COMP 8920: Cryptography	Winter 2025

Lecture 15 — March 4, 2025

Prof. Curtis Bright — *Scribe: Rajat Yadav*

The output of the absorbing phase is $x_{K+1} \oplus x_k \parallel y_{k+1} = f(0^r \parallel y_k)$ and since $y_h = y_k$, this is

$$= f(0^r \parallel y_n) \tag{1}$$
$$= x_{h+1} = y_{h+1} \tag{2}$$

Thus, the output of the absorbing phase for both m and m' is the same, even though $m \neq m'$, so (m, m') is a collision. If l is particularly small, we also have the option of a birthday attak in the hash function directly, which uses $\approx \sqrt{2^l}$ evaluations to find a collision. SHA-3 512 uses the sponge construction with $b = 1600$, $r = 576$, $c = 1024$ so has collision security of 256 and preimage security of 512 bits. That is we expect 2^{256} ops to find a collision and 2^{512} ops to find a preimage of any hash. The SHA3 family supports "extendable output" for which the output length is controllable, e.g. in the hash function SHAKE256.

Message Authentication Codes (MACs)

MACs ensure integrity of a message - Alice sends a message to Bob and appends a tag, depending on a private key shared by Alice & Bob. If the message was corrupted, that would be detected, as the tag would no longer match. A common way of constructing, a MAC is to use an unkeyed hash function and incorporate a secret key as a part of the message to be hashed. However, this must be done carefully we'll sho whow to how to break some simple approaches of this.

Suppose we make a keyed hash function h_k by taking an interated hash function h and setting IV $= K$, keeping k secret. Say x has a length that is a a multiple of t and $c : \mathbb{Z}_2^{m+t} \to \mathbb{Z}_2^m$ is the compression function to build h. We also assume K has m bits. We'll show if an adversary has a single message x and a valid tag $t = h_{k(x)}$, they will be able to generate a valid tags for other messages. Let x' be any bitstring of length t, and consider $m = x \parallel x'$. Its tag is $h_k(x \parallel x') = c(h_{k(x)} \parallel x')$ so $h_k(m) = c(t \parallel x')$, and Eve knows c, t, x' Thus, (m, t) is a valid pair that is a forgery of Eves done without knowing the key k. Even if $|x|$ is not divisible by t, a similar attack can be used. Say $y = x \parallel \text{pad}(x)$ is the padding of x in the preprocessing step. Here $|y| = rt$ for some integer r. Take w any bitstring of length t, and define $x' = x \parallel \text{pad}(x) \parallel w$. The preprocessing step for this x' would give

$$y' = x' \parallel \text{pad}(x') = x \parallel \text{pad}(x) \parallel w \parallel \text{pad}(x') \tag{3}$$
$$= y_1 y_2 \ldots y_{r'} \text{ where } |y_i| = t \text{ and } r' < r. \tag{4}$$

Note $h_{k(x)} = z_r$ where

$$z_r = c(z_{r-1} \parallel y_r) \tag{5}$$
$$z_{r-1} = c(z_{r-2} \parallel y_{r-1}) \tag{6}$$
$$\vdots \tag{7}$$
$$z_0 = \text{IV} = K \tag{8}$$

. Then Eve can compute

$$z_{r+1} = c\left(h_{k(x)} \parallel y_{r+1}\right) \tag{9}$$
$$z_{r+2} = c\left(z_{r+1} \parallel y_{r+2}\right) \tag{10}$$
$$\vdots \tag{11}$$
$$z_{r'} = c\left(z_{r'-1} \parallel y_{r'}\right) = h_{k(x')} \tag{12}$$

So, $(x', z_{r'})$ is a forgery from Eve, even through K is unknown and there is no assumption on the padding length. So, in general we want to avoid letting an advesary compute a message-tag pair (x, y) for unknown key given prior valid pairs

$$(x, y) \tag{13}$$
$$\ldots, \tag{14}$$
$$(x_Q, y_Q) \tag{15}$$

using the same key. The pairs may be pairs observed by Eve, in which casse it is a know message attack or in a chosen message attack Eve has access to a "tag oracle" and can generate tags for messages $x_1, \ldots, x_Q$ that Eve chooses, Here $x \neq x_1, \ldots, x_Q$ and (x, y) is said to be a forgery. If the probability of a forgery is at least ε the advsary is said to be a (ε, Q)-forges. (In known message attack, the attack should work woth probability ε for regardless of the message seen). The attacks we just saw are therefore $(1, 1)$-forgery attacks. Another obvious attack is to choose a key $k \in \mathbb{K}$ at random and output $(x, h_k(x))$ for arbitary x. This would be a $\left(\frac{1}{|\mathbb{K}|}, 0\right)$-forgery attack.

One standarized MAC is algorithm called HMAC constructs a MAC from a unkeyed hash function like SHA-1. The version we'll describe uses a 512 bit key k and 512-bit constants ipad $= 36 \ldots 36$ and opad $= 5C \ldots 5C$ (written in hex). If x is the message to be authenticated, the 160 bit MAC is computed as

$$\text{HMAC}_{k(x)} = \text{SHA-1}((k \oplus \text{opad}) \parallel \text{SHA-1}((k \oplus \text{ipad}) \parallel x)) \tag{16}$$

One can a fixed-size message and a collision-resistant hash function. It is also very efficient, as it uses only one call to SHA-1 on a long message (the "outer") SHA-1 takes constant time as it is on messages of length 512 + 160.