CS641A Mid Sem

Gargi Sarkar, Anindya Ganguly, Utkarsh Srivastava

TOTAL POINTS

37 / 50

QUESTION 1

1 DES algorithm 18 / 25

✓ + 5 pts Discuss differentials at $$L_0R_0$$, $$L_1R_1$$, $$L_2R_2$$, and S-Boxes for the first 2 rounds.

✓ + 10 pts Use the new S-Box design to conclude S-Box output XOR $$0000$$ occurs with probability $$\frac{32}{64}=\frac{1}{2}$$ for certain differentials

✓ + 5 pts Mention the 2-round iterative characteristics with probability $$\frac{1}{2}$$

✓ + 3 pts Use the above 2-round iterative characteristics to form 14-round characteristics with probability $$\frac{1}{128}$$

+ 2 pts Discuss no. of pairs required to break this 16-round DES using $$p=\frac{1}{128}$$

+ 0 pts Wrong or NA

- 5 Point adjustment

S-Box analysis is shown for input differential 000010, not for 000100.

QUESTION 2

2 Diffie Hellman 19 / 25

✓ + 3 pts State the existence of _disjoint cycles_ for a permutation $$p \in S_n$$

✓ + 5 pts Describe a method to efficiently compute the _disjoint cycles_ of a permutation $$p$$

✓ + 2 pts State _disjoint cycles_ of the pair $$(g, g^c)$$ and/or $$(g, g^d)$$

✓ + 10 pts Describe how to form a system of linear modular equations from the _disjoint cycles_ of $$(g, g^c)$$ and/or $$(g, g^d)$$ to compute $$c$$ and/or $$d$$

✓ - 3 pts No explanation of why $$c \equiv r_i \, (mod \, l_i)$$ follows after finding the differences in positions

✓ + 2 pts State how to compute $$c$$ or $$d$$ from the above equations

✓ + 3 pts Correctness of computed $$c$$ or $$d$$. Use the fact that the order $$l$$ of $$g$$ is $$lcm$$ of the order of its _disjoint cycles_

+ 0 pts Incorrect or NA

- 3 Point adjustment

Vague explanation of algorithm to compute disjoint cycles

QUESTION 3

3 References 0 / 0

✓ + 0 pts Correct

CS641
Modern Cryptology
Indian Institute of Technology, Kanpur

Mid Semester Examination

Group Name: Enciphered
Anindya Ganguly (21111261),
Gargi Sarkar (21111263),
Utkarsh Srivastava (21111063)

Submission Deadline:
March 3, 2022, 23:55hrs

Question 1

Consider a variant of DES algorithm in which all the S-boxes are replaced. The new S-boxes are all identical and defined as follows.

> Let $b_1, b_2, \cdots, b_6$ represent the six input bits to an S-box. Its output is $b_1 \oplus (b_2 \cdot b_3 \cdot b_4), (b_3 \cdot b_4 \cdot b_5) \oplus b_6, b_1 \oplus (b_4 \cdot b_5 \cdot b_2), (b_5 \cdot b_2 \cdot b_3) \oplus b_6$.

Here '$\oplus$' is bitwise XOR operation, and '$\cdot$' is bitwise multiplication. Design an algorithm to break 16-round DES with new S-boxes as efficiently as possible.

Solution

We will break this 16-round DES using differential cryptanalysis and new S-Boxes. Differential cryptanalysis is a chosen-plaintext attack. We assume that an attacker has a large number of tuples (x, x , y, y), where the x-or value x' = x xis fixed. The plaintext elements (i.e., x and x) are encrypted using the same unknown key, K, yielding the cipher texts y and y, respectively. For each of these tuples, we will begin to decrypt the ciphertexts y and y ,using all possible candidate keys for the last round of the cipher. For each candidate key, we compute the values of certain state bits, and determine if their x-or has a certain value (namely, the most likely value for the given input x-or). Whenever it does, we increment a counter corresponding to the particular candidate key. At the end of this

	0000	**0001**	**0010**	**0011**	**0100**	**0101**	**0110**	**0111**	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**
00	0000	0000	0000	0000	0000	0000	0000	0100	0000	0000	0000	0010	0000	0001	1000	1111
01	0101	0101	0101	0101	0101	0101	0101	0001	0101	0101	0101	0010	0101	0100	1101	1010
10	1010	1010	1010	1010	1010	1010	1010	1110	1010	1010	1010	1000	1010	1011	0010	0101
11	1111	1111	1111	1111	1111	1111	1111	1011	1111	1111	1111	1100	1111	1110	0111	0000

Table 1: S-box

process, we hope that the candidate key that has the highest frequency count contains the correct values for these key bits.

The manually computed s box is presented in table1, according to the information given in the inquiry for the S BOX.

1. Predicting the XOR of round n-2 output with high likelihood.

The values of L_n, R and R_{n-1} are known. We don't know the two values or their XOR at L_{n-1} which will be the same as the R_{n-2}. If we can somehow find the XOR of output of round n - 2 i.e. R_{n-2} with as high probability as possible, we can easily figure out the key for nth round (K_n).

We know that the new S-Boxes are non-linear, which is why knowing the output XOR to an S-Box is problematic because one cannot remark on the input XOR values. However, if we look at the nature of these new S-boxes, we can extrapolate that if the XOR of the two inputs to either of the S-Boxes (which are all similar) is 000010, then 32 pairings out of 64 possible pairs have the XOR of the output as 000010.

When we take random input pairings with 000010 as the input XOR to any one S-Box, we may anticipate the output XOR to be 1110 with a chance of 32/64 = 1/2. We can anticipate the XOR of the output of this S-Box with probability 1/2 if the input XOR to remaining S-Boxes is all zeroes. [1]

Consider a 2-round DES with XOR at R_0 of 0000 0000 (Hexadecimal representation of 32-bit values) and XOR at L_0 of 0000 0002 (Hexadecimal representation of 32-bit values). Then, since XOR of the right half is all zeroes, XOR at L_1 will be 0000 0000 and XOR at R_1 will be 0000 0002.

Expansion box will now receive the output XOR at R_1 (0000 0002), which will yield the 48-bit value 0000 0000 0004. (in Hexadecimal). We can see that if we divide this value into 8 groups of size 6, the first 7 S-boxes will have all zeros and the last S-box (S8) will have the value 000100. (binary). This S8 box will output 0000 (binary) with probability 1/2, and the subsequent S-boxes will likewise output 0000 (binary), resulting in the final

Figure 1: Sample Difference Pairs of the S-Box [1]

X	Y	ΔY		
		$\Delta X = 1011$	$\Delta X = 1000$	$\Delta X = 0100$
0000	1110	0010	1101	1100
0001	0100	0010	1110	1011
0010	1101	0111	0101	0110
0011	0001	0010	1011	1001
0100	0010	0101	0111	1100
0101	1111	1111	0110	1011
0110	1011	0010	1011	0110
0111	1000	1101	1111	1001
1000	0011	0010	1101	0110
1001	1010	0111	1110	0011
1010	0110	0010	0101	0110
1011	1100	0010	1011	1011
1100	0101	1101	0111	0110
1101	1001	0010	0110	0011
1110	0000	1111	1011	0110
1111	0111	0101	1111	1011

Figure 2: Difference distribution table [2]

		Output Difference															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Input Difference	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	2	0	0	0	2	0	2	4	0	4	2	0	0
	2	0	0	0	2	0	6	2	2	0	2	0	0	0	0	2	0
	3	0	0	2	0	2	0	0	0	0	4	2	0	2	0	0	4
	4	0	0	0	2	0	0	6	0	0	2	0	4	2	0	0	0
	5	0	4	0	0	0	2	2	0	0	0	4	0	2	0	0	2
	6	0	0	0	4	0	4	0	0	0	0	0	0	2	2	2	2
	7	0	0	2	2	2	0	2	0	0	2	2	0	0	0	0	4
	8	0	0	0	0	0	0	2	2	0	0	0	4	0	4	2	2
	9	0	2	0	0	2	0	0	4	2	0	2	2	2	0	0	0
	A	0	2	2	0	0	0	0	0	6	0	0	2	0	0	4	0
	B	0	0	8	0	0	2	0	2	0	0	0	0	0	2	0	2
	C	0	2	0	0	2	2	2	0	0	0	0	2	0	6	0	0
	D	0	4	0	0	0	0	0	4	2	0	2	0	2	0	2	0
	E	0	0	2	4	2	0	0	0	6	0	0	0	0	0	2	0
	F	0	2	0	0	6	0	0	0	0	4	0	2	0	0	2	0

output after permutation being 0000 0000 at R_2 and 0000 0002 at L_2.
This forms the basis for 2-round iterative characteristics with x_0 = 0000 0002, y_0 = 0000 0000, p_1 = 1, x_1 = 0000 0000, y_1 = 0000 0002, p_2 = 1/2, x_2 = 0000 0002 and y_2 = 0000 0000. We get the following 2-round characteristics.

$$(\bar{0}0002, \bar{0}\bar{0}, 1, \bar{0}\bar{0}, \bar{0}0002, \frac{1}{2}, \bar{0}0002, \bar{0}\bar{0})$$

The probability of the above 2- round characteristics is $\frac{1}{2}$.

We will concatenate this 7 times to get 7*2 i.e. 14-round characteristic. The probability of 14-round characteristic will be

$$(\frac{1}{2})^7$$

=

$$\frac{1}{128}$$

which is far better than the brute force. Thus, using this 2-round characteristic 7 times, we will get 14-round characteristic which can be used to find the key k_n.

2.16th round output XOR prediction
We know the XOR at R_{n-2} i.e. R_{14} with probability $\frac{1}{128}$ which is same as L_{15}. We also know the value at R_{16}, hence we know the output XOR at permutation box which in turn gives us the XOR of output of S-boxes for 16th round.

3. Extraction of last round key (k_n)
Define

$$X_i = \{(\beta, \beta') | \beta \oplus \beta' = \beta_i \oplus \beta_i' and S_i(\beta) \oplus S_i(\beta') = \gamma\}$$

pair $(\beta_i, \beta_i') \in X_i$ whenever our guess for $\gamma_i \oplus \gamma_i' = \gamma$ is correct . which happens with probability $\frac{14}{64}$

Define

$$K_i = \{k | \alpha_i \oplus k = \beta and (\gamma, \gamma') \in X_i for some \beta'\}$$

Since, $(\beta_i, \beta_i') \in X_i$ with probabilty $\geq \frac{14}{64}$, we have $k_(4, i) \in K_i$ with probability $\geq \frac{14}{64}$.

Let $E(R_3) = \alpha_1\alpha_2...\alpha_8$ and $E(R'_3)) = \alpha'_1\alpha'_2...\alpha'_8$ with $|\alpha_i| = 6 = |\alpha'_i|$

R_3 and R'_3 are right-halves of output of third round on the plaintexts L_0R_0 and $L'_0R'_0$ = L'_0R_0

Let $\beta_i = \alpha_i \oplus k_{(4,i)}$ and $\beta'_i = \alpha'_i \oplus k_{(4,i)}$, $|\beta_i| = 6 = |\beta'_i|$

$k_4 = k_{(4,1)}k_{(4,2)}...k_{(4,8)}$.

Let $\gamma_i = S_i(\beta_i)$ and $\gamma'_i = S_i(\beta'_i)$,$|\gamma_i| = 4 = |\gamma'_i|$.

We Know α_i, α'_i and $\beta_i \oplus \beta'_i = \alpha_i \oplus \alpha'_i$.

we also know a value γ such $\gamma_i \oplus \gamma'_i = \gamma$ with probability $\frac{14}{64}$

We have $|K_i| = |X_i|$ since α_i and $\beta \oplus \beta'$ is fixed for (β,β') $\in X_i$.

Therefore, $|K_i| \leq 16$ as per property of S-boxes.

We cannot use the method for three rounds here: If we compute another K'_i and take its intersection with K_i, $K_{(4,i)}$ may get dropped out since it is not guaranteed to be present in both.

1 DES algorithm 18 / 25

✓ **+ 5 pts** **Discuss differentials at $$L_0R_0$$, $$L_1R_1$$, $$L_2R_2$$, and S-Boxes for the first 2 rounds.**

✓ **+ 10 pts** **Use the new S-Box design to conclude S-Box output XOR $$0000$$ occurs with probability $$\frac{32}{64}=\frac{1}{2}$$ for certain differentials**

✓ **+ 5 pts** **Mention the 2-round iterative characteristics with probability $$\frac{1}{2}$$**

✓ **+ 3 pts** **Use the above 2-round iterative characteristics to form 14-round characteristics with probability $$\frac{1}{128}$$**

+ **2 pts** Discuss no. of pairs required to break this 16-round DES using $$p=\frac{1}{128}$$

+ **0 pts** Wrong or NA

- 5 Point adjustment

S-Box analysis is shown for input differential 000010, not for 000100.

Question 2

Suppose Anubha and Braj decide to do key-exchange using Diffie-Hellman scheme except for the choice of group used. Instead of using F_p^* as in Diffie-Hellman, they use S_n, the group of permutations of numbers in the range $[1, n]$. It is well-known that $|S| = n!$ and therefore, even for $n = 100$, the group has very large size. The key-exchange happens as follows:

> An element $g \in S_n$ is chosen such that g has large order, say l. Anubha randomly chooses a random number $c \in [1, l-1]$, and sends g^c to Braj. Braj choses another random number $d \in [1, l-1]$ and sends g^d to Anubha. Anubha computes $k = (g^d)^c$ and Braj computes $k = (g^c)^d$.

Show that an attacker Ela can compute the key k efficiently.

Solution

General Idea

Here, we observe that the underlying assumption for the DH scheme over symmetric group is a discrete logarithm problem over symmetric group S_n. Suppose $\mathcal{O}$ denotes the oracle for computing the $h = g^\alpha$ for a given g and α, and $\mathcal{O}^{-1}$ denotes the oracle for computing the discrete log, that is for a given h and g, it will compute α.

In the problem Anubha (A) and Braj (B) run the oracle $\mathcal{O}$ privately to compute g^c and g^d respectively. Our task is to design a oracle $\mathcal{O}^{-1}$which will runs in polynomial time and computationally feasible to retrieve c and d. After computing c and d, we will invoke $((g, c \cdot d))$ to the oracle $\mathcal{O}$ for computing $g^{c \cdot d}$. Hence, we able to solve the question.

The next section roughly sketched the idea for constructing $\mathcal{O}^{-1}$. In addition complexity analysis of each steps has been analyzed. The discussion ends with a mathematical justification of our claims.

Cryptanalysis

As we know that g and h are publicly available and goal is to compute α. Any elements $e \in S_n$ can represented via cycle notation or a list of images $[e(1), e(2), \cdots, e(n)]$.

Phase: 1 Suppose g and h can be decomposed into disjoint cycles

$$g = g_1 \circ g_2 \circ \cdots \circ g_r$$

$$h = h_1 \circ h_2 \circ \cdots \circ h_s$$

where $\circ$ denotes the composition operation. Note that each every $i \in \{1, 2, \cdots, n\}$ lies in exactly one cycle.

- **Time Complexity:** The decomposition techniques requires $O(n)$-time. Without loss of any generality we are assuming that g acts on $1, 2, \cdots, n$. Let us start from $i = 1$, do a look-up and compute the image of i under g. Now, when image is equal to i, stop the cycle and do $i + 1$, and for image not equal to i, append $g(i)$ at the end of the cycle and repeat the process for index i. So at most n look up is required.

Phase: 2 Maintaining arrays G and H.

- The i-th index of $G[i]$ contains
  1. the index j of the cycle g_j having i
  2. the location of i within this cycle $(1 \leq i \leq n)$

  Basically $G[i]$ can be consider as a tuple $(j, p(i))$ which illustrates that element i appears in cycle g_j at a position $p(i)$.
- In similar fashion $H[i]$ will be constructed. Like above $H[i] = (k, p(i))$ means that element i appears in cycle h_k at location $p(i)$. So $H[i]$ contains:
  1. the index k of the cycle h_k having i
  2. the location of i within this cycle $(1 \leq i \leq n)$

- **Time Complexity:** The arrays G and H each of them have $2n$ integers. Clearly construction of G and H require $O(n)$ time.

Phase: 3 Again maintain two array called $X[k]$, $Y[k]$, where $X[k]$ has the first element of each cycle h_k of h and $Y[k]$ has the second element of each cycle h_k of h. Note that $X[k] =$

$Y[k]$ holds for length-one cycles. Our previously constructed array G helps to find the cycle of g containing $X[k]$ and $Y[k]$ each $k \in [n]$. Use array $Z[k]$ to maintain the difference between their location that means $Z[k] = p(Y[k]) - p(X[k])$ for all $k \in [n]$. Then we calculate the length of the cycle containing the element i and put it in the array W.

- **Time Complexity:**Since g^{α} has at most n-cycle, so size of $X[k]$ and $Y[k]$ is at most n. Clearly $X[k]$ and $Y[k]$ lies to the same cycle $g_{k'}$ of g for some k'. To perform this, needs a look up in array G to identify which cycle of g the value $X[k]$ lies. Thus it requires $O(n)$ look up. Look up the location numbers of $Y[i]$ and $X[i]$ and subtract. This needs $O(n)$ operations and $O(n)$ look up.

Phase: 4 To obtain the value of α, we need to call CRT. Because, right now we got

$$\alpha \equiv Z[i] \mod W[i] \text{ for } 1 \leq i \leq |Z|.$$

So we have at most $|Z$-linear equations, where for any i, j $\gcd(W_i, W_j) \neq 1$

- **Time Complexity:** Here we analyze the time complexity for computing n-modular linear equation. Let us consider first two linear congruence

$$\alpha \equiv Z[1] \mod W[1]$$

$$\alpha \equiv Z[2] \mod W[2].$$

Suppose α_1 be the solution of two linear equations. That means $\alpha \equiv \alpha_1 \mod (lcm(W[1], W[2]))$. Solving these two linear equations use extended Euclidean algorithm, which need $O(\log(W[1]) \cdot \log(W[2]))$ time. That is the best time complexity is $O(\log^2 n)$

Now again

$$\alpha \equiv \alpha_1 \mod (lcm(W[1], W[2]))$$

$$\alpha \equiv Z[3] \mod W[3].$$

Suppose α_2 denote the solution of above two linear equations. Like above, computing α_2 needs $O(\log^2 n)$ time. Thus to solve $|Z| \approx O(n)$ equations, we need to

perform $(n-1)$ times extended Euclidean algorithm. Thus the time complexity

$$O(\sum_{k=1}^{n-1} k \cdot \log^2 n) = O(n^2 \log^2 n).$$

Hence the time complexity for computing α is $O(n^2 \log^2 n)$.

Correctness

As per our construction

$$g = g_1 \circ g_2 \circ \cdots \circ g_r.$$

Also we are assuming that the above algorithm returns $\overline{\alpha}$

$$\overline{\alpha} \equiv Z[i] \mod W[i] \ \ \forall i \in [n]$$

$$\Rightarrow W[i] | \alpha - Z[i] \ \Rightarrow \overline{\alpha} = t_i \cdot W[i] + Z[i] \ \forall i.$$

Also observe that, $ord(g_i) = W[i]$. We use these facts below.

In the correctness part we show that

$$g^{\overline{\alpha}} = h$$

$$g^{\overline{\alpha}} = (g_1 \circ g_2 \circ \cdots \circ g_r)^{\overline{\alpha}}$$

Remember that g_i's are disjoint cycle of g. Thus we can write,

$$g^{\overline{\alpha}} = (g_1^{\overline{\alpha}} \circ g_2^{\overline{\alpha}} \circ \cdots \circ g_r^{\overline{\alpha}})$$

$$= (g_1^{t_1 W[1]+Z[1]} \circ g_2^{t_2 W[2]+Z[2]} \circ \cdots \circ g_r^{t_r W[r]+Z[r]})$$

$$= (g_1^{Z[1]} \circ g_2^{Z[2]} \circ \cdots \circ g_r^{Z[r]}).$$

To establish $g^{\overline{\alpha}} = h$, we initially compute the image $X[i]$ under $g^{\overline{\alpha}}$ and later prove that the image is $Y[i]$ $\forall i$. Recall the array G, assume $G[X[i]] = (l, p(X[i]))$, that is

- the index l of the cycle g_l containing $X[i]$

- the position of $X[i]$ within this cycle ($|X| \leq n$)

Also, it is known that $X[i]$ and $Y[i]$ belong to the same cycle of g. Thus similarly, we can write $G[Y[i]] = (l, p(Y[i]))$, that is

- the index l of the cycle g_l containing $Y[i]$
- the position of $Y[i]$ within this cycle ($|Y| \leq n$).

Now using these two we have

$$(g_1^{Z[1]} \circ g_2^{Z[2]} \circ \cdots \circ g_r^{Z[r]})(X[i]) = g_l^{D[i]}(X[i]).$$

The image of $X[i]$ under $g^{lZ[i]}$ is compared by moving right by $Z[i]$ position inside g_l. Finally $X[i]$ maps terminate to the cycle entry at position $p(X[i]) + p(Y[i]) - p(X[i]) = p(Y[i])$. Let us draw the conclusion, so we finally obtain $g_l^{D[i]}(X[i] = Y[i]$. Therefore, in general we can conclude that, for all i, $g^{\overline{\alpha}} = h$ holds.

2 Diffie Hellman 19 / 25

✓ + 3 pts State the existence of _disjoint cycles_ for a permutation $$p \in S_n$$

✓ + 5 pts Describe a method to efficiently compute the _disjoint cycles_ of a permutation $$p$$

✓ + 2 pts State _disjoint cycles_ of the pair $$(g, g^c)$$ and/or $$(g, g^d)$$

✓ + 10 pts Describe how to form a system of linear modular equations from the _disjoint cycles_ of $$(g, g^c)$$ and/or $$(g, g^d)$$ to compute $$c$$ and/or $$d$$

✓ - 3 pts No explanation of why $$c \equiv r_i \, (mod \, l_i)$$ follows after finding the differences in positions

✓ + 2 pts State how to compute $$c$$ or $$d$$ from the above equations

✓ + 3 pts Correctness of computed $$c$$ or $$d$$. Use the fact that the order $$l$$ of $$g$$ is $$lcm$$ of the order of its _disjoint cycles_

+ 0 pts Incorrect or NA

- 3 Point adjustment

Vague explanation of algorithm to compute disjoint cycles

References

[1] H. M. Heys, A tutorial on linear and differential cryptanalysis, Cryptologia 26 (3) (2002) 189–221.

[2] M. I. González Vasco, A. Robinson, R. Steinwandt, Cryptanalysis of a proposal based on the discrete logarithm problem inside sn, Cryptography 2 (3) (2018) 16.

3 References 0 / 0

✓ + 0 pts Correct