Homework 7 – Problem 1

1. We saw that a typical round of AES consisted of the following operations:
   for each byte, $b$ in state
       ByteSub($b$)
   ShiftRow(state)
   if (i < Nr)
       MixCol(state)
   AddRoundKey($i$, state)

For the 128 bit key, 128 bit block size version of Rijndael, using lookup tables to reduce the computations required and assuming basic operations (32 bit lookup, 32 bit xor, etc) all take about .001 microseconds and your code/data budget is under 16 MB, design a implementation of the round operations that is faster than implementing each of the primitive operations (ByteSub, ShiftRow, MixCol).

How long does each round take (about)?

Counter mode use of AES is used by selecting a nonce ($n$) and constructing cipher blocks $AES_K(n||ctr)$, $AES_K(n||ctr+1)$, $AES_K(n||ctr+2)$,.... The resulting bits are xored into the plaintext (as with the stream cipher). What properties of AES make this safe? Can the keystream be generated in parallel and stored for later use? What performance properties does this mode have over ECB?
Let the input to round and the round key be

\[
\begin{array}{cccc}
  a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\
  a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\
  a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\
  a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \\
\end{array}
\quad \quad
\begin{array}{cccc}
  k_{0,0} & k_{0,1} & k_{0,2} & k_{0,3} \\
  k_{1,0} & k_{1,1} & k_{1,2} & k_{1,3} \\
  k_{2,0} & k_{2,1} & k_{2,2} & k_{2,3} \\
  k_{3,0} & k_{3,1} & k_{3,2} & k_{3,3} \\
\end{array}
\]

and let \( S[a] \) be the table which contains the values of ByteSub then after the Shiftrow operation, the “state” is

\[
\begin{array}{cccc}
  S[a_{0,0}] & S[a_{0,1}] & S[a_{0,2}] & S[a_{0,3}] \\
  S[a_{1,0}] & S[a_{1,1}] & S[a_{1,2}] & S[a_{1,3}] \\
  S[a_{2,0}] & S[a_{2,1}] & S[a_{2,2}] & S[a_{2,3}] \\
  S[a_{3,0}] & S[a_{3,1}] & S[a_{3,2}] & S[a_{3,3}] \\
\end{array}
\]
MixCol transforms a column as specified:
\[ [t_0, t_1, t_2, t_3]^T \rightarrow t_0 [2,1,1,3]^T \oplus t_1 [3,2,1,1]^T \oplus t_2 [1,3,2,1]^T \oplus t_3 [1,1,3,2]^T \]

So after MixCol the column \( j \) of the state matrix (for 128 bit key, 128 bit block) is:
\[ S[a_{0,j}] [2,1,1,3]^T \oplus S[a_{0,j-1}] [3,2,1,1]^T \oplus S[a_{0,j-2}] [1,3,2,1]^T \oplus S[a_{0,j-3}] [1,1,3,2]^T \]

Setting
\[ T_0(a) = S[a] [2,1,1,3]^T, \quad T_1(a) = S[a] [3,2,1,1]^T, \quad T_2(a) = S[a] [1,3,2,1]^T, \quad T_3(a) = S[a] [1,1,3,2]^T \]

The output state, expressed as four columns, \( b_j \), resulting from an input state \( a_{ij} \)
and the key expressed as columns \( k_j \), is:
\[ b_j = k_j \oplus T_0(a_{0,j}) \oplus T_1(a_{1,j-1}) \oplus T_2(a_{2,j-2}) \oplus T_3(a_{3,j-3}) \quad \text{........... RoundTransform} \]

The \( T_i \) tables are 1KB each for a total of 4 KB. With these RoundTransform requires about 16 lookups on \( T_i \) plus about 15 index/lookup operations to obtain the \( a_{ij} \) and 4 32 bit Xors for about 35 operations (44 is more accurate). So each round requires about .04 microseconds and the 10 full rounds with loop testing etc requires about .5 microseconds to produce 128 bits of output for a predicted speed of about 32 MB/sec.
This is pretty close to the benchmarked results we mentioned in the first lecture although that machine was slower.

Counter mode if it is to be safe, must be immune from an attack for which related input produces related output. The keystream can be computed both in parallel and in advance of actual use. ECB enables parallel encryption but not pre-computation of the bulk of the encryption.
Homework 7 – Problem 2, 3

2. Show that \( f(x) = x^2 \mod pq \) is a One-Way Function but is not Collision Resistant, where \( p \) and \( q \) are prime (you don’t know \( p \) and \( q \), just their product).

3. Linear Feedback Shift Registers Cryptosystem: Suppose \( X \) is a cryptosystem implemented by a 5 element linear feedback shift register which generates a pseudo random stream \( s_0, s_1, s_2, \ldots \) so

\[
s_{n+5} = a_4 s_{n+4} \oplus a_3 s_{n+3} \oplus a_2 s_{n+2} \oplus a_1 s_{n+1} \oplus a_0 s_n
\]

If the first 10 output bits of the pseudo random generator are 1110100010, what are the next 3 bits? Assume \( n \) is the register length. About how many consecutive bits do you need to break a LFSR? How does this compare to a stream generator on an \( n \) bit state that is not linear?
If $f(x) = x^2 \pmod{pq}$, and $p$ and $q$ are not known [but $pq$ is known], inverting requires finding a square root mod $pq$, the first lecture explained that this was hard; so $f(x)$ is a OWF.

However $f(x)=f(pq-x)$ so $f$ is not collision resistant (or second pre-image resistant)
Homework 7 – Answer 3

1110100010 0101, \( a_4 = a_2 = a_1 = a_0 = 1, \ a_3 = 0 \)

Proof: Note \( s_{n+5} = a_4 \ s_{n+4} \oplus a_3 \ s_{n+3} \oplus a_2 \ s_{n+2} \oplus a_1 \ s_{n+1} \oplus a_0 \ s_n \)

<table>
<thead>
<tr>
<th>Eq</th>
<th>LHS</th>
<th>( a_4 )</th>
<th>( a_3 )</th>
<th>( a_2 )</th>
<th>( a_1 )</th>
<th>( a_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1,</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

A: \( 5 \& 6 \Rightarrow a_4 \oplus a_3 \oplus a_2 = 0 \)  
B: \( 6 \& 7 \Rightarrow a_3 \oplus a_2 \oplus a_1 = 0 \)  
C: \( 7 \& 8 \Rightarrow a_2 \oplus a_1 \oplus a_0 = 1 \)  
D: \( 8 \& 9 \Rightarrow a_4 \oplus a_1 \oplus a_0 = 1 \)

\[
\begin{align*}
A&B\rightarrow a_4 &= a_1 \\
B&C\rightarrow a_0 &= a_3 \oplus 1 \\
C&D\rightarrow a_4 &= a_2 \\
\text{So R: } &a_4 = a_1 = a_2 \\
&\text{and } a_0 = a_3 \oplus 1
\end{align*}
\]

Plugging R into B, we get \( a_0 = 1, \ a_3 = 0 \). 
Finally, plugging this into 9, we get \( a_4 = a_0 = 1 = a_1 = a_2 \).
Homework 7 – Answer 3

Thus the recurrence relation is:

\[ s_{n+5} = s_{n+4} \oplus s_{n+2} \oplus s_{n+1} \oplus s_n \]

What are the next 3 bits?

0101

How many consecutive bits do you need to break a LFSR?

2n, if the equations are not redundant

How does this compare to a stream generator that is not linear?

Couldn’t solve for internal state efficiently with a non-linear one. Arbitrary stream generator with n bits of state requires seeing \(2^n\) bits before you can predict next bit.
4. Given \( i = 64, j = 245 \) and \( S \) is as stated below, what are the next 4 bytes of output of RC4? Estimate the speed of encrypting the next 4 bytes of output of an RC4 cipher on a computer in which assignment addition and logical AND requires \( 0.001 \) microseconds.

5. Suppose two parties share a secret key \( k \) and wish to communicate a series of “yes/no” answers over a public channel without disclosing the answers. Design a protocol to do this using a MAC. Be careful to make sure the adversary cannot figure out all the answers if they know whether the “code” for a few of the yes/no answers.
Homework 7 Problem 4 data

S[0...127]:
0x08 0xa5 0xe9 0x09 0x45 0xc0 0xed 0xf1
0x5d 0xfd 0x34 0xc3 0xe4 0x7b 0x9d 0x96
0x38 0x76 0x7c 0x49 0x8f 0xd9 0x35 0xcc
0x99 0xb0 0x2d 0x97 0xe7 0x1d 0xa9 0x16
0x7d 0x10 0x8c 0x89 0x51 0xa1 0xd7 0x5b
0x3d 0x1c 0x23 0x1e 0xe0 0xb2 0x84 0xa8
0xc5 0x24 0x86 0xb9 0x07 0xac 0xf0 0x52
0x32 0x92 0xda 0x06 0xe4 0xd4 0x82 0xd5
0xdb 0xae 0x04 0x4c 0x36 0xc6 0x19 0x2e
0xb4 0x2c 0x69 0xc7 0xce 0x71 0x91 0xa6
0xde 0x22 0x59 0xf4 0x54 0x25 0x42 0x0d
0xff 0x03 0xa0 0x44 0x87 0x37 0x8e 0x12
0x30 0x33 0x58 0x3a 0x81 0xf3 0x8d 0x9f
0xbd 0xc4 0x95 0x73 0x93 0x55 0x41 0xb6
0x90 0x63 0x9c 0x18 0x77 0xdd 0xe3 0xc9
0x8a 0xb1 0x7f 0xee 0xe5 0xad 0x05 0xa0

S[128...255]:
0x6d 0x15 0xc2 0xab 0x7a 0xa4 0x3f 0x00
0x48 0xa3 0xd1 0x4a 0x75 0xb7 0x85 0xd8
0xfb 0xfe 0xf2 0xe6 0x13 0x56 0xec 0xa7
0x9a 0xe2 0x64 0x53 0x5f 0x65 0xd3 0xc8
0x68 0x74 0x02 0xdc 0x6f 0x43 0xe1 0x8b
0xbf 0xa2 0x2a 0x80 0xbb 0x6a 0x28 0x78
0x17 0xf6 0xfc 0x67 0xb3 0x9e 0xcb 0x31
0xf9 0x9a 0x9b 0x2b 0xb8 0x1a 0x3e 0xf8
0xda 0xa1 0x6b 0x20 0x11 0x4b 0x3b 0x0b
0xb4 0x2c 0x69 0xc7 0xce 0x71 0x91 0xa6
0x90 0x55 0x42 0x0d 0x22 0x59 0xf4 0x54
0x87 0x37 0x8e 0x12 0x03 0xa0 0x44 0x95
0x73 0x93 0x55 0x41 0xbd 0xc4 0x95 0x73
0x93 0x55 0x41 0xb6 0x6c 0x21 0x70 0x4f
0x98 0x9a 0x63 0x9c 0x18 0x77 0xdd 0xe3
0x8a 0xb1 0x7f 0xee 0xe5 0xad 0x05 0xa0
0xd0 0xdf 0x2f 0x0c 0x29 0x66 0xd6 0xe8
Homework 7 -- Answer 4

First step by hand.

1. \(i = 65\), \(S[65] = 0xae = 174\)
2. \(j = 245 + 174 = 419 = 163 \mod 256\)
3. Swap \(S[65] = 0xae\) and \(S[163] = 0xdc\)
4. \(S[65] + S[163] = 174 + 220 = 138 \mod 256\)
5. Output \(S[138] = 0xd1\)

There are about 10 operations (a closer estimate is about 16 operations) for each byte of output, so each byte takes about .01 microseconds for an output speed of 100 MB/sec. Using the more accurate .016 microseconds/byte, the speed is 60 MB/sec, which is amazingly close to benchmark result in Lecture 4.

\[\text{Iteration}\]

\[
\begin{align*}
1: & 0x65, j: 0x7d, \text{ out: } 0xbe \\
2: & 0x08, j: 0x43, \text{ out: } 0xd3 \\
3: & 0x67, j: 0x77, \text{ out: } 0x88 \\
4: & 0x65, j: 0x63, \text{ out: } 0xd1
\end{align*}
\]
Homework 7 – Answer 5

Let $\text{HMAC}_K(x) = \text{SHA-1}(k \oplus \text{outputpad}) || \text{SHA-1}(k \oplus \text{inputpad}) || x$.

Let's say we want to transmit yes/no answers in the stream $b_1, b_2, \ldots$ with input $a_1, a_2, \ldots$ where say $a_i = 1$ for a yes answer and 0 for a no answer.

$b_i = \text{HMAC}_K(i + a_i)$. The sequence $b_i$ satisfies the requirements. This assumes the “question number” is not subject to attack by an adversary. If you’re worried about question substitution, you should also hash in the question.