CS 346 Class Notes

Mark Lindberg

Feb 22, 2016

Last Time:

Experiment	Security notion
$\mathsf{Mac\text{-}sforge}_{\mathcal{A},\Pi}(n)$	Strongly secure mac.
$\mathsf{PrivK}^{\mathsf{CPA}}_{\mathcal{A},\Pi}(n)$	CPA-secure encryption scheme.
$\mathsf{PrivK}^{\mathsf{CCA}}_{\mathcal{A},\Pi}(n)$	CCA-secure encryption scheme.
$\mathsf{Enc\text{-}forge}_{\mathcal{A},\Pi}(n)$	Unforgeable encryption scheme.

An authenticated encryption scheme should be

1. Unforgeable.
2. CCA-secure.

We'll get the above by combining a CPA-secure scheme with a strongly secure MAC.

This Time:

In the $\mathsf{Mac\text{-}sforge}_{\mathcal{A},\Pi}(n)$ experiment the adversary $\mathcal{A}$ gets access to Mac_k oracle which outputs (m, t) The adversary succeeds if it can output a pair (m, t) which is new such that $\mathsf{Vrfy}_k(m, t) = 1$.

In the $\mathsf{PrivK}^{\mathsf{CPA}}_{\mathcal{A},\Pi}(n)$ experiment, the challenger generates k. The adversary gets access to Enc_k oracle. The adversary picks m_0, m_1 of the same length, sends them to the challenger. The challenger picks a random bit b, encrypts $c = \mathsf{Enc}_k(b)$, and sends back c. The adversary then has more oracle access to Enc_k. (Not on m_0 or m_1, though.)

In the $\mathsf{PrivK}^{\mathsf{CCA}}_{\mathcal{A},\Pi}(n)$ experiment, much is the same as the previous experiment, except that the adversary also gets access to a Dec_k oracle, though they cannot call it on the c produced by the challenger.

In the $\mathsf{Enc\text{-}forge}_{\mathcal{A},\Pi}(n)$ experiment, the adversary gets access to Enc_k oracle. The adversary outputs c. The adversary succeeds if

1. $\mathsf{Dec}_k(c) = 1$. (c is a valid ciphertext.)
2. $\mathsf{Dec}_k(c)$ is not a message we queried the oracle on previously.

"Encrypt-then-authenticate" paradigm. Let $\Pi_E = (\mathsf{Enc}, \mathsf{Dec})$ be a CPA-secure encryption scheme, and $\Pi_M = (\mathsf{Mac}, \mathsf{Vrfy})$ be a strongly secure MAC. (Gen is dropped because we can use the same one for both, but don't want name collisions.) Then $\Pi := (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$.

Gen' generates independent n-bit keys k_E and k_M. Let $k = (k_E, k_M)$.
$\mathsf{Enc}'_k(m) = (c, \mathsf{Mac}_{k_M}(c))$ where $c = \mathsf{Enc}_{k_E}(m)$.
$\mathsf{Dec}'_k((c,t))$. First, verify that $\mathsf{Vrfy}_{k_M}(c,t) = 1$. If not, return $\perp$. If so, then return $\mathsf{Dec}_{k_E}(c)$.
THEOREM: Π' is an authentication scheme as defined above.

Intuitive proof outline: Consider an adversary $\mathcal{A}$ in the CCA experiment. We'll show that whp (with high probability), all calls to Dec'_k one outputs $\perp$ unless they correspond to a call to Enc'_k. A call comes to Enc'_k, which comes from the use of Π_M, which yields unforgeablity. Consequently, the ability to call Dec'_k is "useless", so the CPA security of Π_E will be enough.

Some proof details:

"Valid query" event. Call to Dec'_k with (c,t) such that

1. (c,t) is not output of prior Enc'_k call.

2. $\mathsf{Dec}'_k(c,t) \neq \perp$.

Claim: $\Pr[\text{valid query}] \leq \mathtt{negl}(n)$.

Assume $\mathcal{A}$ makes $\leq q(n)$ Dec queries, where q is polynomial.

Simulation argument: Construct an adversary $\mathcal{A}_M$ from $\mathcal{A}$, in the $\mathsf{Mac\text{-}sforge}_{\mathcal{A}_M,\Pi_M}(n)$ experiment. It has access to Mac_{k_M}. At the beginning, $\mathcal{A}_M$ chooses a random k_E when $\mathcal{A}$ call $\mathsf{Enc}'_k(m)$. $\mathcal{A}_M$ simulates this call

1. can compute $c = \mathsf{Enc}_{k_E}(m)$.

2. uses oracle to get t.

If (c,t) is the output of a previous call to $\mathsf{Enc}'_k(m)$, return m. Else, if this is the ith nontrivial query, where i is a random number we picked, halt and output (c,t). Else, return $\perp$.

$$\Pr\left[\mathsf{Mac\text{-}sforge}_{\mathcal{A}_M,\Pi_M}(m) = 1\right] \geq \frac{\Pr[\text{valid query}]}{q(n)}$$

This is definitely the most complicated/confusing proof we've seen so far...