CS 346 Class Notes

Mark Lindberg

Feb 10, 2016

Last Time:
Message Authentication Code (MAC)
$\mathsf{Gen}(1^n) \to n$-bit key k.
$\mathsf{Mac}_k(m) \to$ tag t.
$\mathsf{Vrfy}_k(m,t) \to$ valid/invalid.
This Time:
$\mathsf{Mac\text{-}forge}_{\mathcal{A},t}(n)$, $\mathcal{A}$ gets access to Mac_k oracle, eventually outputs (m,t).

$\mathcal{A}$ "succeeds" if $\mathsf{Vrfy}_k(m,t)$ outputs valid AND $m \notin Q$, where Q is the set uf all messages passed to the Mac_k oracle.

The Mac Π is secure if $\forall$ PPT $\mathcal{A}$, $\Pr[A \text{ succeeds}] = \mathtt{negl}(m)$.

Today we will examine several Macs.

A first secure Mac for fixed-length messages of length n.

Assume F is a PRF. Let m be an n-bit message. Gen will work as normal, generating an n-bit key.

A natural first urge is to set $\mathsf{Mac}_k(m) = F_k(m)$. We will go ahead and do this.

This is a deterministic Mac, so we can use the "canonical verification", which is the Vrfy algorithm defined above.

Proof that Π defined here is a secure Mac.

Proof by contradiction sketch: If Π were not secure, F would not be a PRF. Assume Π is not secure. Then there is a PPT adversary $\mathcal{A}$ such that $\Pr[A \text{ succeeds}] = f(m)$, such that $f(m)$ is non-negligible.

Actual proof presented, direct proof. Let $\mathcal{A}$ be an arbitrary PPT adversary in the experiment $\mathsf{Mac\text{-}forge}_{\mathcal{A},\Pi}(n)$. Let $h(n)$ denote the success probability of A. Construct a PT distinguisher D for F based on $\mathcal{A}$.

The advantage of D is $\Pr[D^{F_k}(1^n) = 1] - \Pr[D^f(1^n) = 1]$.

D will simulate $\mathcal{A}$, using its oracle to answer $\mathcal{A}$'s queries to Mac_k. Finally, D gets output (m,t) of $\mathcal{A}$. D should output 1 when $\mathcal{A}$ succeeds. This involves a single oracle call, for Vrfy, maintaining Q.

Scenario 1: D's oracle is F_k. Then $\Pr[D^{F_k}(1^m) = 1] = \Pr[\mathsf{Mac\text{-}forge}_{\mathcal{A},\Pi}(m) = 1]$.

Scenario 2: D's oracle is f. $\Pr[D^f(1^n) = 1] \le \frac{1}{2^n}$.

Now we will examine secure $\mathsf{Mac}\ \Pi$ for arbitrary-length messages. It will be based on the secure fixed-length $\mathsf{Mac}\ \Pi'$ $(\mathsf{Mac}', \mathsf{Vrfy}')$ shown above.

First idea for Mac_k. Chop m into n-bit blocks, $m_1, m_2, \ldots, m_d$, let $t_i = \mathsf{Mac}'(m_i)$, and use $(t_1, t_2, \ldots, t_d)$ as the tag.

This is bad. This can be easily broken using a reordering attack. Present $m = m_1, m_2$, get tag t_1, t_2. Then message $m' = m_2, m_1$ will have tag t_2, t_1, which will pass Vrfy.

To combat this attack, break m into $\frac{n}{2}$-bit blocks $m_1, \ldots, m_d$, then $t_i = \mathsf{Mac}'_k(\langle i \rangle \mid\mid m_i)$, where $\langle i \rangle$ is the $\frac{n}{2}$-bit binary encoding of i. This prevents the reordering attack.

This scheme is still insecure. Since we have an arbitrary-length message Mac, we can use a truncation attack, and present $m = m_1, m_2, m_3$, get (t_1, t_2, t_3). Then we can present $m' = m_1, m_2$. The tag (t_1, t_2) will be valid for m'.

To prevent the truncation attack, we will include the length ℓ of the full message in the calculation. We will chop our message into $\frac{n}{3}$-bit blocks. Then $t_i = \mathsf{Mac}'_k(\langle \ell \rangle \mid\mid \langle i \rangle \mid\mid m_i)$. Note: We pad the last block with $0's$ if necessary. The tag will be $(t_1, \ldots)$. By this point, we are sending 4ℓ bits.

Unfortunately, even this scheme is still insecure. It can be attacked with a "mix and match" attack. For example, get tag $t = (t_1, t_2, t_3)$ for $m = m_1, m_2, m_3$. Take another message, same length, $m' = m_4, m_5, m_6$, get tag $t' = (t_4, t_5, t_6)$. Then (t_1, t_5, t_6) is a valid tag for m_1, m_5, m_6, which has never been queried from the oracle before.

Finally, let's fix all of this! We'll chop our message $m = m_1, \ldots, m_d$ into $\frac{n}{4}$ bit blocks, and pick a random $\frac{n}{4}$-bit value r for the entire message, and $t_i = \mathsf{Mac}'_k(r \mid\mid \langle \ell \rangle \mid\mid \langle i \rangle \mid\mid m_i)$. $\mathsf{Mac}_k(m) = (r, t_1, \ldots, t_d)$. At this point, this is not a deterministic Mac, so Vrfy has to behave slightly differently, taking into account the random r passed to it. It can reconstruct the tag as above, with this slight extra step.

This is secure!

Proof-ish. Fix the PPT adversary $\mathcal{A}$ in the forging experiment. We need to show that $\Pr[\mathcal{A}]$ succeeding is negligible. More formally, $\Pr[\mathsf{Mac\text{-}forge}_{\mathcal{A},=Pi}(m) = 1]$ is `negl`.

Fix a message m. There are three events of interest in the experiment $\mathsf{Mac\text{-}forge}_{\mathcal{A},\Pi}(m)$.

E_1: $\mathcal{A}$ succeeds.

E_2: Some r repeats.

E_3: Some $(r||\langle \ell \rangle||\langle i \rangle||m_i)$ is passed to Mac'_k when checking $\mathcal{A}$'s output is "new".

$$\begin{aligned} \Pr[E_1] &= \Pr[E_1 \wedge E_2] + \Pr[E_1 \wedge \overline{E_2} \wedge E_3] + \Pr[E_1 \wedge \overline{E_2} \wedge \overline{E_3}] \\ &\leq \Pr[E_2] + \Pr[E_1 \wedge E_3] + \Pr[E_1 \wedge \overline{E_2} \wedge \overline{E_3}] \end{aligned}$$

Proof to be completed at the beginning of the next class.