Notes for Lecture 9

Notes scribed by Joel Weinberger, posted March 1, 2009

Summary

Last time, we showed that combining a CPA-secure encryption with a secure MAC gives a CCA-secure encryption scheme. Today we shall see that such a combination has to be done carefully, or the security guarantee on the combined encryption scheme will not hold.

We then begin to talk about *cryptographic hash functions*, and their construction via the *Merkle-Damgård transform*.

1 Combining Encryption and Authentication

1.1 Encrypt-Then-Authenticate

Let (E, D) be an encryption scheme and (Tag, V) be a MAC.

Last time we considered their *encrypt-then-authenticate* combination defined as follows:

Construction (E_1, D_1)

- Key: a pair (K_1, K_2) where K_1 is a key for (E, D) and K_2 is a key for (Tag, V)
- $E_1((K_1, K_2), M)$:
  - $C := E(K_1, M)$
  - $T := Tag(K_2, C)$
  - return (C, T)
- $D_1((K_1, K_2), (C, T)$:
  - if $V(K_2, C, T)$ then return $D(K_1, C)$
  - else return 'ERROR'

and we proved that if (E, D) is CPA-secure and (Tag, V) is existentially unforgeable under a chosen message attack then (E_1, D_1) is CCA-secure.

Such a result is not provable if (E, D) and (Tag, V) are combined in different ways.

1.2 Encrypt-And-Authenticate

Consider the following alternative composition:

Construction (E_2, D_2)

- Key: a pair (K_1, K_2) where K_1 is a key for (E, D) and K_2 is a key for (Tag, V)
- $E_2((K_1, K_2), M) := E(K_1, M), Tag(K_2, M)$
- $D_2((K_1, K_2), (C, T)$:
  - $M := D(K_1, C)$
  - if $V(K_2, M, T)$ then return M
  - else return 'ERROR'

The problem with this construction is that a MAC (Tag, V) can be secure even if $Tag()$ is deterministic. (E.g. CBC-MAC.) But if the construction (E_2, D_2) is instantiated with a deterministic $Tag()$, then it cannot even guarantee security for 2 encryptions (much less CPA-security or CCA security).

A more theoretical problem with this construction is that a MAC (Tag, V) can be secure even if $Tag(K, M)$ *completely gives away* M, and in such a case (E_2, D_2) is completely broken.

1.3 Authenticate-Then-Encrypt

Finally, consider the following scheme:

Construction (E_3, D_3)

- Key: a pair (K_1, K_2) where K_1 is a key for (E, D) and K_2 is a key for (Tag, V)
- $E_3((K_1, K_2), M)$:
  - $T := Tag(K_2, M)$
  - return $E(K_1, (M, T))$
- $D_3((K_1, K_2), C$:
  - $(M, T) := D(K_1, C)$
  - if $V(K_2, M, T)$ then return M
  - else return 'ERROR'

The problem with this construction is rather subtle.

First of all, the major problem of the construction (E_2, D_2), in which we lost even security for two encryptions, does not occur.

Exercise 1 *Show that if (E, D) is (t, ϵ) CPA-secure and E, D, Tag, V all have running time at most r, then (E_3, D_3) is $(t/O(r), \epsilon)$ CPA secure*

It is possible, however, that (E, D) is CPA-secure and (Tag, V) is existentially unforgeable under chosen message attack, and yet (E_3, D_3) is not CCA-secure.

Suppose that (Tag, V) is such that, in $Tag(K, M)$, the first bit is ignored in the verification. This seems reasonable in the case that some padding is needed to fill out a network protocol, for example. Further, suppose (E, D) is counter mode with a pseudo-random function, F_{K_1}.

Take the encryption of $M_1, ..., M_l$ with keys K_1 and K_2 for E and V, respectively. Pick a random r. Then, by the definition of the encryption scheme in counter mode, the encryption of E, T will be:

$$r, F_{K_1}(r) \oplus M_1, F_{K_1}(r+1) \oplus M_2, ..., F_{K_1}(r+l-1) \oplus M_l, F_{K_1}(r+l) \oplus T_1, F_{K_1}(r+l+1) \oplus T_2, ...$$

Consider this CCA attack. Let $e_1 = (1, 0, ..., 0)$. The attacker sees

$$r, C_1, C_2, ..., C_l, C_{l+1}, ..., C_{l+c}$$

Take $e_1 \oplus C_{l+1}$ (or any of the other tags). Clearly, this is not the original message encryption, as a bit has been changed, so the Oracle will give you a decryption of it in a CCA attack. Since all that was modified in the ciphertext was the first bit, which was padding, the attacker has just used the Oracle to get the original message.

2 Cryptographic Hash Functions

2.1 Definition and Birthday Attack

Definition 1 (Collision-Resistant Hash Function) *A function $H : \{0,1\}^k \times \{0,1\}^L \to \{0,1\}^\ell$ is a (t, ϵ) secure collision resistant hash function if $L > \ell$ and for every algorithm A of complexity $\leq t$ we have*

$$\mathbb{P}[A(s) = (x, x') : H^s(x) = H^s(x')] \leq \epsilon \tag{1}$$

The idea is that, for every key (or *seed*) $s \in \{0,1\}^k$ we have a length-decreasing function $H^s : \{0,1\}^L \to \{0,1\}^\ell$. By the pigeon-hole principle, such functions cannot be injective. An efficient algorithm, however, cannot find *collisions* (pairs of inputs that produce the same output) even given the seed s.

The main *security parameter* in a construction of collision-resistent hash functions (that is, the parameter that one needs to increase in order to hope to achieve larger t and smaller ϵ) is the output length ℓ.

It is easy to see that if H has running time r and output length ℓ then we can find collisions in time $O(r \cdot 2^\ell)$ by computing values of H in any order until we find a collision. (By the pigeon-hole principle, a collision will be found in $2^\ell + 1$ attempts or fewer.)

If, specifically, we attack H by trying a sequence of *randomly chosen* inputs until we find a collision, then we can show that with only $2^{\ell/2}$ attempts we already have a constant probability of finding a collision. (The presence of a collision can be tested by sorting the outputs. Overall, this takes time $O(2^{\ell/2} \cdot \ell + 2^{\ell/2} \cdot r) = O(r \cdot 2^{\ell/2})$.) The calculation can be generalized to the following:

Consider A given $H^s(\cdot)$. Define A as picking m random strings $x_1 ... x_m$ and generating their respective outputs from H^s. We want to check the probability $\mathbb{P}[\text{collision}]$ that $H^s(x_1) ... H^s(x_m)$ contains a repeated value. However, it is easier to begin by checking the probability of whether the m choices *do not* contain a collision, $\mathbb{P}[\text{no collision}]$:

$$\mathbb{P}[\text{no collision}] = 1 \cdot \left(1 - \frac{1}{2^\ell}\right) \cdot \left(1 - \frac{2}{2^\ell}\right) \cdots \left(1 - \frac{(\mathrm{m}-1)}{2^\ell}\right)$$

$$\begin{aligned}
\mathbb{P}[\text{collision}] &= 1 - \mathbb{P}[\text{no collision}] \\
&\approx e^{-\mathbb{P}[\text{no collision}]} \\
&= e^{-\frac{1}{2^\ell}} \cdot e^{-\frac{2}{2^\ell}} \cdot \ldots \cdot e^{-\frac{(m-1)}{2^\ell}} \\
&= e^{-\frac{1+2+\ldots+(m-1)}{2^\ell}} \\
&\approx e^{-\frac{m^2}{2^{\ell+1}}}
\end{aligned}$$

Note that this is a constant if $m \approx 2^{\ell/2}$.

(This calculation is called the "birthday paradox," because it establishes that in a set of n elements uniformly distributed, if you pick elements $x_1 ... x_{\sqrt{n}}$, where x_i is uniformally distributed in the set and x_i is independent, then there is a constant probability that $\exists i, j : x_i = x_j$. Specifically, in the case of birthdays, if there are 23 people in a room, then there is a probability over $\frac{1}{2}$ that two of the people in the room have the same birthday.)

Exercise 2 *If $H : \{0,1\}^k \times \{0,1\}^L \to \{0,1\}^\ell$ is a (t, ϵ) secure collision resistant hash*

function computable in time r, then

$$\frac{t^2}{\epsilon} \leq O(r \cdot 2^\ell) \tag{2}$$

This should be contrasted with the case of pseudorandom generators and pseudorandom functions, where the security parameter is the seed (or key) length k, and the only known generic attack is the one described in a previous exercise which gives

$$\frac{t}{\epsilon} \leq O(m2^\ell) \tag{3}$$

This means that if one wants a (t, ϵ) secure pseudorandom generator or function where, say, $t = 2^{80}$ and $\epsilon = 2^{-40}$, then it is somewhat plausible that a key of 128 bits might suffice. The same level of security for a collision-resistant hash function, however, requires a key of at least about 200 bits.

To make matters worse, attacks which are significantly faster than the generic birthday attack have been found for the two constructions of hash functions which are most used in practice: MD5 and SHA-1. MD5, which has $\ell = 128$, is completely broken by such new attacks. Implementing the new attacks on SHA-1 (which has $\ell = 160$) is not yet feasible but is about 1,000 times faster than the birthday attack.

There is a process under way to define new standards for collision-resistant hash functions.

2.2 The Merkle-Damgård Transform

In practice, it is convenient to have collision-resistant hash functions $H : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^\ell$ in which the input is allowed to be (essentially) arbitrarily long.

The Merkle-Damgård transform is a generic way to transform a hash function that has $L = 2\ell$ into one that is able to handle messages of arbitrary length.

Let $H : \{0,1\}^k \times \{0,1\}^{2\ell} \to \{0,1\}^\ell$ a hash functions that compresses by a factor of two.

Then $H^s_{MD}(M)$ is defined as follows (IV is a fixed ℓ-bit string, for example the all-zero string):

- Let L be the length of M
- divide M into blocks $M_1, \ldots, M_B$ of length ℓ, where $B = \lceil L/\ell \rceil$
- $h_0 := IV$

- for $i := 1$ to B : $h_i := H^s(M_i, h_{i-1})$
- return $H^s(L, h_B)$

We can provide the following security analysis:

Theorem 2 *If H is (t, ϵ)-secure and has running time r, then H_{MD} has security $(t - O(rL/\ell), \epsilon)$ when used to hash messages of length up to L.*

Proof:

Suppose that A, given s, has a probability ϵ of finding messages M, M' where $M = M_1, ..., M_B, M' = M'_1, ..., M'_B$ and $H^s_{MD}(M) = H^s_{MD}(M')$, where $B = L/\ell, B' = L'/\ell$. Let us construct an algorithm A' which:

1. Given s, runs A to find collisions for M, M' in H^s_{MD}. Assume that running A takes time t_{MD}.
2. Computes both $H^s_{MD}(M) and H^s_{MD}(M')$. We know that this takes time on the order of $r \cdot 2L/\ell$, where r is the time of running H^s and assuming without loss of generality that $L \geq L'$ since the Merkle-Damgård Transform, by definition, runs H^s once for each of the L/ℓ blocks of M and M'.
3. Finds a collision of H^s that is guaranteed to exist (we will prove this below) in the Merkle-Damgård algorithm. This will take the time of running the algorithm once for each message M, M', which is on the order of $r \cdot 2L/\ell$, as above.

In order to show that there must be a collision in H^s if there is a collision in H^s_{MD}, recall that the last computation of the MD algorithm hashes $M_{B+1} = L$. We need to consider two cases:

1. $L \neq L'$: By the definition of the problem, we have $H^s_{MD}(M) = H^s_{MD}(M')$. The last step of the alrogithm for M and M' generates, respectively, $H^s(L, h_B) = H^s_{MD}(M)$ and $H^s(L', h'_{B'}) = H^s_{MD}(M') = H^s_{MD}(M)$. This is a collision of H^s on different inputs.
2. $L = L'$: This implies that $B = B'$ and $h_{B+1} = h'_{B'+1}$, as well. Because $M \neq M'$ but $|M| \neq |M'|$, $\exists i, 0 \leq i \leq B : h_i \neq h'_i$.

   Let $j \leq B + 1$ be the largest index where $h_j \neq h'_j$. If $j = B + 1$, then h_{j+1} and h'_{j+1} are two (different) colliding strings because $H^s(h_j) = h_{j+1} = H^s_{MD}(M) = H^s_{MD}(M') = h'_{j+1} = H^s(h'_j)$.

   If $j \leq B$, then that implies $h_{j+1} = h'_{j+1}$. Thus, h_j and h'_j are two different strings whose hashes collide.

This shows that a collision in H^s_{MD} implies a collision in H^s, thus guaranteeing for our proof that by executing each stage of the Merkle-Damgård Transform, we can find a collision in H^s if there is a collision in H^s_{MD}.

Returning to algorithm A', we now have established the steps necessary to find collisions of a hash function H^s by finding collisions in H^s_{MD}. Note that by the definition of the problem, because it uses algorithm A which finds collisions with probability ϵ, A' will also find collisions with probability ϵ. By the definition of a collision-resistant hash function, we also know that it must have time complexity t.

Examining the steps of the algorithm, we know that the time complexity of A is $t = t_{MD} + 2r \cdot L/\ell + 2r \cdot L/\ell = 4r \cdot L/\ell = t_{MD} + O(rL/\ell)$. From this, we solve for t_{MD} and get $t_{MD} = t - O(rL/\ell)$. Thus, we conclude that if A finds a collision with probability ϵ, it is $(t - O(RL/\ell), \epsilon)$ secure.

□