathbb{Z}_2^i \to \mathbb{Z}_2^m$ will also be.*

Proof. (by contraposition)

Suppose we have a collision (x, x') of h. Let $y(x) = y_1 \dots y_k$ and $y(x') = y'_1 \dots y'_\ell$. If $k = \ell$, following the same strategy as before, we obtain a collision for c or obtain that $y = y'$. But $y = y'$ implies that $x = x'$ (since y injective), which would be a contradiction. Thus, we suppose $\ell \neq k$ and WLOG that $\ell > k$. We have $g_k = g'_\ell$ (because it's a collision), so $c(g_{k-1}||y_k) = c(g'_{\ell-1}||y'_\ell)$ which is either a collision for c or $g_{k-1} = g'_{\ell-1}$ and $y_k = y'_\ell$.

Assuming we don't find collisions for c in this way, we eventually achieve

$$y_k = y'_\ell,\ y_{k-1} = y'_{\ell-1},\ \dots,\ y_1 = y'_{\ell-k+1}$$

which implies that $y_1 y_2 \dots y_k$ is a suffix of $y'_1 \dots y'_\ell$, which is not possible due to the construction of y. □

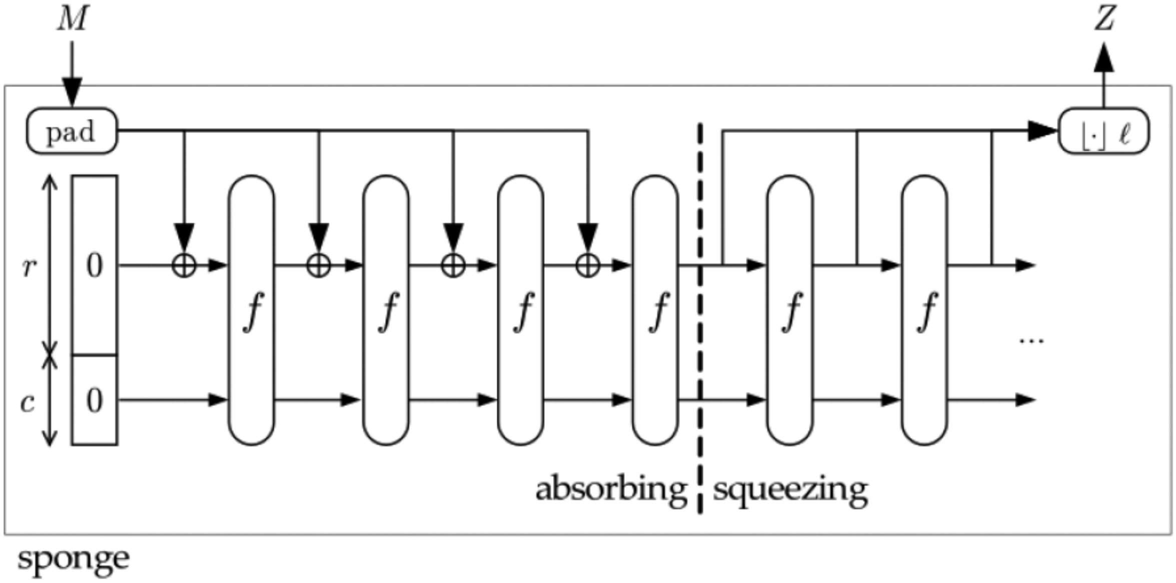

Figure 1: Sponge function circuit diagram (source)

Many common hash functions follow the Merkle-Damgård approach, including MD4, MD5, SHA-0 (all broken), and SHA-1, SHA-2. The first collision for SHA-1 was found in 2017 with about $2^{80}/10^5$ trials, 10^5 times faster than brute force, as SHA-1 uses a 160-bit digest. The padding scheme of SHA-1 extends the input by at most one extra 512-bit block, and uses a compression function $c : \mathbb{Z}_2^{160+512} \to \mathbb{Z}_2^{512}$. So the message is split into 512-bit blocks. The function c uses bitwise operations and integer mod 2^{32}.

The sponge construction

The sponge construction is another way of constructing a hash function and is used in SHA-3. Instead of a compression function, the sponge construction is built from a function $f : \mathbb{Z}_2^b \to \mathbb{Z}_2^b$ that is typically a bijection. The parameter b is called the **width**, and $b = r + c$ where r is the **bitrate** and c is the **capacity**. The larger c is, the more secure the hash function will be against birthday attacks. The message is processed r bits at a time, so the smaller r is, the less efficient the function will be.

The sponge function f uses the circuit as depicted in Figure 1.

Note the advantage of this construction is that it can produce digests of arbitrary size, just by increasing the length of the squeezing phase. However, note that the security of the function will be dependent on c, so increasing the message digest length can't be said to make the function more secure in general.

We will now show how to find a collision of this hash function in an expected $2^{\frac{c}{2}}$ evaluations. The collision is known as an **internal collision** since it will be a collision in the b-bit state of the sponge function, rather than just a collision of its output. Suppose $x_0 = \underbrace{0\ldots0}_{r} = 0^r$ and $y_0 = \underbrace{0\ldots0}_{c} = 0^c$

and we perform the following iteration:

$$\begin{aligned} f(x_0||y_0) &= x_1||y_1 \\ f(x_1||y_1) &= x_2||y_2 \\ &\vdots \\ f(x_0||y_{k-1}) &= x_k||y_k \end{aligned}$$

where $|x_i| = r$, and $|y_i| = c$. We stop wehn we find a repeated y-value, say $y_k = y_h$ where $h < k$. This is just a birthday attack on f where inputs always start with 0^r. We expect a collision after $2^{\frac{c}{2}}$ iterations. After this, we have a pair (y_k, y_h) with $f(0^r||y_k) = f(0^r||y_h)$.

Now consider the messages $m = x_0|| \dots ||x_h$ and $m' = x_0|| \dots ||x_k$. To find the digest for m, note

$$\begin{aligned} &\text{State 1: } f(x_0 \oplus x_0||y_0) = f(0^r||y_0) = x_1||y_1 \\ &\text{State 2: } f(x_1 \oplus x_1||y_1) = f(0^r||y_1) = x_2||y_2 \\ &\text{State 3: } f(x_2 \oplus x_2||y_2) = f(0^r||y_2) = x_3||y_3 \end{aligned}$$

But, similarly for m', the k^{th} state is $x_k||y_k$, so the output of the absorbing phase here
is $x_{k+1}||y_{k+1} = f(x_k \oplus x_k||y_k) = f(0^r||y_k)$.