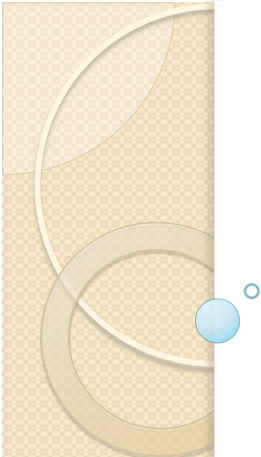


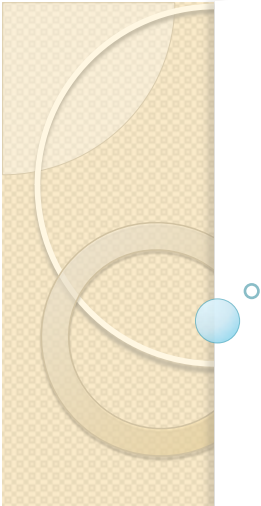
✓ 2.2 Random J. Protocol-Designer has been told to design a scheme to prevent messages from being modified by an intruder. Random J. decides to append to each message a hash of that message. Why doesn't this solve the problem? (We know of a protocol that uses this technique in an attempt to gain security.)

Anyone who knows which hash function is being used can forge a message.



✓2.3 Suppose Alice, Bob, and Carol want to use secret key technology to authenticate each other. If they all used the same secret key K , then Bob could impersonate Carol to Alice (actually any of the three can impersonate the other to the third). Suppose instead that each had their own secret key, so Alice uses K_A , Bob uses K_B , and Carol uses K_C . This means that each one, to prove his or her identity, responds to a challenge with a function of his or her secret key and the challenge. Is this more secure than having them all use the same secret key K ? (Hint: what does Alice need to know in order to verify Carol's answer to Alice's challenge?)

Without the use of public key technology, they will still need to know each other's keys to do the verification, which means they can still forge each other's messages.



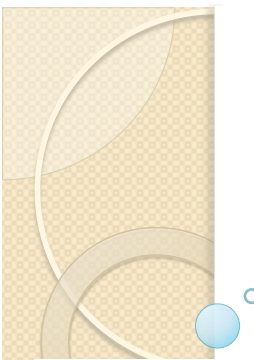
✓ 2.4 As described in §2.6.4 *Downline Load Security*, it is common, for performance reasons, to sign a message digest of a message rather than the message itself. Why is it so important that it be difficult to find two messages with the same message digest?

You can forge a signature on any message with the same hash as one that has been legitimately signed.



2.5 What's wrong with the protocol in §2.4.4 *Authentication*? (Hint: assume Alice can open two connections to Bob.)

Alice doesn't need to know K_{AB} ; when Bob challenges with r_B , Alice just opens a second connection to Bob and challenges him with r_B , then uses his response to respond to his first-connection challenge. She can abort the second connection.



2.6 Assume a cryptographic algorithm in which the performance for the good guys (the ones that know the key) grows linearly with the length of the key, and for which the only way to break it is a brute-force attack of trying all possible keys. Suppose the performance for the good guys is adequate (e.g., it can encrypt and decrypt as fast as the bits can be transmitted over the wire) at a certain size key. Then suppose advances in computer technology make computers twice as fast. Given that both the good guys and the bad guys get faster computers, does this advance in computer speed work to the advantage of the good guys, the bad guys, or does it not make any difference?

If the good guys keep the same key size, then it's an advantage for the bad guys. But if the good guys double the key size, doubling their work while still taking the same amount of time as it used to, then it's much worse for the bad guys, since their work squares.

For example, suppose the good guys were using an n -bit key. With a computer twice as fast, they can use a $2n$ -bit key with the same performance, since doubling the length of the key just doubles their work. However, the bad guys have 2^n times as much work to do with a key twice as long, so it works to the advantage of the good guys.

2. This problem explores the use of a one-time pad version of the Vigenere cipher. In this scheme, the key is a stream of random numbers between 0 and 26. For example, if the key is 3 19 5..., then the first letter of plaintext is encrypted with a shift of 3 letters, the second with a shift of 19 letters, and so on.

a. Encrypt the plaintext sendmoremoney with the key stream 9 0 1 7 23 15 21 14 11 11 2 8 9.

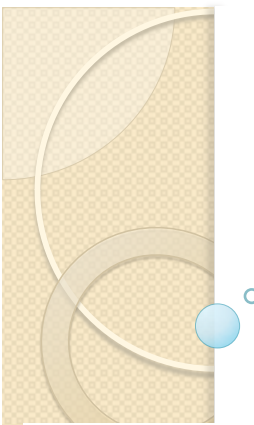
b. Using the ciphertext produced in part a, find a key so that the ciphertext decrypts to the plaintext cashnotneeded.

2. a.

s	e	n	d	m	o	r	e	m	o	n	e	y
18	4	13	3	12	14	17	4	12	14	13	4	24
9	0	1	7	23	15	21	14	11	11	2	8	9
1	4	14	10	9	3	12	18	23	25	15	12	7
B	E	O	K	J	D	M	S	X	Z	P	M	H

b.

c	a	s	h	n	o	t	n	e	e	d	e	d
2	0	18	7	13	14	19	13	4	4	3	4	3
25	4	22	3	22	15	19	5	19	21	12	8	4
1	4	14	10	9	3	12	18	23	25	15	12	7
B	E	O	K	J	D	M	S	X	Z	P	M	H



✓ 3.3 How many DES keys, on the average, encrypt a particular plaintext block to a particular ciphertext block?

There are 2^{56} possible keys and 2^{64} possible ciphertext blocks for a particular plaintext block. So only about $2^{56-64} = 1/256$ of the possible ciphertext blocks can be obtained by with a DES key.

✓ 3.6 Are all the 56 bits of the DES key used an equal number of times in the K_i ? Specify, for each of the K_i , which bits are not used.

Counting all the rounds, there are $8 \times 16 = 128$ unused bits, and 128 is not a multiple of 56, so the bits of the key are used unequally. Of the 28 bits of C_i , bits 9, 18, 22, and 25 are not used in K_i . Of the 28 bits of D_i , bits 35, 38, 43, and 54 are not used in K_i . Translating this back to the bits of C_0 and D_0 by round we have the following unused bits:

Bits unused once: 1 3 10 12 14 16 25 27 31 33 46 48

Bits unused twice: 5 6 7 8 18 20 21 23 29 35 37 39 41 42 44 50 51 52 54 56

Bits unused three times: 2 4 13 15 17 19 22 28 30 32 34 36 38 40 43 45 47 49 53 55

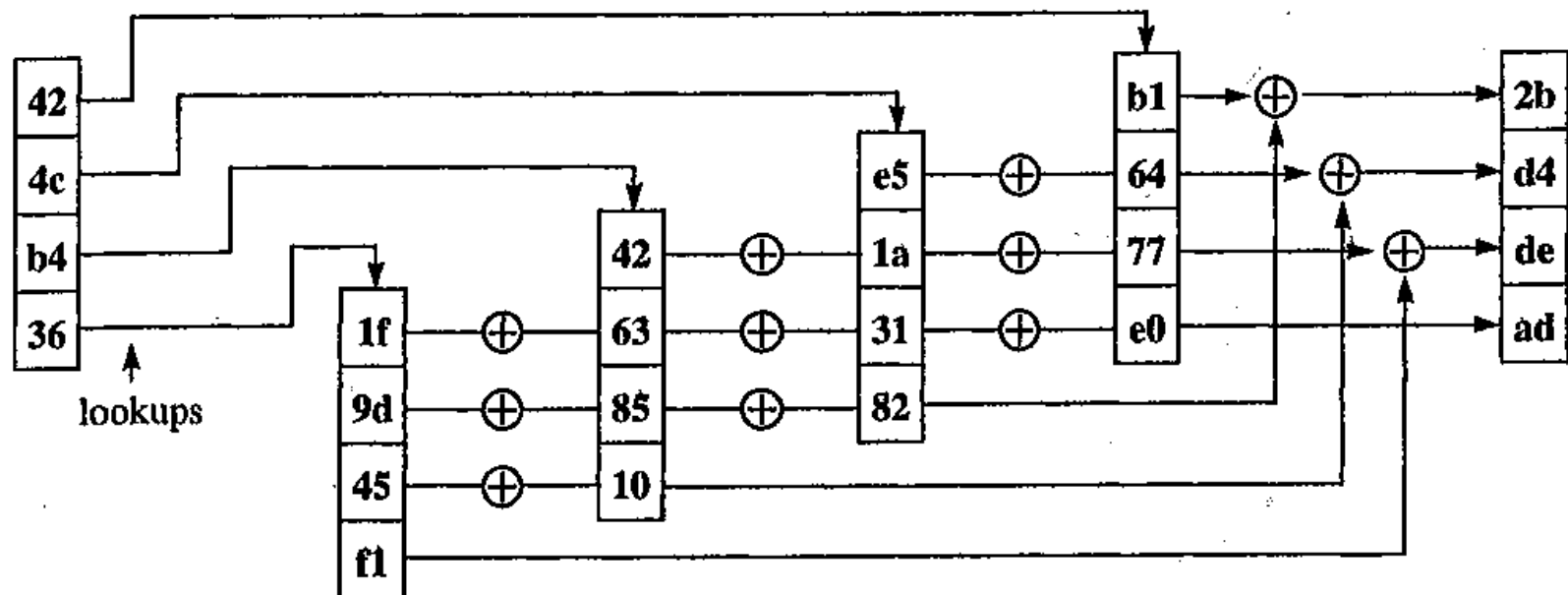
Bits unused four times: 9 11 24 26

round	unused bits from C_0				unused bits from D_0			
1	10	19	23	26	36	39	44	55
2	11	20	24	27	37	40	45	56
3	13	22	26	1	39	42	47	30
4	15	24	28	3	41	44	49	32
5	17	26	2	5	43	46	51	34
6	19	28	4	7	45	48	53	36
7	21	2	6	9	47	50	55	38
8	23	4	8	11	49	52	29	40
9	24	5	9	12	50	53	30	41
10	26	7	11	14	52	55	32	43
11	28	9	13	16	54	29	34	45
12	2	11	15	18	56	31	36	47
13	4	13	17	20	30	33	38	49
14	6	15	19	22	32	35	40	51
15	8	17	21	24	34	37	42	53
16	9	18	22	25	35	38	43	54

3.11 Show that DES encryption and decryption are identical except for the order of the 48-bit keys. Hint: running a round backwards is the same as running it forwards but with the halves swapped (see §3.3.4 *A DES Round*), and DES has a swap after round 16 when run forwards (see §3.3.1 *DES Overview*).

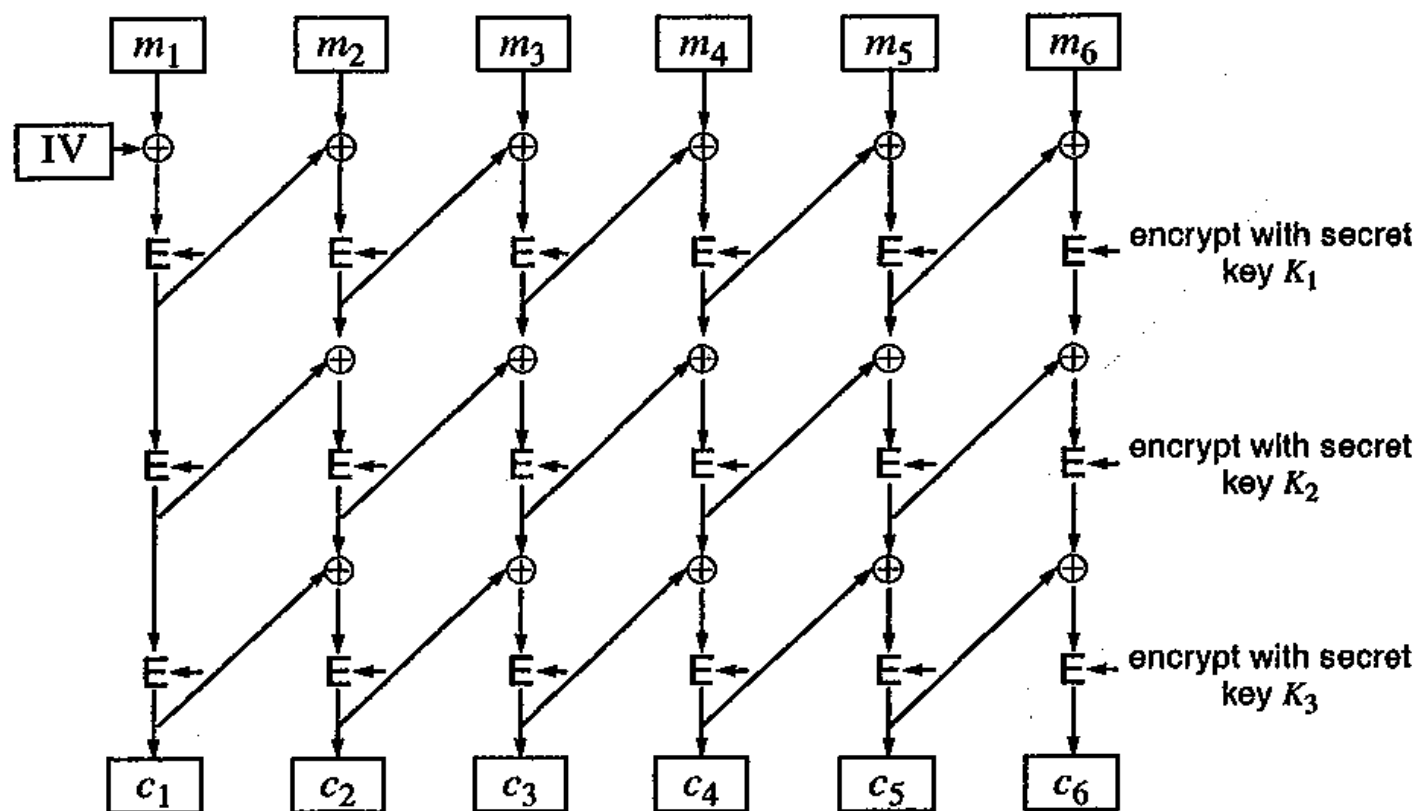
It's sufficient to show that if we follow a DES encryption with another DES encryption but with the 48-bit keys in reverse order, we get back with what we started with. Suggestively call the two encryptions E and D. We will try to collapse pieces at the end of E with corresponding ones at the beginning of D until nothing is left of D and E. First, we see that the final permutation of E is reversed by the initial permutation of D. Now we have Round 16 of E followed by a swap of halves followed by Round 1 of D (which uses the same 48-bit key because the key order is reversed between E and D). By the hint, a swap of halves followed by a round is the same as the reverse of that round followed by a swap of halves. So Round 16 of E followed by a swap of halves followed by Round 1 of D collapses to a swap of halves. Continuing with lower rounds of E and the matching higher rounds of D eventually collapses all the Rounds, leaving only the swap of halves from E. But this now collapses with the swap of halves from D, leaving only the initial permutation from E and the final permutation from D. These collapse to nothing, and we're done!

3.12 Verify the *MixColumn* result in Figure 3-25 by using the same method (in conjunction with Figure 3-28's table) to compute *InvMixColumn* of the *MixColumn* result and checking that you produce the *MixColumn* input.

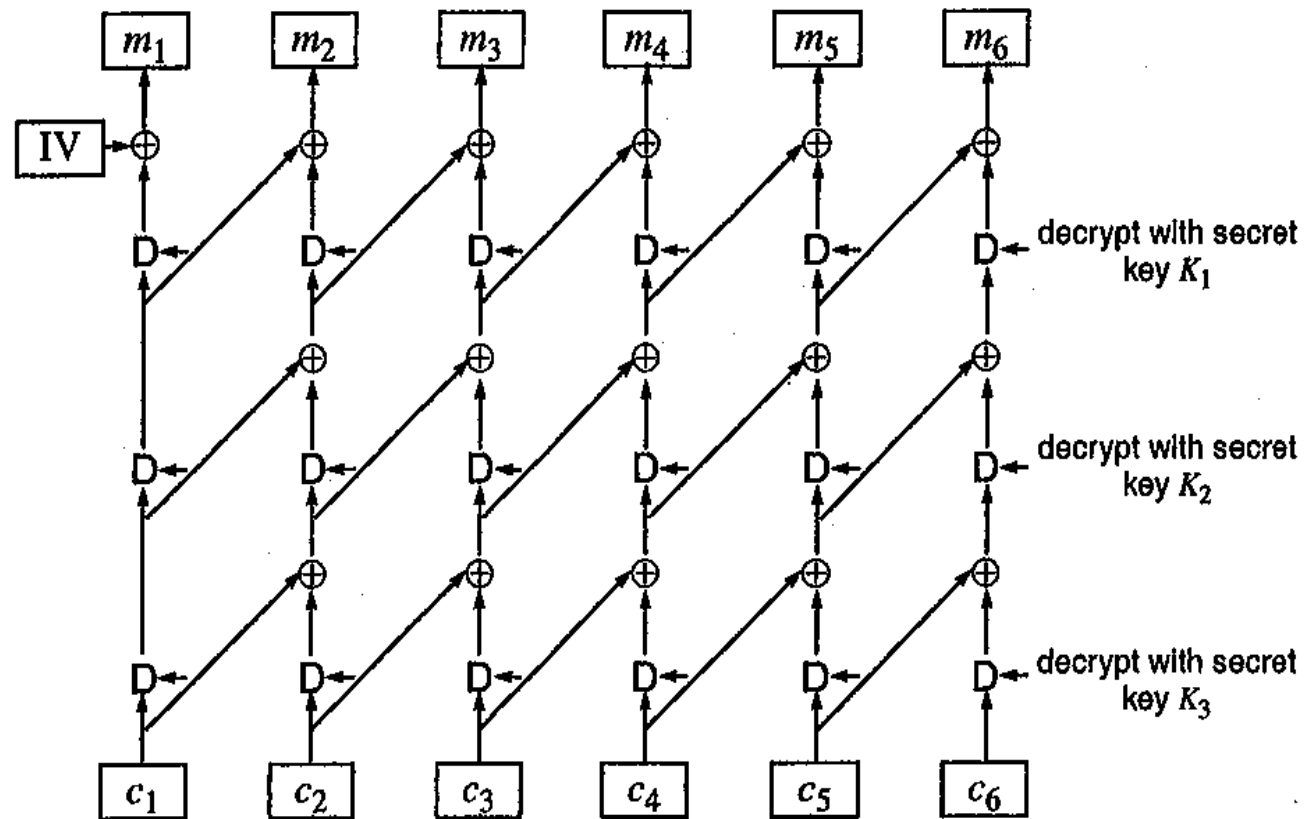


4.5 Let's assume that someone does triple encryption by using *EEE* with CBC on the inside. Suppose an attacker modifies bit x of ciphertext block n . How does this affect the decrypted plaintext?

Triple encryption using EEE with CBC on the inside is three successive CBC encryptions:



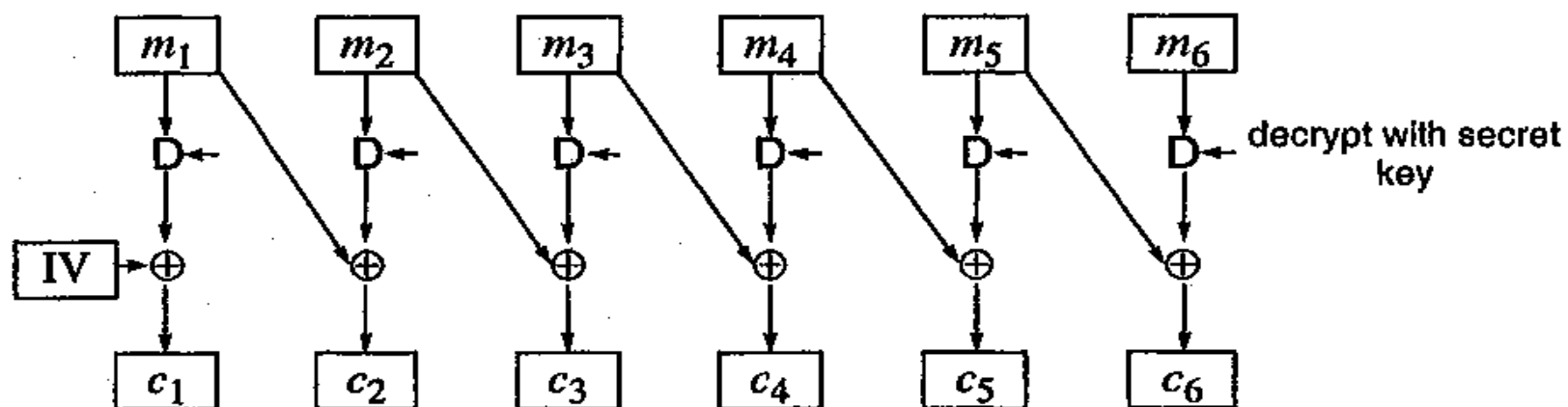
Decryption is the inverse operation:

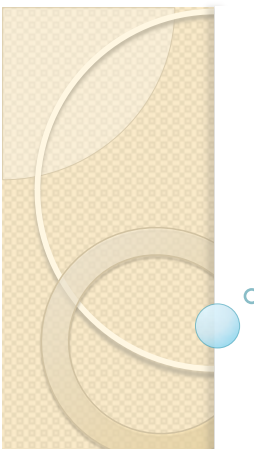


From this it can be seen that a modification to ciphertext block n propagates to plaintext blocks n through $n+3$. No other plaintext blocks are affected.

4.6 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?


It would certainly work, in the sense of allowing encryption and decryption of messages. In this encryption scheme, block n of the plaintext is \oplus 'd with the output of the D step for block $n+1$ to get block $n+1$ of ciphertext.





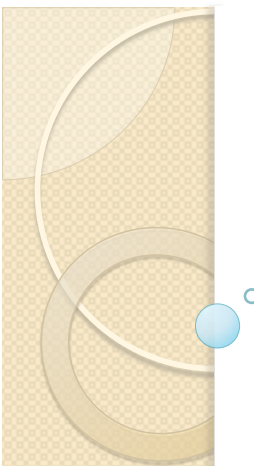
One problem with this is that if Trudy knows the plaintext and ciphertext for a set of messages, she can mix and match the blocks of those messages almost as easily as with ECB. The reason is that block n of plaintext \oplus 'd with block $n+1$ of ciphertext is D of block $n+1$ of plaintext, and once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

More seriously, since block $n+1$ of ciphertext depends only on blocks n and $n+1$ of plaintext, 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 \oplus 'd with block $n+1$ of ciphertext to get block n of plaintext.



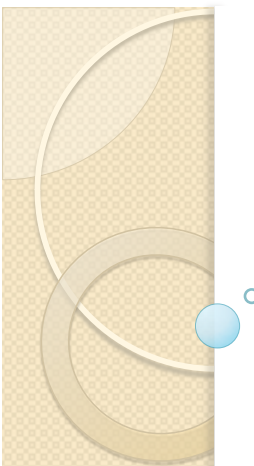
5. In 8-bit CFB mode, if one bit of a ciphertext block is corrupted during transmission, how many blocks in total will be affected in decryption assuming DES is used?

The error will affect the decryption of the ciphertext block. In addition, the error block will stay in the IV for another $64/8=8$ blocks. In total, 9 blocks will be affected.



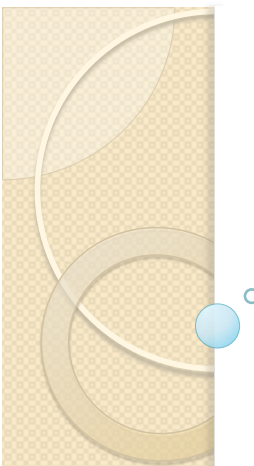
✓ 5.2 Message digests are reasonably fast, but here's a much faster function to compute. Take your message, divide it into 128-bit chunks, and \oplus all the chunks together to get a 128-bit result. Do the standard message digest on the result. Is this a good message digest function?

No. It's fairly easy to generate another message with the same 128-bit \oplus .



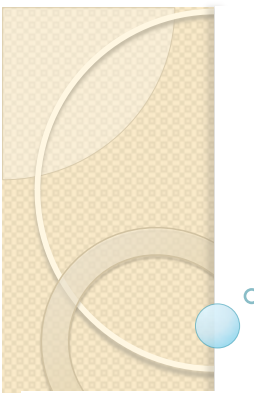
5.12 Assume a good 128-bit message digest function. Assume there is a particular value, d , for the message digest and you'd like to find a message that has a message digest of d . Given that there are many more 2000-bit messages that map to a particular 128-bit message digest than 1000-bit messages, would you theoretically have to test fewer 2000-bit messages to find one that has a message digest of d than if you were to test 1000-bit messages?

No. The expected number of messages you'd need to try is 2^{128} in either case.



5.16 We mentioned in §4.2.2 *Computing a MIC with a Hash* that using $\text{MD4}(K_{AB}|m)$ as a MIC is not secure. This is not a problem if MD2 is used instead of MD4. Why is that the case?

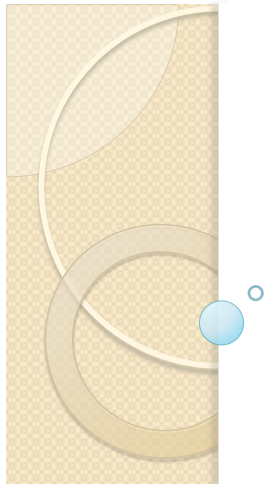
MD2 computes a message checksum based on the entire padded message, which it appends to the padded message before the final pass that computes the hash. So if you want to start with $\text{MD2}(K_{AB}|m)$ and incrementally compute $\text{MD2}(K_{AB}|m|n)$ where n is your part of the message, n would have to start with the padding and message checksum for $K_{AB}|m$, but you don't know the message checksum and you can't compute it because you don't know K_{AB} .



5.18 In §5.2.3.1 *Generating a One-Time Pad*, we generate a pseudo-random stream of MD-sized blocks. This stream must eventually repeat (since only $2^{\text{MD-size}}$ different blocks can be generated). Will the first block necessarily be the first to be repeated? How does this compare to OFB (see Chapter 4 *Modes of Operation* Homework Problem 2)?

The argument for Homework Problem 4.2 was based on the ability to reverse each step. Since message digests are not invertible, that argument fails for this problem, and we can't easily predict which block will be the first to repeat.

Answer to Problem 4.2: IV will be the first block to be repeated. To see this, note that the previous block to any given block must be the decryption of the given block. So if two blocks are equal, their respective previous blocks are also equal (unless one of them doesn't have a previous because it is the first—namely IV)



5.19 How do you decrypt the encryption specified in §5.2.3.2 *Mixing In the Plaintext*?

Since you know K_{AB} and IV and the c_i s, you can compute the b_i s. Then $p_i = c_i \oplus b_i$.



7. Steganography for fun: What's the message embedded in the following letter?

3rd March

Dear George,

Greetings to all at Oxford. Many thanks for your letter and for the Summer examination package. All Entry Forms and Fees Forms should be ready for final despatch to the Syndicate by Friday 20th or at the very latest, I'm told, by the 21st. Admin has improved here, though there's room for improvement still; just give us all two or three more years and we'll really show you! Please don't let these wretched 16t proposals destroy your basis O and A pattern. Certainly this sort of change, if implemented immediately, would bring chaos.

Sincerely yours.

your package ready Friday 21st room three Please destroy this immediately