University of Maryland
CMSC456 — Introduction to Cryptography
Professor Jonathan Katz

Lecture 18

1 Modes of Encryption

We mentioned last time the concept of *modes of encryption* which are used to encrypt arbitrary-length messages, and gave some examples. One thing we did not stress in the last class was the precise definition of left-or-right indistinguishability where modes of encryption are concerned. As usual, we allow the adversary to access the left-or-right oracle LR as many times as it likes. However, we *must* restrict the adversary as follows: when the adversary submits a query (M_0, M_1) to the LR oracle we require that $|M_0| = |M_1|$. (In any of the encryption schemes given in the previous lecture, it should be clear that it is easy to break the encryption scheme if this restriction is removed; in fact, it is *impossible* to construct any encryption scheme which is secure when the adversary is allowed to submit messages of different lengths.) We stress that we impose *no* restriction on the lengths of messages in *different* queries; thus, the first query (M_0, M_1) and the second query (M'_0, M'_1) can have $|M_0| \neq |M'_0|$.

In all of what follows we assume a PRF $F : \{0,1\}^k \times \{0,1\}^m \rightarrow \{0,1\}^m$ and the message M to be encrypted is parsed into a sequence of m-bit blocks $M_1, \ldots, M_\ell$.

Cipher-block chaining (CBC) mode.

```
$\mathcal{E}_s(M)$
 $C_0 \leftarrow \{0,1\}^m$
 For $i = 1$ to $\ell$
     $C_i = F_s(C_{i-1} \oplus M_i)$
 Output $C_0, C_1, \ldots, C_\ell$

$\mathcal{D}_s(C_0, \ldots, C_\ell)$
 For $i = 1$ to $\ell$
   $M_i = C_{i-1} \oplus F_s^{-1}(C_i)$
 Output $M_1, \ldots, M_\ell$
```

The random block C_0 is called an *initialization vector*, and is chosen randomly each time a new message is encrypted. Since this is a *randomized* encryption scheme, it might potentially be secure in the sense of indistinguishability. In fact, this *is* a secure mode of encryption; see reference [1] in the previous lecture notes for details.

The ciphertext here is longer than the message by an *additive* factor of m bits (as opposed to the multiplicative factor we had in our initial mode of encryption). This is essentially optimal (note that a scheme with no ciphertext expansion would be deterministic and hence insecure). In the CBC mode, decryption needs to compute F_s^{-1}; thus, F must be a keyed *permutation*, and it must be efficient to invert (given a particular key s).

Cipher feedback (CFB) mode.

```
$\mathcal{E}_s(M)$
  $C_0 \leftarrow \{0,1\}^m$
  For $i = 1$ to $\ell$
      $C_i = M_i \oplus F_s(C_{i-1})$
  Output $C_0, C_1, \ldots, C_\ell$

$\mathcal{D}_s(C_0, \ldots, C_\ell)$
  For $i = 1$ to $\ell$
    $M_i = C_i \oplus F_s(C_{i-1})$
  Output $M_1, \ldots, M_\ell$
```

Again we have a random initialization vector chosen each time a new message is encrypted. Like CBC mode, this scheme is secure and the ciphertext is longer than the message by only m bits. An advantage of this scheme over CBC mode is that F does not need to be a keyed permutation (in fact, we never need to invert F in order to decrypt). This is useful in some contexts for reasons of efficiency and more importantly allows us to use PRFs F which are *not* permutations.

2 Message Authentication

Thus far in this class, the only cryptographic *application* we have considered is encryption; i.e., keeping the contents of a transmitted message (or messages) hidden from an eavesdropping adversary. For this application — at least thus far — the adversary has been *passive* (even though our definition of security allowed an adversary to mount chosen plaintext attacks and interact repeatedly with a LR oracle, this was meant to model potential eavesdropping of multiple ciphertexts). But there are other important applications which seek to protect against an *active* adversary who tries to interfere with the communication between the sender and receiver. (Again, in the case of encryption the adversary was assumed only to eavesdrop, but never to interfere in any other way with transmissions.) This brings us to the important application of message authentication.

Informally, a message authentication scheme (or MAC — message authentication code) allows a receiver to be convinced that the message(s) he receives are actually sent by the claimed sender. In other words, imagine a sender $\mathcal{S}$ and a receiver $\mathcal{R}$ who wish to communicate in such a way that $\mathcal{R}$ is guaranteed that messages he receives are unaltered, and were actually sent by $\mathcal{S}$. As a real world example, imagine a general who wants the ability to send commands to her soldiers in such a way that the soldiers can be convinced that the orders originated from her. Or you can imagine a bank which needs to ensure that messages it receives from clients were actually issued by those clients. MACs allow them to do this.

It is important to recognize that encryption is completely orthogonal to message authentication. You can — and often do — have one without the other. It should also be clear that encryption generally provides no message authentication whatsoever. As an example, consider the one-time pad which provides perfect secrecy when a single message is encrypted. But even when $\mathcal{S}$ and $\mathcal{R}$ share a key s, an adversary can send any "message" C to the receiver and this will be interpreted by $\mathcal{R}$ as a real message $C \oplus s$ that was sent by $\mathcal{S}$. Even worse, if the sender encrypts message M and sends it (i.e., sends $C = M \oplus s$), an adversary can flip the last bit of C — giving C' — and then the receiver will obtain a message M' which differs from M in the final bit. Imagine the damage this could cause if C were an encryption of, say, the amount of a bank withdrawal!

Before continuing, we give a formal definition of our goal (always an important thing to do). A *message authentication code* consists of two algorithms: the MAC algorithm which takes a secret key s and a message M and returns a "tag" tag; and the verification algorithm Vrfy which takes a key s, a message M, and a "tag" tag and returns 1 if the tag is valid and 0 otherwise. The scheme is used as follows: the sender and receiver secretly share a random key s. When the sender wishes to send message M to the receiver, the sender computes $\mathsf{tag} \leftarrow \textsc{mac}_s(M)$ and sends $\langle M, \mathsf{tag}\rangle$. When the receiver gets transmission $\langle M', \mathsf{tag}'\rangle$, the receiver will not accept the message (i.e., will not be convinced that the message was sent by the sender) unless $\mathsf{Vrfy}_s(M', \mathsf{tag}') = 1$. Of course, we require that the scheme be *correct*, so that if $\textsc{mac}_s(M)$ outputs tag, then $\mathsf{Vrfy}_s(M, \mathsf{tag})$ should always output 1.

With this in mind, we can define an appropriate notion of security. Informally, an adversary should be unable to "forge" a valid tag on any message that was *not* sent by $\mathcal{S}$. That is, an adversary should be unable to "fool" the receiver into accepting *any* message M which did not come from $\mathcal{S}$. Optimally, we would like to prevent the adversary from "fooling" the receiver even after the adversary sees multiple tags computed on many different messages. In fact, we will require something even stronger: that the adversary be unable to "fool" the receive, even after seeing tags for *multiple messages of the adversary's choice.*

A little more formally, we will define an *oracle* $\textsc{mac}_s(\cdot)$ which takes as input a message M and returns as output the tag for that message, computed with respect to key s (one can think of this oracle as representing the sender). We will allow an adversary to interact with this oracle as many times as it likes, submitting multiple messages of its choice and receiving in return the correspoding tags. We say the adversary "breaks" the scheme if it can forge a valid tag for a message M that it never submitted to its oracle; i.e., if the adversary can output a pair $\langle M, \mathsf{tag}\rangle$ (where M was never submitted to $\textsc{mac}_s(\cdot)$) such that $\mathsf{Vrfy}_s(M, \mathsf{tag}) = 1$. We say the scheme is secure if no adversary can "break" the scheme with high probability. The formal definition follows.

Definition 1 *A message authentication scheme* $(\textsc{mac}, \mathsf{Vrfy})$ *is* (t, ϵ)*-secure if, for all adversaries A running in time at most t, we have:*

$$\Pr[s \leftarrow \{0,1\}^k; \langle M, \mathsf{tag}\rangle \leftarrow A^{\mathrm{MAC}_s(\cdot)} : \mathsf{Vrfy}_s(M, \mathsf{tag}) = 1 \wedge M \notin \mathcal{M}] \leq \epsilon,$$

where $\mathcal{M}$ is the set of messages that A submitted to its oracle $\textsc{mac}_s(\cdot)$.

Just to give some practice with the definition, we first examine an insecure scheme: the one-time pad example from above. In this case, we may view the function $\textsc{mac}_s(M)$ as returning $M \oplus s$; algorithm $\mathsf{Vrfy}_s(M, \mathsf{tag})$ returns 1 if and only if $M \oplus \mathsf{tag} = s$. Of course, this scheme is not secure, and an adversary can forge a valid tag on any message M of its choice as follows: the adversary first submits M' (where $M' \neq M$) to its oracle, and receives in return tag. The adversary then outputs its forgery $\langle M, \mathsf{tag} \oplus M \oplus M'\rangle$. To see that this fools the receiver, note that

$$(\mathsf{tag} \oplus M \oplus M') \oplus M = \mathsf{tag} \oplus M' = s,$$

where this is true since tag was a valid tag for message M'.