University of Maryland
CMSC456 — Introduction to Cryptography
Professor Jonathan Katz

Lecture 22

1 Message Authentication Codes used in Practice

Recall from our previous discussion on MACs that it is "easy" to construct a MAC for *short* messages using a PRF. Constructing a MAC for *longer* messages is more difficult. Last time, we gave one construction — the XOR-MAC — that was secure for arbitrarily-long messages. We review this construction, discuss its security, and then give some other MACs that are widely used in practice.

XOR-MAC. We describe this scheme in more generality than we did last time. Let $F : \{0,1\}^k \times \{0,1\}^m \rightarrow \{0,1\}^n$ be a (t, ϵ)-PRF; the sender and receiver will share a random key $s \in \{0,1\}^k$. Fix some parameter $\ell < m$ (we see below where this parameter comes into play). Let the notation $\langle i \rangle$ denote the $(\ell - 1)$-bit representation of integer i in binary. To authenticate message M, parse M as a sequence of blocks $M_1, \ldots, M_t$ each $(m - \ell)$-bits long. Choose a random value $r \in \{0,1\}^{m-1}$ and compute:

$$\mathsf{tag} = F_s(0 \circ r) \oplus F_s(1 \circ \langle 1 \rangle \circ M_1) \oplus F_s(1 \circ \langle 2 \rangle \circ M_2) \oplus \cdots \oplus F_s(1 \circ \langle t \rangle \circ M_t).$$

The complete tag is $\langle r, \mathsf{tag} \rangle$ (the receiver needs r in order to verify).

We may note a few interesting points about this scheme. First, it is randomized; we saw last time how the deterministic version of this scheme is *not* secure. Second, the message M is assumed to have length which is a multiple of $(m - \ell)$. This restriction is not really that severe, since there are secure[1] ways to pad a message so that its length becomes a multiple of $(m - \ell)$; however, such padding may lead to slight loss of efficiency (since more computations of F are required to MAC a longer message). Finally, the maximum message-length supported by this scheme is $(m - \ell) \cdot (2^{\ell-1} - 1)$ bits (since the counter $\langle i \rangle$ included with each block should not "cycle").

We did not mention last time the exact security result for this scheme, so we do so here.

Theorem 1 *For any adversary attacking the XOR-MAC scheme running in time (roughly) t and requesting at most q tags from its MAC oracle, the probability of successfully forging a new, valid message/tag pair is at most $2q^2 \cdot 2^{-m} + 2^{-n} + \epsilon$.*

Roughly speaking, the first term in the above bound comes from the probability of a "collision" in the random value r (recall the "birthday problem" from previous lectures and the notes on probability); the second term comes from the fact that tag is n bits long, and the

[1] Note that padding with, say, all zeros is *not* secure, for the following reason: say $m - \ell = 64$. Then the MAC of $M = 1$ and $M' = 10$ would be identical (since they are both padded out to 10^{63}), and then the receiver cannot unambiguously tell which message was intended.

adversary can always guess a correct tag with probability 2^{-n}; and the final term comes from the security of the PRF.

We do not give a proof here. For more detail about the scheme and a full proof of security, see [1].

CBC-MAC. A widely-used MAC is based on the CBC mode of encryption that we discussed previously. However, note that this connection is entirely fortuitous — there is not reason, in general, to assume that a good mode of encryption will give rise to a secure MAC (and vice versa). In fact, the CBC-MAC differs slightly (and has different security properties) from the CBC mode of encryption.

Assume $F : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ is a (t,ϵ)-PRF (note that the input and output lengths are now assumed to be the same, for convenience only). We may define CBC-MAC as follows: the sender and receiver share a random key $s \in \{0,1\}^k$. Let m be some fixed parameter; the authentication scheme will *only* be defined for messages of length $n \cdot m$ (i.e., m blocks, each n bits long). To authenticate a message $X = x_1, \ldots, x_m$, the sender sets $y_0 = 0^n$ and (for $i = 1$ to m) sets $y_i = F_s(x_i \oplus y_{i-1})$. The tag is simply y_m. Specification of the verification algorithm is left to the reader.

This scheme — in contrast to the CBC mode of encryption — is deterministic; in the context of message authentication this does not necessarily present a problem. Note also that, in contrast to the XOR-MAC, this construction works for *fixed* message lengths only. In fact, it is completely insecure when variable-length messages are used. As a simple example of an attack, say the adversary requests a tag on message $x_1 \in \{0,1\}^n$ — receiving t — and then requests a tag on message t — receiving t'. Note that that t' is a valid tag for the message $x_1, 0^n$ (we leave verification of this fact to the reader). The following theorem therefore refers only to the case where fixed-length messages (m blocks long) are authenticated.

Theorem 2 *For any adversary attacking the CBC-MAC, running in time (roughly) t and requesting at most q tags from its MAC oracle, the probability of successfully forging a new, valid message/tag pair is at most $\frac{q^2m^2}{2^{n-1}} + 2^{-n} + \epsilon$.*

Roughly speaking, the first term corresponds to a sort of collision (we do not discuss details here); the second term comes from the fact that the adversary can always "guess" an n-bit tag correctly with probability 2^{-n}; and the third term comes from the security of the PRF. Again, we do not provide details here, but refer the reader to the well-written paper [2] which describes the CBC-MAC and gives a full proof.

Hash-and-MAC. This scheme is not used in practice, but variants (i.e., UMAC) are. But the real reason for presenting this scheme is to introduce the notion of *collision-resistant hash functions*, a useful cryptographic primitive that will come up again later in the course.

Assume a hash function $H : \{0,1\}^* \to \{0,1\}^n$ that compresses arbitrary-length inputs to an n-bit output. We say that x, x' represents a *collision for H* if $x \neq x'$ but $H(x) = H(x')$. Informally, H is *collision-resistant* if it is "infeasible" to find a collision for H; more formally:

Definition 1 *(Informal) H is (t,ϵ)-collision resistant if for all A running in time at most t, we have:*

$$\Pr[(x,x') \leftarrow A : x \neq x' \wedge H(x) = H(x')] < \epsilon.$$

We note that the above definition would need to be adapted to give a rigorous, complexity-theoretic definition of collision-resistance, but we do not give such a definition in this course.

Collision-resistant hash functions are very useful, and have many applications. Interestingly, collision-resistant hash function are the first primitive we have seen so far that *cannot* be constructed from an arbitrary one-way permutation (this statement is slightly informal; ask me if you are interested in the exact statement of this result). All other primitives we have seen thus far — encryption, PRGs, PRFs, PRPs, message authentication — can be constructed based on any one-way function. Yet collision-resistance seems to be a strictly stronger assumption than one-wayness.

On the other hand, collision-resistant hash functions *can* be constructed based on *specific* assumptions such as RSA, hardness of factoring, and hardness of computing discrete logarithms. From a practical point of view, there are many efficient constructions of (what are believed to be) collision-resistant hash functions; the most well-known of these are SHA-1 and MD5. The situation here is analogous to that of PRFs: we know how to construct PRFs from any one-way function but in practice we use block ciphers like DES or AES which we believe make good PRFs.

References

[1] M. Bellare, R. Guerin, and P. Rogaway. XOR MACs: New Methods for Message Authentication Using Finite Pseudorandom Functions. Crypto '95. Available at `http://www-cse.ucsd.edu/users/mihir/papers/xormacs.html`.

[2] M. Bellare, J. Kilian, and P. Rogaway. The Security of the Cipher Block Chaining Message Authentication Code. *Journal of Computer and System Sciences*, 61(3): 362–399 (2000). Preliminary version in Crypto '94. Available at `http://www-cse.ucsd.edu/users/mihir/papers/cbc.html`.