Admin:
- Pset #3 & groups posted.
- Get your final project team & topic selected!

Today:
- **Message Authentication Codes (MAC's)**
  - HMAC
  - CBC-MAC
  - PRE-MAC
  - One-time MAC (problem statement)
- **AEAD (Authenticated Encryption with Associated Data)**
  - EAX mode (ref paper: pages 1-10 only)
  - Encrypt-then-MAC
- **Finite fields & number theory**

Readings:
- Katz/Lindell Chapter 4
- Paar/Pa21 Chapter 12
MAC (Message Authentication Code)

- Not confidentiality, but integrity (recall "CIA")
- Alice wants to send messages to Bob, such that Bob can verify that messages originated with Alice & arrive unmodified.
- Alice & Bob share a secret key $K$
- Orthogonal to confidentiality; typically do both (e.g. encrypt, then append MAC for integrity)
- Need additional methods (e.g. counters) to protect against replay attacks

Alice $\xrightarrow{K} M, \text{MAC}_K(M)$ \quad Bob

[Here $M$ is message to be authenticated, which could be ciphertext resulting from encryption.]

- Alice computes $\text{MAC}_K(M)$ & appends it to $M$.
- Bob recomputes $\text{MAC}_K(M)$ & verifies it agrees with what is received. If $\neq$, reject message.

If MAC has $t$ bits, then Adv has prob $2^{-t}$ of successful forgery. Good MAC is (keyed) PRF.
Adversary (Eve) wants to forge $M', MAC_k(M')$ pair that Bob accepts, without Eve knowing $K$.
- She may hear a number of valid ($M, MAC_k(M)$) pairs first, possibly even with $M$'s of her choice (chosen msg attacks).
- She wants to forge for $M'$ for which she hasn't seen ($M', MAC_k(M')$) valid pair.

Two common methods:

$$HMAC(K, M) = h(h(K || h(K_2 || M)))$$
where $K_1 = K \oplus \text{opad}$ \quad \{ opad, ipad are fixed constants \\
$K_2 = K \oplus \text{ipad}$

$$CBC-MAC(K, M) \equiv \text{last block of CBC enc. of } M$$

Note: $IV=0$

Something like this is necessary...
**MAC using random oracle (PRF):**

\[
\text{MAC}_k(M) = h(K || M)
\]

(OK if \(h\) is indistinguishable from RO, which means, as we saw, for sequential hash \(h\)s, that last block may need special treatment.)

**One-Time MAC (problem stmt):**

- Can we achieve security against unbounded Eve, as we did for confidentiality with OTP, except here for integrity?

Here key \(K\) may be "use-once" [as it was for OTP].

\[ \begin{array}{ccc}
A & \xrightarrow{K} & B \\
M, T & \xrightarrow{K} & T = \text{MAC}_k(M) \quad ("tag")
\end{array} \]

- Eve can learn \(M, T\) then try to replace \(M, T\) with \(M', T'\) (where \(M' \neq M\)) that Bob accepts.
- Eve is computationally unbounded.
<table>
<thead>
<tr>
<th></th>
<th>Confidentiality</th>
<th>Integrity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unconditional</td>
<td>OTP ✓</td>
<td>One-time MAC?</td>
</tr>
<tr>
<td>Conventional</td>
<td>Block ciphers</td>
<td>MAC (HMAC) ✓</td>
</tr>
<tr>
<td>(symmetric key)</td>
<td>(AES) ✓</td>
<td></td>
</tr>
<tr>
<td>Public-key</td>
<td>PK enc.</td>
<td>Digital signature</td>
</tr>
<tr>
<td>(asymmetric)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: digital signatures are unforgeable, but also have nonrepudiation, since only one copy of signing key exists.
EAX mode

See pgs 1-10 of The EAX Mode of Operation by Bellare, Rogaway, & Wagner

Figure 3

Encrypt-then-MAC

\[ C = \text{Enc}(K_1, M) \]
\[ T = \text{MAC}(K_2, H \| C) \]

xmit: H, C, T

Two passes

Two keys

C, not M!
Finite fields:

System \((S, +, \cdot)\) s.t.

- \(S\) is a **finite** set containing "0" & "1"
- \((S, +)\) is an abelian (commutative) group with identity 0
  \[ ((a+b)+c) = (a+(b+c)) \] associative
  \[ a + 0 = 0 + a = a \] identity 0
  \[ (\forall a)(\exists b) \ a + b = 0 \] (additive) inverses \(b = -a\)
  \[ a + b = b + a \] commutative
- \((S^*, \cdot)\) is an abelian group with identity 1
  \(S^*\) = nonzero elements of \(S\)
  \[ (a \cdot b) \cdot c = a \cdot (b \cdot c) \] associative
  \[ a \cdot 1 = 1 \cdot a = a \] identity 1
  \[ (\forall a \in S^*)(\exists b \in S^*) \ a \cdot b = 1 \] (multiplicative) inverses \(b = a^{-1}\)
  \[ a \cdot b = b \cdot a \] commutative
- Distributive laws:
  \[ a \cdot (b + c) = a \cdot b + a \cdot c \]
  \[ (b + c) \cdot a = b \cdot a + c \cdot a \] (follows)

Familiar fields: \(\mathbb{R}\) (reals) are **infinite**

\(\mathbb{C}\) (complex)

For crypto, we're usually interested in **finite** fields, such as \(\mathbb{Z}_p\) (integers mod prime \(p\))
Over field, usual algorithms work (mostly).

E.g. solving linear eqns:

\[ ax + b = 0 \pmod{\rho} \]

⇒ \[ x = a^{-1} \cdot (-b) \pmod{\rho} \] is soln.

\[ 3x + 5 = 6 \pmod{7} \]

\[ 3x = 1 \pmod{7} \]

\[ x = 5 \pmod{7} \]
Notation: GF(q) is the finite field ("Galois field") with q elements

Theorem: GF(q) exists whenever
\[ q = p^k, \ p \text{ prime, } k \geq 1 \]

Two cases:
1. GF(p) - work modulo prime p
   \[ \mathbb{Z}_p = \text{integers mod } p = \{0, 1, \ldots, p-1\} \]
   \[ \mathbb{Z}_p^* = \mathbb{Z}_p - \{0\} = \{1, 2, \ldots, p-1\} \]

2. GF(p^k) : k \geq 1
   work with polynomials of degree < k
   with coefficients from GF(p)
   modulo fixed irreducible polynomial of degree k
   Common case is GF(2^k)

Note: all operations can be performed efficiently
   (inverses to be demonstrated)
Construction of $GF(2^3) = GF(8)$

Has 4 elements.

Is not arithmetic mod 4, (where 2 has no multi inverse)

Elements are polynomials of degree 2 with coefficients
mod 2 (i.e. in $GF(2)$):

$$\begin{array}{c|cccc}
0 & 0 & 1 & x & x+1 \\
1 & 0 & 0 & 0 & 0 \\
x & 0 & 1 & x & x+1 \\
x+1 & 0 & x+1 & 1 & 0 \\
\end{array}$$

Addition is component-wise according to powers, as usual

$$(x) + (x+1) = (2x+1) = 1 \pmod{2}$$

Multiplication is modulo $x^3 + x + 1$

which is irreducible (doesn't factor)

$$x^2 \mod (x^3 + x + 1) \text{ is } x+1 \pmod{2} \text{ (note that } x \equiv -x \text{ coets mod 2)}$$
"Repeated squaring" to compute $a^b$ in field

(Here $b$ is a non-negative integer)

$$a^b = \begin{cases} 
1 & \text{if } b = 0 \\
(a^{b/2})^2 & \text{if } b > 0, \ b \text{ even} \\
a \cdot a^{b-1} & \text{if } b \text{ odd}
\end{cases}$$

Requires $\leq 2 \cdot \lg(b)$ multiplications in field (efficient)

$\approx$ a few milliseconds for $a^b \pmod p$, 1024-bit integers

$\approx \Theta(k^3)$ time for $k$-bit inputs

Computing (multiplicative) inverses:

**Theorem:** (For $\text{GF}(p)$ called "Fermat's Little Theorem")

In $\text{GF}(q)$ $(\forall a \in \text{GF}(q)^\times) \ a^{q-1} = 1$

**Corollary:** $(\forall a \in \text{GF}(q)) \ a^q = a$

**Corollary:** $(\forall a \in \text{GF}(q)^\times) \ a^{-1} = a^{q-2}$

**Example:** $3^{-1} \pmod 7$

$$= 3^5 \pmod 7$$

$$= 5 \pmod 7$$