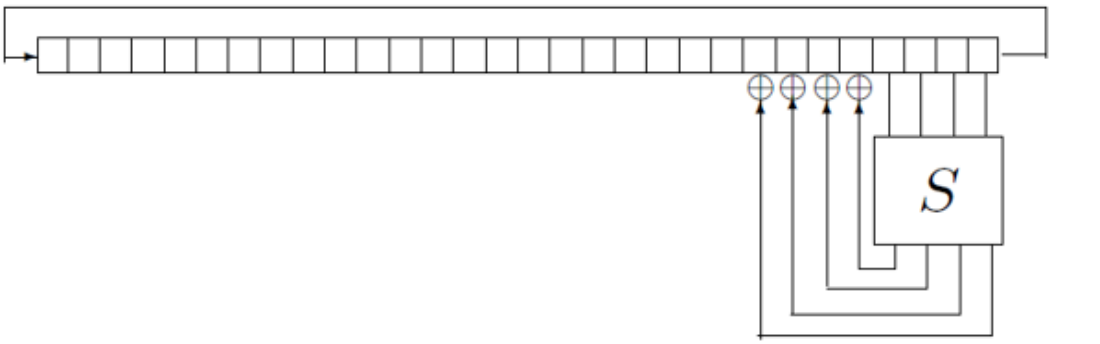
Cryptography-Final

3180103772 张溢弛

1. **You are operating an intercept radar that is posing as a fire-control radar, challenging aircraft to identify themselves. You send challenges to hostile aircraft and analyze their responses. The aircraft are using a 32-bit block cryptosystem of the following form in which the round function picks four bits, looks them up in an s-box feeds them back into the next four bits over and shifts circularly by four bits. The system is keyed by the contents of the s-box. You are trying to break the system so that your own aircraft can respond to challenges from the opponent.(40 points total)**



Q1: If the s-box need not be invertible, how many bits of information does it contain? (10 points)

Q2: If the s-box is invertible but otherwise selected randomly--entries selected at random without replacement--how many bits of information does it contain? (10 points)

Q3: Since the system shifts 4-bits per clock tick, it takes eight clock ticks to produce on full-round of encryption. Let us assume that the system runs for eight full rounds or 64 clock ticks. Describe how you would break the system (recover the s-box) if you could make an arbitrary number of requests to the aircraft to encrypt challenges of your choosing. (20 points)

Solution:

Q1:

- **[Lemma]** An s-box is a list of n-items each m-bits long. If the n-items are all **independent** it contains $n \times m$ bits of information. (Told by Diffie from E-mail)
- If the s-box is not invertible, it means that the feedback function is not invertible. The function is a 4-bit to 4-bit mapping so input has 16 different kinds and the s-box has 16 results (which can be the same) of 16 inputs. To be not invertible, the 16 results in the s-box must have some repeats otherwise it is a one-to-one mapping and invertible. But the number of results in the S-box is still 16 so that the whole s-box contains 64 bits of information totally.

- **Another solution:** we could solve the problem using the Shallnon comentropy, which is $H(X) = -\sum_{x \in X} p(x) \log_2 p(x)$, and if the S-box need not to be invertible, we have 16^{16} different kinds of S-box so $p(x) = \frac{1}{16^{16}}$, so $H(X) = -16^{16} \times \frac{1}{16^{16}} \times \log_2 16^{16} = 64$ bits. So the result is 64 bits.

Q2:

- This time the s-box is invertible so it must be a one-to-one mapping it means each 4-bit binary string in the 16 probable inputs can have one a specific result. **It has 16! different one-to-one mapping** from the 4-bit binary input to the result. And in each input, whether how it is selected, it must have 16 different answers, so the result is 64 bits of information in the S-box in total, but the n-items are not independent because they must be different from each other.
- Therefore, using the Shallnon comentropy, we find that $H(X) = -16! \times \frac{1}{16!} \times \log_2 16! = \log_2 16!$ And an approximation of the result is 45 bits if the S-box is invertible ans selected randomly

Q3:

- **[Principle of the cryptisystem]** First we should know that each time the function f will produce a tuple with 4 element by the input (x_0, x_1, x_2, x_3), we could mark the result as $f(x_0, x_1, x_2, x_3)$. In each clock tick, we will do such a calculation, that is $(x'_4, x'_5, x'_6, x'_7) = (x_4, x_5, x_6, x_7) \oplus f(x_0, x_1, x_2, x_3)$ and then shifts circularly by four bits so after a full round each four-bits group will be $(x'_n, x'_{n+1}, x'_{n+2}, x'_{n+3}) = (x_n, x_{n+1}, x_{n+2}, x_{n+3}) \oplus f(x_0, x_1, x_2, x_3)$ while the tuples (x0,x1,x2,x3) is still changing after each clock tick **because of the circular shift**, or we could write the expression in such a form, that is $(x'_{n+4}, x'_{n+5}, x'_{n+6}, x'_{n+7}) = (x_{n+4}, x_{n+5}, x_{n+6}, x_{n+7}) \oplus f(x'_n, x'_{n+1}, x'_{n+2}, x'_{n+3})$ so it's really a complex encryption algorithm.
- In this problem we could only get the result of 8 full rounds in this cryptosystem with any 32-bit input and we should break the system and recover the S-box. We should also know that it's a **DES-like cryptography** and we could use some method just like how we break the DES algorithm.
- We could divide the 32-bits into 8 blocks and each block contains 4 bits, called L0 to L7
- So the methods to break this system can be
    - **A normal method:exhaustion with backtracking.** The most clumsy method I firstly work out is to use the method of exhaustion and try each possible input of 32-bits (2^{32} in total) and get the each result after 8 full rounds. Besides, the s-box has 16^{16} different choices(it's obvious), and we could use a **backtracking** to check whether each of the possible result of s-box is right or not compared with the 2^{32} outputs, the s-box that matches each outputs correctly after calculate the 32-bit input in 8 full rounds will be a right answer. The pseudo code of this method is

```
function check():
    Generate all the S-boxs;
    Generate all the possible inputs their outputs;
    for each s-box:
        for each input:
            temp_output=f(s-box,input)
            if temp_output==output of this input:
                return s-box

function f(s-box,input):
    execute the encryption algorithm of the cryptosystemusing the s-box and the 32-bit input in 8 full rounds.
    return output
```

- **A better solution:** If the cryptosystem doesn't have XOR calculation in the function feedback, we will get $f^{57}(L0)$ to $f^{64}(L0)$ as the result of 8 full rounds running of the system, each one contains 4 bits. That way, we could let L0=0000 and we will finally find a $f^{i}(L0) = 0000$ after times of interations, that will be the cycle period of the algorithm and the cryptosystem will be broken up using the cycle period. Then we can recover the remaining blocks in the s-box using an exhaustion way. However, it is an idealized method.
- **Differential cryptanalysis**: is a general form of cryptanalysis applicable primarily to block ciphers, but also to stream ciphers and cryptographic hash functions. In the broadest sense, it is the study of how differences in information input can affect the resultant difference at the output. In the case of a block cipher, it refers to a set of techniques for tracing differences through the network of transformation, discovering where the cipher exhibits non-random behavior, and exploiting such properties to recover the secret key (cryptography key). To solve the problem, we could catch the output values and make a **differential table** of the unknow S-box, and then we can recover the S-box from the differential table.
- **Linear cryptanalysis**: It is a most powerful way to attack the DES-like cryptography. linear cryptanalysis is a general form of cryptanalysis based on finding affine approximations to the action of a cipher. Attacks have been developed for block ciphers and stream ciphers. Linear cryptanalysis is one of the two most widely used attacks on block ciphers; the other being differential cryptanalysis.
- **Side-channel attack**: a side-channel attack is any attack based on information gained from the implementation of a cryptosystem, rather than weaknesses in the implemented encryption algorithm itself (e.g. cryptanalysis and software bugs). Timing information, power consumption, electromagnetic leaks or even sound can provide an extra source of information, which can be exploited. In this problem we could also use side-channel attack to break the cryptosystem and recover the S-box.

2. **A sequence of bits is balanced if it has the same number of 0s and 1s. A function is called balanced if the sequence of it outputs over all inputs is balanced.**

   Demonstrate that every column in the table of an invertible n-bit to n-bit function must be balanced. Show that linear functions (other than 0) are balanced.(30 points)

Solution:

- Demonstrate that every column in the table of an invertible n-bit to n-bit function must be balanced.
  - A n-bit to n-bit boolean function is invertible if and only if it is a one-to-one mapping，which means a result n-bit is from only one n-bit input and one n-bit input has only one n-bit input.
  - As an n-bit to n-bit boolean function, the number of distinct invertible boolean function is thus $2^n!$ correspongding to the $2^n!$ ways the truth table can be completed with 2^n distinct entries of F.
  - Since the function table for F is a complete listing of the 2^n n-bit binary string, each output variable f_i of function F has an equal number of 0's and 1's, which means f_i is a balances boolean function. In another word, each column in the function table must be balanced.
  - In a n-bit to n-bit invertible function there are 2^n possible different input and 2^n different output so in each column of the function table, there are 2^{n-1} 0s and 2^{n-1} 1s **because the 2^n binary strings consists of every condition of n-bits**, so it's symmetric and have same number of 0s and 1s.
- Show that linear functions (other than 0) are balanced
  - Method 1: using the **probablity theory**
    - **Introduce**: A balanced boolean function is a function whose output yields as many 0s as 1s over its input and it means that for a random input string of inputs, the probability of getting 1 as result is $\frac{1}{2}$,such a linear function could be feedback functions(used in LFSR or NLFSR) or functions that calculate the **exclusive**(XOR) of some of the inputs.
    - A linear function from n-bit to n-bit can be represented as $f(x_1, x_2, \ldots, x_n) = \begin{bmatrix} f_1(x_1, x_2, \ldots, x_n) \\ \ldots \\ f_n(x_1, x_2, \ldots, x_n) \end{bmatrix} = (y_1, y_2, \ldots, y_n)$ where $f_i(x_1, x_2, \ldots, x_n)$ are all linear boolean functions that calculate the exclusive of some of the input that is $f_i(x_1, x_2, \ldots, x_n) = \sum_{j=1}^{n} k_{ij}x_j (mod 2)$ where k_{ij} can be 0 or 1 to simplify the calculate (because of modulu 2, we can do $k_{ij} = k_{ij}2$ when $k_{ij} >= 2$)in the probability $\frac{1}{2}$ and x_j is a random input which can be 0 and 1(but we should know that f_i can't be zero function), what more the result will be in modulu 2 to make sure the result is still 0 or 1. We can write the linear function in another formula that $f_i(x_1, x_2, \ldots, x_n) = x_{i_1} \oplus x_{i_2} \oplus \ldots \oplus x_{i_k}$ here $\oplus$ means addition in 0 and 1 without carry. So the probability of the result is $\frac{1}{2}$ 0 and $\frac{1}{2}$ 1.

- We can proof this conclusion using the probability thoery: $E(f_i(x_1, x_2, \ldots, x_n)) = E(\sum_{j=1}^{n} k_{ij}x_j) = (\sum_{j=1}^{n} k_{ij}E(x_j)) = \frac{\sum_{j=1}^{n} k_{ij}}{2} = \frac{1}{2}$ and k_{ij} can be any positive integers, so the f_i **could be 0 or 1 in the same probability.**
- So in a macroscopically condition, the result of the whole linear function will have same number of 0s and 1s. That means the linear functions(other than 0) are balanced.

◦ Method 2: using **Mathematical induction**

- Focus on the dimension of function f start from 1(zero is the special condition of dimension 0)
- When k=1, it's obviously balanced because $f^1(x) = x0$ and the possible of the function has the same number of 0s and 1s.
- Then we assum that when k=n, this conclusion is right.
- When k=n+1, the linear function f is $f^{n+1}(x) = f^n(x) \oplus x_{n+1}$, for any possible input x=(x1,x2,...,xn) that make $f^n(x) = 0$, if $x_{n+1} = 0$ the result is 0 otherwise 1, for any possible input x=(x1,x2,...,xn) that make $f^n(x) = 1$, if $x_{n+1} = 0$ the result is 1 otherwise 0, all the possible value of x is in the same probablity. So the number of 0s and 1s in the all possible output still has same number of 0s and 1s.
- Using the Mathematical induction, we could know that for any linear function with dimension no less than 1 is balanced.

3. **Calculate bulges of S-box 5, Row 3, Column 2.(30 points)**

S-BOX 5

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
00	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
01	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
10	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

Solution

- **Definition 1**: In cryptography, an **S-box** (substitution-box) is a basic component of symmetric key algorithms which performs substitution. S-Box5 is used in DES algorithm, mapping 6-bit input to 4-bit output. Given a 6-bit input, the 4-bit output is found by selecting the row using the outer two bits (the first and last bits), and the column using the inner four bits.
- **Definition2**: Bugles: bugle is a relative concept of index of coincidence, but $bulge = \frac{agreement - disagreement}{agreement + disagreement}$ where agreement means the number of the index of

two strings which has the same value.

- We could also write the bulge in a Fourier transform $b_l(f) = \frac{1}{2^n}\sum_{i=0}^{2^n-1}(-1)^{f(i)\oplus l(i)}$
- The third row of S-Box 5 is

w	x	y	z	out	origin number
0	0	0	0	0100	4
0	0	0	1	0010	2
0	0	1	0	0001	1
0	0	1	1	1011	11
0	1	0	0	1010	10
0	1	0	1	1101	13
0	1	1	0	0111	7
0	1	1	1	1000	8
1	0	0	0	1111	15
1	0	0	1	1001	9
1	0	1	0	1100	12
1	0	1	1	0101	5
1	1	0	0	0110	6
1	1	0	1	0011	3
1	1	1	0	0000	0
1	1	1	1	1110	14

- So the Row 3 column 2 of the S-Box 5 is

w	x	y	z	out-2
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

- So the bulge of Row 3 column 2 of the S-Box 5 is (using 16 linear functions)

variables		Agree	Disagree	bulges
none	0000000000000000	8	8	0
z	0101010101010101	6	10	-0.25
y	0011001100110011	8	8	0
y+z	0110011001100110	6	10	-0.25
x	0000111100001111	8	8	0
x+z	0101101001011010	6	10	-0.25
x+y	0011110000111100	8	8	0
x+y+z	0110100101101001	6	10	-0.25
w	0000000011111111	10	6	0.25
w+z	0101010110101010	8	8	0

variables		Agree	Disagree	bulges
w+y	0011001111001100	6	10	-0.25
w+y+z	0110011010011001	12	4	0.5
w+x	0000111111110000	10	6	0.25
w+x+z	0101101010100101	8	8	0
w+x+y	0011110011000011	6	10	-0.25
w+x+y+z	0110100110010110	4	12	-0.5