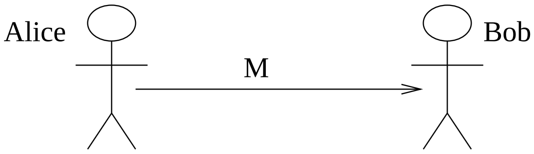
U.C. Berkeley — CS276: Cryptography
Professors Luca Trevisan and David Wagner

Lecture 14
Scribe: Dennis Geels

Lecture Notes: 3/7/2002

1 Message Authentication

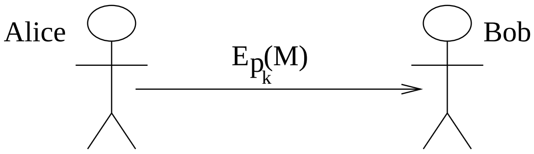
Suppose Alice sends a message M to Bob and wants to prevent anyone from tampering with the message. For example, Bob could be a bank, Alice could be that bank's customer, and M could be the message "please send \$100 to Amazon". Alice (and probably Bob) would prefer that a third party, Eve, could not alter the message to read, for example, "please send \$100 to Eve", or "please send \$300 to Amazon".



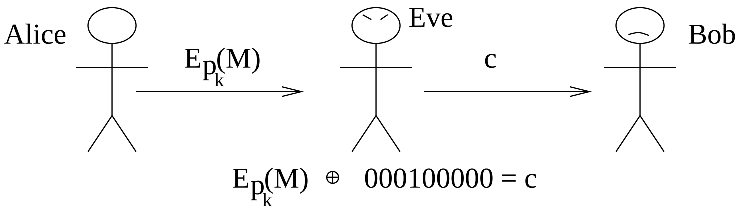
Alice and Bob need a Message Authentication scheme that lets Bob verify that the messages he receives were written by Alice and were not altered in transit.

1.1 One Approach - Encryption

Accepted practice once dictated that Alice could protect the message simply by encrypting it.



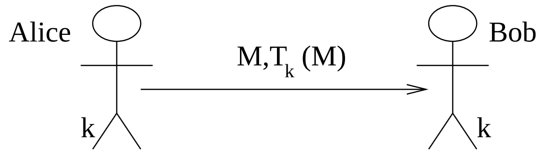
Unfortunately this scheme may allow Eve to trick Bob by modifying the ciphertext $E_{p_k}(M)$ in transit. For example, consider a OTP or the Goldwasser-Micali encryption algorithm, which encrypt bits of the message independently. Eve could flip a few bits of the cipher text, producing a new message c. Bob decrypts c, producing a new plaintext with the same bits flipped.



If Eve chooses bits carefully, she could trick Bob into believing that Alice sent the message "please send \$300 to Amazon".

1.2 A Better Approach - MACs

Instead, Alice and Bob could use a keyed, cryptographically-secure tag algorithm T, with shared key k. Alice appends to the message M its tag $T_k(M)$. Bob receives a message m with tag t and accepts m if and only if he can verify the combination: $V_k(m,t) = 1$.



We define a Message Authentication Code (MAC) as a pair of tag and verify algorithms (T, V) such that messages Bob can always verify messages truly sent by Alice:

$$T : K \times \{0,1\}^* \to S$$

$$V : K \times \{0,1\}^* \times S \to \{0,1\}$$

$$V_k(m, T_k(m)) = 1 \qquad \forall m, k$$

We then define two notions of security for MACs.

Suppose an adversary A^{T_k} makes q queries $m_1, \ldots, m_q$ and receives tags $t_1, \ldots, t_q$. Then make define the adversary's advantage as follows:

$$\mathbf{Adv}_{weak} A = \mathbf{Pr}_{k \in K}[A \text{ outputs } (m,t) \text{ s.t. } V_k(m,t) = 1, \quad \forall i \ m \neq m_i]$$

$$\mathbf{Adv}_{strong} A = \mathbf{Pr}_{k \in K}[A \text{ outputs } (m,t) \text{ s.t. } V_k(m,t) = 1, \quad \forall i \ (m,t) \neq (m_i, t_i)]$$

The difference between $\mathbf{Adv}_{weak} A$ and $\mathbf{Adv}_{strong} A$ is the restriction against reusing tags in the latter.

Definition 1 *(T, V) is a (t, q, μ, ϵ)-weakly-secure MAC if $\mathbf{Adv}_{weak} A < \epsilon \quad \forall$ adversaries A running in time t, making q queries of total length μ.*

Definition 2 *(T, V) is a (t, q, μ, ϵ)-strongly-secure MAC if $\mathbf{Adv}_{strong} A < \epsilon \quad \forall$ adversaries A running in time t, making q queries of total length μ.*

Note: most texts only mention one type of security for MACs. It is generally the first (weak) version.

Clearly strongly-secure MACs are weakly-secure. The converse is not always true. Consider the following simple, deterministic MAC: Pick a good T. Then define V as follows:

$$V_k(m,t) = 1 \textbf{ iff } T_k(m) = t \tag{1}$$

Strongly-secure and weakly-secure are equivalent for this kind of MAC.

The definitions above handle every important aspect of security for MACs except replay attacks: Eve could confuse Bob by saving one of Alice's valid messages and sending it to Bob at a later point. Bob will accept the message, which is tagged correctly, but Alice did not send it (not twice, at least). This weakness results from our restriction that Bob is stateless - V_k retains no state across messages.

To thwart replay attacks, Alice can add sequence numbers to her messages. Bob would then accept only messages that both have valid tags and also have a unique (monotonically increasing) sequence number. There are several techniques to reduce the amount of state required on Bob's end. For instance, one could timestamp messages and exploit loose synchronization between Alice and Bob to bound the amount of time Bob must remember a previously sent message. This approach requires a MAC that is secure in the strong sense.

1.3 Using a PRF as a MAC

One way to construct a MAC (T, V) is to let T be a cryptographically-secure PRF and to define V as in equation 1.

LEMMA 1
*If $R : X \to Y$ is a random function, and $T_R = R$, $V_R(m,t) = 1$ **iff** $T_R(m) = t$, then (T_R, V_R) is a $(\infty, q, \mu, \frac{1}{|Y|})$-strongly-secure MAC.*

PROOF: Because R is a random function, we could implement T_R by choosing $T_R(m)$ only after seeing m, as long as we choose uniformly at random from all possible outputs. Suppose the adversary A outputs the message and tag (m,t). We know that m must not be one of A's queries, so we have not yet chosen $T_R(m)$. Therefore it must be chosen independently at random from the set of all possible outputs Y:

$$\begin{aligned}&\mathbf{Pr}_R[A \text{ outputs } (m,t) \text{ s.t. } V_R(m,t) = 1, \quad \forall i\ (m,t) \neq (m_i, t_i)] \\ &\qquad = \mathbf{Pr}_{y \in Y}[y = t] \leq \frac{1}{|Y|}\end{aligned}$$

□

THEOREM 2
*If $F : K \times X \to Y$ is a (t, q, μ, ϵ)-secure PRF, $T_k = F_k$, $V_k(m,t) = 1$ **iff** $T_k(m) = t$, then (T_k, V_k) is a $(t - O(1), q-1, \mu - O(1), \epsilon + \frac{1}{|Y|})$-strongly-secure MAC.*

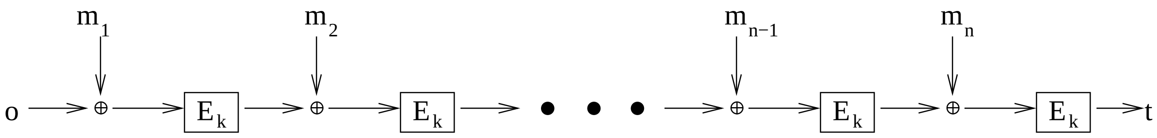
In other words, an MAC based on a PRF gives up only an additional $\frac{1}{|Y|}$ advantage over the internal PRF. This theorem can be proved using a simulation argument much like the others we've seen: an adversary that can break the MAC's security could be used to build an adversary that can break the security of the PRF.

1.4 Variable-length MACs

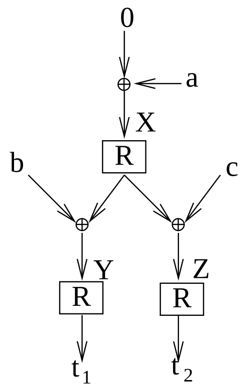
Now we present a MAC scheme for variable-length messages. CBC-MAC chains together iterations of a fixed-length MAC as follows:

DEFINITION 3 *CBC-MAC[E_k](m) $= E_k(m_n \oplus E_k(m_{n-1} \oplus \cdots E_k(m_2 \oplus E_k(m_1))\cdots))$, where E_k is a MAC and $m = (m_1, \ldots, m_n)$ is a message.*

To analyze the security of CBC-MAC, assume that E_k is a random function R. Then an adversary gains an advantage over pure chance only if it finds two messages that pass the identical input to R at some point of their computation. Otherwise, the output of R, and therefore the MAC, is completely independent from all other inputs.



To bound the probability of finding messages with such a collision on inputs to R, arrange the messages as a forest of trees in which trivial collisions are assigned to the same node. For example, the following diagram shows the tree corresponding to the two messages $m_1 = (a, b)$ and $m_2 = (a, c)$. R is invoked on three distinct inputs, X, Y, and Z. The advantage of an adversary (beyond pure chance) is then the probability that X, Y, and Z are not all unique.



Consider the general case of q messages of m blocks each. Each node in the forest (invocation of R) causes a collision in a child node (a following invocation of R) if the parent's output XOR the next message block equals the input of any other node in the forest. For example, if $Y = R(X) \oplus \text{ b } \in \{X, Z\}$ in the example above. We can bound that probability by the fraction of the output space consumed by the $q \cdot m$ nodes in the forest: $qm/2^{64}$.

Thus the probability of a collision at any child of a node with n children can be bounded by $nqm/2^{64}$. Define n_i to be the number of children of node i. Then the total probability of collision is:

$$\mathbf{Pr}[\text{collision}] \leq \sum_i \mathbf{Pr}[\text{collision at node } i] \leq \sum_i \frac{n_i qm}{2^{64}} = \frac{qm}{2^{64}} \sum_i n_i \leq \frac{qm}{2^{64}} qm = \frac{(qm)^2}{2^{64}}$$

A little formalism would produce the following lemma:

Lemma 3
If $R : \{0,1\}^l \to \{0,1\}^l$ is a random function, CBC-MAC[R] on m-block messages is $(\infty, q, qml, (1.5)\frac{(qm)^2}{2^l})$-secure.

So CBC-MAC is secure in fixed-length message space. In general, it is secure in prefix-free message spaces. Unfortunately the MAC of two messages x and y, where x is a prefix of y, are not necessarily independent. This shortcoming can be overcome, for instance by preventing prefix messages.

1.5 LOR-CCA Security

We now explore an extension to LOR-security to account for chosen ciphertext attacks. In the realm of public-key cryptography, CCA security was related to the concept of Non-malleability. Naturally we may assume that MACs may help in the symmetric case.

A few suggestions for adding MACs to an LOR-CPA-secure encryption algorithm (E,D):

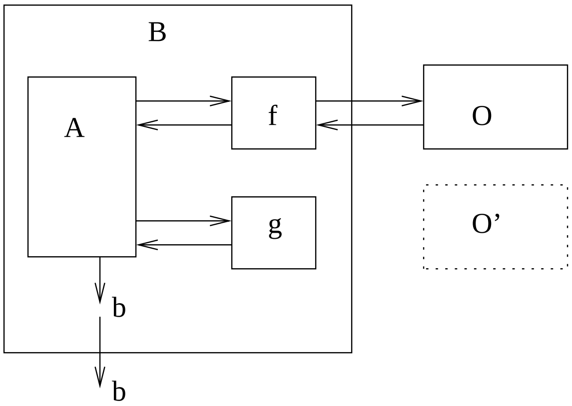
(i)	Encrypt then MAC:	$E_k(m)\|\|T_{k'}(E_k(m))$	LOR-CCA secure
(ii)	MAC then encrypt:	$E_k(m\|\|T_{k'}(m))$	not secure in general
(iii)	Encrypt and MAC:	$E_k(m)\|\|T_{k'}(m)$	not secure

Given (E, D), a (t, q, μ, ϵ)-secure LOR-CPA encryption system and (T, V), a $(t, q + 1, \mu', \epsilon')$-strongly-secure MAC, define

$$E'_{kk'}(m) = E_k(m)||T_{k'}(E_k(m))$$

$$D'_{kk'}(c||t) = D_k(c) \text{ if } V_{k'}(c, t) = 1$$

D' should refuse to decrypt (returning some sort of error) if it cannot verify the MAC.



THEOREM 4
(E', D') is $(t - O(\mu + \mu'), q, \mu, \epsilon + q\epsilon')$-secure in the LOR-CCA sense.

We assume that, because (T, V) is secure, an adversary would never query its decryption oracle with a valid tag. Our proof replaces a true oracle with one that crashes and burns in the off-chance that it receives a valid $(c||t)$ pair. We then have a good chance of using an adversary to (E', D') to break (E, D).

PROOF: If $A^{O,O'}$ breaks E', D', then let B^O be an adversary to (E, D) with no decryption oracle (we're using a CPA) defined as follows:

(1) pick $k' \in K$
(2) let $b = A^{f,g}$, where $f(x) = O(x)||T_k(O(x))$ and $g(y)$ always returns a decryption error.
(3) output b

B^O has the same advantage as $A^{O,O'}$ unless a BAD™ thing happens: $A^{O,O'}$ manages to provide a good tag to its fake decryption oracle g. We can bound the probability of this event as usual. Thus if $A^{O,O'}$ breaks (E', D'), we can break (E, D) □