CS 346 Class Notes

Mark Lindberg

Feb 10, 2016

New HW out, due Wed, Feb 24.

Last Time:

We had CPA-secure schemes with CBC, OFB, and CTR block ciphers. Note that for each of these, the IV is sent along with the ciphertext.

Today:

How do you decrypt with CBC, given (IV, c_1, c_2, c_3)? We rely on the invertibility of F_k. Then $m_1 = IV \oplus F_k^{-1}(c_1)$. $m_2 = c_1 \oplus F_k^{-1}(c_2)$, and $m_3 = c_2 \oplus F_k^{-1}(c_3)$.

In OFB mode, we don't need the inverse of F_k. $m_1 = c_1 \oplus F_k(IV)$, $m_2 = c_2 \oplus F_k^2(IV)$, $m_3 = F_k^3(IV)$, where F_k^n means applying F_k n times to the given value.

In CTR mode, again, we don't need any inverse. CTR mode is parallelizable. $m_1 = F_k(CTR + 1) \oplus c_1$, $m_2 = F_k(CTR + 2) \oplus c_2$, etc.

CBC mode has the advantage that if you reuse an IV, we have problems where the prefix of the message is equal, but as soon as the messages differ, they are distinct. In CTR mode, a reused counter can be fatal, due to the properties of $\oplus$.

Now, a stronger security notion: CCA-security. Chosen Ciphertext Attack. $\mathsf{PrivK}^{\mathsf{CCA}}_{\mathcal{A},\Pi}$.

1. Before determining m_0, m_1, $\mathcal{A}$ gets access to Dec_k oracle in addition to Enc_k.
2. After receiving a challenge ciphertext c, It again has access to Dec_k and Enc_k, with the exception that Dec_k cannot be called on c, because that would tell us immediately whether m_0 or m_1 was used.

A scheme is CCA-secure if $\forall$ PPT adversaries $\mathcal{A}$, $\Pr[\mathsf{PrivK}^{\mathsf{CCA}}_{\mathcal{A},\Pi} = 1] \leq \frac{1}{2} + \mathtt{negl}$.

None of the schemes we've seen so far are CCA-secure.

Consider $\mathsf{Enc}_k(m) = (r, F_k(r) \oplus m)$, the first CPA-secure scheme we saw. This scheme is not CCA-secure. $\mathcal{A}$ can set $m_0 = 0^n$ and $m_1 = 1^n$. We will get challenge ciphertext (r, s), where $s = F_k(r) \oplus m_b$. Call $\mathsf{Dec}_k(r, \overline{s})$, where $\overline{s} = \neg s$.

$$F_k(r) \oplus m_b = s$$
$$1^n \oplus F_k(r) \oplus m_b = s \oplus 1^n = \overline{s}$$
$$= F_k(r) \oplus (m_b \oplus 1^n) = F_k(r) \oplus \overline{m_b}$$

Then $\mathsf{Dec}_k(r, \overline{s}) = F_k(r) \oplus F_k(r) \oplus \overline{m_b} = \overline{m_b}$, which makes it trivial to determine which message was sent.

We can do the same thing on each block individually for CTR mode, as it's literally the exact same thing, with $r = CTR + 1$, etc.

With CBC mode, we will begin by examining 1-block messages. $\mathcal{A}$: $m_0 = 0^n$, and $m_1 = 1^n$. Then we get (IV, s), where $s = F_k(m_b \oplus IV)$. Now we make a call to Dec_k. Here, $(IV, \overline{s})$ is not so helpful for us. Instead, we call $(\overline{IV}, s)$.

$$\mathsf{Dec}_k(\overline{IV}, s) = F_k^{-1}(s) \oplus \overline{IV}$$
$$ARGH.$$

To achieve CCA security, we will first develop another tool–MAC. Message Authentication Code. (Chapter 4)

With a one-time-pad, someone can tamper with the message and it will be undetectable. Now we care about the integrity of the message. We want to be able to detect any tampering with the messages.

MAC: Encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$. $\mathsf{Gen}(1^n)$ produces a key, usually a random n-bit string. Gen is PPT.

Tag Generation Algorithm. Mac: Takes a message $m \in \{0,1\}^*$, or $\{0,1\}^{\ell(n)}$ for fixed length), k, returns TAC. Mac is PPT.

Vrfy: Takes a key k message m, tag t, and outputs 1 if it's valid or 0 if it's invalid. Vrfy is a deterministic polynomial time algorithm.

Correctness: $\mathsf{Vrfy}_k(m, \mathsf{Mac}_k(m)) = 1$.

What is a secure Mac? As in chapter 3, we will define "secure" in terms of suitable experiments.

The Message Authentication Experiment: $\mathsf{Mac} - forge_{\mathcal{A},\Pi}(n)$.

1. Choose k, a random n-bit string.
2. $\mathcal{A}$ gets access to Mac_k oracle, then outputs (m, t).
3. $\mathcal{A}$ succeeds if $\mathsf{Vrfy}_k(m, t) = 1$, where $m \notin Q$, where Q is the set of messages passed to the oracle in step 2. The experiment outputs 1 if it succeeds in this step.

Mac Π is existentially unforgeable under an adaptive chosen-message attack ("secure") if $\forall$ PPT $\mathcal{A}$, $\Pr[\mathsf{Mac} - forge_{\mathcal{A},\Pi}(n) = 1] \leq \mathtt{negl}$.

"Strong" version. We call the experiment $\mathsf{Mac} - sforge(n)$, and change the previous experiment to say to say $(m, t) \notin Q$ where m, t are input-output pairs for oracle calls. Basically, we cannot forge an "alternate" tag for a message for a message, having seen a tag already.

There is no difference in the case where Mac is deterministic.

$\mathsf{Vrfy}_k(m, t)$ just checks $\mathsf{Mac}_k(m) = t$, in the canonical, common, practical version.

Might want to look at exercise 4.2 in the text, for fun. Vrfy oracle can be simulated by using the Mac oracle. It leads to an equivalent definition. 4.3 is in the HW.