Step 5) Compute a_5, a_9, a_{10}, $\gcd(a_{10} - a_5, n)$, store a_5, a_{10}, etc.

In the above example we succeed at the sixth step since $\gcd(a_{12} - a_6, n) = 23$. If $n = pq$ and $p < \sqrt{n}$ then the algorithm takes time $O(\sqrt{p})$ (from a random walk through $\mathbf{F}_p$) $= O(\sqrt[4]{n}) = e^{O(\frac{1}{4}\log n)}$. Trivial division takes time $O(\sqrt{n}) = e^{O(\frac{1}{2}\log n)}$ and the number field sieve takes time $e^{O((\log n)^{1/3}(\log\log n)^{2/3})}$.

We can use the same idea to solve the discrete log problem for elliptic curves over finite fields. Let $E : y^2 = x^3 + 17x + 1$ over $\mathbf{F}_{101}$. The point $G = (0, 1)$ generates $E(\mathbf{F}_{101})$. In addition, $103G = \emptyset$ so the multiples of the points work mod 103. The point $Q = (5, 98) = nG$ for some n; find n. Let $x(\text{point})$ denote the x-coordinate of a point, so $x(Q) = 5$.

Let's take a random walk through $E(\mathbf{F}_{101})$. Let $v_0 = [0, 0]$ and $P_0 = \emptyset$. The vector $v_i = [a, b]$ means $P_i = a_iQ + b_iG$ where a_i, b_i are defined mod 103.
If $x(P_i) \leq 33$ or $P_i = \emptyset$ then $P_{i+1} = Q + P_i$ and $v_{i+1} = v_i + [1, 0]$.
If $33 < x(P_i) < 68$ then $P_{i+1} = 2P_i$ and $v_{i+1} = 2v_i$.
If $68 \leq x(P_i)$ then $P_{i+1} = G + P_i$ and $v_{i+1} = v_i + [0, 1]$.

When $P_{2j} = P_j$, quit. Then $P_{2j} = a_{2j}Q + b_{2j}G = a_jQ + b_jG = P_j$. So $(a_{2j} - a_j)Q = (b_j - b_{2j})G$ and $Q = (b_j - b_{2j})(a_{2j} - a_j)^{-1}G$ where $(a_{2j} - a_j)^{-1}$ is reduced mod 103.

i	P_i	$[a, b]$
	0	$[0, 0]$
1	$(5, 98)$	$[1, 0]$
2	$(68, 60)$	$[2, 0]$
3	$(63, 29)$	$[2, 1]$
4	$(12, 32)$	$[4, 2]$
5	$(8, 89)$	$[5, 2]$
6	$(97, 77)$	$[6, 2]$
7	$(62, 77)$	$[6, 3]$
8	$(53, 81)$	$[12, 6]$
9	$(97, 77)$	$[24, 12]$
10	$(62, 66)$	$[24, 13]$
11	$(53, 81)$	$[48, 26]$
12	$(97, 77)$	$[96, 52]$

Note that $P_{12} = P_6$ so $6Q + 2G = 96Q + 52G$. Thus $-90Q = 50G$ and $Q = (-90)^{-1}50G$. We have $(-90)^{-1}50 \equiv 91 (\text{mod} 103)$ so $Q = 91G$.

We could do this in $\mathbf{F}_p^*$ also, where p is prime. Let g generate $\mathbf{F}_p^*$ and say $q \in \mathbf{F}_p^*$. Solve $g^x = q$. Let $p_0 = 1$. If $p_i < p/3$ then $p_{i+1} = p_iq$, if $p/3 < p_i < 2p/3$ then $p_{i+1} = p_i^2$, if $2p/3 < p_i$ then $p_{i+1} = p_ig$.

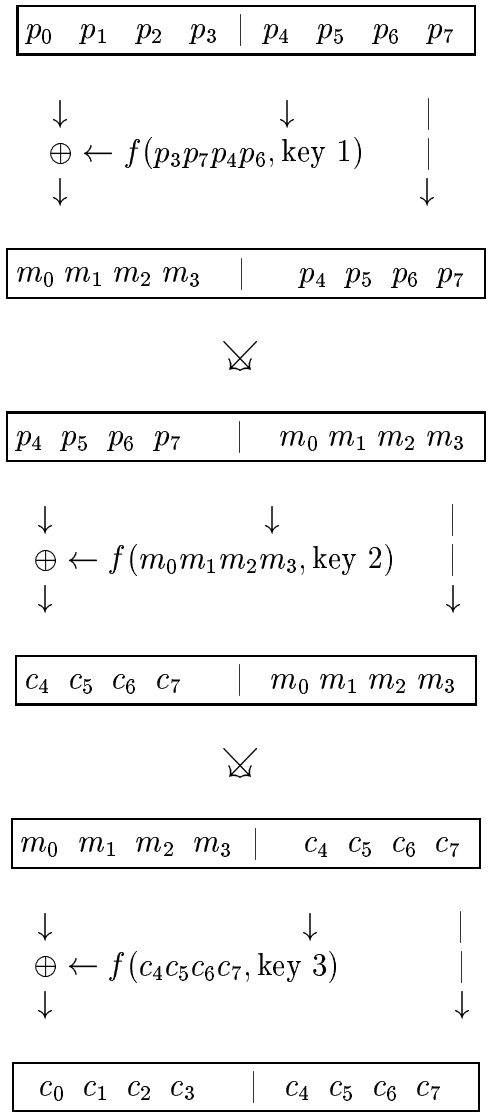
Cryptanalysis of DES

First we will come up with a slightly different Baby DES that will be more suitable for demonstrating several cryptanalytic techniques. Last time we had 2 round Baby DES. We started with an initial permutation IP and ended with permutation IP^{-1}. Linear and differential cryptanalysis are known and chosen plaintext attacks. In both cases, the enemy

knows some PT and matching CT. So the enemy knows IP(PT) and IP^{-1}(CT). So IP and IP^{-1} don't contribute to the challenge so we will leave them out.

3 round Baby DES has a ten bit key $k_0k_1k_2k_3k_4k_5k_6k_7k_8k_9$ and three subkeys, key1 $= k_0k_6k_8k_3k_7k_2k_9k_5$, key2 $= k_7k_2k_5k_4k_9k_1k_8k_0$ and key3 $= k_9k_1k_0k_6k_8k_3k_5k_7$.

We will denote the PT by $p_0p_1p_2p_3p_4p_5p_6p_7$, the CT by $c_0c_1c_2c_3c_4c_5c_6c_7$ and some intermediate bits by $m_0m_1m_2m_3$. The function $f(\cdot, \text{key}i)$ is the same as the function Π_{T_i} described in the earlier description of Baby DES.



Linear cryptanalysis

Linear cryptanalysis is an idea of Matsui's published in 1992. It is a known PT attack. Let the PT block be $p_0 \ldots p_{n-1}$, the key be $k_0 \ldots k_{m-1}$ and the corresponding CT be $c_0 \ldots c_{n-1}$. Let's say that the linear equation
$p_{\alpha_1} + p_{\alpha_2} + \ldots + p_{\alpha_a} + c_{\beta_1} + \ldots + c_{\beta_b} + k_{\gamma_1} + \ldots + k_{\gamma_g} = x$ (where $x = 0$ or 1, $1 \leq a, b \leq n$, $1 \leq g \leq m$), holds with probability $p > 1/2$ over all PT/key pairs. So $x + p_{\alpha_1} + \ldots + c_{\beta_b}$ $= k_{\gamma_1} + \ldots + k_{\gamma_g}$ with $p > 1/2$. Then compute $x + p_{\alpha_1} + \ldots + c_{\beta_b}$ over all intercepted PT/CT

pairs. If it's 0 most of the time, assume $k_{\gamma_1} + \ldots + k_{\gamma_g} = 0$. If it's 1 most of the time, assume $k_{\gamma_1} + \ldots + k_{\gamma_g} = 1$. This gives a relation on the key bits. Try to get several relations.

Interestingly, if encryption were linear, these equations would all hold with probability exactly 1/2, so linear cryptanalysis exploits the non-linearity of encryption.

Linear cryptanalysis of 3-round Baby DES

Let S_0 and S_1 be the functions from four bits to two bits corresponding to the operations of the S-boxes, $S[0]$ and $S[1]$. Note that every $+$ should be an $\oplus$. After expanding and adding the key we have

$$\begin{array}{c|cc|c} a_0 & a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 & a_7 \end{array}. \text{ Let } S_0(a_0a_1a_2a_3) = b_0b_1, \quad S_1(a_4a_5a_6a_7) = b_2b_3.$$

S_0

a_0	a_1	a_2	a_3	b_0	b_1
0	0	0	0	0	1
0	0	0	1	1	1
0	0	1	0	0	0
0	0	1	1	1	0
0	1	0	0	1	1
0	1	0	1	0	1
0	1	1	0	1	0
0	1	1	1	0	0
1	0	0	1	0	0
1	0	0	0	1	1
1	0	1	0	1	0
1	0	1	1	0	1
1	1	0	0	0	1
1	1	0	1	1	1
1	1	1	0	1	1
1	1	1	1	1	0

S_1

a_4	a_5	a_6	a_7	b_2	b_3
0	0	0	0	0	1
0	0	0	1	1	0
0	0	1	0	0	1
0	0	1	1	0	0
0	1	0	0	1	0
0	1	0	1	0	1
0	1	1	0	1	1
0	1	1	1	1	1
1	0	0	0	1	1
1	0	0	1	1	0
1	0	1	0	0	0
1	0	1	1	0	1
1	1	0	0	0	1
1	1	0	1	0	0
1	1	1	0	0	0
1	1	1	1	1	1

We are interested in sums of a_i's and b_j's where the outputs are mostly 0's or mostly 1's. So for S_0, we compute the output of $x_0a_0 + x_1a_1 + x_2a_2 + x_3a_3 + x_4b_0 + x_5b_1$ ($x_i \in \{0,1\}$) where not all of x_0, x_1, x_2, x_3 are 0 and x_4, x_5 are not both 0. In the table below are those sums of a_i's and b_j's whose outputs of 0's and 1's are most unevenly distributed. The second column gives the 16 outputs from the 16 lines in the above table.

$$\begin{aligned}
a_2 + b_1 &= 1111\ 1111\ 0110\ 1101, && = 1, \quad p = 13/16, && \text{I} \\
a_0 + a_1 + a_2 + a_3 + b_0 + b_1 &= 1111\ 1111\ 1010\ 1111, && = 1, \quad p = 14/16, && \text{II} \\
a_2 + a_3 + b_0 &= 0011\ 1100\ 0000\ 0001, && = 0, \quad p = 11/16, && \text{III} \\
a_0 + a_2 + a_3 + b_0 &= 0011\ 1100\ 1111\ 1110, && = 1, \quad p = 11/16, && \text{IV}
\end{aligned}$$

Do the same thing for S_1.

$$\begin{aligned}
a_4 + a_6 + a_7 + b_3 &= 0100\ 0001\ 0000\ 0000, && = 0, \quad p = 14/16, && \text{V} \\
a_5 + a_6 + b_2 &= 0111\ 0111\ 1111\ 1101, && = 1, \quad p = 13/16, && \text{VI} \\
a_6 + b_2 + b_3 &= 0101\ 1111\ 0110\ 1011, && = 1, \quad p = 11/16, && \text{VII} \\
a_4 + a_5 + b_2 &= 0100\ 0100\ 0011\ 0001, && = 0, \quad p = 11/16, && \text{VIII}
\end{aligned}$$

Also $a_0 + a_1 + a_3 + b_0 = 0, 13/16$, but it's the same as I+II.
Recall 3-round Baby DES. Key: $k_0k_1k_2k_3k_4k_5k_6k_7k_8k_9$, key1: $k_0k_6k_8k_3k_7k_2k_9k_5$, key3: $k_9k_1k_0k_6k_8k_3k_5k_7$. First round expansion:

$$\begin{array}{c|cc|c} p_7+k_0 & p_4+k_6 & p_5+k_8 & p_6+k_3 \\ p_5+k_7 & p_6+k_2 & p_7+k_9 & p_4+k_5 \end{array} \text{ output : } b_0b_1b_2b_3 \overset{P4}{\nearrow} \begin{array}{cccc} b_1 & b_3 & b_2 & b_0 \\ \underline{+p_0} & \underline{p_1} & \underline{p_2} & \underline{p_3} \\ m_0 & m_1 & m_2 & m_3 \end{array}$$

Last round expansion:

$$\begin{array}{c|cc|c} c_7+k_9 & c_4+k_1 & c_5+k_0 & c_6+k_6 \\ c_5+k_8 & c_6+k_3 & c_7+k_5 & c_4+k_7 \end{array} \text{ output : } b'_0b'_1b'_2b'_3 \overset{P4}{\nearrow} \begin{array}{cccc} b'_1 & b'_3 & b'_2 & b'_0 \\ \underline{+m_0} & \underline{m_1} & \underline{m_2} & \underline{m_3} \\ c_0 & c_1 & c_2 & c_3 \end{array}$$

Relation I is $a_2 + b_1 = 1, p = 13/16$.
In the first round $a_2 + b_1 = (p_5 + k_8) + (m_0 + p_0) = 1, p = 13/16$.
In the last round $a_2 + b_1 = (c_5 + k_0) + (m_0 + c_0) = 1, p = 13/16$.
Adding we get $p_5+c_5+k_8+k_0+p_0+c_0 = 0$. What's the probability of that equation holding true? Either both of the above were 1: $(13/16)^2$, or both were 0: $(3/16)^2$. So the probability is $(13/16)^2 + (1 - 13/16)^2 \approx .70$.

$$k_0 + k_8 = p_0 + c_0 + p_5 + c_5, \quad p \approx .70 \quad \text{from I}$$

Relation II is $a_0 + a_1 + a_2 + a_3 + b_0 + b_1 = 1, p = 14/16$.
First round $p_7 + k_0 + p_4 + k_6 + p_5 + k_8 + p_6 + k_3 + p_3 + m_3 + p_0 + m_0 = 1, p = 14/16$.
Last round $c_7 + k_9 + c_4 + k_1 + c_5 + k_0 + c_6 + k_6 + c_3 + m_3 + c_0 + m_0 = 1, p = 14/16$.
Adding we get $p_0 + c_0 + p_3 + c_3 + p_4 + c_4 + p_5 + c_5 + p_6 + c_6 + p_7 + c_7 + k_1 + k_3 + k_8 + k_9 = 0$ with probability $(14/16)^2 + (1 - 14/16)^2 \approx .78$

$$\text{II : } \; k_1 + k_3 + k_8 + k_9 = p_0 + c_0 + p_3 + c_3 + p_4 + c_4 + p_5 + c_5 + p_6 + c_6 + p_7 + c_7, \; p \approx .78$$

$$\text{From relation III : } k_0 + k_3 + k_6 + k_8 = p_3 + c_3 + p_5 + c_5 + p_6 + c_6, \quad p \approx .57$$

$$\text{IV : } k_0 + k_3 + k_8 + k_9 = p_3 + c_3 + p_5 + c_5 + p_6 + c_6 + p_7 + c_7, \quad p \approx .57$$

$$\text{V : } k_8 + k_9 = p_1 + c_1 + p_4 + c_4 + p_5 + c_5 + p_7 + c_7, \quad p \approx .78$$

$$\text{VI : } k_2 + k_3 + k_5 + k_9 = p_2 + c_2 + p_6 + c_6 + p_7 + c_7, \quad p \approx .70$$

$$\text{VII : } k_5 + k_9 = p_1 + c_1 + p_2 + c_2 + p_7 + c_7, \quad p \approx .57$$

$$\text{VIII : } k_2 + k_3 + k_7 + k_8 = p_2 + c_2 + p_5 + c_5 + p_6 + c_6, \quad p \approx .57$$

Say you have an unfair coin with probabilities .78 and .22. How many times must you flip it before you decide which is the .78 side with 90% certainty? The answer is 5. If the probabilities are instead .57 and .43, how many times must you flip for 90% certainty? 83.

So if your key has 10 bits, you could use enough matched PT/CT pairs to feel certain that you got correct relations from I, II, V, VI. Then there would be 6 free variables, so you could use brute force on the 2^6 possibilities. Or you could use a lot more matched PT/CT and feel certain you got all eight relations. Then use brute force on the remaining 2^2 possibilities.

Statistics determines everything. For a given key and 80 PT/CT pairs, the odds are about .65 that you will get all eight relations right.

With more than three rounds, this linear cryptanalysis gets slightly more complicated, though not much. You need about 2^{47} known PT/CT pairs to solve for a DES key. This is faster than brute force. If the PT's are not random (like standard English) then you know something about the p_i's and you can make a CT-only attack. You need more than 2^{47}, however.

Differential cryptanalysis

This is an idea of Biham and Shamir. It can be used in an attempt to cryptanalyze cryptosystems like DES, RC5, and FEAL. It is usually a chosen PT attack, so it is usually unrealistic. You can use it if you have an enormous amount of known PT. With enough PT, you'll find ones you would have chosen. As with linear cryptanalysis, you use many PT/CT pairs to try to solve for the key.

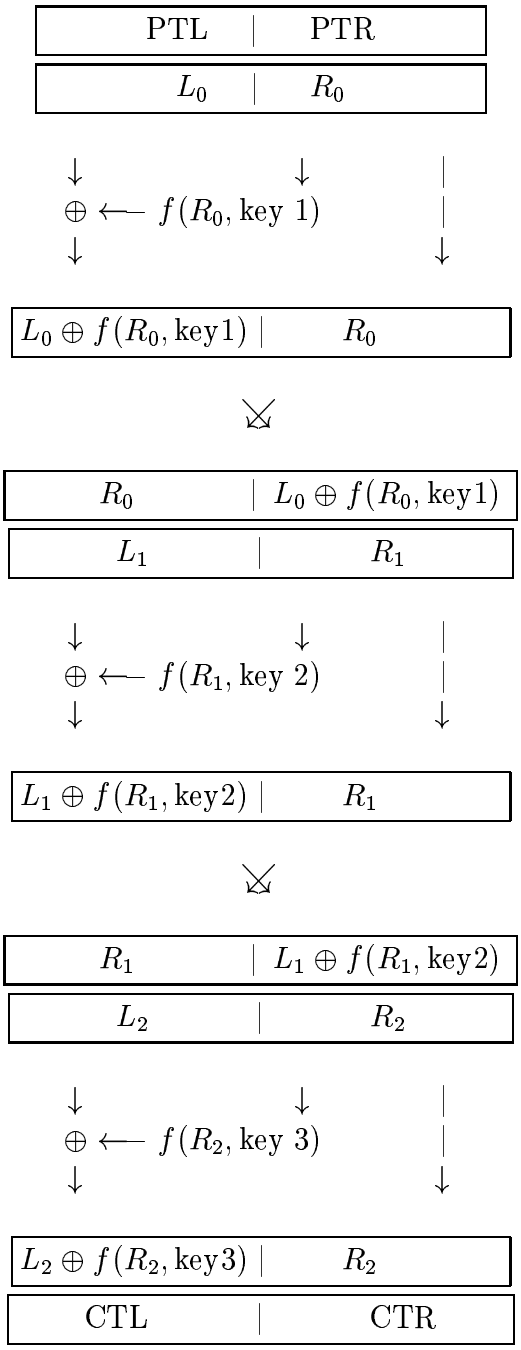
Typically you choose two PTs that differ at specified bits and are the same at the rest and look at the difference between the corresponding two CT's and deduce information about the key.

Here is how differential cryptanalysis could be used to cryptanalyze 3-round Baby DES. The idea is the same for cryptanalyzing 3-round DES.
Let's say a plaintext is PT=0001 0100 and the corresponding ciphertext is CT=0111 1111 and a second plaintext is PT*=0011 0100 and its corresponding ciphertext is CT*=0001 1100. Let PTR be the right half of PT, etc. Note PTR=PTR* but PTL $\neq$ PTL*. See the figure on the following page.

$$\begin{aligned} \text{CTL} = 0111 &= L_2 + f(R_2, \text{key3}) = R_1 + f(R_2, \text{key3}) \\ &= L_0 + f(R_0, \text{key 1}) + f(R_2, \text{key3}) \\ &= \text{PTL} + f(\text{PTR}, \text{key 1}) + f(\text{CTR}, \text{key3}) \\ &= 0001 + f(\text{PTR}, \text{key1}) + f(1111, \text{key3}). \end{aligned}$$

Similarly $\text{CTL}^* = 0001 = 0011 + f(\text{PTR}^*, \text{key1}) + f(1100, \text{key3})$.

So we have $\text{CTL} + \text{CTL}^* = 0111 + 0001 = 0110$ but also $\text{CTL} + \text{CTL}^* = 0001 + 0011 + f(1111, \text{key3}) + f(1100, \text{key3})$. Note that since $\text{PTR} = \text{PTR}^*$, two of the terms dropped out. So $f(1111, \text{key3}) + f(1100, \text{key3}) = 0100$.
In general $f(\text{CTR}, \text{key3}) + f(\text{CTR}^*, \text{key3}) = \text{PTL} + \text{PTL}^* + \text{CTL} + \text{CTL}^*$ and the right half of that equation is known.



Aside 1 on $f(\text{CTR}, key3)$. We have CTR= $c_4c_5c_6c_7$ and key3=$k_9k_1k_0k_6k_8k_3k_5k_7$. To do the function f you first expand CTR and add the key and get

$$\begin{array}{c|cc|c} c_7+k_9 & c_4+k_1 & c_5+k_0 & c_6+k_6 \\ c_5+k_8 & c_6+k_3 & c_7+k_5 & c_4+k_7. \end{array}$$

The first row is the input to S-box S_0 and the second row is the input to S_1. Let's denote the output of S_0 by ab and the output of S_1 by cd (each is a pair of bits, of course) Recall P4(abcd)=(bdca). So $f(\text{CTR}, key3) = bdca$.

Aside 2 on $f(\text{CTR}, key3)$. $P4^{-1}(wxyz) = (zwyx)$. You can $P4^{-1}$ or XOR in either order. So $P4^{-1}(f(\alpha) + f(\beta)) = P4^{-1}(f(\alpha)) + P4^{-1}(f(\beta))$ (where α and β are 4-bit strings).

From earlier we have $0100 = f(1111, key3) + f(1100, key3)$. Now $P4^{-1}(0100)$ is 0001. From Aside 2 we have

$00 = S_0(c_7c_4c_5c_6 + k_9k_1k_0k_6) + S_0(c_7^*c_4^*c_5^*c_6^* + k_9k_1k_0k_6)$

and $01 = S_1(c_5c_6c_7c_4 + k_8k_3k_5k_7) + S_1(c_5^*c_6^*c_7^*c_4^* + k_8k_3k_5k_7)$.

Let's look at the first one. We have

$00 = S_0(1111 + k_9k_1k_0k_6) + S_0(0110 + k_9k_1k_0k_6)$ so we want to find all four bit strings $----$ with $S_0(1111 + ----) + S_0(0110 + ----) = 00$

can− di− date	$\alpha =$ 1111⊕ cand	$\beta =$ 0110⊕ cand	$S_0(\alpha)$	$S_0(\beta)$	$S_0(\alpha) \oplus S_0(\beta)$
∗0000	1111	0110	10	10	00
0001	1110	0111	11	00	11
∗0010	1101	0100	11	11	00
∗0011	1100	0101	01	01	00
0100	1011	0010	01	00	01
∗0101	1010	0011	10	10	00
0110	1001	0000	11	01	10
0111	1000	0001	00	11	11
1000	0111	1110	00	11	11
∗1001	1110	1111	10	10	00
∗1010	0101	1100	01	01	00
∗1011	0100	1101	11	11	00
∗1100	0011	1010	10	10	00
1101	0010	1011	00	01	01
1110	0001	1000	11	00	11
1111	0000	1001	01	11	10

We find $k_9k_1k_0k_6$ is in the set $\{0000, 0010, 0011, 0101, 1001, 1010, 1011, 1100\}$.

We can do the same kind of thing to find all $k_8k_3k_5k_7$ with the property that $S_1(1111 + k_8k_3k_5k_7) + S_1(1001 + k_8k_3k_5k_7) = 01$. We can write a computer program to do this for us. We find $k_8k_3k_5k_7$ is in the set $\{1110, 1100, 1010, 1000, 0110, 0101, 0100, 0011, 0010, 0000\}$.

Now we use a second pair of PT's with PTR= PTR* and their corresponding CT's. The new pair of pairs is PT=0001 0000, CT=0110 1111, PT*= 0011 0000, CT*=0100 0100. We add the four right hand sides together and get $0000 = f(\cdot) + f(\cdot)$. Then we $P4^{-1}$ this and get 0000 again. The first two $00 = S_0(\cdot) + S_0(\cdot)$ and the second two $00 = S_1(\cdot) + S_1(\cdot)$. We have CTR=1111 and we extend that and get $\frac{1111}{1111}$. We also have CTR*=0100 and we extend that and get $\frac{0010}{1000}$.

Thus we want to find $k_9k_1k_0k_6 = ----$ such that $S_0(1111, ----) + S_0(0010, ----) = 00$ and $k_8k_3k_5k_7 = ----$ such that $S_1(1111, ----) + S_0(1000, ----) = 00$. We run this through a computer program and find that $k_9k_1k_0k_6$ is in $\{0110, 1011\}$ and that $k_8k_3k_5k_7$ is in $\{1010, 1101, 0000, 0010, 0100, 0101, 0011, 0111\}$. But these 4-bit keybit

strings must be in the two large sets from the first two pairs of matched PT/CT. So they must be in the intersections. Thus we see that $k_9k_1k_0k_6$ is in $\{1011\}$ and $k_8k_3k_5k_7$ is in $\{1010, 0000, 0010, 0100, 0101, 0011\}$.

Now we can pick more pairs of matched plaintext and ciphertext where the right halves of the two plaintexts are the same and narrow down the possibilities even more. We continue until we find out what these keybits are. Eventually we discover that $k_8k_3k_5k_7 = 0010$. So we have $k_0k_1k_2k_3k_4k_5k_6k_7k_8k_9 = 10?0?11001$. To determine the two unknown bits, we can try all four possibilities and see, for example, which send PT=0001 0100 to CT=0111 1111.

With 3 rounds, the correct keybits appeared in every candidate set. With more rounds, things don't work out so nicely. You come up with sets of candidates which do not all necessarily contain the correct keybits. You will, however, know the keybits will appear in some given percentage of these candidate sets. The incorrect keystrings will appear far less frequently so you need to search for the keystring appearing with that frequency. Then you can use brute force to find the other bits.

Other attacks on DES

Recall that a block of plaintext has 64 bits. This could be 8 ASCII characters. There are versions of ASCII where the 8th bit is a parity check. It's determined by the first 7 bits. This leads to a CT only attack. Given a block of CT, decrypt it with all 2^{56} possible keys to get tentative PT's. Keep only the keys that give ASCII PT. Note that there are eight, 8-bit substrings of a given PT. Check each to see if the parity bit is correct. On average, it will only be correct $1/2^8$ of the time. So you will be left with $2^{56}/2^8 = 2^{48}$ candidate keys.

Do this again for the second CT. After that there will be 2^{40} candidate keys. With 7 or 8 different blocks of CT, you should get the key. Note this does not take much longer than a brute force attack with known PT.

Meet in the middle attack for double DES

This is an idea of Diffie and Hellman. Let's add some notation. Encryption in the usual way with DES and a 56 bit key will be denoted $CT = E_{key}(PT)$ and decryption by $PT = D_{key}(CT)$. For double DES you have $CT = E_{key1}(E_{key2}(PT))$ and $PT = D_{key2}(D_{key1}(CT))$.

For this attack you need 2 known PT/CT pairs, call them PT_1/CT_1 and PT_2/CT_2. Encrypt PT_1 with single DES with every key and store the outputs. Decrypt CT_1 with single DES with every key and store the outputs. Find key pairs $key_{1,i}, key_{2,i}$ where $E_{key_{1,i}}(PT_1) = D_{key_{2,i}}(CT_2)$. This gives you a collection of possible key pairs $(key_{1,i}, key_{2,i})$. For each possible key pair compute $E_{key_{1,i}}(PT_2)$ and $D_{key_{2,i}}(CT_2)$. Those will probably only agree once, at the correct pair, i.e. at $key_{1,j} = key1$ and $key_{2,j} = key2$.

This involves a lot of storage, but it shows that double DES is not much more secure than single DES. For that reason, many use triple DES. You can use a meet in the middle attack on triple DES to show that triple DES with 3 keys is not much better than triple DES with 2 keys and 2 keys are easier to agree on so people tend to do: $CT = E_{key1}(D_{key2}(E_{key1}(PT)))$.