University of Maryland
CMSC456 — Introduction to Cryptography
Professor Jonathan Katz

Lecture 19

1 A Secure Message Authentication Code

We saw last time the definition of security for a message authentication scheme: that an adversary should be unable to forge a valid tag on *any* message that was not explicitly validated by the legitimate sender. More formally, the adversary has access to an oracle $\text{MAC}_{sk}(\cdot)$, and can request tags for multiple messages of his choice. The adversary "succeeds" if it outputs a pair $\langle M, \mathsf{tag}\rangle$ for which $\mathsf{Vrfy}_{sk}(M, \mathsf{tag}) = 1$ and furthermore M was *not* submitted to the $\text{MAC}_{sk}(\cdot)$ oracle (if the adversary succeeds in this way, we say that it has forged a valid tag on message M). A message authentication code is (t, ϵ)-secure if any adversary running in time t has success probability less than ϵ (the complete, formal definition is in the previous lecture notes).

Note that the above definition says nothing about "replay attacks" whereby an adversary re-sends a message that was previously sent by the legitimate sender. In all the examples we have discussed so far, the Vrfy algorithm was stateless so that if a pair $\langle M, \mathsf{tag}\rangle$ is ever accepted by the receiver, then it will be accepted later on if the adversary replays it. Although preventing replay attacks of this sort is important, the best way to prevent such attacks is often application-specific (what is the sender supposed to do if he legitimately wants to repeat a previous message?) and so we do not discuss it much here. For the record, though, it is not all that difficult to prevent these attacks in practice. For example, if there is a global clock the sender can authenticate the concatenation of a message and a timestamp; if the adversary replays the message at some later time it will be evident that this is a replay because the timestamp will be out of date. This neglects a lot of details (what of the clocks of the sender and receiver differ by a few seconds?) but gives a hint of the way replay attacks can be handled.

We actually have all the tools necessary to construct a secure message authentication code for short messages. Let $F : \{0,1\}^k \times \{0,1\}^m \rightarrow \{0,1\}^n$ be a (t, ϵ)-secure PRF. We can authenticate m-bit messages as follows: the sender and receiver share a secret key $sk \in \{0,1\}^k$. To authenticate a message $M \in \{0,1\}^m$, the sender computes $\mathsf{tag} = F_{sk}(M)$. When the receiver receives $\langle M, \mathsf{tag}\rangle$, he outputs 1 iff $F_{sk}(M) = \mathsf{tag}$. It should be clear, on an intuitive level, that this is a secure MAC (when n is sufficiently long); for completeness, we give a proof here.

Theorem 1 *The* MAC *outlined above is* $(t, \epsilon + 2^{-n})$*-secure.*

Proof Let A be an adversary attacking the scheme; i.e., A attempts to forge a valid tag on a new message. Recall that A interacts with the scheme by requesting a bunch of tags to be computed on various message, and at some point outputs an attempted forgery $\langle M, \mathsf{tag}\rangle$.

We show how to convert any such adversary into an adversary B that tries to distinguish F from a truly random function.

B is given access to a function oracle, where this function is either a completely random function or else an instance of $F_{sk}(\cdot)$ for randomly-chosen sk. B will simulate an instance of the MAC for A in the following way: when A requests authentication of a message M', B submits M' to its function oracle and receives back some value tag' which it then returns to A. Finally, A outputs $\langle M, \mathsf{tag}\rangle$ and is done. B checks whether A ever requested that message M be authenticated; if so, B guesses "random function" (i.e., outputs 0) and stops. Otherwise, B submits M to its oracle and receives back some value tag^*. If $\mathsf{tag} = \mathsf{tag}^*$, then B guesses "pseudorandom function" (i.e., outputs 1); otherwise, B guesses "random function" (i.e., outputs 0).

Let us analyze $\Pr[sk \leftarrow \{0,1\}^k : B^{F_{sk}(\cdot)} = 1]$. In this case, B is given access to a pseudorandom function, and the simulation for A is exactly equivalent to an execution of A attacking the actual MAC. So:

$$\begin{aligned}&\Pr[sk \leftarrow \{0,1\}^k : B^{F_{sk}(\cdot)} = 1]\\ &\quad= \Pr[sk \leftarrow \{0,1\}^k; \langle M, \mathsf{tag}\rangle \leftarrow A^{\mathrm{MAC}_{sk}(\cdot)} : \mathsf{Vrfy}_{sk}(M, \mathsf{tag}) = 1 \wedge M \notin \mathcal{M}],\end{aligned}$$

the success probability of A.

On the other hand, consider $\Pr[F \leftarrow \mathsf{Rand}^{m\to n} : B^{F(\cdot)} = 1]$. This is exactly the probability that A can forge a valid tag on a previously-unqueried message, when the MAC is instantiated with a *completely random* function. A little thought should show that since F is competely random, A cannot possible predict the correct tag on a non-queried message with probability better than 2^{-n} (since n is the output length of the function). So, $\Pr[F \leftarrow \mathsf{Rand}^{m\to n} : B^{F(\cdot)} = 1] \leq 2^{-n}$.

Since we know that F is a (t,ϵ)-PRF, we have:

$$\left|\Pr[sk \leftarrow \{0,1\}^k : B^{F_{sk}(\cdot)} = 1] - \Pr[F \leftarrow \mathsf{Rand}^{m\to n} : B^{F(\cdot)} = 1]\right| \leq \epsilon.$$

This immediately implies that

$$\Pr[sk \leftarrow \{0,1\}^k; \langle M, \mathsf{tag}\rangle \leftarrow A^{\mathrm{MAC}_{sk}(\cdot)} : \mathsf{Vrfy}_{sk}(M, \mathsf{tag}) = 1 \wedge M \notin \mathcal{M}] \leq \epsilon + 2^{-n},$$

which proves the theorem. ■

2 MACs for Arbitrary-Length Messages

The scheme of the previous section was secure, but only allowed authentication of short, fixed-length messages. As in the case of encryption, we would like to extend this to allow authentication of longer messages. It is much more difficult to do so here. To see this, recall that in the case of encryption we could always use the scheme in which we simply break the message into blocks and encrypt each block separately (using an indistinguishable encryption scheme) — this approach may not be the most efficient, but at least it will be secure! But this does *not* work in the case of authentication. Consider extending the scheme of the previous section in this way. So when an attacker requests an authentication of $M = M_1 \circ M_2$ the sender computes and returns $\mathsf{tag} = \mathsf{tag}_1 \circ \mathsf{tag}_2$, where $\mathsf{tag}_1 = F_{sk}(M_1)$ and $\mathsf{tag}_2 = F_{sk}(M_2)$. But it is easy to forge a new tag in this scheme: note that $\mathsf{tag}_2 \circ \mathsf{tag}_1$ is a valid tag on $M' = M_2 \circ M_2 \neq M$!

Hmm…this scheme was no good since it allowed an adversary to "reshuffle" message blocks. So maybe we can enforce some ordering on these blocks[1]? For example, we might compute $\text{MAC}_{sk}(M_1 \circ M_2)$ as follows: $F_{sk}(\langle 1 \rangle \circ M_1) \circ F_{sk}(\langle 2 \rangle M_2)$. (Where $\langle i \rangle$ represents some fixed-length encoding of the integer i, and M_1, M_2 are of the appropriate lengths.) This certainly prevents the re-ordering attack of the previous scheme. But is it secure? Definitely not. An adversary who obtains the tag $\mathsf{tag}_1 \circ \mathsf{tag}_2$ for message $M_1 \circ M_2$ can immediately forge a valid tag for message M_1 (it should be obvious how). It should also be clear that an adversary who obtains multiple tags on different messages can "piece together" different parts of these messages to create a valid tag on some new message.

It seems that a basic flaw in these schemes is that the tags are computed and transmitted in a blockwise fashion, which gives the adversary too much to play with. Maybe a variation on the last scheme will work. Specifically, instead of outputting the concatenation of tag_1 and tag_2, why not output their xor? In this case, we have $\text{MAC}_{sk}(M_1 \circ M_2) = F_{sk}(\langle 1 \rangle \circ M_1) \oplus F_{sk}(\langle 2 \rangle \circ M_2)$ (and verification is performed in the obvious way). It certainly seems harder to attack this scheme than either of our previous schemes.

But of course we should never be satisfied with a construction that "looks secure" (it seems that adversaries are always more clever than designers…). We prefer a scheme that is provably secure! And for good reason: this last scheme is in fact *not* secure, via the following attack. Say an adversary obtains tag tag_1 for message M_1, tag tag_2 for message $M_1 \circ M_2$, and tag tag_3 for message M_1'. Then the adversary has learned the following:

$$\begin{aligned} \mathsf{tag}_1 &= F_{sk}(\langle 1 \rangle \circ M_1) \\ \mathsf{tag}_2 &= F_{sk}(\langle 1 \rangle \circ M_1) \oplus F_{sk}(\langle 2 \rangle \circ M_2) \\ \mathsf{tag}_3 &= F_{sk}(\langle 1 \rangle \circ M_1'). \end{aligned}$$

Since $\mathsf{tag}_1 \oplus \mathsf{tag}_2 = F_{sk}(\langle 2 \rangle \circ M_2)$, the adversary can forge the valid tag $\mathsf{tag}_1 \oplus \mathsf{tag}_2 \oplus \mathsf{tag}_3$ on the (new) message $M_1' \circ M_2$, and the scheme is not secure.

This brings us to a scheme (the *XOR-MAC*) for arbitrary-length messages which *is* provably secure. It is very close to this last scheme, except that it is randomized. As above, let $F : \{0,1\}^k \times \{0,1\}^m \to \{0,1\}^n$ be a (t, ϵ)-PRF. The notation $\langle i \rangle$ will denote the $m/2$-bit representation of integer i in binary (we will be limited to messages at most $2^{m/2} - 1$ blocks long, not a serious restriction in practice). To authenticate a messgae M, parse M as $M_1 \circ \cdots \circ M_\ell$, where $|M_i| = m/2$. Choose a random value $r \in \{0,1\}^{m-1}$ and compute:

$$\mathsf{tag}' = F_{sk}(0 \circ r) \oplus F_{sk}(\langle 1 \rangle \circ M_1) \oplus \cdots \oplus F_{sk}(\langle \ell \rangle \circ M_\ell).$$

The complete tag is $\mathsf{tag} = (r, \mathsf{tag}')$. Specifying the verification algorithm is left as an exercise for the reader. For a detailed description of the XOR-MAC and a proof of security, see [1].

[1] The discussion in the next few paragraphs is informal, until we get to a scheme which is actually secure.

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`.