MATH 314 - Class Notes

10/19/2015

Scribe: Noah Day

Summary: Today in class we discussed how to encrypt and decrypt DES, as well as how to use SageMathCloud to solve for the key.

Notes:

Differential Cryptonalysis:

3 Round DES
The goal: To find the key
Pick a plaintext (L_0, R_0)
Run SDES to get the output (L_3, R_3)
Pick a second plaintext (L_0^*, R_0^*) to get (L_3^*, R_3^*)
In the last class (10/14) it was shown that $(R_3^*, R_3^*) \oplus (L_0^*, L_0^*) = f(L_3, K_3) \oplus f(L_3^*, K_3)$
*Note that $(R_3^*, R_3^*) \oplus (L_0^*, L_0^*)$ is everything that is currently known

First, take $f(L_3, K_3)$ and run it through the expander
$E(L_3) \oplus K_3$ is then split into s_1 and s_2, which then come together to make $f(L_3, K_3)$ Then take $f(L_3^*, K_3)$ and run it through the expander
$E(L_3^*) \oplus K_3$ is then split into s_1 and s_2, which then come together to make $f(L_3^*, K_3)$ We know that the XOR of these inputs ends up being $E(L_3) \oplus E((L_3^*) \oplus K_3) = E(L_3) \oplus E(L_3^*)$
We also notice that the 1st 4 bits of the input turn into the 1st 3 bits of the ouput
We now focus on s_1, and search over all the possible pairs of inputs to s_1 that produce the pairs for $E(L_3) \oplus E(L_3^*)$

Example:
$L_3 =$ 101110
$L_3^* =$ 000010
$(R_3, R_3^*) \oplus (L_0, L_0^*) =$ 100001
$E(L_3) =$10111110
$E(L_3^*) =$10111110
The inputs to the s_1 boxes must xor to 1011, while the outputs must xor to 100
The possible pairs that xor to 1011 are (0000, 1011) and (0001, 1010)
Take the pair (0000, 1011) and put them both into s_1
When 0000 is put into s_1 the result is then 001
When 1011 is put into s_1 the result is then 010
Add 001 and 010 together to get 011, which is not equal to 100, and the second pair of numbers must be tried
Take the pair (0001, 1010) and put them both into s_1
When 0001 is put into s_1 the result is then 010
When 1010 is put into s_1 the result is then 110
Add 001 and 010 together to get 100, which is the correct value that is needed

This process is repeated for all possible pairs, to find that the only good pairs are (`0001`, `1010`) and (`1010`, `0001`)
This means that the first 4 bits of $E(L_3) \oplus K_3$ are either `1010` or `0001`
When this no longer yields any results, then try a different plaintext and continue through the process

SageMathCloud:

This code is from the handout on SMC
The function `randomKey()` generates and defines a random key to work with
First try a plaintext of all 0's
Then run 3 rounds of SDES with the key and encrypt it
Pick a second plaintext, with the second half the exact same as the first plaintext, and then run that through the same process
Split that into (L_0, R_0) and (L_3, R_3)
Then expand (L_3) and (L_3^*) and combine them
Search over pairs that would sum up to the xor by using the function's first 4 bits
Find the pairs and run the process again
Leave the last 6 bits the same, and run again until there is a match, and then repeat the entire process over with s_2

DES:

Data Encryption Standard
It is made of 64-bit blocks, with a 56 bit key
There are three stages of doing DES:

1. Create an intial 64 bit number (m) message and apply an initial permutation of these 64 bit. The IP is public knowledge.

2. Split $m_0 = IP(m)$ into L_0R_0 (32 bits each)
   Repeat 16 times:
   $L_i = R_i - 1$
   $R_i = L_i - 1 \oplus f(R_i - 1, K_i)$

3. Switch the left and right to obtain $R_{16}L_{16}$ and apply $IP^{-1}(R_{16}, L_{16}) = c(\textit{ciphertext})$

DES F:

1. Use the expander R_{i-1} (which is 32 bits) into a 48 bit string

2. xor $E(R_{i-1})$ with K_i

3. Break into 8 different pieces (each consisting of 6 bits) which are fed into 8 different s boxes (concentrate outputs)

Heart of the security of DES is the choice of sboxes

- chosen to be nonlinear

- chosen to avoid all sorts of potential paterns

DES is secure against differential cryptonanlysis, as it takes just as long as brute force
If DES was 15$\geq$ rounds then cryptanalysis would work
DES was designed very securely (however, the only flaw is that it has a short key length)
In the 1990's and early 2000's, people built special computers to brute force DES, and could break a DES key in about a day
People also wanted to make DES stronger, and wanted to possibly encrypt a message multiple times with DES
Therefore a double DES is equal to $E_{k2}(E_k,(P))$
Is that possible?

- Let E_k be the encryption function for DES with the key being k
- Affline's double encryption is the same as encrypting with a 3rd different key, which is similar to every other hill cipher
- For most keys K_1 and K_2, $E_{k2}{*}E_{k1} \neq E_{k3}$

Double DES is vulnerable to a meet in the middle attack (discussed in the next set of notes)