

Forum:[Hack Forums](#)

Thread: [\[Challenge\] Linear cryptanalysis: break 2-round linear DES.](#)

Phizo - July 26, 2017, 3:40 am

It has been a while since I have posted anything hacking related. I thought this might be fun for some of you newbies out there who may be interested in some beginner-level cryptanalysis.

Context

LDES (linear DES) is an example of a block cipher which includes a 2-round Feistel network. It operates on 4-bit blocks and 2-bit keys. The important characteristic of LDES is that the S-box has been replaced with a linear operation, which makes the whole cipher linear. The purpose of this exercise is to help you understand the importance of the S-box in DES. Without the S-box, the cipher becomes linear and can be completely and trivially broken.

The following diagrams illustrate how LDES functions. On the left, we have the 2-round Feistel network which takes the 4-bit message M , applies a single bitwise left shift, and splits the message into two registers, $\{A, B\}$. The rounds are computed such that $A[n] = B[n-1]$ and $B[n] = A[n-1] \oplus F(B[n-1], K[n])$ with $0 < n \leq 2$ (note that $\oplus$ represents XOR). K represents a key which is distributed as subkeys by the key scheduler for each round in the Feistel network, which simply expands each n 'th bit of K into $K[n]$ for $1 \leq n \leq 2$. Following the two rounds, a P-box is introduced (permutes the registers), and finally a single bitwise right shift of the 4-bit M produces the 4-bit ciphertext C . The F (Feistel) function is the heart of (L)DES. F is given by $F : X \times Y \rightarrow Z$ such that X is a partial (2-bit) block given by register B , Y is a 3-bit subkey, and Z is a "scrambled" partial (2-bit) block. E represents the expansion function: the 2-bit input X is expanded to a 3-bit X which is XOR'd against the subkey Y also passed into F . The resulting 3-bit output is input for the linear operation which replaces the S-box. This operation is simply the bitwise complement/NOT of the last two bits of I : $J[1] = I[2] + 1 \bmod 2$; $J[2] = I[3] + 1 \bmod 2$.

Goal

Given the following known plaintext-ciphertext pairs:

Quote: $E(0000) = 0110$

$E(1000) = 1110$

$E(0100) = 0011$

$E(0010) = 1100$

$E(0001) = 0111$

You are to solve the following (i.e. compute the corresponding plaintexts):

Quote: $E(1100)$

$E(1010)$

$E(1001)$

$E(0110)$

$E(0101)$

$E(0011)$

$E(0111)$

$E(1011)$

$E(1101)$

E(1110)
E(1111)

This demonstrates that if an adversary knows $E(0000)$, $E(1000)$, $E(0100)$, $E(0010)$, and $E(0001)$, he can encrypt any message. In general, for a linear cipher with block size n , if an adversary knows $n+1$ plaintext-ciphertext pairs, he can encrypt/decrypt any message without even knowing the key.

A large hint as to how one solves this is given by the following property: for a linear encryption function E , we have $E(X \oplus Y) = E(X) \oplus E(Y)$.

In truth, you don't need to really understand the intricates of how this cipher works in order to break it, but context is good regardless. Solving this is actually very trivial.

Please put your solutions in spoiler tags so every participant gets a chance. Show all necessary working.

If you have any questions, feel free to post them. I will provide another hint depending on how people do, and I will post the solution maybe in a week or so.

Good luck.

Jesse Pinkman. - July 27, 2017, 2:20 am

This seems like a very interesting challenge. I'm curious to see if anyone is able to solve it?

Bronze5 - July 27, 2017, 8:50 pm

Sounds rllly hard .-. but im down thx

????Kosdff - July 27, 2017, 8:54 pm

Welp. This just broke my brain.

Phizo - July 28, 2017, 6:57 pm

Glad to finally see some interested individuals. I thought no one would make an attempt, which would be unfortunate, as I was looking to post more challenges in the future. Honestly, as I said in the last paragraph, this really is not too difficult. The algorithm of the cipher itself will be challenging to interpret for someone who is new to this sort of stuff, but as I said, you don't really need to understand the algorithm in order to solve this challenge. It's just a bit of context. It is easier than it seems.

I will provide some hints, as I want to see some people actually attempt/solve this and learn something:

Spoiler ([Click to View](#))

What do you notice about each of the plaintexts in the five plaintext-ciphertext pairs provided? What is significant about their values, collectively?

An even bigger hint:

Spoiler ([Click to View](#))

Can some of the plaintext-ciphertext pairs be combined in order to compute a new pair? How is the aforementioned pattern useful here?

100 views and no attempts. Don't any of you want to learn some real hacking? You will never evolve unless you challenge yourself. I would like to post more challenges in the future, but there is no point if no one is willing to participate. Throw your lame RATs away and do something interesting

that's actually worth your while. :)

Baud - July 29, 2017, 3:39 am

I'm currently giving it a try with little to no cryptography knowledge but by using logic and bitwise arithmetics. I will update this in a few hours, I may be able to get this right.

I'd love to see more of these challenges by the way, it's a good excuse for me to use my brain and a chance to get to learn more about cryptography.

Phizo - July 29, 2017, 5:25 am

(Jul 29, 2017 03:39 AM)Baud Wrote: [\[CITE\]](#) I'm currently giving it a try with little to no cryptography knowledge but by using logic and bitwise arithmetics. I will update this in a few hours, I may be able to get this right.

I'd love to see more of these challenges by the way, it's a good excuse for me to use my brain and a chance to get to learn more about cryptography.

Glad to see a participant. You honestly don't need to be very familiar with cryptography in order to solve this. All you need to know is how to compute $C = A \text{ XOR } B$. If you don't know, that is effectively $C = (A + B) \bmod 2$. You won't need any other bitwise operation for this. Keep the property I gave in mind, as it is the key to solving this: for a linear encryption function E , $E(X \oplus Y) = E(X) \oplus E(Y)$. This is because a linear Boolean function is defined as $f(x[0], x[2], \dots, x[n]) = \sum c[i]x[i]$ where $c[i], x[i] \in \{0, 1\}$ – note that it is addition modulo 2, i.e. XOR. The polynomial must have a degree of 1. If you don't understand this, don't worry, as it is contextual more than anything. So long as you can utilise the property I gave, you should be able to solve this easily. If you haven't already, take a look at the hints I provided in the post prior to yours. Good luck. :)

Also, good, I will post more in that case. Even if a few amount of people are interested and benefit from these challenges, I'm happy to post some more.

Bronze5 - July 29, 2017, 6:04 am

Shit this is acutally hard ;-(for me). I read like 6 pages on cyrpthograpy just for this still not getting near, but thx for the tips.

Phizo - July 29, 2017, 6:09 am

(Jul 29, 2017 06:04 AM)Bronze5 Wrote: [\[CITE\]](#) Shit this is acutally hard ;-(for me). I read like 6 pages on cyrpthograpy just for this still not getting near, but thx for the tips.

I don't think you will find anything on this in particular online, in a book, etc. I haven't found this problem online. You don't need to read up on cryptography. Everything you need is in this thread, especially in the post prior to yours. :)

Asdethprime - July 29, 2017, 7:36 am

$E(1100) = E(1000) \text{ XOR } E(0100)$
 $= 1110 \text{ XOR } 0011$
 $= 1101$

$E(1010) = E(1000) \text{ XOR } E(0010)$
 $= 1110 \text{ XOR } 1100$
 $= 0010$

$E(1001) = E(1000) \text{ XOR } E(0001)$
 $= 0011 \text{ XOR } 0111$
 $= 0100$

That's what I have got so far. Is this how it's done? I know very little about encryption, really a topic I

should really read up on.

Phizo - July 29, 2017, 7:44 am

(Jul 29, 2017 07:36 AM)Asdethprime Wrote: [\[CITE\]](#) -snip-

That's what I have got so far. Is this how it's done? I know very little about encryption, really a topic I should really read up on.

Firstly, please wrap your solution in spoiler tags ([sp0iler]...[/sp0iler]).

You're very close. That is the principle, yes. However, it's still not correct. I will give you part of the solution so you can try to determine what else you are missing:

Spoiler ([Click to View](#))

Quote: $E(1100) = 1011$

$E(1010) = 0100$

$E(1001) = 1111$

Here is a good hint for you: compute $E(0111)$. Why is this different to the ones you have attempted?

Baud - July 29, 2017, 7:52 am

This is a placeholder because I just found the solution and I'm going to write it here as soon as I come back home, I solved all the numbers.

Edit: well... after seeing what Phizo posted I just found out my solution is not correct, still, I'm going to post it here to see if at least I went close enough.

Spoiler ([Click to View](#)) Code $E(1100) = E(1000) \text{ xor } E(0100) =$

$E(1000) = 1110$

$E(0100) = 0011$

1101

$E(1010) = E(1000) \text{ xor } E(0010) =$

$E(1000) = 1110$

$E(0010) = 1100$

0010

$E(1001) = E(1000) \text{ xor } E(0001) =$

$E(1000) = 1110$

$E(0001) = 0111$

1001

$E(0110) = E(0100) \text{ xor } E(0010) =$

$E(0100) = 0011$

$E(0010) = 1100$

1111

$E(0101) = E(0100) \text{ xor } E(0001) =$

$E(0100) = 0011$

$E(0001) = 0111$

0100

$E(0011) = E(0010) \text{ xor } E(0001) =$

$E(0010) = 1100$

$E(0001) = 0111$

1011

$E(0111) = E(0100) \text{ xor } E(0010) \text{ xor } E(0001) =$

$E(0100) = 0011$

$E(0010) = 1100$

$E(0001) = 0111$

1000

$E(1011) = E(1000) \text{ xor } E(0010) \text{ xor } E(0001) =$

$E(1000) = 1110$

$E(0010) = 1100$

$E(0001) = 0111$

0101

$E(1101) = E(1000) \text{ xor } E(0100) \text{ xor } E(0001) =$

$E(1000) = 1110$

$E(0100) = 0011$

$E(0001) = 0111$

1010

$E(1110) = E(1000) \text{ xor } E(0100) \text{ xor } E(0010) =$

$E(1000) = 1110$

$E(0100) = 0011$

$E(0010) = 1100$

0001

$E(1111) = E(1000) \text{ xor } E(0100) \text{ xor } E(0010) \text{ xor } E(0001) =$

$E(1000) = 1110$

$E(0100) = 0011$

$E(0010) = 1100$

$E(0001) = 0111$

0110

Phizo - July 29, 2017, 8:15 am

(Jul 29, 2017 07:52 AM)Baud Wrote: [\[CITE\]](#) This is a placeholder because I just found the solution and I'm going to write it here as soon as I come back home, I solved all the numbers.

Edit: well... after seeing what Phizo posted I just found out my solution is not correct, still, I'm going to post it here to see if at least I went close enough.

Spoiler ([Click to View](#))Code $E(1100) = E(1000) \text{ xor } E(0100) =$

$E(1000) = 1110$

$E(0100) = 0011$

1101

$E(1010) = E(1000) \text{ xor } E(0010) =$

$E(1000) = 1110$

$E(0010) = 1100$

0010

$E(1001) = E(1000) \text{ xor } E(0001) =$

$E(1000) = 1110$

$E(0001) = 0111$

1001

$E(0110) = E(0100) \text{ xor } E(0010) =$
 $E(0100) = 0011$
 $E(0010) = 1100$
1111

$E(0101) = E(0100) \text{ xor } E(0001) =$
 $E(0100) = 0011$
 $E(0001) = 0111$
0100

$E(0011) = E(0010) \text{ xor } E(0001) =$
 $E(0010) = 1100$
 $E(0001) = 0111$
1011

$E(0111) = E(0100) \text{ xor } E(0010) \text{ xor } E(0001) =$
 $E(0100) = 0011$
 $E(0010) = 1100$
 $E(0001) = 0111$
1000

$E(1011) = E(1000) \text{ xor } E(0010) \text{ xor } E(0001) =$
 $E(1000) = 1110$
 $E(0010) = 1100$
 $E(0001) = 0111$
0101

$E(1101) = E(1000) \text{ xor } E(0100) \text{ xor } E(0001) =$
 $E(1000) = 1110$
 $E(0100) = 0011$
 $E(0001) = 0111$
1010

$E(1110) = E(1000) \text{ xor } E(0100) \text{ xor } E(0010) =$
 $E(1000) = 1110$
 $E(0100) = 0011$
 $E(0010) = 1100$
0001

$E(1111) = E(1000) \text{ xor } E(0100) \text{ xor } E(0010) \text{ xor } E(0001) =$
 $E(1000) = 1110$
 $E(0100) = 0011$
 $E(0010) = 1100$
 $E(0001) = 0111$
0110

I'm very sorry. I should have said this earlier: to anyone who is verifying a solution, you can just give me a few correct pairs and I'll know you've solved it. I understand doing them all is a bit tedious. The equation is the same for all of them, but the inputs are, of course, different. This means if you need to call E N times to solve one, you should expect to call it N times for all of them. It will be the same amount every time. This is also a hint. A good portion of your computed pairs are correct, Baud. Try and find the minimum amount of pairs you need to use in order to compute them all.

I have actually already told you which one of your solutions are correct in the post above yours. So, you should be able to easily infer the solution now. :)

Baud - July 29, 2017, 9:16 am

(Jul 29, 2017 08:15 AM)Phizo Wrote: [\[CITE\]](#) I'm very sorry. I should have said this earlier: to anyone who is verifying a solution, you can just give me a few correct pairs and I'll know you've solved it. I understand doing them all is a bit tedious. The equation is the same for all of them, but the inputs are, of course, different. This means if you need to call E N times to solve one, you should expect to call it N times for all of them. It will be the same amount every time. This is also a hint. A good portion of your computed pairs are correct, Baud. Try and find the minimum amount of pairs you need to use in order to compute them all.

I have actually already told you which one of your solutions are correct in the post above yours. So, you should be able to easily infer the solution now. :)

Oh don't worry I was actually pretty fast in doing the calculations, I'm used to XOR bits manually since I used to be an ASM programmer. Maybe I finally figured it out, this is the error I did before:

Spoiler ([Click to View](#)) The wrong pairs were the first ones, because I need to use 3 pairs for each calculation while I only used 2, is that correct? Because I tried doing this and I finally obtained the results you posted above, what I forgot to do is to XOR for E(0000) which is 0110, so the results now would be:

Code E(1001) = E(1000) xor E(0001) xor E(0000) =

E(1000) = 1110

E(0001) = 0111

E(0000) = 0110

E(1001) = 1111

E(1100) = E(1000) xor E(0100) xor E(0000) =

E(1000) = 1110

E(0001) = 0011

E(0000) = 0110

E(1001) = 1011

E(0110) = E(0100) xor E(0010) xor E(0000) =

E(0100) = 0011

E(0010) = 1100

E(0000) = 0110

E(0110) = 1001

It's also possible to calculate E(1111) with just 3 pairs now, you just need to calculate the value of a number like E(1100) then xor for E(0010) and again for E(0001), so we can say that E(1111) is equal to 0000.

Phizo - July 29, 2017, 10:01 am

(Jul 29, 2017 09:16 AM)Baud Wrote: [\[CITE\]](#) Oh don't worry I was actually pretty fast in doing the calculations, I'm used to XOR bits manually since I used to be an ASM programmer. Maybe I finally figured it out, this is the error I did before:

Spoiler ([Click to View](#)) The wrong pairs were the first ones, because I need to use 3 pairs for each calculation while I only used 2, is that correct? Because I tried doing this and I finally obtained the

results you posted above, what I forgot to do is to XOR for E(0000) which is 0110, so the results now would be:

Code E(1001) = E(1000) xor E(0001) xor E(0000) =

E(1000) = 1110

E(0001) = 0111

E(0000) = 0110

E(1001) = 1111

E(1100) = E(1000) xor E(0100) xor E(0000) =

E(1000) = 1110

E(0001) = 0011

E(0000) = 0110

E(1001) = 1011

E(0110) = E(0100) xor E(0010) xor E(0000) =

E(0100) = 0011

E(0010) = 1100

E(0000) = 0110

E(0110) = 1001

It's also possible to calculate E(1111) with just 3 pairs now, you just need to calculate the value of a number like E(1100) then xor for E(0010) and again for E(0001), so we can say that E(1111) is equal to 0000.

You are correct. Good work! I knew you would get it after the last post. Thanks for participating. I hope you enjoyed it. :)

Baud - July 29, 2017, 10:37 am

(Jul 29, 2017 10:01 AM)Phizo Wrote: [\[CITE\]](#) You are correct. Good work! I knew you would get it after the last post. Thanks for participating. I hope you enjoyed it. :)

I'm so glad I finally got it right, it was fun and definitely satisfying in the end. If you post more in the future I'd be more than happy to participate again and hopefully learn new things, I love me some challenges so they are always welcome, thank you for this one!

Dr. Gonzo - July 29, 2017, 12:21 pm

We need more challenges like this going on. Good job man, Keep it up!

Phizo - July 30, 2017, 5:49 am

(Jul 29, 2017 10:37 AM)Baud Wrote: [\[CITE\]](#) I'm so glad I finally got it right, it was fun and definitely satisfying in the end. If you post more in the future I'd be more than happy to participate again and hopefully learn new things, I love me some challenges so they are always welcome, thank you for this one!

I'm glad it was satisfying. I will definitely make some more challenges. I am thinking about what to do next, but I will likely post one soon. Seeing as you are very curious and eager to learn, I will emphasise why the S-box is so important to DES and similar block ciphers. Two properties of a secure cipher identified by a smart person named Claude Shannon are dubbed "diffusion" and "confusion."

Diffusion refers to the act of diffusing the statistical structure of the plaintext over the ciphertext such that each bit of ciphertext depends on several bits of plaintext. In general, this is achieved by the repeated invocation of $f(P(b))$, where $P(b)$ represents a permutation on some input bits b (i.e. P-box), and f is some function which transforms the permutation of that input. With the Feistel

network in DES (and some other block ciphers), this is achieved well by the accumulation of all rounds. This challenge only demonstrates a 2-round Feistel network, but real DES uses 16-round Feistel network. Each round takes a permutation of the input bits, applies F (the Feistel function) to it, and repeats for all rounds. Good diffusion implies that a single bit change in plaintext should statistically change half of the ciphertext bits on average and vice versa. This is called the "avalanche effect."

Confusion refers to the act of making the statistical relationship between the ciphertext and the key as complex as possible by making each bit in the ciphertext depend on several bits of the key. In the case of a binary block cipher like DES, this is achieved by introducing non-linear table whose output depends on the input (the key). This table is the S-box. In real DES, a 48-bit subkey is divided into eight 6-bit blocks as input for each S-box (there are eight). Each S-box outputs a 4-bit digit using a non-linear transformation (each row in an S-box is a non-linear Boolean function), which is drastically different to its input. Of course, with this challenge, the implementation is much simpler and the S-box is removed, making this mini-DES completely linear. The S-box is the core of DES's security, and this challenge is aimed to illustrate that.

Take the following S-box for example:

Code unsigned int S(unsigned int i)

```
{
    unsigned int Sbox[] =
    {
        0x00, 0x00, 0x00, 0x01,
        0x00, 0x00, 0x02, 0x03
    };
};
```

return Sbox[i];

```
}
```

Representing it as in the challenge (3-bit input; 2-bit output):

Quote: S(000) = 00, S(001) = 00, S(010) = 00, S(011) = 01,

S(100) = 00, S(101) = 00, S(110) = 10, S(111) = 11

As you can see, the relationship between input and output is no longer linear.

If we were to substitute the linear function (which hypothetically replaced the S-box) with the S-box, the one dubbed "linear" in the diagram, you will no longer be able to compute the pairs because the property $E(X \oplus Y) = E(X) \oplus E(Y)$ no longer holds because non-linearity has been introduced into the cipher.

Quote: $E(1100) = 0011$:

$E(1100) \rightarrow E(1000) \oplus E(0100) \oplus E(0000) = 0010 \oplus 1000 \oplus 1111 = 0100$

$0100 \neq 0011$ – linear relationship not satisfied.

$E(1010) = 1010$:

$E(1010) \rightarrow E(1000) \oplus E(0010) \oplus E(0000) = 0010 \oplus 0001 \oplus 1111 = 1100$

$1100 \neq 1010$ – linear relationship not satisfied.

$E(1001) = 0110$:

$E(1001) \rightarrow E(1000) \oplus E(0001) \oplus E(0000) = 0010 \oplus 0100 \oplus 1111 = 1001$

$1001 \neq 0110$ – linear relationship not satisfied.

You only need one counterexample in order to prove the relationship no longer holds, but funnily enough, if you compute them all, you will notice that it does not hold for any. If it appeared to hold, it would merely be coincidental. This is why the S-box is so important for this block cipher and others.

EDIT: Also, do you do any programming still? You mentioned you used to do Assembly. I'm deciding if my next challenge will involve some programming.

(Jul 29, 2017 12:21 PM)Great White Shark Wrote: [\[CITE\]](#) We need more challenges like this going on. Good job man, Keep it up!
Thanks. :)

mutybyte - July 31, 2017, 7:35 pm

All these things goes out of my mind. So hard for me. Any right direction for me ?

tulamba - July 31, 2017, 7:41 pm

Great Job Phizo. I hope you will do AES next ;)

Baud - July 31, 2017, 8:07 pm

(Jul 30, 2017 05:49 AM)Phizo Wrote: [\[CITE\]](#) [Too long to quote]

Thank you very much for clearing this up, I know very little about cryptography and I just started looking more seriously into it thanks to a book I have laying around in my collection, so most of these concepts were actually new to me and your explanation helped me understanding a lot of things.

The only thing I did not find clear is the definition of P-box, for example, would there be a P-box on the scheme of this linear DES algorithm? How is it used?

I'm also examining the other challenge you posted a few days ago, it's much more difficult than this one at least for me but I'm slowly learning everything I need to give it a try. If I have any questions regarding this challenge I will post them on its own thread so that other users can see them as well.

(Jul 30, 2017 05:49 AM)Phizo Wrote: [\[CITE\]](#) EDIT: Also, do you do any programming still? You mentioned you used to do Assembly. I'm deciding if my next challenge will involve some programming.

Yes, I do know how to program in several programming languages including Assembly (Zilog 80 and ARM9+Thumb architectures), C, C#, python, and some others as well, so it would actually be interesting to solve a challenge which include programming elements as well.

Phizo - August 2, 2017, 5:07 pm

(Jul 31, 2017 08:07 PM)Baud Wrote: [\[CITE\]](#) Thank you very much for clearing this up, I know very little about cryptography and I just started looking more seriously into it thanks to a book I have laying around in my collection, so most of these concepts were actually new to me and your explanation helped me understanding a lot of things.

The only thing I did not find clear is the definition of P-box, for example, would there be a P-box on the scheme of this linear DES algorithm? How is it used?

I'm also examining the other challenge you posted a few days ago, it's much more difficult than this one at least for me but I'm slowly learning everything I need to give it a try. If I have any questions regarding this challenge I will post them on its own thread so that other users can see them as well.

Yes, I do know how to program in several programming languages including Assembly (Zilog 80 and ARM9+Thumb architectures), C, C#, python, and some others as well, so it would actually be interesting to solve a challenge which include programming elements as well.

My pleasure. :)

The P-box is essentially a function which outputs a permutation of its input. In real DES (but not this proposed version), there exists a P-box which permutes the output bits of the S-box, so that the bits are dispersed across four other S-boxes in the next round (recall there are eight S-boxes in DES). This provides confusion and diffusion. Even though I don't believe they're classified as P-boxes, you can say the first and last permutations in this (and real) DES as P-boxes. The first thing that happens with the 4-bit message M you put in is they're split into 2-bit registers and swapped (permuted). This happens again at the end just before outputting the 4-bit ciphertext C. Also, each round performs a permutation: $A[n] = B[n-1]$, $B[n] = A[n-1] \oplus F(B[n-1], K[n])$. They aren't P-boxes as such, but they're effectively the same thing.

Here is a diagram of the F (Feistel) function in real DES. The "P" represents the P-box, which takes the output given by the S-boxes and outputs a permutation of them.

As for the other challenge I posted (CBC bitflipping), I'm not sure if I would recommend doing that one. It will require a fair amount of time/code. I somewhat regret posting it, as I doubt anyone will do that one. It's better if you've done the previous challenges (it's from a set of challenges). I have another challenge on the way, though. It will be a tad harder than this one. I have written the problem (not that it requires much code). I'm currently in the process of solving it myself, as I should make sure I can before I post it. I don't doubt I can, though, as I know how to. It's just a matter of doing it. Hopefully, I will post it soon. :)

EDIT: Here is my next challenge: <https://hackforums.net/showthread.php?tid=5685376>

Baud - August 3, 2017, 7:22 am

(Aug 2, 2017 05:07 PM)Phizo Wrote: [\[CITE\]](#) My pleasure. :)

The P-box is essentially a function which outputs a permutation of its input. In real DES (but not this proposed version), there exists a P-box which permutes the output bits of the S-box, so that the bits are dispersed across four other S-boxes in the next round (recall there are eight S-boxes in DES). This provides confusion and diffusion. Even though I don't believe they're classified as P-boxes, you can say the first and last permutations in this (and real) DES as P-boxes. The first thing that happens with the 4-bit message M you put in is they're split into 2-bit registers and swapped (permuted). This happens again at the end just before outputting the 4-bit ciphertext C. Also, each round performs a permutation: $A[n] = B[n-1]$, $B[n] = A[n-1] \oplus F(B[n-1], K[n])$. They aren't P-boxes as such, but they're effectively the same thing.

Ah okay now it's much clearer, I had the wrong concept of permutation in mind so that's what confused me, it's easier than I thought, thank you once again.

(Aug 2, 2017 05:07 PM)Phizo Wrote: [\[CITE\]](#) As for the other challenge I posted (CBC bitflipping), I'm not sure if I would recommend doing that one. It will require a fair amount of time/code. I somewhat regret posting it, as I doubt anyone will do that one. It's better if you've done the previous challenges (it's from a set of challenges).

It really does look difficult, I'm going to postpone that one and spend some time studying the basics I'm missing and solving the previous exercises before giving it another try, I can safely say I'm not competent enough for it yet.