CS558: Scribe Notes

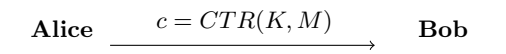
February 2, 2017

Short history on MACs

Macs have been used throughout history. An example of this was in 8000 BCE, with the concept of seals to authenticate transactions. In addition, things like signatures also can be seen as MACs.

Attempt at construction of a MAC

Let's start by trying to construct a MAC algorithm. We begin by taking a block cipher operated in counter mode.



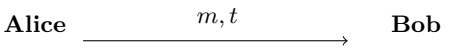
As we attempt to construct a MAC, we start with this simple scheme. For some message M, and key K, we have Alice send the ciphertext $c = CTR(K, M)$ in order to authenticate. Now, is this a good scheme? The answer is no, since the scheme is malleable. We can easily show this by flipping any bit in the cipher text. The result of this is flipping the corresponding bit in the message.

Now, for a scheme we have two properties we want to achieve: confidentiality, and authentication. The above scheme is confidential, since our adversary has no idea what the message is exactly, however it fails at authentication, since the adversary can easily change what the message is.

For a MAC, there are also two properties we want to achieve. The first, obviously is authentication. We want to guarantee that the message is actually from Alice. The second is integrity: we want to make sure that the message has not been modified in some way. We note that authentication implies integrity, however it may not be the case that integrity implies authentication.

So in order to achieve our two guarantees of authentication and confidentiality, we have Alice send over m, the message (in the showing the construction of a

mac, we don't really need confidentiality, we want to show authentication! Of course, in a real applications we would want to encrypt our message). So now we have



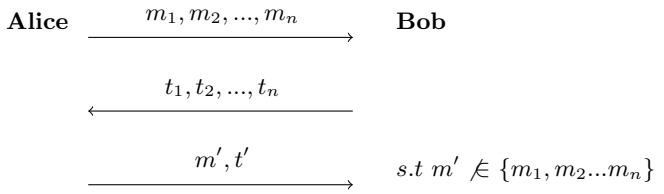
where Alice generates some tag $t = MAC(m, k)$, and Bob can verify with $Ver(k, t, m) = 1$

Now, for one construction of a MAC, we have $MAC(k, m) = PRF_k(m) = t$, meaning we simply generate the tag by throwing our message through a pseudo random function. This is secure since we have a pseudo random function, an adversary will not know what the output is. Now, what about using MD5? Shown in lab, this does not work since MD5 is not pseudorandom. We can easily conduct length extension attacks on it.

Formal definition of a MAC

Formally, a MAC needs to satisfy both correctness, and security/unforgeability. We define correctness as $Ver_k(m, MAC_k(m)) = 1$. In other words, it works. The idea is that if we verify message m, and its tag $MAC_k(m)$, our Ver algorithm will accept these inputs since they are both correct.

For security, since this is cryptography, we say the MAC is secure if the adversary wins the following game with probability $1/2^n$, where n is the length of the tag



Note: we want to make sure that the adversary does not pick a message he has already queried

HMACs

So, how do we take a hash function which is vulnerable to length extension attacks, and make it viable for a MAC? We use the HMAC construction.

$$H(k \oplus opad || H(k \oplus ipad || m))$$

where k is the key H is the hash function, m is the message, and $oplus, iplus$ are public constants.
Note: also commonly used is

$$H(k||H(k||m))$$

, however there is no provable security on this construction.