COMP 8920: Cryptography Winter 2025

Lecture 16 — March 6, 2025

Prof. Curtis Bright *Scribe: Rajat Yadav*

Last class, we saw how to generate a MAC using a hash function like SHA-1. Another way of generating MAC is to prepend the key to a message and use a hash function based on the sponge construction like SHA-3, which is not susceptible to a length extension attack, as knowledge of $h_{k(x)}$ doesn't help you compute $h_{k(x\|x')}$ unlike in the merkel-damgard construction.

Another popular way of constructing a MAC uses a block cipher in CBC mode with a fixed IV. Recall in CBC mode each ciphertext block y_i is XORed with the next plaintext before encyption, so far $x = x_1, \ldots, x_n$. We have

$$y_0 = \text{IV} \tag{1}$$
$$y_1 = e_k(y_0 \oplus x_1) \tag{2}$$
$$\vdots \tag{3}$$
$$y_n = e_k(y_{n-1} \oplus x_n) \tag{4}$$

CBC-MAC(x, k) are discarded. The best known attack in CBC-MAC is a birthday chosen message attack. Eve requests the tags of a large number of messages and if a duplicate is ever found, Eve needs only one more request in order to forge a tag.

Suppose the block length is t and let $x_3, x_4, \ldots, x_n \in \mathbb{Z}_z^t$ be fixed. Eve choses Q distinct elements of $\mathbb{Z}_2^t$ (where $Q \approx 1.17\sqrt{2^x}$) and call them $x_1^1, x_2^2, \ldots, x_Q^1$; these will form the first block of the messages she'll construct. Also let $x_1^2, \ldots, x_2^2$ be chosen randomly in $\mathbb{Z}_2^t$ (these will be the 2^{nd} blocks).

Now define

$$x^i = x_1^i \parallel x_2^i \parallel x_3 \parallel x_4 \parallel \ldots \parallel x_n \tag{5}$$

for $1 \leq i \leq Q$. Eve requests tags for all x^i, and with 50% chance 2 have the same tag.

In the process of finding the tags for x_i, the oracle will find the values $y_0^i, \ldots, y_{n(i)}$ and output y_n^i as the tag. Suppose 2 tags match i.e. $y_n^i = y_n^j$ for $1 \leq i < j \leq Q$. Because the last n-2 blocks of the x_s^i are identical, this implies that

$$y_2^i = y_2^j, \wedge \tag{6}$$
$$y_2^i = e_k(y_i^i \oplus x_2^i) \wedge y_2^j = e_k(y_j^j \oplus x_2^j) \tag{7}$$

so applying Q_k to both sides, we get

$$y_1^i \oplus x_2^i = y_1^j \oplus x_2^j \tag{8}$$

. Since $x_2^i \wedge x_2^j$ were chosen randomly, with $\frac{1}{2}$ probability this will happen. Now let $\delta \in \mathbb{Z}_2^t$ be a nonzero bitstring. XOR δ with second block of $x^i \wedge x^j$ to get

$$u = x_1^i \parallel (x_2^i \oplus \delta) \parallel x_3 \parallel \ldots \parallel x_n \tag{9}$$
$$x = x_1^j \parallel x_2^j \oplus \delta \parallel x_3 \parallel \ldots \parallel x_n \tag{10}$$

Eve requests the tag for u, but then can use it as forgery for the tag of v, since u and v have the same tag. Note $u \neq v$ since $x_1^i \neq x_1^j$. This attack is a $\left(\frac{1}{2}, O\left(2^{\frac{t}{2}}\right)\right)$ - forger. So for, we've only used a MAC to provide data integrity, but it is often used with encryption, achieving secrecy as well. This is called **authenticated encryption**, and there are 3 obvious ways to combine a MAC with encryption. (We'll use seprate keys for the MAC and encryption.)

1. MAC-and-encrypt

   For message x, compute $z = h_{k_1}(x)$ and $y = e_{k_2}(x)$ and send (y, z). Bob decrypt y and checks z is a valid tag for y's encryption.

2. MAC-then-encrypt

   Still $z = h_{k_1}(x)$ but the plaintext to encrypt incorportates the z: $y = e_{k_2}(x \parallel z)$, and only y is sent. Bob decrypts y to get $x \parallel z$ and checks z is the tag of x.

3. Encrypt-then-MAC

   Now first compute $y = e_{k_2}(x)$ and then the tag is $z = h_{k_1}(y)$, and send (y, z). Bob checks z is the tag of y, and if so, will decrypt y.

Method 3 is usually the best it can be shown if the MAC and encryption are individually secure, then the method is also secure, but this is not always true is methods 1 and 2. Also, onlly in method 3 can decryption be skipped if the tag is invalid. The CCM (counter with CBC-MAC) mode is a NIST standard providing authenticated encryption. It combines CTR mode a tag computed by CBC-MAC.

Suppose $x = x_1, \ldots, x_n$ is the plaintext with each $x_i \in \mathbb{Z}_2^m$. As in CTR mode, a initial value ctr is chosen by Alice and sent to Bob in plaintext. It is important that ctr is never repeated with same key, otherwise Eve can learn the XOR of your messages encrypted with same key and ctr.

Starting from ctr, we construct $T_0, \ldots, T_n$ with $T_i = (\text{ctr} + i) \mod 2^m$. The ciphertext blocks are encrypted with

$$y_i = x_i \oplus e_k(Y_i). \tag{11}$$

Then compute the tag temp = CBC-MAC(x, k) and $y' = \text{temp} \oplus e_k(T_0)$ The encrypted tag is appended to the ciphertext, $y = y_1 \parallel \ldots \parallel y_n \parallel y'$. Bob computes $e_{k(T_0)}, \ldots, e_{k(T_n)}$ and then finds $x_i = y_i \oplus r_{k(T_1)} \ldots x_n = y_n \oplus e_k(T_n)$, obtaining x. Then Bob computes CBC-MAC(x, k) and XORs it with $e_{k(T_0)}$ to check the tag is valid.

We now discuss public-key cryptography and the RSA cryptosystem, the first example of a cryptosystem where the encryption and decryption keys are different, with the encryption key publically known. Thus, anyone can securely send a message to anyone for which they have the public key. When Alice wants to send a message to Bob, it is essential she has his actual public key and not an attacker's. In practice, public keys are digitally signed using certificates to verfy their authenticity.