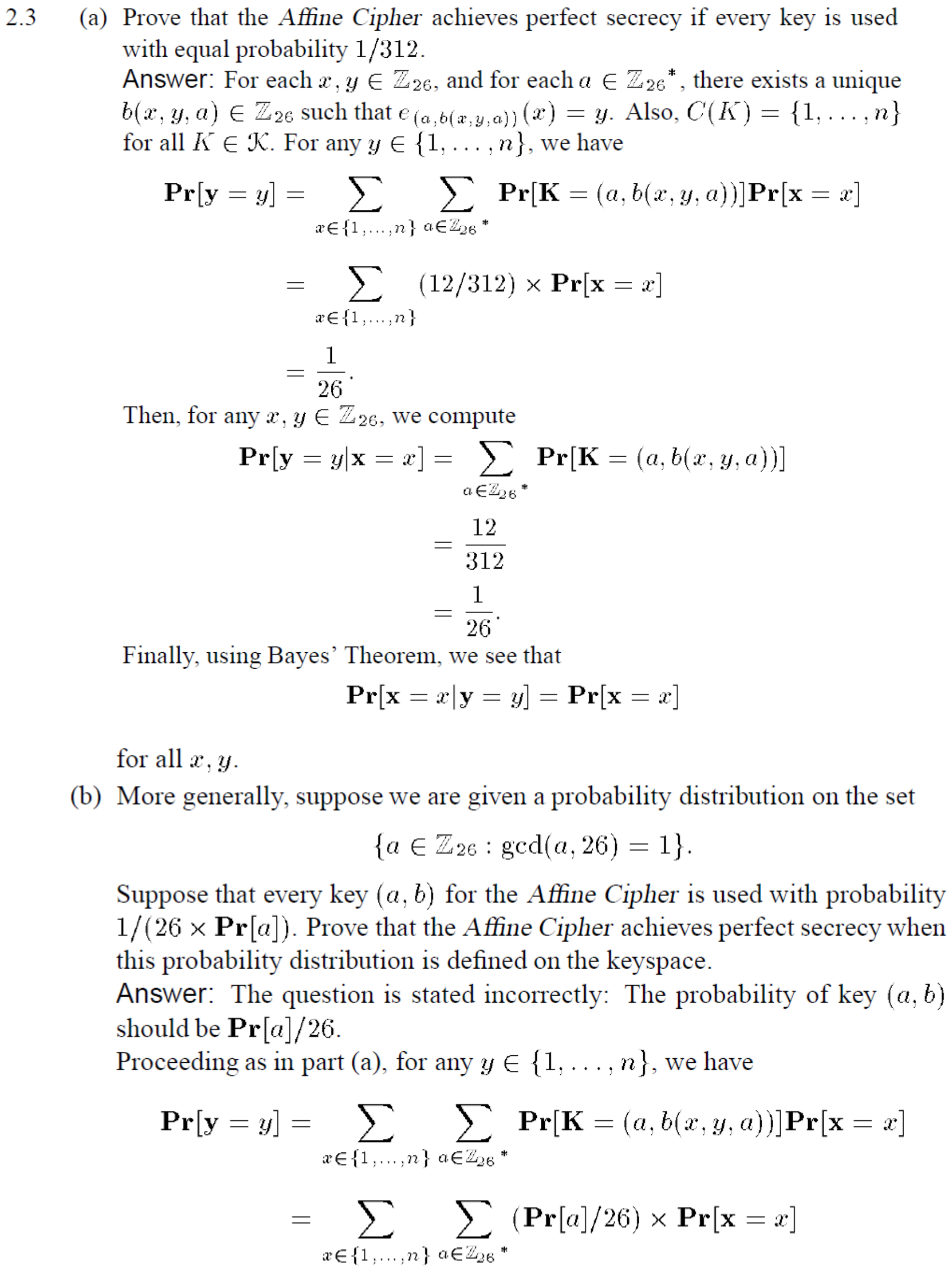
Electrical and Computer Engineering, University of Arizona
ECE 471/571: Fundamentals of Information and Network Security

Spring 2024
Instructor: Ming Li
Assignment #2 Solutions

Each problem worths 10 points.

Problem 2.3 from Stinson's book



2.3 (a) Prove that the *Affine Cipher* achieves perfect secrecy if every key is used with equal probability $1/312$.

Answer: For each $x, y \in \mathbb{Z}_{26}$, and for each $a \in {\mathbb{Z}_{26}}^*$, there exists a unique $b(x, y, a) \in \mathbb{Z}_{26}$ such that $e_{(a,b(x,y,a))}(x) = y$. Also, $C(K) = \{1, \dots, n\}$ for all $K \in \mathcal{K}$. For any $y \in \{1, \dots, n\}$, we have

$$\begin{aligned}\mathbf{Pr}[\mathbf{y} = y] &= \sum_{x \in \{1,\dots,n\}} \sum_{a \in {\mathbb{Z}_{26}}^*} \mathbf{Pr}[\mathbf{K} = (a, b(x, y, a))]\mathbf{Pr}[\mathbf{x} = x] \\ &= \sum_{x \in \{1,\dots,n\}} (12/312) \times \mathbf{Pr}[\mathbf{x} = x] \\ &= \frac{1}{26}.\end{aligned}$$

Then, for any $x, y \in \mathbb{Z}_{26}$, we compute

$$\begin{aligned}\mathbf{Pr}[\mathbf{y} = y|\mathbf{x} = x] &= \sum_{a \in {\mathbb{Z}_{26}}^*} \mathbf{Pr}[\mathbf{K} = (a, b(x, y, a))] \\ &= \frac{12}{312} \\ &= \frac{1}{26}.\end{aligned}$$

Finally, using Bayes' Theorem, we see that

$$\mathbf{Pr}[\mathbf{x} = x|\mathbf{y} = y] = \mathbf{Pr}[\mathbf{x} = x]$$

for all x, y.

(b) More generally, suppose we are given a probability distribution on the set

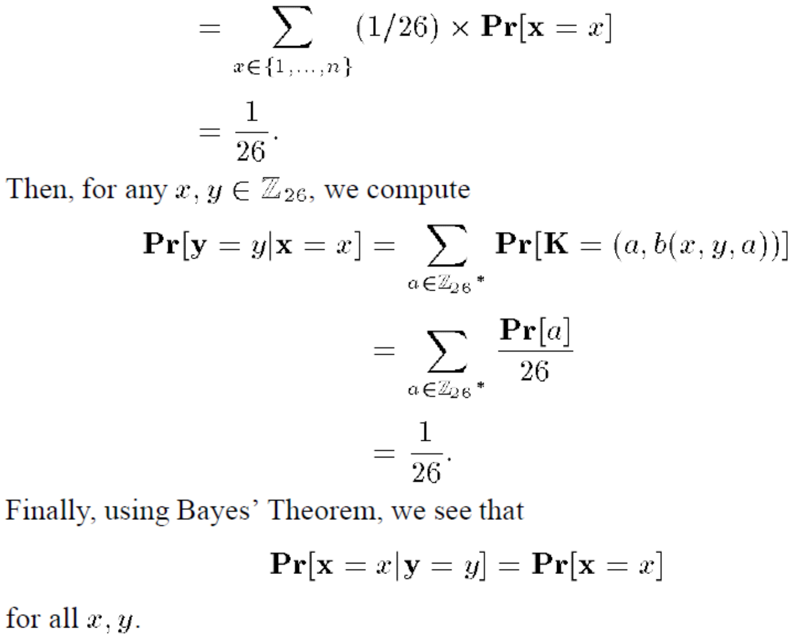
$$\{a \in \mathbb{Z}_{26} : \gcd(a, 26) = 1\}.$$

Suppose that every key (a, b) for the *Affine Cipher* is used with probability $1/(26 \times \mathbf{Pr}[a])$. Prove that the *Affine Cipher* achieves perfect secrecy when this probability distribution is defined on the keyspace.

Answer: The question is stated incorrectly: The probability of key (a, b) should be $\mathbf{Pr}[a]/26$.

Proceeding as in part (a), for any $y \in \{1, \dots, n\}$, we have

$$\begin{aligned}\mathbf{Pr}[\mathbf{y} = y] &= \sum_{x \in \{1,\dots,n\}} \sum_{a \in {\mathbb{Z}_{26}}^*} \mathbf{Pr}[\mathbf{K} = (a, b(x, y, a))]\mathbf{Pr}[\mathbf{x} = x] \\ &= \sum_{x \in \{1,\dots,n\}} \sum_{a \in {\mathbb{Z}_{26}}^*} (\mathbf{Pr}[a]/26) \times \mathbf{Pr}[\mathbf{x} = x]\end{aligned}$$



$$= \sum_{x\in\{1,\dots,n\}} (1/26) \times \mathbf{Pr}[\mathbf{x} = x]$$

$$= \frac{1}{26}.$$

Then, for any $x, y \in \mathbb{Z}_{26}$, we compute

$$\mathbf{Pr}[\mathbf{y} = y|\mathbf{x} = x] = \sum_{a\in\mathbb{Z}_{26}{}^*} \mathbf{Pr}[\mathbf{K} = (a, b(x, y, a))]$$

$$= \sum_{a\in\mathbb{Z}_{26}{}^*} \frac{\mathbf{Pr}[a]}{26}$$

$$= \frac{1}{26}.$$

Finally, using Bayes' Theorem, we see that

$$\mathbf{Pr}[\mathbf{x} = x|\mathbf{y} = y] = \mathbf{Pr}[\mathbf{x} = x]$$

for all x, y.

Problem 2.13 from Stinson's book

Answer:

It is not perfectly secret. Since if we look at the encryption table, Pr(X=a|Y= 4) = 0, which does not equal to Pr(X=a) in the general case which may be larger than 0. Or Pr(X=b|Y= 1) = Pr(X=c|Y= 2)=0.

An alternative is to compute Pr(X=x|Y= y) for all x and y to see if it equals to the prior of X, for any given prior, using the Bayes formula. As long as we find out one case where Pr(X=x|Y= y) is not equal to Pr(X=x) we can say it is not perfectly secret. But this would be more complex.

Problem 4.2 from the textbook.
Solution:
Because of the key schedule, the round functions used in rounds 9 through 16 are mirror images of the round functions used in rounds 1 through 8. From this fact we see that encryption and decryption are identical. We are given a ciphertext c. Let m' = c. Ask the encryption oracle to encrypt m'. The ciphertext returned by the oracle will be the decryption of c.

Problem 4.5 from textbook.
Solution:
For $1 \leq i \leq 128$, take $c_i \in \{0, 1\}^{128}$ to be the string containing a 1 in position i and then zeros elsewhere. Obtain the decryption of these 128 ciphertexts. Let $m_1, m_2, \dots, m_{128}$ be the corresponding plaintexts. Now, given any ciphertext c which does not consist of all zeros, there is a unique nonempty subset of the ci's which we can XOR together to obtain c. Let $I(c) \subseteq \{1, 2, \dots, 128\}$ denote this subset. Observe

$$c = \bigoplus_{i\in I(c)} c_i = \bigoplus_{i\in I(c)} E(m_i) = E\left(\bigoplus_{i\in I(c)} m_i\right)$$

Let **0** be the all-zero string. Note that **0** = **0** ⊕ **0**. From this we obtain E(**0**) = E(**0** ⊕ **0**) = E(**0**) ⊕ E(**0**) = **0**. Thus, the plaintext of c = **0** is m = **0**. Hence we can decrypt every c$_i$ ∈ {0, 1}^{128}.

Problem 4.7 from the textbook.
Solution:
The reasoning for the Feistel cipher, as shown in Figure 4.3, applies in the case of DES. We only have to show the effect of the IP and IP$_{-1}$ functions. For encryption, the input to the final IP$_{-1}$ is RE$_{16}$ || LE$_{16}$. The output of that stage is the ciphertext. On decryption, the first step is to take the ciphertext and pass it through IP. Because IP is the inverse of IP$_{-1}$, the result of this operation is just RE$_{16}$ || LE$_{16}$, which is equivalent to LD$_0$ || RD$_0$. Then, we follow the same reasoning as with the Feistel cipher to reach a point where LE$_0$ = RD$_{16}$ and RE$_0$ = LD$_{16}$. Decryption is completed by passing LD$_0$ || RD$_0$ through IP$_{-1}$. Again, because IP is the inverse of IP$_{-1}$, passing the plaintext through IP as the first step of encryption yields LD$_0$ || RD$_0$, thus showing that decryption is the inverse of encryption.

Problem 6.6 from the textbook.
Solution:
a. AddRoundKey
b. The MixColumn step, because this is where the different bytes interact with each other.
c. The ByteSub step, because it contributes nonlinearity to AES.
d. The ShiftRow step, because it permutes the bytes.
e. There is no wholesale swapping of rows or columns. AES does not require this step because: The MixColumn step causes every byte in a column to alter every other byte in the column, so there is no need to swap rows; The ShiftRow step moves bytes from one column to another, so there is no need to swap columns

Problem 7.4 from the textbook
Solution:
a. No. For example, suppose C1 is corrupted. The output block P3 depends only on the input blocks C2 and C3.

b. An error in P1 affects C1. But since C1 is input to the calculation of C2, C2 is affected. This effect carries through indefinitely, so that all ciphertext blocks are affected. However, at the receiving end, the decryption algorithm restores the correct plaintext for blocks except the one in error. You can show this by writing out the equations for the decryption. Therefore, the error only effects the corresponding decrypted plaintext block.

Additional Problem 1: If the useful life of DES was about 20 years (1977-1999), how long do you predict the useful life of AES (128-bits key) to be? Justify your answer.

Solution:
(Here the exact answer is not as important as how the answer was derived)
Part of the reason for demise of DES has been the increase in computing power, so in 1999 it became feasible to perform an exhaustive key search of 56-bit space. The AES key length of 128, 192, or 256 bits

should be immune to exhaustive search for many more years, even if Moore's law continues to hold (and computing power doubles every 1.5 or 2 years). The end of AES occurs when 2^128 times of the per AES enc/dec time divided by 2^n, where n is the number of 1.5 or 2 year periods, is short, e.g., one month.

For example: If we assume the speed of PCs approximately doubles every 2 years (the key length increases by 1 bit to maintain the same security), then it needs (128-56)*2=144 years to reach 128 bits, since 1999. That is, AES 128 will be useful before the year 2143.

Additional Problem 2: What is the output of the first round of DES when the plaintext and the key are both all zeros? What if the plaintext and the key are all ones?

Solution: The output of the first round of des for all-zero input is (in HEX mode)

L^1: 000000000, R^1: D8D8DBBC

For an input of all ones

L^1: FFFFFFFF, R^1: 27272443

Additional Problem 3: An important property which makes DES secure is that the S-boxes are nonlinear. Verify the nonlinearity of the S-boxes by computing the output of box S_1, for several pairs of inputs. Show that

$$S_1(x_1) \oplus S_1(x_2) \neq S_1(x_1 \oplus x_2)$$

for

a. $x_1 = 000000$, $x_2 = 000001$
b. $x_1 = 111111$, $x_2 = 100000$
c. $x_1 = 101010$, $x_2 = 010101$

Solution:

a. $1110 \neq 0000$
b. $1001 \neq 1000$
c. $1010 \neq 1101$

Additional Problem 4: Assume that bit 57 of a 64 plaintext block is 1 with all other bits equal to zero. Let the key be all zeros.

a. How many S-boxes get different inputs compared to the case of an all-zero plaintext, in the first round of DES?
b. What is the number of output bits which are different compared to the input after the first round?
c. How many output bits have actually changed after the first round compared to the case of an all-zero plaintext (consider only one round). Does DES exhibit the *avalanche effect* (small changes in the plaintext yield significant changes in the ciphertext)?

Solution:

a. Two S-boxes (s-box 1, 8) get different inputs compared to the all-zero case

b. For all-zero input
S-box 1: 1110, S-box 8: 1101
For bit 57 = 1
S-box 1: 0011 (differ by three), S-box 8: 0001 (differ by two)

c.
For all-zero input
L^1: 00000000000000000000000000000000,
R^1: 11011000110110001101101110111100

For bit 57 being equal to 1
L^1: **1**0000000000000000000000000000000,
R^1: 1101**0**000**0**1011000**0**101101110**0**111**1**0

In total, six bits have been changed after the first round. Though this is not an indication of a strong avalanche effect, this is only the first round of DES and many more bits will change on the second round.

Additional Problem 5: Consider the following alternative method of encrypting a message. To encrypt a message, use the algorithm for doing a CBC decrypt. To decrypt a message, use the algorithm for doing a CBC encrypt. Would this work? What are the security implications of this, if any, as contrasted with the "normal" CBC?

Answer:
It would certainly work, in the sense of allowing encryption and decryption of messages.

One problem with this is that if someone knows the plaintext and ciphertext for a set of messages, he/she can mix and match the blocks of those messages almost as easily as with ECB. The reason is that block n of plaintext XOR'd with block n+1 of ciphertext is D of block n+1 of plaintext, and once the attacker knows D of a desired block of plaintext, he/she can XOR it with the plaintext of the previous block to produce correct ciphertext.

More seriously, since block n+1 of ciphertext depends only on block n and n+1 of ciphertext, patterns of ciphertext blocks indicate patterns in the plaintext, which provides a big clue for cryptanalysis. And if D of block n+1 of plaintext is known, it can be XOR'd with block n+1 of ciphertext to get block n of plaintext.

Bonus Problem (10% extra points)
Find a key *K* such that

$$DES_K(x) = DES_K^{-1}(y), \forall x, y$$

Such a key is sometimes called a “weak” key. How many weak keys can you find? To solve this problem you need to look up the exact key schedule generation algorithm for DES. For details refer to http://www.itl.nist.gov/fipspubs/fip46-2.htm Show your work or you will receive zero credit!

Solution:

For the stated property to hold, the double encryption of the plaintext with the same key shall yield the plaintext. This means that the encryption of the ciphertext is equivalent to decrypting it. In turn, this implies that the decryption key schedule is identical to the encryption key schedule.

$K^1 = K^{16}$, $K^2 = K^{15}$, … ,$K^8 = K^9$.

While several solutions that satisfy the identical key schedule condition may be possible, we list some of the obvious ones.

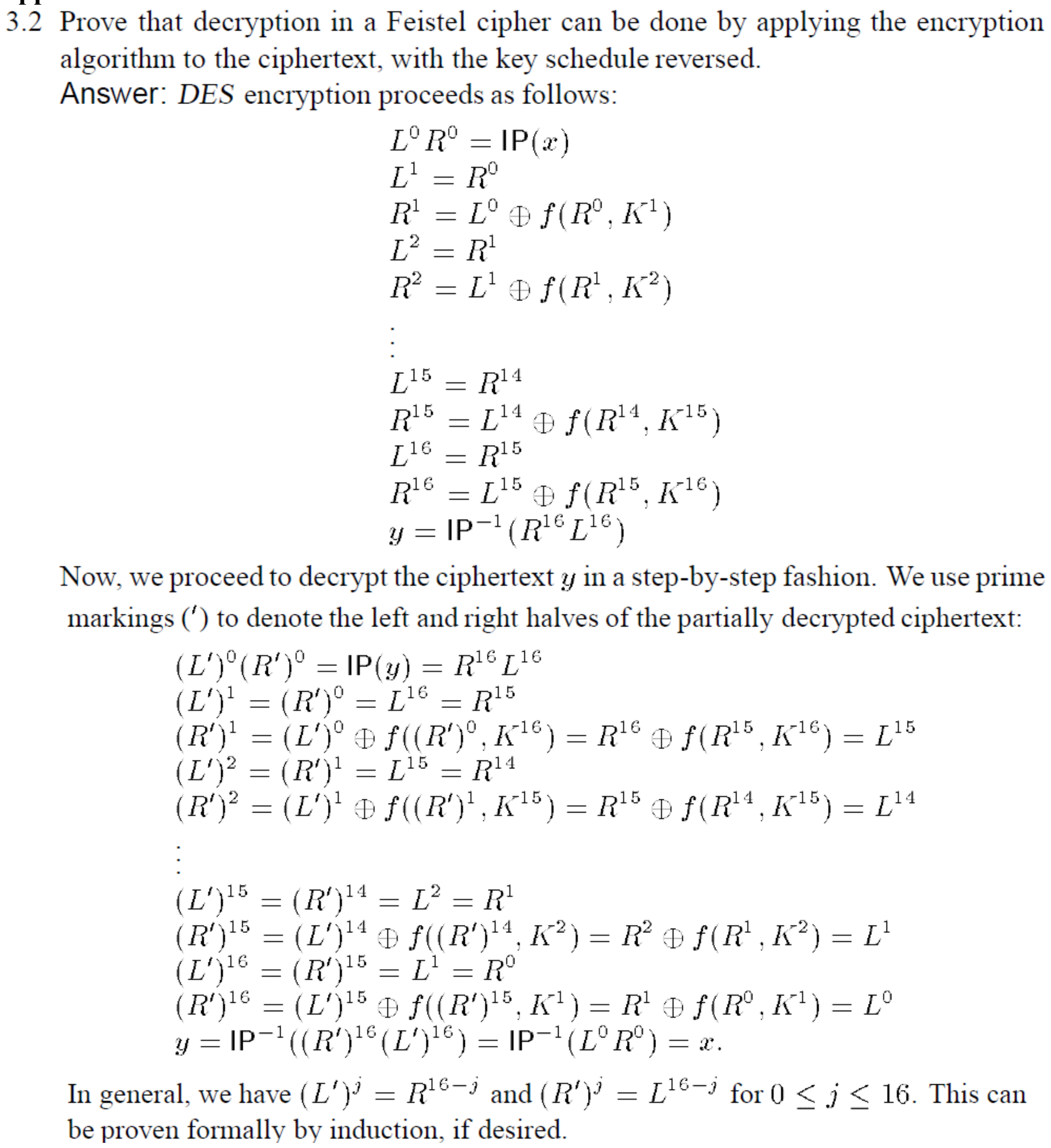
Consider as Ci and Di the input to the left shift registers of the key schedule generator. Weak keys can be obtained if

i) Ci = All zeros, Di = All zeros
ii) Ci = All ones, Di = All ones
iii) Ci = All zeros, Di = All ones
iv) Ci = All ones, Di = All zeros

For each case the actual weak key can be found by executing the key generation algorithm and computing the appropriate parity bits. These combinations yield four weak keys as follows (HEX mode).

i) 0x0101010101010101
ii) 0xFEFEFEFEFEFEFEFE
iii) 0x1F1F1F1F0E0E0E0E
iv) 0xE0E0E0E0F1F1F1F1

Appendix: An alternative solution of Problem 2:



3.2 Prove that decryption in a Feistel cipher can be done by applying the encryption algorithm to the ciphertext, with the key schedule reversed.

Answer: *DES* encryption proceeds as follows:

$$
\begin{aligned}
&L^0R^0 = \mathsf{IP}(x)\\
&L^1 = R^0\\
&R^1 = L^0 \oplus f(R^0, K^1)\\
&L^2 = R^1\\
&R^2 = L^1 \oplus f(R^1, K^2)\\
&\vdots\\
&L^{15} = R^{14}\\
&R^{15} = L^{14} \oplus f(R^{14}, K^{15})\\
&L^{16} = R^{15}\\
&R^{16} = L^{15} \oplus f(R^{15}, K^{16})\\
&y = \mathsf{IP}^{-1}(R^{16}L^{16})
\end{aligned}
$$

Now, we proceed to decrypt the ciphertext y in a step-by-step fashion. We use prime markings ($'$) to denote the left and right halves of the partially decrypted ciphertext:

$$
\begin{aligned}
&(L')^0(R')^0 = \mathsf{IP}(y) = R^{16}L^{16}\\
&(L')^1 = (R')^0 = L^{16} = R^{15}\\
&(R')^1 = (L')^0 \oplus f((R')^0, K^{16}) = R^{16} \oplus f(R^{15}, K^{16}) = L^{15}\\
&(L')^2 = (R')^1 = L^{15} = R^{14}\\
&(R')^2 = (L')^1 \oplus f((R')^1, K^{15}) = R^{15} \oplus f(R^{14}, K^{15}) = L^{14}\\
&\vdots\\
&(L')^{15} = (R')^{14} = L^2 = R^1\\
&(R')^{15} = (L')^{14} \oplus f((R')^{14}, K^2) = R^2 \oplus f(R^1, K^2) = L^1\\
&(L')^{16} = (R')^{15} = L^1 = R^0\\
&(R')^{16} = (L')^{15} \oplus f((R')^{15}, K^1) = R^1 \oplus f(R^0, K^1) = L^0\\
&y = \mathsf{IP}^{-1}((R')^{16}(L')^{16}) = \mathsf{IP}^{-1}(L^0R^0) = x.
\end{aligned}
$$

In general, we have $(L')^j = R^{16-j}$ and $(R')^j = L^{16-j}$ for $0 \leq j \leq 16$. This can be proven formally by induction, if desired.