Administrative Notes

The material on finite fields is taking longer than I anticipated, so we are running behind the schedule in the syllabus. This is not a problem: the material is important, relevant, and unusual, so we will take enough time to do it right. We might get some of the time back in the RSA section, because there is a little overlap.

Representing Field Elements

How does one work explicitly in $GF(q)$

The easiest case is when $q$ is itself prime (7, 11, . . . , $2^{32}, 582, 657 − 1$) (the latter being the largest known prime, according to Wikipedia), rather than a power of a prime: this is just modular arithmetic.

The prime power case is more complicated. Some new (to us) notation will help. Suppose that $K$ is a field. Let $K[x]$ denote the (ring of) polynomials with coefficients in $K$. For example, if $K = GF(3)$, then $x^3 + 2x + x + 2$ is an element of $K[x]$.

A polynomial $p(x) \in K[x]$ is irreducible if the only way to factor it as $p(x) = m(x)n(x)$ is for one of $n$ or $m$ to be a constant, that is, an element of $K$, that is, a polynomial of degree 0.

An ideal is a subset of a ring that is closed under addition within itself and multiplication by elements of the ring. Most of our examples, if not all of the examples of ideals in information theory, are principal ideals, meaning that there is an element $i$ such that every element of the ideal is a multiple of $i$; denote such an ideal as $(i)$. For example, in $\mathbb{Z}/12\mathbb{Z}$, the ideal $(2)$ consists of the elements 2, 4, 6, 8, and 10.

The quotient of a ring $A$ by an ideal $I$, written $A/I$ and called $A$ “mod” $I$, is the set of equivalence classes for the relation $a \equiv b \iff a - b \in I$. The rings $\mathbb{Z}/n\mathbb{Z}$ form an important example. As the notation implies, the set $n\mathbb{Z}$ is an ideal in the ring of integers $\mathbb{Z}$. When, for example, $n = 15$, the equivalence classes consist of sets like $\bar{1} = \{1, 16, 31, \ldots\}$ or $\bar{7} = \{7, 22, 37, \ldots\}$. The arithmetic operations from the ring defined arithmetic operations on the quotient by defining $\bar{a} + \bar{b}$ to be $\bar{a + b}$ and $\bar{ab} = \bar{a} \bar{b}$.

The situation for polynomial rings is no different. A key example is when $K = GF(2)$ and $p(x) = x^2 + x + 1$. The quotient $GF(2)[x]/(p(x))$ is a ring; in fact, it is isomorphic to $GF(4)$. There are four equivalence classes: 0, 1, $\bar{x}$, and $\bar{1} + \bar{x}$. Which class contains $x^2$? Since $x^2 - (x + 1)$ is in $(p(x))$, $x^2$ is in $\bar{x + 1}$. Similarly, find the class containing $x^3 + x$ using the Euclidean algorithm: $x^3 + x = x(x^2 + x + 1) + x^2$, so $(x^3 + x) - x^2$ is in $(p(x))$, and $x^3 + x = \bar{x + 1}$. 

1
It is a theorem that if \( p(x) \) is irreducible in \( K[x] \) then \( K[x]/(p(x)) \) is a field. (One proof of this is based on the extended Euclidean algorithm, because the irredicibility of \( p \) means that its GCD with any other polynomial \( n \) is 1, so the extended algorithm finds the inverse to \( n \).) Suppose that the degree of \( p \) is \( n \). Then the elements of this field are linear combinations (coefficients in \( K \)) of \( 1, \bar{x}, \bar{x}^2, \ldots, \bar{x}^{n-1} \). No higher powers are needed, because \( \bar{x}^n \) is equivalent to a sum of lower powers, using \( p(x) \).

Computation only uses the coefficients, and \( a_{n-1}\bar{x}^{n-1} + \cdots + a_1\bar{x} + a_0 \) can be stored as \( (a_{n-1}, a_{n-2}, \ldots, a_1, a_0) \). This makes the isomorphism of \( K[x]/(p(x)) \) with \( K^n \) explicit. This is called the polynomial representation.

Also, \( \bar{x} \) serves as the generator, like \( \theta \) in the Rijndael field. This leads to some work, since in a field with \( q \) elements the group generated by a generator is cyclic of order \( q - 1 \). Thus, in \( GF(4) \), one can write any element except 0 as \( \alpha^j \), with \( j \) either 1, 2, or 3. To wit: \( \alpha^1 = \alpha, \alpha^2 = \alpha + 1, \) and \( \alpha^3 = \alpha^2 + \alpha = 1 \). This is called the generator representation.

Further simplification occurs in the binary case, when each coefficient is 0 or 1: run them together into a bit string, like 001011010111, which represents an element in \( GF(2^{12}) \). When the degree is a multiple of three, these are combined into octal digits, as 01327, and when the degree is a multiple of four, these are combined into hexadecimal digits, as 0x2E7. These are the octal and hexadecimal representations.

In Rijndael, the element with hexadecimal representation 0x63, which is binary 01100011, is used during each round.

---

**AES**

AES is the Advanced Encryption Standard, also known as Rijndael. It is a block cipher with block size 128 and several supported key sizes. The encryption and decryption are different, unlike DES.

Each round of AES consists of four functions:

1. byte substitution
2. permutation
3. arithmetic over the Rijndael field
4. XOR with a round key

Stallings discusses a simplified AES for educational purposes, starting on page 165. All we will do today is outline the encryption function.

Rijndael’s 128-bit block is conveniently represented as a sequence of 16 8-bit bytes, which in turn fit into a \( 4 \times 4 \) array; the same applies to a 128-bit key. The array is filled first by columns, then by rows, like this:

<table>
<thead>
<tr>
<th>0</th>
<th>4</th>
<th>8</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>9</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>10</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>11</td>
<td>15</td>
</tr>
</tbody>
</table>

---

2
The initial value of the *state* is the input block, and each operation modifies the state. The final state is the output block.

---

**Substitute Bytes**

This operation uses an $S$-box, similar to that of DES. The $S$ box is $16 \times 16$. The first four bits of the byte select a row, and the last four select a column. The difference is that the $S$-box is constructed using the arithmetic of the Rijndael field. In other words, the substitute value determined by the $S$-box can be computed using linear algebra, rather than table-lookup.

A mere substitution provides no security *per se*, but does change simple ASCII text into something less obviously readable.

---

**Shift Rows**

The state array is subjected to bytewise shifts as illustrated below.

<table>
<thead>
<tr>
<th>0</th>
<th>4</th>
<th>8</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>9</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>14</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>15</td>
<td>3</td>
<td>7</td>
<td>11</td>
</tr>
</tbody>
</table>

---

**Mix Columns**

Each byte of a column of the state is transformed into a linear combination of all of the bytes in the column, as follows (the matrix entries are hexadecimal representations of elements of the Rijndael field):

\[
\begin{array}{cccc}
02 & 03 & 01 & 01 s_00 & s_01 & s_02 & s_03 \\
01 & 02 & 03 & 01 s_10 & s_11 & s_12 & s_13 \\
01 & 01 & 02 & 03 s_20 & s_21 & s_22 & s_23 \\
03 & 01 & 01 & 02 s_30 & s_31 & s_32 & s_33 \\
\end{array}
\]

These matrix values were chosen to ease hardware implementation.

---

**Add Round Key**

The round key is XORed with the state. Security is enhanced because a different round key is used in each round.