Solutions Manual

Information Security: Principles and Practice 2nd edition

by Mark Stamp

April 25, 2011

A Note to Instructors

The solutions here are reasonably complete, although not every problem has been solved in every detail. I've included additional comments and references where it seemed appropriate.

Please do not post these solutions online (or otherwise distribute). It may be inevitable that solutions will be available via Google, but I'd prefer that they not be mine.

Chapter 1

1. Confidentiality: In the banking example, Bob doesn't want Alice to know how little money he has in his account. Integrity: The bank (and customers) are concerned that their account balances do not change in an unauthorized manner. Availability is crucial for any online business.

   a. See the text.

   b. Confidentiality without integrity is of little use, since you would not be able to trust the data.

   c. Any transaction that is not secret, for example, the transfer of funds between federal reserve banks in the U.S.

   d. Availability is critical for any online business, such as amazon.com.

2. The bank's primary concern is with the integrity of transactions, while it's customers probably have roughly equal concern with both.

3. a. For example, payment of fees requires confidentiality of credit card information.

   b. Integrity is required for financial transactions and when actually playing games.

   c. If the service is not available on demand, it won't be too popular with users.

4. a. For securing financial transactions and to prevent cheating when making moves.

   b. Users need to authenticate to use the service, and they need to be restricted in their actions once they have been authenticated.

   c. The transactions occur over a network, and security protocols are required for all secure transactions over a network.

   d. Yes. If the software is subject to attack, then the financial information is vulnerable. Also, viruses and other malware is a concern for everyone involved.

5. a. Privacy concerns arise in many contexts. For example, a patient is likely to be concerned about the privacy of medical records.

   b. Confidentiality—in the sense that it is used in this question—is required of a government employee who handles classified information. Others who must protect confidentiality include lawyers, doctors, and priests, among many others.

6. a. The tags could reveal a great deal of information about you.

   b. For example, a terrorist could set a bomb to explode when a particular patter of RFID tag information is detected—a pattern that could be highly correlated with a particular target.

7. a. We'll see in later chapters that protocols are similar to crypto in this sense.

   b. Firewalls, intrusion detection, and, in general, any strategy that relies on "defense in depth."

8. a. In essence, the paper says something like, "Public key crypto ought be possible, but we don't know how to do it."

   b. It is the Diffie-Hellman key exchange described in Chapter 4 of this book.

   c. It might sound plausible, but it is not a feasible approach (see the paper, "On the (im)possibility of obfuscating programs").

9. a. See Figure 6.1.

   b. The German Navy had a limited number of refueling tankers (for their submarines). The only source of information on the whereabouts of these tankers was Enigma decrypts. The Allies sank these tankers soon after the Americans entered the war, and it severely limited the range of German U-boats.

10. a. They could have changed to another cipher or they could have continued to use the Enigma, but used it to send false information.

    b. They had a strong belief in the invincibility of the machine. Another important factor is that the Nazi system was not one that encouraged people to raise such ticklish questions. Military people with knowledge about ciphers were probably the most likely to realize that something was amiss, but these very same people were also the most likely to suffer in the event that problems were discovered. In other words, would you want to be the person in the Nazi cipher bureau to tell Hitler that his main cipher system was fatally flawed?

11. a. Some laptops have a thumbprint reader.

    b. Many companies require that you have a badge, which is used for authentication.

    c. An ATM card requires both something you have (the card) and something you know (your PIN).

12. a. CAPTCHAs are everywhere, so this is easy.

    b. Most CAPTCHAs involve recognizing distorted text. For these, you could use an optical character recognition (OCR) system.

    c. You could pay people to solve CAPTCHAs for you.

    d. Most widely-used modern CAPTCHAs are fairly strong—user friendlyness (or not) is a matter of taste.

    e. Well, I don't know about you, but I think they are annoying. To add insult to injury, they are, IMHO, used far too often, and they are usually overly complex when considering the sensitivity of the application.

13. a. Least secure, most user-friendly.

    b. Slightly more secure, but less user-friendly.

    c. Much more secure, but far less user-friendly.

d. The most secure (since Trudy, posing as Alice, would not know why the protocol had failed), but also the most user-unfriendly. It's often the case that the more secure you make something, the less user-friendly it becomes. Much of real-world security engineering is a matter of balancing the competing demands of security and user-friendliness.

14. a. It's not easy to find examples of such attacks.

    b. Steal the entire ATM machine.

15. a. Some percentage of the bugs will affect the security of the system.

    b. Bugs might weaken security or provide a way for Trudy to attack the system. Bugs often lead to unintended "features" and Trudy could take advantage of such a feature.

    c. Bugs often result in unintended "features" and Trudy might be able to take advantage of one of these to break the security.

16. a. Well, I use a Mac, and it's not common to have such problems. This is not to say that Mac is inherently more secure than Windows. We'll have more to say about this issue in later chapters.

    b. Malware could be used to obtain commercial secrets by infecting computers within a particular company—unscrupulous rival companies might pay for such info. Or malware could be used to "take over" computers so that the machines could be used, for example, to send spam email. Then a spammer could pay the virus writer to gain access to compromised machines.

17. a. Find a real-world example of a salami attack. These are discussed in Chapter 12.

    b. There is a bug in the attack—the decimal point is in the wrong place so they steal large and noticeable sums instead of small slices.

18. a. Most applications software today is closed source.

    b. Linux is a good example.

    c. Look at the source code.

    d. Reverse engineer the code (disassemble it, debug it, etc.).

    e. Look at the source code.

    f. In principle, she could reverse engineer it, but more likely, she would complain to the company that sells the software.

    g. It's debatable, as we'll see when we discuss the topic in Chapter 12.

19. a. Windows XP is said to have 40,000,000 lines of code, and it emphasizes features over security. There have been many security issues with XP.

    b. IPSec is a good example (discussed in Chapter 10).

20. a. Someone who buys the book might redistribute copies to other people.

b. This is actually very difficult. Special "readers" like the Kindle are one reasonable approach.

c. A special purpose device such as Kindle can still be broken by a really motivated attacker.

21. a. On their summary slide they mention the following:
   - Exploited physical security holes
   - Reverse engineered the CharlieTicket
   - Wrote code to analyze & generate magcards
   - Wrote a toolchain for analyzing 13.56MHz RFID transactions using the USRP+GNUradio
   - Attacked problems with the MIFARE Classic cards
   - Wrote brute forcer-generator to crack keys on an FPGA
   - Developed software to reduce MQ to SAT, allowing key recovery
   - Wrote code to read and clone MIFARE cards (given the key)

   b. You could make a reasonable argument either way. However, the Boston transit authority had been notified well in advance, and the slides had appeared online (as I recall), so I would argue that the legal action accomplished nothing other than greatly increasing the fame of the authors.

   c. War dialing refers to dialing lots of phone numbers looking for accessible modems. War driving refers to driving around looking for open (or poorly-secured) Wifi networks. War carting is an amusing part of the PowerPoint presentations.

   d. Nice!

Chapter 2

1. a. See text.

   b. Unfortunately, there are many such examples. For one, the crypto used in GSM was designed in violation of Kerckhoffs' Principle. The crypto algorithms are weak.

   c. The design of any security features should be open and available for scrutiny.

2. a. It is a simple substitution, and it is broken using frequency counts of symbols.

   b. They find a pirate's buried treasure.

3. The plaintext is "SpongeBob SquarePants."

4. The shift is 22 (or, equivalently, -4) and the plaintext is "you are terminated."

5. a. The answer is 2^{47} seconds $\approx 4.462 \times 10^6$ years.

   b. The answer is 2^{71} seconds $\approx 74.872 \times 10^{12}$ years.

   c. The answer is 2^{215} seconds $\approx 1.67 \times 10^{57}$ years.

6. To unscramble, use the permutation 4,9,1,5,7,10,2,6,3,8. For example, the first line unscrambles to "If at first you don't succeed, try and try again."

7. One improvement would be to use a permutation only once (or a very limited number of times). The codebook could also be more comprehensive to avoid giving away too much context.

8. a. Confusion obscures the relationship between plaintext and ciphertext while diffusion spreads the plaintext statistics throughout the ciphertext.

   b. Both the simple substitution and one-time pad.

   c. Double transposition.

   d. The ciphers from the election of 1876.

9. The key is: QGZAFOLBVMKJYWTCRHXPDUENIS and the plaintext is from *Alice in Wonderland* (with punctuation added):

   "The time has come," the Walrus said,
   "To talk of many things:
   Of shoes—and ships—and sealing wax—
   Of cabbages—and kings—
   And why the sea is boiling hot—
   And whether pigs have wings."

   "But wait a bit," the Oysters cried,
   "Before we have our chat;

For some of us are out of breath,
And all of us are fat!"
"No hurry!" said the Carpenter.
They thanked him much for that.

"A loaf of bread," the Walrus said,
"Is what we chiefly need:
Pepper and vinegar besides
Are very good indeed—
Now, if you're ready, Oysters dear,
We can begin to feed."

10. The key is ZGYHXIWJVKULTMSARBQCPDOENF and the plaintext is from *Alice in Wonderland*: Never imagine yourself not to be otherwise than what it might appear to others that what you were or might have been was not otherwise than what you had been would have appeared to them to be otherwise.

11. The key is KFAZSROBCWDINUELTHQGXVPJMY and the plaintext is from a speech by President Franklin D. Roosevelt shortly after the attack on Pearl Harbor: The United States was at peace with that nation and at the solicitation of Japan was still in conversation with its government and its emperor looking toward the maintenance of peace in the Pacific. Indeed, one hour after Japanese air squadrons had commenced bombing in Oahu, the Japanese ambassador to the United States and his colleague delivered to the Secretary of State a formal reply to a recent American message. While this reply stated that it seemed useless to continue the existing diplomatic negotiations it contained no threat or hint of war or armed attack.

12. Programming problem.

13. Programming problem.

14. The ciphertext is LEALETHRAWERGTOE.

15. Put the ciphertext in a 7×10 array. Then the letters of "there" will all appear (in scrambled order) in one row. This gives a start on the column permutation. Once the column perms are known, the row perms are easily determined. The answer is a quote from President Kennedy: "There are some who say that communism is the wave of the future. Let them come to Berlin."

16. Divide and conquer—try to undo the column permutations first.

17. After the first columnar transposition, we write out the intermediate ciphertext

```
tkatawacdatn
```

and after the second transposition, we have the ciphertext

TCNKWATADAAT.

18. The words "shriek" and "strike" work. Perhaps the best way to solve this is to find all 6-letter dictionary words that can be made from the allowed letters. Then any pair that XOR to the same thing as KHHLTK XOR KTHLLE are possible solutions.

19. a. Convert the plaintext "thrill" to binary and also convert the ciphertext KITLKE to binary. Then XOR these together to recover the key. In this case we have

	t	h	r	i	l	l
plaintext:	111	001	101	010	100	100
ciphertext:	011	010	111	100	011	000
key:	100	011	010	110	111	100
	l	k	i	s	t	l

b. For "tiller" we have

	t	i	l	l	e	r
plaintext:	111	010	100	100	000	101
ciphertext:	011	010	111	100	011	000
key:	100	000	011	000	011	101
	l	e	k	e	k	r

20. Let x be the 64-bit key. One approach would be to simply repeat x 16 times. However, this would be equivalent to using a one-time pad 16 times so it would be very insecure. A better idea would be to design a function f that produces a 64-bit output and use $x, f(x), f(f(x)), \ldots$. Of course, the security would depend on the choice of the function f. In Chapter 3 we'll discuss stream ciphers, which are used to stretch a short key into a long stream of bits that can be used like a one-time pad.

21. One simple idea would be to specify a complete codebook, then given a key, XOR the key with all elements of the ciphertext. For small codebooks (say, 4 or 8 bits), this would be inherently insecure and it would be impractical for large codebooks of, say, 64 bit blocks (since it would be impossible to store such a big codebook).

22. Once upon a time or maybe twice

23. From the hint, we have that $7 = 19a+b \pmod{26}$ and $4 = 14a+b \pmod{26}$. Solving, we find the encryption function is $f(x) = 11x + 6 \pmod{26}$ which implies the decryption function is $f^{-1}(x) = 19(x - 6) \pmod{26}$. The plaintext message is: If you bow at all bow low.

24. The plaintext is: Spoon feeding in the long run teaches us nothing but the shape of the spoon.

25. a. The probability of XX is $3/4 \cdot 3/4 = 0.5625$ and YY has probability $1/16 = 0.0625$, so the total probability of a match is 0.625.

b. 5two messages—one encrypted and one not—what fraction The same, 0.625.

c. The same, 0.625.

d. They will match about 1/2 of the time.

e. See the wiki article.

f. Guess different lengths for the keyword and treat each of the corresponding sets of columns as distinct messages. When you guess the correct length, each of these "messages" will be encrypted with a simple substitution and so the IC will be the same as for English—when the guess is incorrect, the IC should be close to that of random text.

26. a. Guess possible plaintext messages and encrypt them with the public key.

b. Pad the message with random bits.

c. The attacker doesn't know the key that was used to encrypt the message.

27. a. Trudy computes $h(x)$ for all reasonable salary values x. When she finds a value that gives the desired hash, she's recovered Alice's salary.

b. We know something about possible input values (salaries), so it limits our search.

c. Include a random value r and compute the hash as, say, $y = h(x, r)$. Then Trudy must guess both x and r and the forward search is no longer feasible, provide that r is selected from a large enough space.

28. a. You must try half of the keys, on average, for a work factor of 2^{39}.

b. Do the exhaustive key search, and for each putative key, "decrypt" the ciphertext. Since you know the corresponding plaintext, you'll know when you've got the correct key.

c. This is not so easy, since you will have to do some work to tell when you've got the correct key. If you know the plaintext is English, you could, in principle, visually inspect each putative decrypt until you see one that looks like English, but that would be impractical for such a large key space. To automate the attack, you would need to automatically test the putative decrypts to see if they appear to be English, which requires some additional work.

29. a. 5the correct one? Half of the keys, or 2^{39}.

b. Assuming that the plaintext is English, Trudy must test each putative decrypt for its "English-ness". This can be done using basic statistics of the English language. Depending on the size of the plaintext, this could yield many false alarms.

c. It depends on a lot of factors, but suppose you decrypt m and compute monograph and digraph statistics to see how close it is to English. Then the work is $c \cdot 2^{39}$ for some constant c, and you want to make c small. You can make c smaller by

decrypting fewer bits, but the less you decrypt, the more incorrect keys will pass the test and you will soon get tired of looking at incorrect "decrypts." A better strategy is to have a fast primary test that weeds out most of the incorrect keys, then do a secondary test (i.e., decrypt more plaintext and compute more statistics) for keys that pass the primary test. Then you can determine the optimal size of these tests, so as to minimize the overall work factor.

d. It depends on the details of the specific test proposed, but for the general idea, see the answer to part c.

Chapter 3

1. The keystreams are not chosen uniformly at random, since a relatively small number of keys generate a much larger number of possible keystreams.

2. a. Once the internal state of the cipher repeats, then the keystream will repeat.

   b. It's at least as bad as using a one-time pad more than once.

3. Suppose that Alice uses a stream cipher to encrypt plaintext P, obtaining ciphertext C, and Alice then sends C to Bob. Suppose that Trudy happens to know the plaintext P, but Trudy does not know the key K that was used in the stream cipher.

   a. Since $C = P \oplus K$, we have $K = C \oplus P$.

   b. From part a, Trudy knows the key K, so Trudy can replace C with $C' = P' \oplus K$.

4. a. The X register steps exactly 3/4 of the time. A truth table is the easiest way to answer this, and all of the remaining parts of this question.

   b. The Y register steps exactly 3/4 of the time.

   c. The Z register steps exactly 3/4 of the time.

   d. All 3 registers step when all three "step bits" agree, that is, in the 000 and 111 case, which occur $2/8 = 1/4$ if the time.

   e. Two registers step in all cases except when all 3 step, so 3/4 of the time.

   f. Never, since it is not possible for a single register to be in the majority—there are 3 registers, so a majority consists of 2 or more.

   g. Never, since it is not possible that no register will be in the majority.

5. The keystream bits are

$$k_0k_1k_2\ldots k_n = 10000011011100000111100000011001$$

   and the registers are

$$X = x_0x_1x_2...x_{18} = 0001101000000000000$$

   and

$$Y = y_0y_1y_2...y_{21} = 1111101010101010101010$$

   and

$$Z = z_0z_1z_2...z_{22} = 01101010111100001010101.$$

6. The function is $f(x, y, z) = xy \oplus xz \oplus yz$.

7. a. We only swap elements of the S, and swapping elements of a permutation yields another permutation.

   b. SSL and WEP, and many other places.

8. a. Following the hint, the bound is $2^{16} \cdot 256! \approx 2^{1700}$

   b. It gives the maximum possible length of a keystream sequence before it must repeat, and if the keystream ever repeats, there are serious problems. Of course, there could be short cycles, so a large state space is not sufficient, but it is necessary.

9. Table S after initialization is, in hex,

```
4c 46 51 0e 36 aa 27 24 14 00 70 80 42 7f a7 a9
8e 94 0c 7d 7b 37 78 be af 09 c0 9b e2 0d 3c 6c
04 ed 1f b8 a2 68 db 3f d6 76 ba bc c2 d5 3b ff
39 2e 4e bd fa 18 b2 b1 86 6b fc 1e 73 5b 75 35
1a 13 d7 ad 74 66 f7 b3 ac fb 07 ab e3 41 77 57
53 84 98 e8 dd 31 11 a1 d2 16 82 72 f9 6a 40 33
8c c4 99 02 cf 81 dc 5e 9c 5a b4 22 c1 5f cc 0b
56 30 c7 7c 7a 9a 10 cd 4d 83 65 7e ee ce 71 34
c8 62 fd 49 f4 9f bf 2a 8f 6d 1c 0a c9 ef e4 b9
06 d8 3d 3a bb a8 b5 a4 c3 43 52 4b a0 05 a5 93
90 ae 48 d4 a6 2d 6f a3 63 e1 8a 8d b0 5d b6 23
ea 60 44 69 92 17 2f 0f 50 2b f5 f6 32 b7 29 01
45 12 59 26 d3 c6 87 d9 e5 de 47 5c 6e cb 1d 58
08 4f eb da 55 89 85 f0 e6 e7 f1 c5 03 f8 d0 38
d1 fe e0 67 88 9d 25 ca ec 8b 19 21 15 28 f3 97
79 20 df 91 61 4a 54 1b e9 9e 3e f2 64 2c 96 95
```

Table S after 100 bytes of keystream

```
4c f7 a4 2d c5 41 c4 3f 88 14 f9 55 70 ba 73 4e
7c 3b 1a 42 ad 4f 6c 37 c8 6d fb 0c 87 da 9d 04
d5 66 bc 67 aa 28 98 94 f8 e8 a8 c2 ab 00 fc 82
3a 12 6b 6e e2 d0 06 13 d9 3c 72 e3 91 31 26 e9
7d 75 5b b3 f6 78 5e 52 45 af 2a 6f d3 9f 11 0f
dd e4 e5 a0 ed 77 8c 97 2e f0 85 08 53 7a 92 ca
7b b0 93 ae 56 81 dc 46 9c 5a b4 22 c1 5f cc 0b
cf 30 c7 8e 6a 9a 10 cd 4d 83 65 7e ee ce 71 34
c0 62 fd 49 f4 a2 bf 07 8f 09 1c 0a c9 ef 84 b9
b2 d8 3d 39 bb 7f b5 51 c3 43 d7 4b 76 05 a5 99
90 02 48 d4 a6 0e 1f a3 63 e1 8a 8d 27 5d b6 23
ea 60 44 69 40 17 2f 57 50 2b f5 74 32 b7 29 01
ac d2 59 b1 1e c6 fa 86 db de 47 5c bd cb 1d 58
24 be eb 0d 80 89 ff 16 e6 e7 f1 36 03 d6 18 38
d1 fe e0 b8 9b a9 25 33 ec 8b 19 21 15 68 f3 a1
79 20 df a7 61 4a 54 1b 35 9e 3e f2 64 2c 96 95
```

Table S after 1000 bytes of keystream

```
aa 9a ea 7d 30 0a 21 e3 c8 e9 66 cf 5a f6 f5 72
31 c3 d3 3f a9 75 c6 a1 8f 8a c7 3c 77 ee a0 d6
c9 bc f1 0b ed fc 3b fb 7e 13 7c 95 fd 39 b9 98
e6 03 4e 05 65 63 ff 7b 0c 27 93 dd da 1d 3a fa
e7 73 ce 51 b1 89 86 3d 01 46 52 df 90 a8 8d 5b
b4 48 e1 02 2f d1 60 26 d4 b0 74 11 10 9e 69 f7
d9 9b e2 0f b6 96 2b 9d 22 29 d5 94 cb 6e 62 eb
80 42 20 4d 3e 38 6b 44 fe d0 91 14 2e 34 1f 6f
84 41 59 b5 5f 4b 47 78 cd ad 64 12 00 92 e0 36
db 57 5c 7f d8 1e cc 50 d2 0d 8b a4 b2 37 e5 f2
19 87 bf 06 53 4c 33 88 07 f4 18 a3 c5 c2 9c 35
e4 82 1c b7 49 ca 2d 6d 4a a2 c4 a7 dc de 17 ae
32 f3 58 71 28 7a 16 5e 45 67 6c 70 ef 40 6a 09
a5 ab 97 1a 83 bb be 76 c1 1b 2c 04 43 d7 15 ba
9f 23 ac ec e8 56 f0 25 24 a6 b8 f8 4f 68 55 99
0e 5d 2a 79 54 61 b3 81 bd 8c f9 85 08 af 8e c0
```

10. a. Trudy computes $k_0 = c_0 \oplus p_0$.

    b. Replace c_0 with $p_0' \oplus k_0 = p_0' \oplus (c_0 \oplus p_0)$.

    c. Yes, but Trudy would also need to change the CRC to match.

    d. No. These topics are discussed in detail later chapters.

11. a. The formulas are given in the book.

    b. Yes.

    c. No.

    d. TEA uses "$+$" and "$-$" in place of $\oplus$.

12. a. We have $C = P$.

    b. Here, we find $C = (R_0, L_0 \oplus R_0)$.

    c. This one is a little more involved:

$$C = (L_0 \oplus K_1 \oplus K_3 \oplus \cdots \oplus K_{15}, R_0 \oplus K_2 \oplus K_4 \oplus \cdots \oplus K_{16})$$

    d. This one is downright annoying:

$$C = (R_0 \oplus K_2 \oplus K_3 \oplus K_5 \oplus K_6 \oplus K_8 \oplus K_9 \oplus K_{11} \oplus K_{12} \oplus K_{14} \oplus K_{15},$$
$$L_0 \oplus K_1 \oplus K_3 \oplus K_4 \oplus K_6 \oplus K_7 \oplus K_9 \oplus K_{10} \oplus K_{12} \oplus K_{13} \oplus K_{15} \oplus K_{16})$$

13. a. The subkey which is XORed in each round (or the S-boxes).

    b. Any of the permutations.

14. a. 64

    b. 64

c. 56

d. 48

e. 16

f. 8

g. 6

h. 4

15. It serves on security purpose—it allows for the same code to be used as for decryption as for encryption (the subkey must be used in reverse order). Of course, the swap could have been done as the first step of decryption, but instead it is specified as the last step of encryption.

16. It's the same as the "double DES" attack given in the book except that we must find K_1 and K_2 that satisfy $E(C, K_2) = E(P, K_1)$.

17. The bits of the key K that appear in each subkey are given in the four tables below.

K_1	8	44	29	52	42	14	28	49	1	7	16	36	2	30	22	21	38	50	51	0	31	23	15	35
	19	24	34	47	32	3	41	26	4	46	20	25	53	18	33	55	13	17	39	12	11	54	48	27
K_2	1	37	22	45	35	7	21	42	51	0	9	29	52	23	15	14	31	43	44	50	49	16	8	28
	12	17	27	40	25	55	34	19	24	39	13	18	46	11	26	48	6	10	32	5	4	47	41	20
K_3	44	23	8	31	21	50	7	28	37	43	52	15	38	9	1	0	42	29	30	36	35	2	51	14
	53	3	13	26	11	41	20	5	10	25	54	4	32	24	12	34	47	55	18	46	17	33	27	6
K_4	30	9	51	42	7	36	50	14	23	29	38	1	49	52	44	43	28	15	16	22	21	45	37	0
	39	48	54	12	24	27	6	46	55	11	40	17	18	10	53	20	33	41	4	32	3	19	13	47

K_5	16	52	37	28	50	22	36	0	9	15	49	44	35	38	30	29	14	1	2	8	7	31	23	43
	25	34	40	53	10	13	47	32	41	24	26	3	4	55	39	6	19	27	17	18	48	5	54	33
K_6	2	38	23	14	36	8	22	43	52	1	35	30	21	49	16	15	0	44	45	51	50	42	9	29
	11	20	26	39	55	54	33	18	27	10	12	48	17	41	25	47	5	13	3	4	34	46	40	19
K_7	45	49	9	0	22	51	8	29	38	44	21	16	7	35	2	1	43	30	31	37	36	28	52	15
	24	6	12	25	41	40	19	4	13	55	53	34	3	27	11	33	46	54	48	17	20	32	26	5
K_8	31	35	52	43	8	37	51	15	49	30	7	2	50	21	45	44	29	16	42	23	22	14	38	1
	10	47	53	11	27	26	5	17	54	41	39	20	48	13	24	19	32	40	34	3	6	18	12	46

K_9	49	28	45	36	1	30	44	8	42	23	0	52	43	14	38	37	22	9	35	16	15	7	31	51
	3	40	46	4	20	19	53	10	47	34	32	13	41	6	17	12	25	33	27	55	54	11	5	39
K_{10}	35	14	31	22	44	16	30	51	28	9	43	38	29	0	49	23	8	52	21	2	1	50	42	37
	48	26	32	17	6	5	39	55	33	20	18	54	27	47	3	53	11	19	13	41	40	24	46	25
K_{11}	21	0	42	8	30	2	16	37	14	52	29	49	15	43	35	9	51	38	7	45	44	36	28	23
	34	12	18	3	47	46	25	41	19	6	4	40	13	33	48	39	24	5	54	27	26	10	32	11
K_{12}	7	43	28	51	16	45	2	23	0	38	15	35	1	29	21	52	37	49	50	31	30	22	14	9
	20	53	4	48	33	32	11	27	5	47	17	26	54	19	34	25	10	46	40	13	12	55	18	24

K_{13}	50	29	14	37	2	31	45	9	43	49	1	21	44	15	7	38	23	35	36	42	16	8	0	52
	6	39	17	34	19	18	24	13	46	33	3	12	40	5	20	11	55	32	26	54	53	41	4	10
K_{14}	36	15	0	23	45	42	31	52	29	35	44	7	30	1	50	49	9	21	22	28	2	51	43	38
	47	25	3	20	5	4	10	54	32	19	48	53	26	46	6	24	41	18	12	40	39	27	17	55
K_{15}	22	1	43	9	31	28	42	38	15	21	30	50	16	44	36	35	52	7	8	14	45	37	29	49
	33	11	48	6	46	17	55	40	18	5	34	39	12	32	47	10	27	4	53	26	25	13	3	41
K_{16}	15	51	36	2	49	21	35	31	8	14	23	43	9	37	29	28	45	0	1	7	38	30	22	42
	26	4	41	54	39	10	48	33	11	53	27	32	5	25	40	3	20	24	46	19	18	6	55	34

The number of times that each bit of K appears in a subkey is given below.

bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
count	14	15	12	14	14	14	13	14	13	14	13	14	13	14	14	15	13	14	14	14	13	14	13	14	13	13	14	15
bit	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55
count	13	15	14	13	15	13	13	15	12	13	15	13	14	15	13	15	14	12	15	13	13	15	12	13	15	13	14	14

The number of subkey bits is $16 \cdot 48 = 768$ and the number of key bits is 56. Since 768 is not divisible by 56, it's not possible to use every key bit exactly the same number of times. However, it would be possible to use every key bit either 13 or 14 times, which is more uniform than the actual counts for DES, which vary from 12 to 15.

18. a. The attack is the same, except that we need to recompute the lookup table each time we conduct the attack.

    b. We must include the work to compute the lookup table, since we cannot amortize this work. This gives us a work factor of $2^{56} + 2^{55} \approx 2^{56}$. So it is about twice as much work, on average, if we only have known plaintext instead of chosen plaintext.

19. The functions and their corresponding layers are: Bytesub (nonlinear), Shiftrow (linear mixing), MixColumn (nonlinear), and AddRoundKey (key addition). The confusion parts are ByteSub and MixColumn, which together serve essentially the same purpose as the DES S-boxes. ShiftRow is for diffusion, and this serves a similar purpose as the expansion permutation in DES. AddRoundKey would also be considered a confusion step.

20. a. The ciphertext is `0xbd2b2fa555ae7017`.

    b. TBD

21. The Wikipedia article on the Tiny Encryption Algorithm has a good diagram (as I'm sure many of your students have discovered).

22. a. For the uses discussed in this chapter, IVs chosen in sequence would suffice.

    b. If the IV space is small, it might be advantageous to choose the IVs in sequence, since it will take much longer for an IV to repeat, which is a problem (and particularly serious in CTR mode).

23. See the discussion of CBC mode in the Wikipedia article on "block cipher modes of operation."

24. No. The IV is not secret, so with a single block of known plaintext, Trudy can determine K and then decrypt all blocks.

25. a. The decryption rule is

$$P_0 = D(C_0 \oplus \text{IV}, K), \ P_i = D(C_i \oplus C_{i-1}, K)$$

b. This approach is no better than ECB mode, since Trudy can compute $C_i \oplus C_{i-1}$ to obtain ECB mode data. Consequently, this mode has all of the same dproblems as ECB mode.

26. As long as two consecutive ciphertext blocks are adjacent, the corresponding plaintext block will be correct, even if it is out of order. So, for example, $C_8, C_9, C_1, C_2, \ldots$ will yield $X, P_9, Y, P_2, \ldots$.

27. If you know the key, you can obtain P_i provided that you know C_i and C_{i-1}. So, random access is possible, provided that you have access to the current and previous ciphertext blocks. However, if you want to replace the plaintext P_i with P_i', then CBC mode is impractical—you would need to decrypt and re-encrypt everything.

28. One simple modification would be $C_i = P_i \oplus E(\text{IV} \oplus i, K)$. This would work for random access. Another modification would be $C_i = P_i \oplus E(\text{IV} \oplus C_{i-1}, K)$, which would work for random access decryption, but not if we want to re-encrypt new plaintext (see the previous problem).

29. Blocks 0, 1, and 4 will be correct.

30. a. It should look the same as the example in the book.

b. The jpg format is compressed, so the parts that look the same are actually stored as different bits. Consequently, blocks that appear similar are different, so they encrypt to different blocks, even in ECB mode.

31. a. The same initial plaintext yields same initial ciphertext. Note that as soon as the messages differ, the ciphertext will differ from that point on, even if the plaintexts should agree at some point(s).

b. The same plaintext yields the same ciphertext.

c. CTR is much worse, since the same keystream is used every time (until the key K changes)! This is at least as bad a using a one-time pad multiple times.

32. a. Two messages that start with the same plaintext will yield the same plaintext, up to the point where the messages differ.

b. This is OK in this case.

33. You can pad a partial block to fill it out to the size of a full block. However, it's necessary to be able to remove the padding when decrypted, which is not completely trivial. If you do not want to expand the data, you could use CTR mode on any final partial block. See also Schneier's *Applied Cryptography* for a discussion of "ciphertext stealing" which works for ECB or CBC mode.

34. a. Suppose X is a single block. Then function $F(X)$ is one-way since we cannot obtain X without the key (assuming the cipher is secure). That is, we need to decrypt or break the cipher to obtain X from $F(X)$. If X is multiple blocks, it's even harder to determine X.

    b. Without the key, there is no obvious way to search for such a collision.

35. If only one block changes, then it will be detected, with probability 1. If more than one block changes, the chance of the MACs matching is about $1/2^n$ where n is the length of a block, in bits. For DES, this gives a probability of about $1/2^{64}$.

36. a. No, since we are just decrypting and then re-encrypting with the same key, so we'll get what we started with.

    b. No.

    c. No.

37. Trudy replaces C_0 with $\tilde{C} = C_0 \oplus P_1 \oplus X$.

38. When decrypted, at least one block will decrypt incorrectly. Then the MAC will be incorrect since errors in CBC encryption propagate into the final block (a complete solution should actually "show" this by, say, writing out the MAC formula with one block incorrect and arguing that this propagates into the computed MAC). Note that this depends on the fact that the keys are different.

39. a. When decrypting, blocks 1 and 2 will be incorrect, Then when computing the MAC we are re-encrypting with a different key, so the computed MAC will almost certainly be different than the received MAC, and the tampering will be detected.

    b. No. As long as the keys differ, it is OK.

40. No, this is not a good idea. There is a straightforward meet-in-the-middle attack on double encryption using Cipher A, which has a work factor of about 2^{64}, assuming that we can store a lookup table of size 2^{64}. In comparison, the expected work to break Cipher B is 2^{127}. See the discussion of the meet-in-the-middle attack on double DES. However, note that this attack requires known plaintext.

41. a. Using a meet-in-the-middle attack requires about 32 bits of work. Can you do better?

    b. This is not so easy, and there is no obvious shortcut in this case. Interestingly, this also seems to apply in the extreme case, where you iterate a cipher with a 1-bit key...

42. It's precisely the same strategy as in the 2DES attack, except here we need to guess a 112-bit key.

43. a. Choose IV, $P_0, P_1, \ldots, P_{n-2}$ and, using CBC mode, compute C_{n-2}. Then we want $E(P_{n-1} \oplus C_{n-2}, K) = \text{MAC}$, where MAC is the given MAC value. Since we know K, we choose $P_{n-1} = D(\text{MAC}, K) \oplus C_{n-2}$.

    b. You can choose everything except the final plaintext block.

Chapter 4

1. a. A name and a public key. To be of any value, it must also be signed by a CA.

   b. Just about anything—phone number, office, department, etc.

   c. If any of the information changes, a new certificate is needed.

2. a. Nothing. It's a public key certificate, and public keys are public.

   b. Hash the "message" and decrypt the signed quantity (using the CAs public key), then compare the two.

   c. Nothing. If Bob trusts the CA, he believes that the private key corresponding to the public key in the certificate is held by Alice.

3. a. The same plaintext will encrypt to the same ciphertext (recall the image of Alice encrypted in ECB mode in Chapter 3).

   b. Random padding is necessary to prevent a forward search attack, and due to the random padding, the same plaintext will not yield the same ciphertext.

   c. Yes, you could pad each block with random bits. However, this would reduce the efficiency of the block cipher, since you'd need to use a significant portion of the block for padding (typically, 64 bits are used for block cipher padding).

4. For now, we'll say that Alice computes $[M]_{\text{Alice}} = M^d \bmod N$. However, in the next chapter we'll see that in most situations, she would actually compute the signature as $[h(M)]_{\text{Alice}} = h(M)^d \bmod N$, where h is a cryptographic hash function.

5. Since $M^{de} = M^{ed} \bmod N$, this is a trivial modification to the proof in the text.

6. a. To encrypt: $19^3 = 28 \bmod 33$. To decrypt: $28^7 = 19 \bmod 33$.

   b. The signed result is $S = M^d \bmod N = 25^7 \bmod 33 = 31$. To verify the signature, Bob computes $S^3 \bmod N$ and the signature is verified if the result matches the received value M. In this case, $31^3 = 25 \bmod 33$. Assuming Bob receives the sent message $M = 25$, the signature is verified.

   Note that in practice, a hash function is usually used, so that the hash of the message is signed instead of message itself. This is discussed in Chapter 5.

7. For example, in some protocols, a random "challenge" is signed and sent. If so, Trudy might trick Alice into signing "random" challenge that is actually an encrypted message, say, $X = \{M\}_{\text{Alice}}$. The effect would be that Alice decrypts X and sends M back to Trudy. Cryptanalysis doesn't get any easier than that.

8. a. If $M^3 < N$, then the mod N operation has no effect, so Trudy can simply take the usual cube root of the ciphertext to decrypt. To prevent the attack, pad with bits so that, as a number, $M > N^{1/3}$. Of course, the recipient must know the padding scheme too.

b. Since $3^3 = 27 < 33$, Trudy can take the cube root to obtain M, but $4^3 = 64 = 31 \bmod 33$, so Trudy cannot simply take the cube root.

9. a. If $e = 3$ and $M < N^{1/3}$, then the mod N operation has no effect, so Trudy can take the usual cube root of C to recover M.

   b. The most straightforward solution is to pad with random bits, making sure that a sufficiently high order is set to 1 so that when the padding is included, $M > N^{1/3}$, regardless of the actual message.

10. a. Graph the function $f(n)$ for $1 \leq n \leq 10{,}000$. Easy.

    b. Since about 88 bits of work are required, this is roughly equal to an exhaustive search for an 89-bit symmetric key.

    c. About 119 bits.

    d. A modulus of about 13620 bits is required.

11. Public: p and g. Alice's secret is a and Bob's secret is b.

12. Encrypt the exchange.

13. Bob wants to solve for b in the equation $(g^a)^b \bmod p = X$, but this requires Bob to solve the discrete log problem, where the base is $g^a \bmod p$.

14. Use Diffie-Hellman and encrypt the exchange using the PIN X. That is, Alice computes and sends $E(g^a \bmod p, X)$ and Bob computes and sends $E(g^b \bmod p, X)$. Why is this more secure? To do the man-in-the-middle attack, Trudy must act in real time (i.e., while the exchange is taking place). When Trudy intercepts $E(g^a \bmod p, X)$, she can guess possible values for X, but she has no way to know whether her guess is correct. So, in effect, she gets one chance to guess X, so her chance of success is only 1/10,000, while in the original protocol, her chance of success was 1. This is a big improvement in security for minimal additional work.

15. The private key is known only to the signer.

16. a. Encrypt the Diffie-Hellman exchange using DES.

    b. Encrypt a symmetric key using RSA.

17. This is essentially the same as the non-ECC version.

18. If symmetric keys are used, then it would not be so easy for Bob to confuse Charlie, since Bob does not have access to the symmetric key that Alice and Charlie share.

19. If F is a one way function (i.e., given $F(M)$ it is infeasible to find M), then the attack is infeasible. We'll see in the next chapter that cryptographic hash functions are one way, and also provide other desirable properties.

20. a. First, compute $m^{-1}C = 6 \cdot 20 = 120 = 26 \bmod 47$. Using the superincreasing knapsack in the private key, we find that the plaintext is 1001, in binary.

b. First, compute $m^{-1}C = 6 \cdot 29 = 174 = 33 \bmod 47$. Using the superincreasing knapsack in the private key, we find that the plaintext is 0011.

c. Since $m^{-1}m = 6m = 1 \bmod 47$, $m = 8$. Multiply each element in the superincreasing knapsack by m and reduce mod 47 to obtain the "general" (though not quite general enough—see Chapter 6 for the attack) knapsack, $(24, 40, 33, 43)$. It is easy to verify the encryptions that appear in parts a, and b.

21. a. Public key: $(18, 30, 25, 44)$ and $n = 47$.

b. The result is $C = 18 + 30 + 25 = 73 = 26 \bmod 47$.

22. a. The superincreasing knapsack is $(3, 5, 9, 20)$ and $m^{-1} = 8$.

b. The ciphertext is $C = 74$ or $C = 74 \bmod 47 = 27$.

23. First, you must show that multiplying by m^{-1} converts the problem into the superincreasing domain, then the actual algorithm is straightforward. Note that $C = \sum a_i \cdot W_i$, where $W_i = S_i \cdot m \bmod n$, where S_i are the elements of the superincreasing knapsack. Then

$$m^{-1}C = \sum a_i \cdot W_i \cdot m^{-1} = \sum a_i \cdot S_i \cdot m \cdot m^{-1} = \sum a_i \cdot S_i \bmod n$$

which is the desired result.

24. a. Easy.

b. See the solution to the previous problem—an extra "mod n" makes no difference.

c. It should make no difference. However, I find that the knapsack attack in Chapter 6 seems to work more often when the ciphertext is not reduced modulo n. Is it just my imagination?

25. No. Trudy has no way to determine the secret value $g^{abt} \bmod p$ that Alice and Bob will share (short of solving a discrete log problem, that is). Is there any possible way for Trudy to achieve her goal?

26. a. Then $g^n = 1 \bmod p$ for all n, so g is not a generator.

b. Then $g = -1 \bmod p$, so $g^n \bmod p$ is either 1 or -1 for any n, and g is not a generator.

27. Bad, bad, bad idea...

28. Very little. An attack that only succeeds a small percentage of the time would be devastating for a cryptosystem and the fact that the general problem is hard does not prevent there being some relatively easy cases. For example, there are efficient algorithms that give approximate solutions to the traveling salesman problem, although finding an exact solution in the general case is intractable. So even if the RSA problem is, say, NP-hard, that does not rule out the possibility of finding an efficient attack that is "close enough" to make the system useless in practice.

29. a. No, since the probability is low.

    b. Easily determined by trying all M satisfying $1 < M < 3127$. Note that the problem should ask for non-trivial solutions, since 0 and 1 always work.

30. In this case, double encryption is equivalent to single encryption with the exponent $e = e_0 \cdot e_1$. In general, double encryption in RSA does not increase security; see, for example, Trappe and Washington, p. 159.

31. Yes. Trudy sends the ciphertext $s^e C \pmod N$ and Alice returns $sC^d \pmod N$, from which Trudy can easily recover $M = C^d \pmod N$.

32. a. The message is $M = (34, 73, 71, 71, 76, 69, 83)$, which translates to "Biggles". A good source on the details of this type of attack can be found here: `www.di-mgt.com.au/crt.html`

    b. Since there is no padding, a forward search might be reasonable, especially given that $e = 3$, which implies that each trial encryption would be fast. Factoring the modulus would certainly be feasible.

33. a. Trudy obtains $(C')^d = (M \cdot r)^{ed} = M \cdot r \bmod N$. Using the modular inverse of r, Trudy computes $r^{-1} \cdot M \cdot r = M \bmod N$.

    b. TBD

34. a. This is easy, since $C_0 \cdot C_1 = (M_0^e \cdot M_1^e = (M_0 \cdot M_1)^e \bmod N$

    b. There is certainly no any easy algebraic manipulation that gives the desired result.

    c. It would be possible to compute arbitrary functions on encrypted data. So, for example, you could send some encrypted data to an untrusted computer, which could do computations directly on the encrypted data and return the result to you in encrypted form. The untrusted computer would not have access to the plaintext data or the plaintext results.

35. a. Suppose Alice signs M and sends the signed value, along with M, to Bob. Then Bob is supposed to receive $S = [M]_{\text{Alice}}$ and M. If Bob receives S and $M' \neq M$, then $\{S\}_{\text{Alice}} \neq M'$ so the signature verification fails. On the other hand, if Bob receives $S' \neq S$ and M, then $\{S'\}_{\text{Alice}} \neq M$ and again the signature check fails. Similarly, if Bob receives $S' \neq S$ and $M' \neq M$, then the signature check fails, unless Trudy chooses a nonsense message as discussed in Problem 19 (we'll see how to eliminate this problem in the next chapter).

    b. Only Alice has the private key, so nobody could have faked the signature.

36. a. It's the dictionary definition—Alice cannot repudiate a given transaction.

    b. Almost any commercial activity would work.

37. a. There may be a slight efficiency advantage for the MAC (depending on the comparative efficiency of the block cipher used for the MAC and the hash function

public key method used for the signature). But, if public keys and symmetric keys are available, there is probably no major difference between the two. So the primary issue would simply be the type of keys available (symmetric or public/private).

b. In this case, they must use a signature, since a MAC does not provide non-repudiation.

38. a. Yes.

b. Yes.

c. It's not a good idea, since it requires additional work for no real security benefit—a signature without the MAC would be just as good, and if non-repudiation is not an issue, then either the MAC alone or signature alone would suffice.

39. a. The padding is encrypted, which adds to the computational expense (assuming multiple blocks of data) and it consumes bandwidth.

b. To break the random padding is roughly equivalent to an exhaustive key search. Today, 64 bits might be a reasonable minimum, with 128 bits having a huge margin for safety.

c. TBD

40. a. An easy calculation shows $b = 10$.

b. The point at infinity and $(3,5), (3,6), (4,5), (4,6), (5,4), (5,7), (6,2), (6,9)$.

c. The sum is $P_3 = (3,5)$.

d. We have $(4,5) + (4,5) = (3,6)$ and then compute $(4,5) + (3,6)$.

41. a. We have $7^2 = 2^3 + 11 \cdot 2 + 19 \bmod 167$, since $49 = 49$, regardless of the modulus.

b. Alice sends $12(2,7) = (153,36)$ and Bob sends $31(2,7) = (103,153)$. The shared secret is

$$12(103,153) = 31(153,36) = (137,54).$$

Note that $12 \cdot 31 = 372$, so that $372(2,7) = (137,54)$.

42. a. Depends on values selected.

b. Note that $M = xa + ks \bmod (p-1)$ implies $M = n(p-1) + (xa + ks)$ for some n (where $0 \leq xa + ks < p - 1$) then $g^M = g^{n(p-1)} g^{xa+ks} \bmod p$. Now by Fermat's Little Theorem, $g^{n(p-1)} = 1 \bmod p$ and the result follows.

Chapter 5

1. a. Using such a has function for digital signatures would not be useful, since you'd still have to sign a large message.

   b. This would make digital signature calculations slow.

   c. The online bid example in the book would fail.

   d. The hash would fail when used in digital signatures.

2. a. Suppose that h satisfies the strong collision resistance property, but if fails to satisfy the weak collision resistance property. Then for some x_0 and $h(x_0)$, we can find x_1 such that $h(x_0) = h(x_1)$. But then x_0 and x_1 violate the strong collision resistance assumption.

   b. This is a little bit tricky. Suppose that g is a hash function that provides strong collision resistance (and therefore, weak collision resistance as well) and that g produces an n-bit output. Let (a, b) be a concatenated with b. Then define the hash function h by

$$h(x) = \begin{cases} (1, x) & \text{if } x \text{ is of length } n \\ (0, g(x)) & \text{otherwise.} \end{cases}$$

   Then h is collision resistant, but not one-way.

3. You would want to attack the strong collision resistance. If you could hash $2^{n/2}$ random inputs, then you would expect to find a collision.

4. a. If you hash 2^{10} messages, then you have about 2^{20} comparisons. For each 2^{12} comparisons, you expect to find a collision, so the expected number id 2^8.

   b. The answer is $m^2/2^n$.

5. a. You need about $2^{n/2}$ hashes.

   b. You will need about $\sqrt{10} \cdot 2^{n/2}$, since this number of hashes will result in about $10 \cdot 2^n$ comparisons, and for each 2^n comparisons, we expect to find a collision.

   c. We need $\sqrt{m} \cdot 2^{n/2}$ hashes. Note that this implies that it gets progressively easier for find collisions as more hashes are computed. This is intuitive, since each new hash can be compared to all of the previous hashes.

6. a. About $2^{n(k-1)/k}$ hashes must be computed before we expect to find a k-way collision.

   b. About $2^{1/k} \cdot 2^{n(k-1)/k}$ (see the solution to part c for an explanation).

   c. With y hash values, there are about y^k k-way comparisons. We expect to find one k-way collision for each $2^{n(k-1)}$ comparisons. Therefore, to find m k-way collisions, we need to compute about $m^{1/k} \cdot 2^{n(k-1)/k}$. It is instructive to compare this to the result of the previous problem.

7. No. Trudy can simply find R' such that $h(I, R) = h(E, R')$, where I is the original innocent message and E is Trudy's evil message. So, this is actually easier for Trudy since she doesn't need to make up a bunch of equivalent innocent and evil messages—she just varies the random bits.

8. Examples include 11100111 and 11110100.

9. The only values that work are 11100011 and 11110000.

10. The arrows on the right-hand side (key schedule) all denote 512 bit quantities, the arrow on the left-hand side represents the 192 bits of (a, b, c), while the arrows in the middle all represent 64 bits.

11. a. We have

$$h(M) = F(F(F(A, B_1), B_2), B_3) = F(F(h(B_1), B_2), B_3) = F(h(B_1, B_2), B_3).$$

b. The general case follows by a simple induction.

12. a. It should only require about 2^6 hashes.

b. The following both yield Bobcat hash values of `0xcc6fca4b4a12`.

```
e901 172c 68e1 0c9e c343 5bbe 9ace 0736
8ad2 cb62 4983 cfcf 90ea efd2 0e82 f8e2
46eb c662 ae46 ad5b 1268 bdbc c731 53e7
42d9 acea b106 6b70 d7a9 e359 a3be 0d86
```

and

```
e73e 1f55 e79a b0dd 268e 56e5 823c 136b
4e14 b23b 5956 a55d 914a 978a 4646 2dd5
0d1d 725b 6517 db48 3ff6 dd52 1009 ddab
7e2d 134e 09c1 f550 5d36 d68f bd4b 334b
```

13. Each of a, b, and c are OK, since a secure hash must be secure in all bit positions—otherwise there would be a shortcut attack.

14. a. From the definition of F it is clear that this works if K, M and X are all multiples of the block length.

b. It works any time that (K, M) is a multiple of the block length.

c. TBD

15. A MAC would provide integrity protection as with an HMAC, but there is a subtle difference. If the key is known, it's easy to generate collisions for a MAC, but this is not the case for an HMAC—see the next problem.

16. No, since a hash is one-way.

17. a. One way (supposed to prevent anyone from determining a bid from the corresponding hash) and collision resistance (prevents anyone from changing their bid after submitting the hash).

    b. A forward search—hash all reasonable bids and look for ones that give the same hash as Alice's and/or Bob's bid.

    c. Yes, most definitely.

    d. Alice can should select a random value R_A and submit $h(A, R_A)$, and similarly for the other bidders. When submitting her bid, Alice must submit her bid A and the random padding R_A.

18. a. The collision resistance is important for the work factor and to prevent having multiple email messages with the same hash. One-way is also needed, otherwise you could specify the hash and then determine a message.

    b. The "message" could include a multiple email addresses, in which case Trudy would only need to find one appropriate R to send the message to all of the addresses.

19. a. You can define $h(B) = E(X, B)$, where X is some non-secret constant (e.g., 0). That is, the block to be hashed is used as the key.

    b. If you have more than one block, you could divide message into blocks of n bits, where n is the length of the block cipher key, then iteratively encrypt the blocks. For example, suppose the message is 3 blocks: B_0, B_1, B_2. Then use the block cipher to define $h(B_0, B_1, B_2) = E(E(E(X, B_0), B_1), B_2)$ where X is some non-secret constant. This should be secure since determining the "key" is hard, even if the attacker knows plaintext and ciphertext. As in part a, any initial constant could be used. Also, CBC "encryption" works, where the message is used as the key and the plaintext is a constant. In any case, the block cipher needs to have a large enough block size so that a brute force collision attack is not feasible.

20. Alice could encrypt as $C_0 = P_0 \oplus K$, $C_1 = P_1 \oplus h(K)$, and $C_1 = P_1 \oplus h(h(K))$. This is not strong, since if Trudy knows P_i she can easily determine P_j for all $j \geq i$.

21. Method (i) does not require anything to be stored. Method (ii) uses a stronger key to encrypt the data, making an attack directly on the ciphertext more difficult. Method (ii) could also be advantageous when Alice wants to change her password, since it is not necessary to decrypt/re-encrypt the data.

22. The obvious advantage of key diversification is that almost no storage is required. A possible advantage to the database is that there is no single point of failure. However, if someone can recover the master key, they could probably recover the database too. On the other hand, if the database is distributed (as it is on GSM), then it might be better to use a database. It might also be somewhat easier to change users' keys when using a database.

23. a. Anytime you have lots of data that only changes slightly, an incremental hash would be preferred. For example, suppose you hash all of the files on your hard drive as a way to detect errors. Suppose you later want to hash all of the files again, but only one file should have changed. Using a non-incremental hash, re-computing the hash will be as costly as the original hash computation, whereas an incremental hash will only be as costly as computing the hash of the one altered file.

    b. Assuming that the block sizes are appropriate, let $H(M') = F(h(M), X)$ where F is the "round function" of the hash h.

24. a. Once Alice knows Bob's guess Z, she tries different "keys" K' until she finds one for which the first bit of $W = D(Y, K')$ is not Z. Then she sends the "key" K' to Bob.

    b. Alice sends $h(K)$ along with Y, so she cannot change the key. That is, the hash of the key commits Alice to K and since K is random, there is no forward search attack.

25. The only trick here is to be sure that the bits are correctly entered into the file. I find that a lot of students want to treat this as the string "d131dd02..." instead of hex, which annoys me to no end. I blame this on too much Java.

26. a. Easy.

    b. Both hash to `c321325acff48137d62844e481ab01c5`. Suppose I sign the "recommendation" letter, `rec2.ps`. Then Trudy (or anyone else) can substitute the file `auth2.ps` for the recommendation and the signature verification will still pass.

    c. Depends on the selected messages, but opening either file with a text editor makes it pretty clear what needs to be done.

    d. The basic idea is as follows: Suppose A and B form an MD5 collision, that is, $h(A) = h(B)$. We then create two identical Postscript files, where the "good" letter is, say, T_0 and the "evil" letter is T_1 Then in the first file we set, say, $X = B$ and $Y = B$ in the conditional statement, so that T_0 (the "good" message) will be displayed. In the second file, we let $X = A$ and $Y = B$, so that the "evil" message is displayed. Both messages will hash to the same thing, due to the fact that $h(A) = h(B)$, and the files are identical beyond the "(X)(Y)" clause. Actually, it's slightly more subtle, since things must align properly on 512-bit block boundaries, but it's not difficult to arrange for this to occur.

27. a. See the discussion in the text.

    b. Failure to verify the signature means that the corresponding private key could belong to anybody. For example, Trudy could simply create a public/private key pair, put this public key in a certificate that says "Alice", sign the certificate herself, and keep the private key. If you then use the public key to encrypt a message, only Trudy can decrypt it, not Alice.

c. The CA's job is to create the certificate and make sure that the private key goes to Alice (and no one else). Only the CA could have signed the certificate, so if you trust that the CA did its job, and Alice's private key has not been compromised, then only Alice would have the private key.

d. Nothing. The certificate is public, so anyone can possess and send it. This is an important point that, hopefully, will become painfully clear when we discuss protocols.

28. a. Given M and $S = [h(M)]_{\text{Alice}}$, you compute $h(M)$ and verify that it agrees with $\{S\}_{\text{Alice}}$.

b. If you can recover the private key, then you can forge signatures.

c. Then you can find a collision, that is, you can find $M' \neq M$ such that $h(M') = h(M)$. If you replace M with M', the signature verification will succeed. The point here is that the security of the signature scheme depends on the security of the public key system and the hash function.

29. a. See the previous problem.

b. Suppose Alice sends M and $S = [h(M)]_{\text{Alice}}$. If Bob receives $M' \neq M$ and S, then $h(M') \neq \{S\}_{\text{Alice}}$. Similarly, if Bob receives M and $S' \neq S$, then $h(M) \neq \{S'\}_{\text{Alice}}$. If both are in error, then, almost surely $h(M') \neq \{S'\}_{\text{Alice}}$.

c. This follows from the fact that only signer has the private key.

30. a. Alice computes $S = [h(M)]_{\text{Alice}}$.

b. Alice sends both M and $S = [h(M)]_{\text{Alice}}$ to Bob. Bob verifies that $h(M) = \{S\}_{\text{Alice}}$.

31. a. Guess possible plaintext messages and encrypt each with the public key, then compare the results with the ciphertext.

b. The online bid problem in the book.

c. Random padding, i.e., the same solution to a forward search on public key cryptosystem.

32. a. This follows from the basic properties of a secure cipher.

b. For $M = (M_0, M_1)$, let $Y = E(E(X, M_0), M_1)$. This easily generalizes to messages consisting of any number of blocks.

c. If not, you can construct collisions.

33. a. The secret is $S = 6$ and the line is $2x + 3y = 18$.

b. The secret is $S = 8$ and the line is $5x + 6y = 9 \bmod 13$.

34. a. Two points are needed to determine the line. Given only one point, any possible S will yield a valid line thru Alice's point, and there is no additional information available to decide whether a putative S is correct or not.

b. Same principle as in part a.

35. a. Alice, of course.

b. This is easy—just modify the html file.

36. a. Suppose that you know one share, and consider any given pixel of the share. You have no info about whether the original pixel was black or white, since both are equally likely.

b. For example, a 3 out of 3 scheme is described in the paper "An (3, 3)-Visual Secret Sharing Scheme for Hiding Three Secret Data" by Tsai and Wang.

c. At least for the 3 out of 3 scheme mentioned in the solution to part b, there is a definite decrease in the resolution of the recovered image. Exactly how this varies with m and n is an interesting question.

37. Examples include varying the spacing between lines, change line breaks, change punctuation, etc. A different approach would be to rewrite each in a slightly different way.

38. a. The instructor might have reworded specific sentences, or varied line breaks. varied the spacing between lines (or words), changed fonts, punctuation, etc. In fact, the scheme the author used was to create two slightly different versions of the first page of each chapter (for example, using a slanted font or an italics font for one of the quotes, or slightly altering the punctuation on the page, etc.). With 13 chapters, this gave the author the ability to create 2^{13} distinct watermarks. Since only about 60 manuscripts were needed, the author choose watermarks that differed in as many bit positions as possible. This was, in effect, an error correcting code—up to some threshold, if the mark was damaged, it was still possible to assign the mark to the correct student. Only one pair of (excellent) students working together was able to solve this problem, and they required several hints. It would be interesting if such a scheme could be applied to the actual textbook, instead of just the manuscript, For books, the scheme would be even stronger, since the attacks mentioned below would be more difficult. Of course, it would be difficult to apply to hardcopy versions of a book, but it would be fairly easy to apply to electronic books. Note that if the textbooks are watermarked, the author could determine who is responsible for those illegal pdf versions of the book that are inevitably available via BitTorrent....

b. The best approach would be a collusion attack. That is, carefully compare several copies of the manuscript looking for differences

c. This is easy. Since you know how the scheme works, you can interchange pages that contain crucial watermarking information between various copies of the manuscripts.

d. Randomly shuffle pages between various copies of the manuscript.

39. a. Google "*A Boat Beneath a Sunny Sky*" and you can't miss it.

b. It's an acrostic—the first letter of each line spells the full name of the real Alice.

40. a. Low-order bits don't matter, while high-order bits matter a lot.
    b. Good question...

41. a. Hide the info in the low order RGB bits.
    b. Easy.
    c. They should be indistinguishable.

42. a. The Alice books, in pdf.
    b. Easy.
    c. They should be indistinguishable.

43. a. You could randomize (or zero out) the low order RGB bits. Alternatively, you could replace the bits with some other information of your choosing.
    b. If you randomized the bits, you will get nothing useful.

44. a. Just like the method discussed in the text—it puts the info in the low order RGB bits.
    b. Randomize the bits, zero the bits, replace the bits with some other message, etc.
    c. One option would be to use higher-order RGB bits. Of course, this would make the encoding much trickier, since you could only use combinations of bits that do not affect the image.

45. a. If the image is compressed, there is an encoding scheme, so it would be much more difficult to modify bits.
    b. Your modifications would need to follow the encoding scheme, which would complicate the process.

46. a. Programming problem.
    b. Ideally, you should not be able to detect any difference.
    c. If you cannot find a simple automated attack, then you might have something...

47. a. Programming problem.
    b. Ideally, you should not be able to detect any difference.
    c. It depends on the scheme.

48. a. Symmetric keys and IVs.
    b. Randomly selecting primes (RSA) and generating random exponents (DH).

49. a. Given a sequence of such numbers, the remaining number in the sequence can be determined. If such a sequence was used as a keystream, then a known plaintext attack might be devastating.
    b. Yes, since an attacker might know some plaintext, in which case they would know the keystream bits.

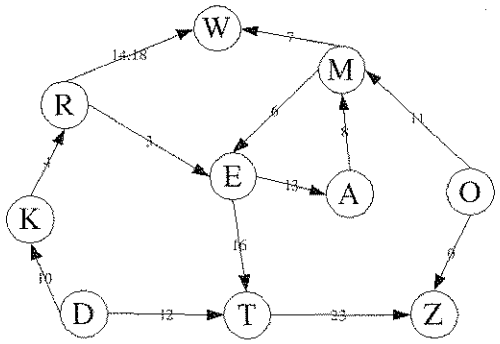
Chapter 6

1. a. The number of choices of rotors is $5 \cdot 4 \cdot 3 \approx 2^{5.9}$. As discussed in the text, the setting of the rings gives a factor of $2^{9.4}$ and the initial positions of the rotors gives another factor of $2^{14.1}$, while a stecker with 10 cables can be wired in $2^{47.1}$ ways. These numbers yield the claimed results. Summing these, we obtain $2^{76.5}$, which is close enough for government work.

   b. The precise number is $2^{29.4}$.

2. Select $2p$ of the 26 letters. Plug in first cable to one of these letters, then $2p-1$ places remain to plug other end. Plug in second cable to one of remaining positions, then $2p-3$ places to plug other end, and so on.

3. All cycles can be obtained from the following graph



   The independent cycles not listed in the text include $S(\mathtt{E}) = P_6 P_{11} P_0^{-1} P_{23} P_{16} S(\mathtt{E})$ and $S(\mathtt{E}) = P_3 P_4 P_{10} P_{12}^{-1} P_{16} S(\mathtt{E})$

4. Each pair reduces the number of settings by a factor of 26. Since there are 2^{29} possible settings, find the smallest n such that $2^{29}/26^n \leq 1$. Solving, we find $n = 7$. With $n = 6$ pairs, the expected number of remaining rotor settings is less than 1.75.

5. a. Suppose that at step i, we press x and y lights up. Let

$$\begin{aligned} R_e &= \text{reflector} \\ R_\ell &= \text{leftmost rotor} \\ R_m &= \text{middle rotor} \\ R_r &= \text{rightmost rotor} \end{aligned}$$

   Then $y = S^{-1} R_r^{-1} R_m^{-1} R_\ell^{-1} R_e R_\ell R_m R_r S(x)$ Where "inverse" is thru the rotor from left to right (i.e., the inverse permutation). Note that the reflector is its own inverse since there is only one way to go thru reflector. At step i, we have

$$y = S^{-1} R_r^{-1} R_m^{-1} R_\ell^{-1} R_e R_\ell R_m R_r S(x)$$

Then, also at step i,

$$x = S^{-1}R_r^{-1}R_m^{-1}R_\ell^{-1}R_eR_\ell R_mR_rS(y)$$

since $R_e = R_e^{-1}$.

b. The primary advantage is that the same circuitry and same settings (i.e., key) are used for both encryption and decryption.

6. a. Let x be a plaintext letter and y the corresponding ciphertext letter. Then

$$y = S^{-1}R_r^{-1}R_m^{-1}R_\ell^{-1}R_eR_\ell R_mR_rS(x)$$

or

$$R_\ell R_mR_rS(y) = R_eR_\ell R_mR_rS(x)$$

Suppose $x = y$. Then $z = R_e(z)$, where $z = R_\ell R_mR_rS(x)$. But this implies that the reflector permutes a letter to itself, which is not allowed (as it would cause a short circuit). Therefore, $x \neq y$.

b. Suppose that you have a putative crib, that is, you suspect that you know the plaintext that corresponds to some given segment of ciphertext. If it so happens that one of the ciphertext letters matches the corresponding putative plaintext letter, then the crib must be incorrect.

7. Once the rotor settings are known, all P_i are known. Let X_i be the known input and Y_i be the corresponding known output. Then $Y_i = SP_iS(X_i)$ for each known value. Equivalently, $S(Y_i) = P_iS(X_i)$. If either $S(Y_i)$ or $S(X_i)$ is known, then we uniquely determine another stecker value. Of the 22 letters that appear in the table, at least 16 (including $S(\mathtt{E})$) can be found as discussed here. The remaining values could be found by trial decrypts of the ciphertext.

8. a. The initial positions of rotors LMR are `UPS`, respectively.

b. The stecker is `KZGLEUCJIHADXRYWQNSTFVPMOB`, that is the stecker has 10 cables with the following pairs of letters connected: `AK`, `BZ`, `CG`, `DL`, `FU`, `HJ`, `MX`, `NR`, `OY`, `PW`, and the following six letters unsteckered: `EIQSTV`.

9. Initial positions of LMR rotors are BIG, respectively. The plaintext is

```
TAKEABONEFROMADOGWHATREMAINSALICECONSIDEREDTHEBONEWOULD
NTREMAINOFCOURSEIFITOOKITANDTHEDOGWOULDNTREMAINITWOULDC
OMETOBITEMEANDIMSUREISHOULDNTREMAINTHENYOUTHINKNOTHINGW
OULDREMAINSAIDTHEREDQUEENITHINKTHATSTHEANSWERWRONGASUSU
ALSAIDTHEREDQUEENTHEDOGSTEMPERWOULDREMAIN
```

The difficulty here is that you must test English automatically. Using digraph (aka bigram) and trigraph (aka trigram) statistics should be sufficient. For tables of such statistics, see, for example, `www.data-compression.com/english.html`

10. For each possible rotor setting, and each of the 26 possible values of $S(\mathtt{E})$, encrypt EEEE..." and use the corresponding ciphertext to solve for the stecker value in each position. Accumulate a score for each stecker value as discussed in the text. Save the top scoring stecker values for each guess of the settings. Finally, do a trial decrypt for each putative setting (including the recovered stecker) and score the results.

    There are 26 choices for $S(\mathtt{E})$ and 26^3 initial positions for the three rotors, giving 26^4 settings to consider. The work factor is on the order of $26^4 \cdot T$, where T is the amount of data required. To estimate T note that if the rotor and stecker settings are correct, you expect to generate the correct ciphertext slightly more than 12% of the time, while if they are incorrect, you should only see a match about $1/26 \approx 3.8\%$ of the time. We can then use a normal approximation to estimate the value of T needed to ensure only one (or very few) random survivors from the $26^4 \approx 2^{18.8}$ initial settings.

11. Hmmm...

12. The displacements shift. Assuming the shift is left (which corresponds to "up" as discussed in the text), and that the displacements are numbered beginning with 0, we have

$$P_k = (0 + d_k, 1 + d_{k+1}, 2 + d_{k+2}, \ldots, n - 1 + d_{k+n-1})$$

    where all sums are taken mod n.

13. Programming exercise.

14. a. Any IV of the form IV $= (n, 255, V)$ will work.

    b. We have

$$K_n = \text{keystreamByte} - n(n+1)/2 - V - \sum_{i=3}^{n-1} K_i.$$

15. The probability the equation holds is $(253/256)^{251}$. The probability the equation for K_n holds is $(253/256)^{255-n}$.

16. If IV $= (1, 1, 254)$, then the array S is in the same state as when IV $= (2, 253, 0)$ (the example in the text) after the $i = 3$ initialization step.

17. The accepted approach is to discard the first 256 keystream bytes. Another option would be to hash the key and IV together. Other methods that mix the key and IV more completely are certainly possible.

18. a. We have, $k_0 = c_0 \oplus p_0$.

    b. TBD

    c. TBD

    d. TBD

    e. TBD

    f. TBD

g. It's easy to recover a WEP key of any size (assuming the first byte of plaintext in each block is known, which is usually the case). For more details, see, for example, the author's book, *Applied Cryptanalysis: Breaking Ciphers in the Real World.*

19. The key is $K =$ `dcba`.

20. The key is $K =$ `1ff1`.

21. An exhaustive key search reveals that the key is `0xd2a0`.

22. Assuming the indexing is the same as a DES S-box, the input 011101 indexes row 01 and column 1110, which yields output 1111. A difference of 000001 implies that the columns are the same, and the rows are either 0 and 1, or 2 and 3. In this case, the most likely output difference is 0010 which occurs with probability $20/32 = 5/8$.

23. In fact, $y_1 = x_0 \oplus x_2$, that is, this "approximation" is exact.

24. The most biased differences are given below.

input difference	output difference	probability
110100	0010	16/64
000011	0000	14/64
001100	1110	14/64
010000	0111	14/64
011101	1110	14/64
011110	0100	14/64
100100	1000	14/64
100100	1001	14/64
101001	1001	14/64
101010	1001	14/64
110101	1000	14/64
110101	1110	14/64
111110	0111	14/64

25. The most biased differences are given in the table below.

input difference	output difference	probability
001000	0010	48/64
111001	0010	18/64
000010	1001	16/64
110001	0000	16/64

26. The best linear approximation is

$$y_0 \oplus y_1 \oplus y_2 \oplus y_3 = x_1 \oplus 1$$

which holds with probability 50/64. The linear approximators

$$y_0 = x_0 \oplus x_1 \oplus x_2 \oplus x_4 \oplus x_5 \oplus 1$$

and

$$y_3 = x_0 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus 1$$

each hold with probability 46/64. There are several more that hold with probability 44/64.

27. The best linear approximator is

$$y_0 \oplus y_2 = x_2 \oplus x_3$$

which holds with probability 1. All of the following hold with probability 48/64.

$$\begin{aligned}
y_1 &= x_4 \oplus x_5 \oplus 1 \\
y_2 &= x_3 \\
y_1 &= x_2 \\
y_2 &= x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus 1 \\
y_2 &= x_0 \oplus x_4 \ominus x_5 \oplus 1 \\
y_1 &= x_0 \oplus x_3 \oplus 1 \\
y_2 &= x_0 \oplus x_2 \ominus 1 \\
y_1 &= x_0 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus 1
\end{aligned}$$

28. The probability of $(3/4)^3$ is about .42, so $k_0 \oplus k_1 = 0$ will hold "causally" for about $(.42)100 = 42$ of the cases, and for the remaining 58 we get a random match half of the time, or 29 times, for a total of $42 + 29 = 71$. Actual results may differ since we are assuming everything is uniformly random, which may not be the case

29. In this case, $P_1 = 01001010$ and $P_2 = 00110011$. The private key consists of $S = (3, 5, 10, 21, 45, 88, 180, 371)$, $m^{-1} = 14$ (since $m = 56$), and $n = 783$.

30. In this case, $P_1 = 1010001001$ and $P_2 = 0100011001$. The private key consists of $S = (5, 10, 21, 41, 89, 175, 423, 819, 1601, 4123)$, and $m^{-1} = 18$ (since $m = 439$), and $n = 7901$.

31. a. If $d_1 = 0$, choose Y and Z so that $Y^5 < N$ and $Z^4 < N < Z^5$. On the other hand, if $d_1 = 1$, choose Y and Z so that $Y^7 < N$ and $Z^6 < N < Z^7$.

    b. TBD

    c. The higher "roots" become meaningless before you can recover a sufficient number of bits.

32. a. Find minimum variance of $T(C_j) - \tilde{t}_{0\ldots2}$.

b. These would contribute to the "noise," with the net result being that the noise would swamp the "signal."

33. Programming problem.

34. Programming problem.

Chapter 7

1. a. GdnroGm and GdNrGM.

   b. rUWUSUr and RuwusuR.

2. a. "Pokemon, gotta catch them all"

   b. "Four score and seven years ago"

   c. "Give me liberty, or give me death"

   d. "I can't get no satisfaction" or, if you are a complexity theory person, "I can't get no satisfiability" or, if you are a geologist, "I can't get no solifluction" or ....

3. The fraud rate is the rate at which, say, Trudy is erroneously authenticated as Alice. This is often called the false accept rate (FAR). The insult rate is the rate at which Alice, say, attempts to authenticate, but the authentication fails. This is often given as the false reject rate (FRR).

   In statistics lingo, the hypothesis you are testing is the null hypothesis. If a user claiming to be Alice tries to authenticate, you might think it is most sensible to test the hypothesis that this user really is Alice. A Type I error occurs when the null hypothesis is true, but we reject it, which in this case corresponds to the insult rate. A Type II occurs when null hypothesis is false, but we accept it, which corresponds to the fraud rate. Of course, if you think it's more appropriate to hypothesize that anyone trying to authenticate is lying, then the null hypothesis is "Alice is really Trudy." In this scenario, Type I correspond to the fraud rate, while Type II corresponds to the insult rate. It's difficult to come up with a wrong answer to this one!

4. a. "Hello" corresponds to 43556.

   b. I can only find LINK and KINK.

5. a. Since the passwords are not salted, we can pre-compute the hashes of the dictionary words. Denote these hashes as $y_0, y_1 \ldots, y_{2^{20}-1}$. Let $p_0, p_1, \ldots, p_{511}$ be the hashes in the password file. Then the pseudo-code is simply

```
for i = 0 to 2^20 − 1
    for j = 0 to 511
        if y_i == p_j
            password found
        end if
    next j
next i
```

   b. The attack is the same as in part a., except that the hashes can't be pre-computed, so that before each comparison, we need to hash the dictionary word with the appropriate salt value.

6. a. If Trudy gets the file, she does not get the actual passwords.

b. The key must be available and if Trudy can get the file, she can probably also get the key.

c. A salt is a random, non-secret value appended to a password before it's hashed. Salting makes dictionary attacks much more difficult.

7. a. Assuming the hashes for the dictionary have been precomputed, the expected work is $1/4 \cdot 0 + 3/4 \cdot 2^{55} \approx 2^{54.6}$.

b. The expected work is $1/4 \cdot 2^{29} + 3/4 \cdot 2^{55} \approx 2^{54.6}$.

c. The probability is $1 - (3/4)^{1024} \approx 1$.

8. a. The system with the lower fraud rate is more secure.

b. The system with the lower insult rate is more user-friendly.

c. A merchant would likely choose the system that is most user-friendly.

9. a. Criminals believe there is a higher chance of getting caught if they use a card with a photo.

b. The quality and size of the photo is much greater. In addition, anyone who steals a card (or finds a lost card) might not know whether they look similar to the card holder.

10. a. There are 2^{48} passwords.

b. For a brute force attack, about 2^{17} seconds, which is about 36.4 hours.

c. The probability is $1 - (3/4)^{256}$, which is essentially 1.

d. About 2^{32}.

11. a. It is 2^{34} times easier.

b. If Trudy is wise, she will crack the second half first, starting with all nulls, then all 1-character with 6 nulls, all 2-character with 5 nulls, and so on. In this case, Trudy would recover the last 3 characters of the password with minimal work. She could then use these last 3 characters to construct a customized dictionary that would be likely to make the first half easier to crack.

12. a. Follow the same process with the putative password, hashing the first half with S_0 and the second half with S_1. It is a match if the first hash matches Y_0 and the second matches Y_1.

b. The work is 2^{56}.

c. Attack the second half first, and if solved, use the result to build a custom dictionary for the first half. For any password that is at least 9 characters, and somewhat less than 16, this approach would be likely to yield a significant shortcut.

13. a. 1) The email address (which is not secret) is acting as a password. 2) Anyone with temporary access to your email can get your password. 3) Passwords are sent via email, which is subject to snooping.

    b. No, since the plaintext passwords are retained. If the website hashed passwords—as it should—it could not send you your original password.

14. a. If the passwords are hashed, as they should be, then the SA does not have access to Alice's password.

    b. The SA knows the password. Also, it might be difficult for Alice to remember, since she did not choose it, or it might be something easy to guess (e.g., last 4 digits of student ID number).

    c. Yes, provided that the previous password hash (and salt) are retained.

15. a. No, if Trudy has recorded the response for a valid iteration that used R, she can simply replay the proper response if R is repeated.

    b. It seems to be OK.

16. The attacker could intercept the cookie and then replay it.

17. Written assignment.

18. a. If the MAC address appears in the packets, this could be used to indicate that it is the correct machine.

    b. Also require a password or biometric.

    c. The scheme in part a is not very secure—it's easy to spoof the MAC address (this is often done during attacks on WEP). The scheme in part b is better (much better if a strong biometric is used).

19. a. The probability is $1 - (3/4)^6 \approx 0.82$.

    b. The probability is reduced to 0.468.

20. a. The probability is p.

    b. The probability is $1 - (1 - p)^n$. Note that if $n = 1$, then the probability is p, which agrees with part a.

    c. The probability that Trudy can get one (or more) or your passwords is lowest if you have only one password (assuming that $n > 1$). However, a single password puts all of your eggs in one basket, so to speak, which could be a very bad thing. For example, if one of the accounts does not hash passwords, Trudy might have a good chance of obtaining your password from that account, and if you use the same password everywhere, you're hosed. So, as always, it depends.... But, as a general rule, using the same password in lots of places is a bad idea.

21. a. I think it's reasonable, since it's basically the approach I follow. The one difference is that in practice, it is not always possible to choose the same simple password for multiple sites (due to requirements on length, upper-case, special characters, etc.). So, in practice, I have 2 distinct "families" of passwords, where the passwords within each family are closely related.

    b. The different password requirements on different sites can be an issue. Also, it's easy to accidentally use the strong password on non-important site and/or a site that might not treat passwords properly.

22. a. Well-chosen is better, except there is the "all of your eggs in one basket" problem.

    b. One tempting approach is to choose distinct, but related passwords. However, this is a bad idea—if Trudy gets ahold of one of your passwords, she would have a good hint as to your other passwords.

23. a. The work is 2^{56}.

    b. The work is 2^{72}.

    c. The work is 2^{119}.

24. Denote the stored salted password hashes (and salt) as (y_0, s_0), (y_1, s_1), ... Let $d_0, d_1, \ldots, d_{2^n-1}$ be the dictionary words. Compute $h(d_0, s_0), h(d_1, s_0), \ldots, h(d_{2^n-1}, s_0)$, in each case comparing to y_0, stopping if a match is found. The probability of a match is p and in the case of a match, the expected work is 2^{n-1}. If no match, then compute $h(d_0, s_1), h(d_1, s_1), \ldots, h(d_{2^n-1}, s_1)$, in each case comparing to y_1, stopping if a match is found. The probability of a match at this step is $p(1-p)$ and the expected work is $2^n + 2^{n-1}$ (since we must do try all 2^n with s_0). Continuing, we find the expected work is (assuming M hashes in password file) $\sum_{k=0}^{M}(k2^n + 2^{n-1})p(1-p)^k$ from which the result follows by approximating the finite sum with an infinite series.

    Here's a nice alternative proof given by a student, Jay Freeman, in one of my classes:

    Let $W(m)$ be the expected work to recover a single password, given a dictionary of 2^n passwords and a file containing m salted password hashes, with probability p that any one given password is in the dictionary (the same p for all the passwords). The attack is to pick one salted hash from the file, compare it with the appropriately salted hashed passwords from the dictionary one at a time, and if it is not there, pick another salt/hash combination, and press onward in the same manner.

    If the first password chosen is in the dictionary (with probability p), then the expected work to recover it is 2^{n-1}. if it is not in the dictionary (with probability $1-p$), we can only find that out by going through all of the passwords in the dictionary (work 2^n), and then we have to attack what is in essence a new salt/hash file, comprising the $m-1$ salt/hash combinations left over after testing the first one.

    Then $W(m)$ can be written in terms of $W(m-1)$ as

$$W(m) = p2^{n-1} + (1-p)(2^n + W(m-1)).$$

But $W(m-1)$ is certainly less than $W(m)$, because it has one less salt/hash combination to try, so we may substitute $W(m)$ for $W(m-1)$ in the above and change the equal sign to less than, that is,

$$W(m) < p2^{n-1} + (1-p)(2^n + W(m)).$$

Solving this for $W(m)$, and taking care not to invert the sense of the "<" accidentally, we have

$$W(m)(1+p-1) < p2^{n-1} + (1-p)2^n$$

and so on writing 2^n as $2 \cdot 2^{n-1}$ we obtain

$$pW(m) < p2^{n-1} + 2(1-p)2^{n-1}.$$

Finally, dividing by p and factoring out the 2^{n-1} on the right, we have the desired result, namely,

$$W(m) < 2^{n-1}(1 + 2(1-p)/p).$$

25. From the previous problem, for small p we have that the expected work is

$$2^{n-1}(1 + 2(1-p)/p) \approx 2^n/p.$$

The only requirement on M is that it is large enough so that we can safely ignore the case where no password from the dictionary is in the password file. Of course, this depends on p, since the probability that at least one password in the file is also on the dictionary is given by $1 - (1-p)^M$.

26. a. You would expect to find $10^5 \cdot 10^7/10^{10} = 100$ false matches.

    b. The probability is only $10^7/10^{10} = 1/1000$.

27. a. Spit into a bucket on your way in to work.

    b. Security — get someone else's DNA and you would be mis-authenticated.
    Privacy — DNA analysis might be possible, revealing personal info, such as genetic predisposition to disease, etc.

28. a. See definitions in the text.

    b. Authentication is much easier. Each comparison carries a probability of a false match, so the more comparisons the more errors, and identification requires far more comparisons.

29. a. The rate at which Trudy is authenticated as Alice.

    b. The rate at which Alice is not authenticated as Alice.

    c. The error rate when the parameters of the biometric are adjusted so that the fraud rate equals the insult rate. It is useful for comparing different biometrics.

30. a. Wear a long gown, use crutches, etc.

b. Depending on your personal hygiene, wear strong cologne or take a bath.

31. a. Grow a beard, wear a hat pulled down low over your face, wear sunglasses, etc.

b. The casino could place a security guard at the entrance and require customers to take off sunglasses, remove hats, etc., before entering.

c. In principle, the casino could ask people to remove sunglasses, etc., before entering. However, if the casino asks patrons to shave their beards, the casino will surely be sued by the ACLU....

32. a. In one case, they construct a thumb out of "ballistics gel" (a rubber-like substance) and etch the valid user's thumbprint onto it. This is a pretty sophisticated attack and would require considerable effort. At the other extreme, they make a photocopy of the valid user's thumbprint, enhance it slightly with a pen, then press it onto the sensing device with a live thumb. This does not work—which leads to the sophisticated attack mentioned above. However, at the end of the show, they get a low-tech attack to work by simply licking their thumb before pressing the photocopy onto the sensory.

b. The sensor could, for example, also look at the blood vessel pattern in the thumb (using a strong light). This would make it much harder to forge a valid thumbprint. Of course, the tradeoff is that this would make the detector more expensive.

33. a. You could try to build a "ballistics gel" (or rubber) model of the valid user's hand. It would be more difficult to obtain the necessary information, as compared to obtaining a fingerprint.

b. The hand geometry would probably be much easier to break, since hand geometry is inherently less discrimination than a fingerprint. However, in practice, this would certainly be more difficult to obtain a user's hand geometry as compared to a fingerprint.

34. a. TBD

b. TBD

c. Pro: Harder to attack and therefore more robust.
Cons: Scanner would be more expensive and invasive.

d. I'd choose the iris scan, since it is somewhat less invasive, so it has less potential to do harm.

35. a. Depends on the individual, but you should be able to point out some similarities.

b. Depends on the people, but there are probably some fairly obvious differences.

c. As always, it depends...

d. There are a lot of standard statistics that can be extracted from speech, related to pitch and such. These would be a good place to start.

36. a. Building a fake retina that can defeat the scanner is not going to be easy, unless you happen to be in the film *Mission Impossible*.

b. It would be much harder to fake an iris pattern, or obtain someone's iris pattern, etc.

c. Cost.

37. a. We find $d(A, B) = 29/64 = 0.453125$, $d(A, C) = 39/64 = 0.609375$, and $d(B, C) = 34/64 = 0.531250$.

b. User W is Alice (distance 0.156250), user X is Bob (distance 0.156250), and user U is Charlie (distance 0.171875). All other distances are near 0.5, so V and Y are none of the above.

38. a. Easy.

b. If there is no challenge T, then the response is always the same, $h(K)$, which would eliminate the need for Alice to actually posses the SecurID device. So, it would not be a "something you have" form of authentication. Also, if Trudy observed one response, she would be able to replay that response forever to (mis)authenticate as Alice.

c. By using T, we save one message—there is no need to send the challenge. Note that T acts like a challenge R that Alice knows, so there is no need to send it. The down side to using T is that time becomes a security issue—if Alice and Bob's clocks are out of sync, by more than a minute, they cannot communicate. This opens up a new avenue of attack for Trudy.

d. Both are equally secure if implemented correctly.

39. a. Attacks on the hash function to recover K might be worth considering.

b. Denial of service.

c. Shoulder-surf to obtain Alice's PIN, then steal her password generator.

40. a. Perhaps you could use a webcam and require the user to give a sign language version of their password.

b. Press a button on the wireless access point and enter a password.

Chapter 8

1. a. Someone who had automated their testing and accomplish more in a few hours than manual testing could accomplish in several weeks.

   b. For my money, the whole thing is nonsensical, so it's not difficult to find more examples.

2. EAL1 — Functionally Tested: Documentation must be examined and the product will be independently tested to verify it matches documentation and guards against some known threats.
   EAL2 — Structurally Tested: Includes testing in EAL1, but goes a step further in looking at design information.
   EAL3 — Methodically Tested and Checked: Includes testing in EAL2 but ensures more security by being more thorough in the testing of the product. The testing provided by the product creators is selectively confirmed. The product creators must also provide evidence they considered "obvious vulnerabilities."
   EAL4 — Methodically Designed, Tested, and Reviewed: Includes the testing in EAL3 but also evaluates the design process and independently tests for "obvious vulnerabilities."
   EAL5 — Semiformally Designed and Tested: Includes the testing in EAL4, but further strengthens the independent tests by considering "resistance to penetration attackers with a moderate attack potential" (GAO06392). Furthermore, evidence of "planned development, with a rigorous development approach" (GAO06392) must be provided.
   EAL6 — Semiformally Verified Design and Tested: Includes the testing in EAL5 but with consideration of "penetration attackers with a high attack potential" (GAO06392).
   EAL7 — Formally Verified Design and Tested: Includes the testing in EAL6. Also includes verification of formal design, mathematical proofs, and correctness preserving transformations to code. Testing further includes evaluation of testing by the product creators.

3. a. Advantages of capabilities include that it is easier to delete users, and that it is easier to delegate.

   b. ACLs are simpler to implement and with ACLs it is easier to change permissions for a given resource.

4. a. You would need to keep track of who you are acting on behalf of and use their permissions in the appropriate ACLs.

   b. This is gets fairly complicated with ACLs, but is easy with capabilities.

   c. Capabilities—see part b.

5. a. Integrity, non-repudiation, and the Bob's authority to act on Alice's behalf could be time-limited.

   b. It's more complex, so more chances for errors.

6. Many business applications could be considered.

7. It means that a person only knows as much as is necessary to do their job. In other words, you cannot access some information, even if you have the appropriate clearance, unless it is necessary. Compartments are the primary means of enforcing this principle, since you are not allowed to see compartmented info unless you have specifically been given access to the appropriate compartment.

8. a. This is not so easy as with TCP, since the headers are smaller. But you could simply send a series of UDP packets and use the delay between packets to encode information (e.g., "long" delay is a 1 while a "short" delay is a 0).

   b. If you suspect such a covert channel is being used, you could randomly delay packets.

9. a. "High watermark" means that you start out at a low level clearance, and your active clearance gets upgraded as needed—up to the maximum level that you are allowed. The low watermark applies to integrity, and it means that your integrity level goes down to the integrity level of data that you read.

   b. BLP with the weak tranquility property is compatible with a high watermark.

   c. Biba's model is compatible with a low watermark.

10. a. Put a file on the print queue and delete it—similar to the file example in the text. Unless you automate it, the capacity would certainly be significantly less than 1 bit per second.

    b. Using sequence numbers is pretty subtle (as discussed in the text). A timing channel would also be difficult to detect (although it is not specific to TCP). Many other covert channels are possible.

11. a. Query set size control—don't return the answer if too few respondents. N-respondent, $k\%$ dominance rule—essentially, a more refined form of query set size control, where answer is not returned if $k\%$ or more comes from N or fewer respondents. Randomization is just what it says—ad some random noise to the reply.

    b. The first two will increase the difficulty of finding specific information, but probably not prevent such attacks. A weakness of randomization is that it could be a problem if very precise answers are necessary (e.g., if you are doing research on a rare disease).

12. a. Consider an MLS system, where you want to prevent queries at one level from leaking information at a higher level. You might simply check the result of a query against information at higher levels before returning the response.

    b. Malware that attacks the software would be an option.

13. a. IRC is a very natural way to communicate with a large number of hosts.

b. From Trudy's perspective, it would be harder for the good guys to detect control information.

c. Suppose your bots scan a particular social networking site and look some obscure topic. The botmaster could post to that topic and the information could be interpreted as commands.

14. Reading assignment.

15. a. No. It is better to use inference control even if it is weak rather than do nothing. There is very little extra work to add such inference control, and it makes the attacker's job more difficult.

b. Yes. In most cases it would be better to use no encryption rather than use a weak cipher. Encrypted data is a sign that the information is important, so it might be more likely to be filtered, and once it's filtered it could be broken if the cipher is weak.

c. Same answer as in part a.

16. This can be accomplished by simply including confidentiality and integrity labels with all objects, and then apply both BLP and Biba. Lipner's model combines confidentiality and integrity into one coherent model; see the chapter titled "Multilevel Security Models" in the *Handbook of Information Security* for a brief discussion of Lipner's model.

17. "No write up, no read down."

18. a. See the reference for details and examples.

b. EZ Gimpy can be broken at least 80% of the time, perhaps up to 90%. Hard Gimpy is more secure, but attacks are improving.

c. Basically, they are OCR problems, and the state of OCR has improved greatly since these CAPTCHAs were proposed.

19. a. For a typical test-based CAPTCHA, a word is randomly selected from a (large) dictionary and then a randomly-selected distortion (from a large set of possible distortions) is applied, and the result is displayed. The human needs to type in the word.

b. We assume that the attacker knows the dictionary, the set of possible distortions, and has access to the software.

20. Programming problem.

21. a. A good source of info on audio CAPTCHAs (including real-world audio CAPTCHAs and attacks) is the article found here:
`www.captcha.net/Breaking_Audio_CAPTCHAs.pdf`

b. TBD

22. See the article mentioned in the previous problem for some ideas.

23. a. The most difficult for computer to solve is the segmentation problem, that is, the problem of determining letter boundaries. Today, OCR is so strong that if the letter boundaries can be determined, good OCR software can determine the letters.

    b. Once the segmentation problem has been solved, you know the letter boundaries. Today, OCR is so good that if a human can recognize the letter, high-quality OCR can too, with a high probability of success.

    c. There are only 26 letters—even including case, special characters and so on, there are perhaps a hundred or so distinct characters to consider. On the other hand, the possibilities are almost endless when we need to separate letters that have been run together.

24. a. The time spent solving reCAPTCHAs would be more 500,000 hours, but only about half of that would be used to solve OCR problems. So, about 250,000 hours daily.

    b. About 1280 days, or about 3.5 years. Of course, the problem is actually much more difficult than that, since these 32,000,000 book are in 470 different languages. Also, in addition to the 32,000,000 books, the library has more than 62,000,000 manuscripts. In addition, a 10 hour per book assumption is probably overly optimistic. And in reality, it is necessary to solve each OCR problem multiple times (see the next part of this problem), which increases the time factor proportionally. Finally, it is not realistic to think that all CAPTCHAs would be reCAPTCHAs focused on this problem. So, realistically, it is probably a decades-long project (not to mention the fact that the library holdings are constantly growing...).

    c. Each time she solves a CAPTCHA, Trudy could give one correct answer and one incorrect answer. About half of these would be accepted as valid, and in each of these accepted cases, the OCR word would be incorrect. This would have little effect if only Trudy does this, but if she could recruit lots of people to "solve" lots of CAPTCHAs in this way, it would magnify the effect (see the next problem).

    d. They would need to reCAPTCHA each OCR word multiple times before assuming that it is correct. While this makes the system more robust, it also makes it take longer to clean up OCR errors.

25. a. Free email services require that you solve a CAPTCHA to obtain an account, and spammers need lots of accounts.

    b. As of 2011, some websites were claiming a rate of \$0.02 to \$0.05 per CAPTCHA.

    c. It has been reported that people who solve CAPTCHAs are sometimes enticed to do so using free pornography.

26. a. Packet filter — network layer
    Stateful packet filter — transport layer
    Application proxy — application layer.

b. Packet filter — static info, such as IP addresses, protocols, port numbers, etc.
Stateful packet filter — all of the info available to the packet filter, plus TCP connections and other ongoing conversations
Application proxy — all info available to stateful packet filter plus application data

c. Packet filter — TCP ACK scan
Stateful packet filter — Firewalk
Application proxy — DoS attack (i.e., overwhelm the firewall)

27. Some firewalls claim to do "deep packet inspection," which seems to mean that they are like a stateful packet filter that also looks at application data. That is, they look at the application data without all of the processing (and overhead) that would apply to an application proxy.

28. a. Connections could not be immediately terminated and would have to be allowed to timeout.

b. Maybe.

29. a. Each packet that is forwarded through an application proxy is a new packet, that is, the old packet is destroyed and a new one created. Consequently, the TTL field will be set to a default value, and the host that receives the packet will not respond with a time exceeded error message.

b. I asked my magic 8-ball, and it replied, "My sources say no."

30. a. The "time exceeded" messages is not sent.

b. It would mean that errant packets would circulate longer than they would otherwise, but that is probably not a major issue.

c. I asked my magic 8-ball, and it replied, "Concentrate and ask again."

31. It is easier to make sure that one machine is doing things correctly, as opposed to having to make sure that a bunch of machines are all doing the right thing. In other words, it is much simpler, from an administrative point of view.

32. A packet filter would still work, but not a stateful packet filter or an application proxy.

33. a. At a minimum, the length, fragmentation, TTL, source and destination IP fields cannot be encrypted, since routers need to see the information in those fields.

b. Neither the TTL nor fragmentation fields can be integrity protected, since routers can change the values in those fields.

c. Assuming the "data" is also encrypted, then only a rudimentary packet filter would work. If the data is not encrypted, then a reasonable version of any of the firewalls would work.

34. a. Same as part a in the previous problem.

b. Same as part b in the previous problem.

c. In this case, any firewall will work, since the firewall decrypts the packets.

35. Defense in depth is a good general security strategy. For example, you might have both a network-based IDS and host-based IDS to catch any attacks that evade the network-based IDS.

36. a. Signature-based detection is relatively fast and it is effective against known attacks.

b. Anomaly-based detection gives us a chance of detecting previously unseen attacks.

c. It's much harder to defend against the unknown.

37. a. Much quicker development and distribution of signatures—standard systems typically only distribute signatures once a week or less.

b. It works... In fact, it is nearly as effective as a theoretical anomaly-based system—the nearer the delay in distributing signatures is to zero, the nearer it is to being as effective as an anomaly-based system (in the sense of detecting new attacks).

c. It's only based on signatures—none of the anomaly-based statistical issues are present.

d. Marketing appeal—anomaly-based is a lot sexier than signature-based, and with no standard terminology, nobody can complain (at least not too much...).

38. a. Commands issued, typing speed, mouse movements, activity versus inactivity, time of use, among many, many other possibilities.

b. More statistics would give a clearer view of the user's activity.

c. More statistics would make it slower, and it might also tend to give more false alarms, since a legitimate user is more likely to vary from one of the stats.

39. a. Use changes over time, so if these values do not change, you will soon get many false alarms.

b. Trudy can simply "go slow" and eventually convince the IDS that her actions are normal.

c. My personal favorite would be to train an "hidden Markov model", or HMM, on the data. This process is relatively efficient, so the updating step could simply involve re-training, where new data is included and the oldest data is discarded (a sliding window approach). Of course, any other machine learning technique (e.g., neural nets) could be used instead.

40. a. In this example, $S \approx 0.118$, which exceeds the threshold of 0.1 given in the book. Therefore, this would be considered abnormal.

b. TBD

41. a. TBD

b. Since the threshold is 0.1 and $\sqrt{0.1} \approx 0.31$, it will take at least three "normal" iterations before $H_3 > 0.9$.

42. a. The value of $A_3 \geq 0.39$ will trigger an alarm, while $A_3 = 0.36$, with $A_0 = 0.02$, $A_1 = 0.32$, and $A_2 = 0.30$, will stay well below the threshold of $S = 0.1$.

b. See part a.

c. See part a.

Chapter 9

1. Simply replace symmetric key encryption with a hash that uses a key, or, preferably, use an HMAC.

2. The following are examples: 1) Encrypt sender's name along with timestamp and key. 2) Bob remembers all received timestamps within the clock skew., so that replay attacks are not possible.

3. a. In Figure 9.22, replace the signature/public key encryption with symmetric encryption.

   b. Use timestamps instead of nonces.

4. The following attacks will work: 1) Trudy opens one connection to Bob and sends the first message, obtaining Bob's reply. Then she opens a second connection and sends $R+1$ to Bob. She uses Bob's response to complete the first connection, letting second one time out. 2) Trudy can record messages 1 and 3 in a legitimate connection between Alice and Bob, then replay them later. Note that a MiM does not work, here.

5. The obvious thing to do is just repeat the attack over and over—if she can repeat it 500 times without being detected, she's likely to succeed.

6. a. Fewer messages (i.,e., more efficient).

   b. Time is a security-critical parameter (or, simply, "clock skew").

7. a. Yes. The value S is needed in order to determine K, which is used in the final message, and Alice can verify that K was used in the final message. Only Bob knows his private key, so only Bob could determine S from $\{S\}_{\text{Bob}}$.

   b. No. Only public key operations are required.

8. 5and $K = h(S, R_A, R_B)$ is the session key.

   a. Yes. The final message will convince Alice that she has made contact with Bob. Only Bob knows K_{AB}, which is needed to determine S, which is needed to compute K, and if Alice decrypts and finds SRVR, then she knows that the responder must know K.

   b. Yes, but this is somewhat subtle. Note that Trudy (masquerading as Alice) can send a random string of bits in message 3 in place of $E(S, K_{AB})$. Bob would then "decrypt" this random string and whatever the result, he would assume that it is S. However, Trudy cannot determine a valid K without knowing the result S that Bob obtains, and Trudy cannot know this unless she knows K_{AB}. Therefore, Bob authenticates Alice in message 3 when he decrypts $E(\text{CLNT}, K)$ and finds the results is CLNT. Since Trudy cannot compute $E(\text{CLNT}, K)$, Bob does not mistake Trudy for Alice.

9. 1. If Trudy acts within the clock skew, she can send $\{K\}_{\text{Bob}}$ to Bob (as herself) and Bob will respond with $\{K\}_{\text{Trudy}}$ so that Trudy can find K.
   2. Trudy can be a MiM and forward the first message from Alice to Bob, but change $\{K\}_{\text{Bob}}$ to, say, $\{K_T\}_{\text{Bob}}$ where K_T is a key of Trudy's choosing. Then Trudy shares a key with Bob.

10. a. This is not a good idea. If Trudy recovers the session key, then she can determine K. The point of having a session key is that if it gets compromised, only that particular session is affected.

    b. Good.

    c. Bad.

    d. Good.

    e. Good.

11. The protocol in Figure 9.23 is an example (in any case, you'll need to use timestamps). This particular protocol does not provide for anonymity, but if you modify it so that in the first message, "Alice" is inside the encryption (but not the signature), then it provides anonymity.

12. Perhaps surprisingly, this protocol does provide plausible deniability. Consider the following exchange of messages.

Trudy (as Alice)	$\longrightarrow$	I'm Alice, X	$\longrightarrow$	Trudy (as Bob)
Trudy (as Alice)	$\longleftarrow$	Y	$\longleftarrow$	Trudy (as Bob)
Trudy (as Alice)	$\longrightarrow$	$\{S\}_{\text{Bob}}, E(\{X\}_{\text{Alice}}, K)$	$\longrightarrow$	Trudy (as Bob)
Trudy (as Alice)	$\longleftarrow$	$E(\{Y\}_{\text{Bob}}, K)$	$\longleftarrow$	Trudy (as Bob)

    Suppose that Bob observers these messages. Bob can determine S by using his private key and therefore he knows K. He then verifies that by applying Alice's public key to the quantity sent in message 1, he obtains the value that is encrypted with key K in message 3 (and similarly for messages 2 and 4). According to the protocol, this is what he is supposed to see, so it appears to be a legitimate connection. The problem here is that Bob has no way to know that he is actually encrypting X, as opposed to verifying the signature on R_A. To any other observer, the interaction also appears to be valid.

    Notice also that Trudy can simply replay the first two messages, then choose any value S and complete both sides of the protocol. Consequently, it might be argued that this is enough to show that the protocol provides plausible deniability. However, an observer could note that both of the random nonces R_A and R_B are repeated from a previous interaction, and the chance of this is negligible. Note that repeated nonces indicate that a connection was almost certainly forged. Furthermore, if the nonces are not repeated, then Alice and Bob cannot "plausibly deny" the conversation. Therefore, the argument above is required to show that the protocol does indeed provide plausible deniability.

13. Yes, this protocol does provide plausible deniability since Trudy, acting as both Alice and Bob, can construct a valid set of messages.

14. A simple example of such a protocol is the following.

$$\begin{array}{lcccl} \text{Alice} & \longrightarrow & R_A & \longrightarrow & \text{Bob} \\ \text{Alice} & \longleftarrow & [R_A]_{\text{Bob}}, R_B & \longleftarrow & \text{Bob} \\ \text{Alice} & \longrightarrow & [R_B]_{\text{Alice}} & \longrightarrow & \text{Bob} \end{array}$$

To see that this provides plausible deniability, Trudy can generate random X and Y and send

$$\begin{array}{lcccl} \text{Trudy (as Alice)} & \longrightarrow & \{X\}_{\text{Bob}} & \longrightarrow & \text{Trudy (as Bob)} \\ \text{Trudy (as Alice)} & \longleftarrow & X, \{Y\}_{\text{Alice}} & \longleftarrow & \text{Trudy (as Bob)} \\ \text{Trudy (as Alice)} & \longrightarrow & Y & \longrightarrow & \text{Trudy (as Bob)} \end{array}$$

Then to any observer (including Alice or Bob), The value $\{X\}_{\text{Bob}}$ looks like a random nonce and X is the result of "signing" this nonce. Similar comments hold for $\{Y\}_{\text{Alice}}$ and Y.

15. It is actually an advertised feature of some protocols. However, in some usage scenarios, it could cause problems and if that's the case, then such a protocol should not be used.

16. This is the same protocol that appears in Figure 9.10, so the attack is the same as in Figure 9.11. Also, the approach in Figure 9.12 prevents this attack. It is a good exercise to find other modifications that prevent the attack.

17. This is the same protocol as in Figure 9.24; the attack is the same as in Figure 9.25 and a modification that prevents the attack is given in Figure 9.26. It is a good exercise to find other modifications that prevent the attack.

18. a. Bob does not know which public key to use to verify the signature so the protocol is not practical (or "effective").

    b. Secure mutual authentication and session key.

    c. Secure mutual authentication and session key.

    d. Alice is not authenticated since Trudy could replay the message. Bob does not know what K is, so he cannot verify the signature.

    e. Mutual authentication is OK, but the key is likely weak. Presumably, Trudy can guess the time to within, say, five minutes. Then even if the time resolution is to the microsecond, Trudy needs to try less than 2^{30} keys for an exhaustive search.

19. a. Bob does not know it's Alice, so he does not know which key to use, so, this is not practical.

    b. Secure mutual authentication and key.

c. Secure mutual authentication and key.

d. Secure mutual authentication and key.

20. a. No authentication of Alice.

b. Mutual authentication and secure session key.

c. No authentication of Alice or Bob. Even if the authentication were OK, the key K is problematic since Trudy could replace it with a key of her choosing.

d. Mutual authentication, but Trudy can replace $\{\text{"Alice"}, K\}_{\text{Bob}}$ with $\{\text{"Alice"}, K'\}_{\text{Bob}}$, where she knows K'.

e. No authentication of Alice—cannot let "Alice" choose her own challenge!

21. a. Trudy has one chance in $5 \cdot 60 \cdot 1000 = 300{,}000$.

b. There are many possible solutions. The time resolution can be reduced to, say, microseconds. Or the protocol could be modified to so that Bob must return K instead of (or in addition to) $T + 1$, in which case K would act as a session key and, effectively, also as a nonce (of course, a separate nonce could be used too). Or a signature could be used instead of encryption in the second message.

22. a. Trudy conducts a MiM attack, but she sends $\{g^t \bmod p\}_{\text{Alice}}$ to Alice in message 2, in place of $\{g^b \bmod p\}_{\text{Alice}}$. Trudy also opens a connection with Bob, claiming to be Alice, and sends $\{g^a \bmod p\}_{\text{Bob}}$ in place of R_A in message 1. Bob "signs" this value, which decrypts it, and sends $g^a \bmod p$ to Trudy. Trudy then lets this connection die and she can compute the session key that Alice will use. Note that Bob uses a different session key, in spite of the fact that mutual authentication succeeds.

b. Yes, indeed. At the very least, Trudy will be able to decrypt one message from Alice.

23. a. Mimic the protocol the "sign and encrypt" with a timestamp protocol, but include a nonce with the first message, and replace the key K with the nonce in message 2 (actually, there is then no need to encrypt the second message). This requires just two messages.

b. This seems to require three messages, since Bob must send the nonce/challenge in message 2, and Alice must then respond appropriately in message 3. So there is no benefit to this approach as compared to using nonces.

24. a. The third message only requires knowledge of Bob's public key, so it's easy.

b. No, since Trudy does not know $g^b \bmod p$.

25. a. No, since Trudy cannot sign R_B as Alice.

b. Yes, since Trudy can do a MiM attack and get Alice to generate her key as $K_A = g^{at} \bmod p$ and simultaneously get Bob to generate his key as $K_B = g^{bt} \bmod p$,

where Trudy knows t, and hence Trudy knows K_A and K_B. Note that the authentication was successful, so Alice and Bob have no reason to suspect anything is amiss, yet Trudy can read and/or modify all of their encrypted messages.

26. The crucial difference is that Alice chooses K in the latter protocol while Bob chooses K in the former. In the latter case, the clock skew allows Trudy to replay $\{T, K\}_{\text{Bob}}$ and obtain K. In the former protocol, Trudy can find $\{R, K\}_{\text{Alice}}$ and $\{R+1, K\}_{\text{Bob}}$, but neither of these can be replayed to find K, since Bob chooses K.

27. Bob can remember each received timestamps until it is older than the allowed clock skew, and refuse to accept any duplicate timestamps.

28. One easy fix would be to require intervention by the pilot before a challenge is encrypted. Presumably, the pilot would know when he was approaching his own base. However, this is risky, since the pilot might make a mistake by, say, pressing the wrong button, or responding too slowly, etc.

29. a. Yes. If Trudy observes one iteration of the protocol, she can guess PIN numbers until she correctly decrypts, say, $E(\text{Bob}, R_A, K)$. Note that Trudy knows R_A (and "Bob"), so she will know when she has found the correct PIN. SInce there are only 10,000 possible PIN numbers, this is a trivial attack for Trudy.

    b. Yes, it's secure since a brute force attack is not possible.

30. Yes, but this is more subtle than the previous exercise, since Trudy does not know R_A or R_B. In this case, Trudy must guess a PIN number, generate a putative key K_{PIN}, then "decrypt", say, $E(R_A, K_{\text{PIN}})$ and $E(R_A, R_B, K_{\text{PIN}})$. Only when the PIN number is correct will Trudy see the same (unknown) value for R_A in both cases.

31. a. As a passive attacker, Trudy would need to solve the discrete log problem, which we assume is infeasible, so the answer is no.

    b. Trudy can act as a MiM, but she only gets one chance (per iteration) to guess the PIN, since she must do so in real time. This gives her a 1/10,000 chance of success, per iteration. Note that after the fact, she cannot guess the PIN, since she would need to break the discrete log problem, which implies that offline PIN cracking is not feasible in this case. So, her probability of success depends on how many iterations of the protocol she can attack. Of course, once she discovers the PIN, she can continue to break the protocol until the PIN changes.

    It is worth noting that this use of Diffie-Hellman makes a weak PIN-based (or password-based) protocol more secure, at virtually no additional cost.

32. Use RSA to generate a public-private key pair, use it once to establish a session key, then discard the private key. While this would work, it is far less efficient than using the ephemeral Diffie-Hellman approach discussed in the text.

33. It is possible to achieve a similar effect using a cryptographic hash function h. Suppose Alice and Bob share the key K_{AB}. Alice selects a session key K and sends $E(K, K_{AB})$

to Bob. Alice compute $K'_{AB} = h(K_{AB})$ and then forgets K_{AB}. Bob receives $E(K, K_{AB})$ and decrypts it to obtain K. Bob then computes $K'_{AB} = h(K_{AB})$ and forgets K_{AB}. The new "long-term" secret is K'_{AB} and if Trudy somehow gets this she cannot determine K_{AB} (the hash is one-way) and hence she cannot determine K from $E(K, K_{AB})$. At each subsequent connection, this process would be repeated. There is a practical difficulty of staying in sync, but, in principle, this would achieve a similar effect as perfect forward secrecy.

34. Put a soundproof glass window in the cave. Then Bob can watch to be sure that Alice can open the entrance, without learning the phrase (unless he's a good lip reader).

35. a. Make the cave door one-way, and Bob knows which way it opens, say, from side "R" to "S" in the Figure, and Alice is supposed to always choose the R side. But, Bob does not get to see which side Alice actually chooses. Now, if Bob tells Alice to come out the S side, Alice must know the secret phrase (assuming she followed the protocol), but if Bob tells her to come out the R side, she does not need the phrase. Bob must still choose the side at random—if Bob's choice is predictable, Trudy can cheat by simply going down the side that Bob will choose.

    b. This new cave protocol still differs from Fiat-Shamir. In this cave protocol, Alice does not make any random choice, whereas in Fiat-Shamir, she chooses a random value for r. In this sense, the original analogy was somewhat more accurate, since Alice made a random choice as to which side, R or S.

36. a. It is easily verified that Bob accepts this iteration.

    b. It is easily verified that Bob accepts this iteration.

    c. Evidently, Alice chose the same r for both iterations. From the second iteration, third message, we know that $r = 10 \bmod 63$ and from the first iteration, third message, $r \cdot S = 4 \bmod 63$. Then using the hint, $S = r^{-1} \cdot r \cdot S = 19 \cdot 4 = 13 \bmod 63$.

37. a. It is easily verified that, in this case, $y^2 = x \cdot v^e \bmod N$.

    b. In the first message, Trudy choose $x = r^2 \cdot v^{-1} \bmod N = 60$. Then in the third message, Trudy sends $y = r = 10$. It is easily verified that Bob accepts this iteration of the protocol. This shows that Bob's choice of e must not be predictable.

38. a. The public value is $v = S^2 \bmod N = 81 \bmod 55 = 26$.

    b. For $r = 10$, the first message is $r^2 \bmod 55 = 45$.

    c. Alice sends $r = 10$ in message three.

    d. Alice sends $r \cdot S = 90 \bmod 55 = 35$ in message three.

39. We have $y^2 = 30^2 = 900 \bmod 55 = 20$ and $xv^e = 5 \cdot 4 = 20 \bmod 55 = 20$. Since the numbers are so small, by brute force we find $S = 15$, since $15^2 = 225 = 5 \bmod 55$.

40. a. At the first iteration Bob sends $e = 0$, and he receives r in message three. At the next iteration Bob sends $e = 1$, and he receives rS in message three. Bob (or anyone else who observed the messages) can find S from $r^{-1}rS$.

    b. Anyone who knows S can impersonate Alice, thereby breaking the security of the protocol.

41. It is straightforward to verify $y^2 = x \cdot v^e \mod N$ for parts a and b. From the first iteration, Bob knows $r = 657$. Apparently, Alice re-used $r = 657$ in the second iteration, so Bob also knows $rS = 26{,}938$. Bob can therefore recover S by using $r^{-1} = 208 \bmod 27{,}331$ to compute $r^{-1}rS = 208 \cdot 26{,}938 = 249 \bmod 27{,}331$.

Chapter 10

1. a. The signature S_A, since only Alice could have signed it, and Bob can verify the signature. Note that H is included in S_A, and H contains Bob's challenge, R_B, so this provides the replay protection.

   b. Trudy would have to break the Diffie-Hellman key exchange by, presumably, solving an intractable discrete log problem.

   c. Put Trudy in the role of Bob. Then after the 4th message, she and Alice will have agreed on a key $K = g^{at} \bmod p$. This does not break the protocol—the authentication will fail, since Trudy cannot forge the required signature in message 4. So Alice will terminate the protocol, and she'll never use K.

   d. In this version of the protocol, the encryption serves no purpose. However, in the password version, the encryption does have a purpose (see the next problem).

2. a. He must know Alice's password. Note that it is not sufficient for Bob to store the hash of Alice's password—he must know the actual password.

   b. No. The authentication fails, so Alice never sends the final message.

   c. Advantage: convenient (passwords are much easier to set up and administer than certificates).
   Disadvantages: all the usual password cracking issues.

3. a. Bob must have some way to associate a public key with Alice. Typically, this is accomplished by having a database of public keys and Bob simply looks up "Alice's" public key to be sure that it really belongs to Alice. Note that this would work well for a small number of users, but it would probably be impractical if there are a large number of users.

   b. Similar to part a, Alice must have some way to associate"Bob's" public key with Bob.

   c. Advantages: no certificates required which eliminates many PKI issues. Disadvantages: must have knowledge of public keys in advance and must have a way to associate public keys to users (typically, via a database lookup).

4. a. TBD

   b. See the RFCs mentioned in the statement of the next problem. These list the various packets that can appear.

5. a. Messages 1 and 2.

   b. Messages 3 and 4.

   c. The last message.

   d. These messages are for exchanging the crypto parameters that are used for client authentication. Note that if password authentication is used, these are not necessary, which explains why this info is not exchanged in the initial CP/CS messages.

6. a. Surprisingly, none, since the pre-master secret S serves as Alice's challenge to Bob, not the "nonces". Note that in any case S must be chosen at random. These "nonces" are important in the connection protocol, since the ensure that the session keys for different connections differ.

   b. None, since Trudy must know the key K to compute the HMAC, which implies that she must know S.

   c. None, assuming the hash function is secure.

7. Authentication occurs on the last message, so we can leave everything else the same, but change the final message to, say, $[h(\text{msgs}, \text{SRVR}, R_A, R_B), S]_{\text{Bob}}$. Actually, this is costlier than the original protocol and has no benefit.

8. a. Alice sends $g^a \bmod p$ in message 1, Bob sends $g^b \bmod p$ in message 2, and encrypts his certificate in message 2 with the key $K = g^{ab} \bmod p$. A passive attacker cannot do the man-in-the-middle attack that would be needed to decrypt the certificate.

   b. Yes.

9. a. Remove the certificate from message 2, and encrypt S in message 3 with the shared symmetric key.

   b. Scalability. Each pair of users who want to communicate must share a symmetric key, whereas in the public key case, everyone only needs their own public/private key pair (and if we only authenticate the serve, as is commonly done over the Internet, then only the server needs a valid key pair).

10. a. TBD

    b. TBD

11. a. You do not have to rely on the OS.

    b. Applications are "automatically" secure.

12. a. They both accomplish essentially the same thing—authentication, session key, etc.

    b. Complexity, layer at which the operate, etc.

13. a. It should fail when Alice tries to verify the certificate, since it is not Bob's certificate.

    b. Alice's browser will warn her that the certificate is not valid, but it gives her the option to continue. If Alice chooses to continue (as most users wouldd), then the attack will succeed.

14. a. The advantage is that Alice could change her password without changing her Kerberos key. Then there would be no need to securely update her key with the KDC.

b. Storing $E(K_A, K)$ and the additional computation required might be considered ever-so-slight disadvantages, but these are insignificant.

15. a. So that only Bob can decrypt it. The session key K_{AB} that Alice and Bob will share is included in the ticket.

b. Bob will know that the included key is to be used to communicate with Alice.

c. This serves no obvious purpose, since the ticket to Bob is already encrypted with K_B, which only Bob and the KDC know. Also, Alice sends the ticket to Bob without any additional encryption.

d. It would be more efficient (in terms of bandwidth usage) to send it directly to Bob, but then Bob would have to remember this info until Alice contacts him. That is, Bob would have to maintain state, and Kerberos is all about being stateless.

16. a. Ticket Granting Ticket — it serves as a user's "credentials," that is, it enables a user to request ordinary tickets.

b. This enables the KDC to remain stateless. In effect, the KDC distributes the database of TGT information to the clients.

c. The KDC—and only the KDC—can decrypt a TGT, which enables the KDC to "remember" everything it needs to know about the user.

d. Apparently, this serves no purpose, since the TGT is already encrypted and it is freely passed about (with no additional encryption) in subsequent interactions.

17. a. Usually, anonymity with symmetric keys is, at best, difficult (see IPSec, for example). However, in Kerberos, anonymity when requesting a ticket is easy, since all TGTs are encrypted with K_{KDC}, which is known only to the KDC. Consequently, the KDC does not need to know whose TGT it is before decrypting it.

b. At this point, Alice does not have any TGT/credentials, so the trick described in the solution to part a does not apply.

c. The ticket to Bob (which is encrypted) tells Bob that he should be communicating with Alice.

18. a. The KDC makes this possible. Each user shares one key with the KDC, then the KDC acts as a go-between to establish connections between users.

b. The KDC is a TTP, which, in effect, takes the place of the PKI.

19. a. This exhaustive search would, on average, require $(16^5)/2 = 2^{19}$, which is infeasible if the guesses must be entered by hand.

b. Begin by trying each letter in the first position, with wildcards in the remaining 4 positions. After 8 guesses, we expect to have found the first hex digit. Then repeat this for the second and subsequent digits. An average number of only $5 \cdot 8 = 40$ guesses are required. See also the discussion of "linearization attacks" in Chapter 11.

c. Dave would be wise to exploit vouchers that were about to expire. He could use the wildcard feature to cash the voucher, without having possession of the voucher. In fact, physical vouchers were required as part of the record-keeping, but nobody examined them carefully (and they were often in very poor condition). So to satisfy the paperwork requirement, Dave could simply submit a worthless voucher that he found on the floor.

d. The biggest risk is that Dave the actual voucher would appear after Dave had already cashed it. In fact, this happened and eventually led to Dave being discovered.

e. The actual voucher was required as part of the receipt. But for this to be meaningful, someone would need to look at the receipts, which apparently never happened due to the volume of vouchers and the poor condition of old vouchers. It would not be reasonable to identify the purchaser, since race track patrons often want to remain anonymous. A less trivial wildcard feature might be helpful, but it would be important not too make it too cumbersome, or it will create too much work for managers. Some hashing or crypto schemes could prove useful here.

20. Yes, due to the fact that IPSec operates at the network layer whereas SSL is at the application layer. Security at the network layer is inherently more complex than at the application layer—when working at the application layer, the complexities of the network can, for all practical purposes, be ignored.

21. a. Main mode tries to hide identities (although without success in symmetric key mode). Also, main mode MUST be implemented, while aggressive mode SHOULD be implemented.

b. There is no need to know the public keys to begin the protocol. Since everyone knows their private key, you can start the protocol without waiting around to obtain the other side's public key. Of course, at some point, you need the other party's public key (to verify the "proof"), but obtaining the public key can be done in parallel with the protocol.

22. a. Phase 1 does the mutual authentication and establishes a key. Phase 2 establishes the actual IPSec connection (and could be used to establish all sorts of other connections too, but in practice it's not).

b. In symmetric key mode, the IP address is used to determine the client's identity, which effectively does nothing to hide the identity (which is, after all, suppose to be the primary benefit of main mode). In short, the public key mode works as advertised, while the symmetric key mode does not.

23. a. The stated purpose is to make IPSec stateless (or at least more stateless).

b. Information sent in the first messages (CP and CS) must be remembered for computing proofs of identity. So, client and server must maintain state from the first message on.

c. It might be possible to embed information (such as CP and CS) in the cookies, thereby delaying the point at which the client and server must actually maintain state.

24. In signing mode, the signature is used to authenticate. In encryption mode, the proofs need not be signed or encrypted since the symmetric key encryption with the key K is used for authentication. More precisely, private key operations are needed to determine K, since the nonces are encrypted, so knowledge of K provides authentication.

25. In aggressive mode, the crypto parameters g and p must be known before starting, while in main mode, they can be negotiated. In addition, main mode MUST be implemented, while aggressive mode SHOULD be implemented, so you may have no choice but to use main mode, in spite of its almost non-existent advantages over the more efficient aggressive mode.

26. No, it is not a flaw, since authentication will fail if Trudy attempts a MiM attack, so the key will never be used.

27. a. No. This is the sense in which IPSec provides anonymity.

    b. No. This is the sense in which IPSec provides anonymity.

    c. Yes. If Trudy can pretend to be Bob, she can establish a shared key with Alice, then Trudy can use this key to decrypt $E(\text{"Alice"}, \text{proof}_A, K)$ to reveal Alice's identity. Authentication will fail, but not before Trudy has determined Alice's identity.

    d. No. If Trudy pretends to be Alice, the authentication will fail in message 5, so Bob will never send message 6, which is the first place where he identifies himself.

    e. Yes. Since it works for an active attacker, it must also work for an insider.

    f. Yes, Trudy can simply open a connection with (unknown) Bob based on the IP address in Alice's connection with Bob.

28. A passive attacker can determine Alice's identity, unless you consider Alice's (static) IP address to be anonymous. However, if you do consider the IP address to be anonymous, then even an active attacker cannot determine Alice's identity (since the symmetric key K_{AB} is needed to reveal "Alice").

29. Even an active attacker is out of luck in this mode, since the appropriate private keys are needed to reveal identities (in particular, Trudy acting as Bob cannot determine R_A).

30. Same as the main mode case in the previous problem.

31. a. IPSec tunnel mode encapsulates the entire IP packet in another packet. If the source and destination are the same as in the encapsulated packet, the new header will be the same as the original header, so the identities of the endpoints are still known to a passive observer.

b. If the encapsulation occurs at a firewall and the destination of the encapsulated packet is another firewall, a passive attacker only knows that the firewalls are communicated—not which hosts behind the firewalls are communicating. This assumes that ESP with a non-NULL encryption algorithm is used.

32. a. No, since there is no place for the original IP addresses.

b. Yes, but it's less efficient since the IP addresses are duplicated.

33. The NULL encryption algorithm MUST be supported as one of the crypto options in ESP.

34. In AH, some additional fields of the IP header are integrity protected. In ESP (used from host-to-host), firewalls won't know whether NULL encryption is used or not, so they cannot use the header info for filtering.

35. a. The data is encrypted, so the firewalls cannot look at the data to determine whether to filter a packet or not. Note that from the perspective of IP, the data includes things like the TCP header, which has information required by virtually any firewall. Consequently, firewalls cannot make sensible filtering decisions when confronted with ESP-protected data. For this reason, IPSec is often implemented at the firewalls, instead of at the end systems, much to the chagrin of many self-proclaimed security experts.

b. Although the data is not encrypted, this presents the same problem as in part a, since the firewalls have no way to know that NULL encryption was used—the firewalls only know that ESP was used, not which specific "encryption" algorithm.

c. This does not present a problem for the firewalls—they can see that AH is used, and consequently they know that it is safe to look at the data. Note that this implies that when using IPSec from host-to-host, if the data must pass thru a firewall, as a practical matter, the protection is limited to AH.

36. a. It's a very bad idea. In WEP, the keystream is repeated each time the IV repeats (assuming that the long-term key K doesn't change). However, this modified version is even worse, since each packet is encrypted with the same keystream—this is at least as bad as using a one-time pad over and over and over and over and....

b. It would be worse. While the way that the key K and IV are combined in WEP leads to a fairly simple cryptanalytic attack, this attack would be even easier.

37. a. The expected number of IVs until the first one repeats is given by n in the equation $(1-p)^n = 1/2$, where $p = 1/2^{24}$. Solving, we find that $n \approx 11{,}629{,}079$. At 11 Mbps and 1500 bytes per packet, there are about 916 packets per second. So, we would expect to see the first IV again after about $11{,}629{,}079/916 \approx 12{,}696$ seconds, which is about 3.5 hours.

By the birthday problem, some IV will repeat after about 2^{12} packets. Again, at 11 Mbps and 1500 bytes per packet, there are about 916 packets per second. So, we would expect to see a repeated IV after about $2^{12}/916 \approx 4.5$ seconds.

b. In this case, the first repeat occurs after 2^{24} packets have been observed, which takes about $2^{24}/916 \approx 18{,}316$ seconds, which is a little more than 5 hours.

c. This is at least as bad as a one-time pad being used more than once. That is, repeated IVs make cryptanalytic attacks a realistic possibility, and the more repeats for a give IV, the stronger the attacks become.

d. Assuming the first byte of the plaintext in each packet is known, there is a very efficient and straightforward attack to recover the key. Interestingly, the attack is essentially no more difficult regardless of the key length.

38. a. The keystream bits of interest are given by $C \oplus P$, so Trudy replaces C with $X \oplus C \oplus P$.

b. Trudy must also change the CRC "integrity check" to match, which is easy to do (in particular, it does not require knowledge of the key).

39. a. This has all of the inherent problems of passwords. Even worse, the SSID is sent in the clear. So, if Trudy happens to intercept the message from a client to the access point which contains the SSID, she will then know the "password". In addition, there are hacker tools available that will cause WEP to "disassociate", which forces any active clients to resend the SSID. So there is no need for Trudy to wait—she can simply force the SSID to be sent, assuming there is at least one active client using the access point.

b. This is also of limited value, since the MAC address can be spoofed. As with the SSID, there are tools available that make this task easy for attackers.

40. a. So the government could easily listen to phone calls and scan for possible additional terrorist attacks.

As an aside, note that if these reports are true, it suggests that the Russian government was not capable of monitoring GSM traffic (at least not on a large scale) at that time. This is interesting since it is generally believed that GSM was an abysmal security failure, yet this would be some fairly strong evidence that the practical level of security provided by GSM was actually reasonably good.

b. Yes. Recall that the base station decides whether to encrypt or not, and the user has no knowledge as to whether encryption is actually occurring or not.

41. The mobile could send a nonce R in message one, and the base station could return $[R]_{\text{BS}}$, where "BS" is the base station, along with RAND in message two. There are many other reasonable approaches.

42. a. No database of stored keys is required.

b. Single point of failure.

c. The database of keys is distributed between many home networks, so if one is compromised, only a subset of user keys is compromised. If a different "master" key were used for each AuC, then this point would not be a significant issue.

43. Cloning is essentially a replay attack. Assuming that the mobile and base station have public and private key pairs, the following one-message protocol satisfies the criteria, and also provides for anonymity. Here, T is a timestamp and the base station is not authenticated.

$$\text{Mobile} \longrightarrow \{\text{IMSI}, K, [T]_{\text{Mobile}}\}_{\text{BS}} \longrightarrow \text{Base Station}$$

44. The following protocol satisfies the requirements.

$$\begin{array}{ccccc} \text{Mobile} & \longrightarrow & \{\text{IMSI}, K, [T]_{\text{Mobile}}\}_{\text{BS}} & \longrightarrow & \text{Base Station} \\ \text{Mobile} & \longleftarrow & T+1 & \longleftarrow & \text{Base Station} \end{array}$$

Chapter 11

1. Complexity is self-explanatory. Paul Kocher (creator of SSL and all-around security guru) has said that "complexity is the enemy of security." There are many reasons for this: Complex systems are difficult (if not impossible) to analyze, complex software will inevitably have bugs, and so on.

   Extensibility means that the capabilities of a system can be extended. It is difficult (if not impossible) to analyze a system that can change.

   Connectivity creates more avenues for attack and increases the number of potential attackers.

   See Hoglund and McGraw's book, *Exploiting Software*, for a discussion of these topics in the context of software.

2. Validation refers to checking the validity of input. Failure to validate input can result in buffer overflow attacks, Web attacks, etc.

3. There are many to choose from. Mydoom is one interesting example.

4. A race condition is more a matter of timing by the attacker rather than a "race." The attacker tries to exploit an operation between stages, such as in the `mkdir` example in the text. An example of a real-world race condition is discussed here
   `www.securityfocus.com/archive/1/195617`
   For an excellent discussion of race conditions in general, see
   `www-128.ibm.com/developerworks/library-combined/l-sprace.html`

5. a. A change occurs between the point where a condition is checked and the use of that result.

   b. Yes.

   c. The `mkdir` example given in the text is one. Another example is the web form example given at the beginning of the TOCTTOU wiki page.

6. a. If the canary value is overwritten, then there is likely a problem with the return address.

   b. The Microsoft implementation allowed the user to define a handler function to be called when the canary died. It was claimed that the handler function could be specified by the attacker. This would make *all* buffer overflows exploitable, even those that were not exploitable when the canary was not used.

7. Most modern viruses and worms have exploited buffer overflows to do their evil deeds. For example, Slammer and Code Red both exploited buffer overflows.

8. If you overflow a buffer on the heap, it may overwrite other data stored on the heap. A reasonably good discussion of heap overflows can be found at
   `julianor.tripod.com/heap/heaptut.txt`

9. If a signed integer has it's first bit set, then it is a negative number, so it might pass some bounds check. However, if the same bit string is then interpreted as an unsigned integer, it will be a very big number, which will likely overflow array bounds (an example is given in Problem 16).

10. Reading assignment.

11. The user must specify n, and if it is larger than the target array, an overflow will still occur.

12. Possibly. It depends on what action is taken when the buffer overflow is detected. It also depends on how serious it is if the affected processes are killed. For example, if the buffer overflows only affect a few relatively unimportant threads, it might not do too much harm to Alice even if all of these are killed off.

13. a. Yes, since no code executes on the stack.
    b. No, since code cannot execute on the stack.
    c. No code executes on the stack.

14. At a minumum, the following C functions are inherently unsafe:

    `strcpy`, `strcat`, `sprintf`, `gets`.

    The safer alternatives are, respectively,

    `strncpy`, `strncat`, `snprintf`

    (there is no safer alternative to `gets`). Each of these is safer, but not completely safe.

15. a. The results are `buf2 = 22222222` and `buf2 = 11122222`.
    b. Apparently, buffer 1 has overwritten the start of buffer 2.
    c. Trudy might be able to overwrite some important data to, say, change a failed authentication into a successful authentication.

16. a. If `len` is negative there is a problem. The test in the `if` will be passed, but then `memcpy` assumes that `len` is unsigned. So, a negative value for `len` is interpreted as a very large number, which would lead to a buffer overflow.
    b. Explain how an integer overflow might be exploited by Trudy. Trudy can overflow an array and cause problems.

17. a. Programming problem.
    b. S123N456 (which is clearly visible in disassembly.

18. a. The process is not atomic—the value is read, then later it is written.

b. One attack would be to have 2 cards, say one with a value of \$1 and one with a value of \$1000. Insert the cards on top of each other, so that the $x = 1000$ is read. Then insert, say, \$5 and pull out the \$1000 card before pressing the *enter* button. If you can do this, a value of \$1005 would be written to the card that had a \$1 value, and the \$1000 card would still be worth \$1000.

Another attack is to insert the card, wait until the value x is read from the card, then make a transaction on the account before entering y dollars and pressing enter. In this way, $x + y$ is written to the card, even after some of the x dollars have been spent. Note that this attack only works for transactions where the debit card is not required (for example, online fund transfer).

c. Read the value just before writing it.

19. a. Programming problem.

b. The possibilities are endless...

20. a. Programming problem. Note that some authors define a virus to be parasitic code, and such code is fairly challenging to write. However, due to the definition used in this book, there is no such requirement for this problem.

b. Use your imagination.

21. a. Programming problem.

b. Let me count the ways...

22. a. Polymorphic malware mutates the decryptor code.

b. Metamorphic malware is "body polymorphic," that is, the body of the code is mutated. This eliminates the need for encryption, so metamorphic code is not encrypted.

23. a. Different instances of the code perform the same function, but the internal structure differs.

b. Metamorphism is strong means of evading signature detection.

c. If the internal structure of the code differs, an attack on one instance will not necessarily work on other instances. For example, buffer overflow attacks are very delicate, so a change in the internal structure will likely prevent such an attack from working—even if the buffer overflow vulnerability is present in all instances of the code.

24. Randomly insert dead code, change instructions to something equivalent but different, randomly reorder the instructions (where possible), etc.

25. This is much more challenging than the metamorphic generator discussed int he previous problem. The code would have to disassemble itself, and modify itself, and the code that does the modification would need to be modified as well (since it is part of the program). This is nontrivial task and that is likely the reason why hackers have not been able to take much advantage of metamorphism in the realm of worms.

26. a. The standalone generator is much easier. The malware that carries its own generator must be able to morph the generator code.

    b. The malware that carries its own generator would likely be easier to detect. Even if the morphing was sufficient to obscure any signature (a difficult task), it would still be more likely to be susceptible to heuristic analysis or detection via emulation.

27. The biggest issue is that for a polymorphic worm, the morphing engine is part of the encrypted body, so it does not need to be morphed. In contrast, the metamorphic worm needs to morph its own morphing code.

28. a. It's difficult to produce a strong metamorphic generator.

    b. They are very metamorphic, but nevertheless they are very different from "normal" code.

    c. The metamorphic code can be modified to be more like normal code by simply inserting code segments taken from normal programs. At some point, the statistics of the viruses will be too similar to those of the normal programs for reliable detection. That is, the code must be highly metamorphic (to evade signature detection) and sufficiently similar to normal code (to evade statistical analysis). Of course, behavior analysis or anomaly detection might still be possible.

29. a. Weakness of slow worm: more chance that it will be detected Another possible weakness to a slow worm: According to an audience member at a talk at a Defcon, it's "way cooler to infect the entire Internet in 15 seconds."

    b. Weakness of flash worm: more difficult to code, more work required in advance (determine vulnerable IP addresses)

30. a. Yes, it is. Metamorphic generators are readily available, and these are widely used to create distinct copies of malware, that is, copies that have distinct signatures but the same (or similar) function.

    b. Detect the goodware (via signatures) and assume everything else is malware.

31. a. This seems to have been a direct command and control architecture. Some estimates of its size were as high as 8 to 12 million machines, but that seems a bit far fetched. It is notoriously difficult to accurately estimate the size of a botnet, and the people doing the estimating (i.e., security companies) often have a vested interest in making things appear is bleak as possible, so "buyer beware." Mariposa was supposedly dismantled in 2009.

    b. There are many variants, and it is sometimes considered to be a worm. Estimates of the size are as high as 7 million active infections (as of 2011).

    c. Kraken is interesting because it was dismantled but, as of 2010, it was staging a comeback, infecting several hundred thousand machines.

    d. Once, one of the largest botnets (and responsible for a large volume of spam) it seems to have died out.

32. a. Reading assignment.

    b. Presumably, the developers would argue that these have legitimate uses.

33. Stealing corporate secrets.

34. a. Eliminate the competition—present and future.

    b. Watch for unauthorized virus scans, etc.

35. a. Programming-like exercise.

    b. The serial number is checked one character at a time. As a result, incorrect serial numbers that have more leading characters correct will take longer to execute. If the attacker can measure these timing differences in any statistically significant way, then the serial number is vulnerable.

36. The more characters that are correct, the longer the program takes to execute, due to the `flag = false` statement being executed. So, in principle, the basic linearization attack will still work, although the timings need to be more precise (or more timings need to be taken).

37. Possibly OK (or at least very difficult to time), but it depends on inner workings of the hash function, and it would be preferable to not have to rely on such details (which could, conceivably, differ from platform to platform).

38. a. There would still be a statistical difference between the correct and incorrect case.

    b. The random delays would make the linearization attack more difficult, since Trudy would need to repeat each case more times to detect the underlying timing difference, but it does not eliminate the fundamental problem.

39. a. Almost certainly not. It is highly unlikely that `strcmp` was implemented with this attack in mind, so it is likely that there is some timing difference between the correct and incorrect cases.

    b. Possible implementations are given here: `en.wikipedia.org/wiki/Strcmp` Undoubtedly, additional optimizations are used to speed it up even more.

40. a. The correct 7-digit serial number is 8675309.

    b. The expected number of guesses for an arbitrary 7-digit serial number is $7 \cdot 5 = 35$ but it is likely that more guesses were required before the timing difference became apparent.

    c. If the code were not susceptible to a linearization attack, the expected work is $(10^7)/2 > 2^{23}$.

41. a. A salami attack could be designed to truncate any fraction of a cent left over from a calculation, and credit this to the attacker's account.

b. Each transaction will result in an average of 1/2 of a cent, so 1000 transactions has an expected value of 500 cents, or $5. Each weak this would give the attacker $35, and each year, about $1820.

c. In the movie *Office Space*, there was a misplaced decimal point, so the small fraction of a cent was actually a large chunk of salami. Otherwise, an audit might eventually discover the discrepancy.

42. There are some potential problems here. First, the compiler might optimize out the `count = count + 0;` line. Even if the compiler leaves the code as is, it's highly unlikely that the `if` branch takes exactly the same amount of time as the `else` branch—you would need to look very closely at the resulting assembly code to determine whether this is the case. And as long as there is any timing difference, no matter how small, there is still the potential for a successful timing attack. Smaller timing differences only serve to increase the number of tests that must be conducted.

43. This code would do the trick:

```
int main(int argc, const char *argv[])
{
   int i;
   int t;
   char serial[9]="S123N456\n";
   if(strlen(argv[1]) < 8)
   {
      printf("\nError---try again.\n\n");
      exit(0);
   }
   t = 0;
   for(i = 0; i < 8; ++i)
   {
      t += argv[1][i] ^ serial[i];
   }
   if(t == 0)
   {
      printf("\nSerial number is correct!\n\n");
   }
}
```

44. Reading assignment.

Chapter 12

1. a. Programming problem.

   b. DEADbeef.

2. a. Perhaps the most obvious would be to change the `jz` to an unconditional `jmp`. Other approaches are not so easy. You might be able to change `test eax,eax` to, say, `test eax,0`, however, you'd have to be sure that this is the same size as the original. Another possibility is to insert NOP instructions (i.e., "do nothing" instructions) to overwrite all of the instructions from the test instruction, up to location 401045 (which is where you want to jump to). This latter approach would require some testing to make sure that no important instructions were overwritten.

   b. The correct serial number won't work—all others will, but not the correct one!

3. a. Programming exercise.

   b. TBD

4. a. Programming problem.

   b. Programming problem.

5. a. Programming problem.

   b. Obfuscate the custom class loader, for example.

6. a. Programming problem.

   b. Step over the calls to `IsDebuggerPresent()`.

7. a. The output is different for case iii as compared to ii and i.

   b. It must be a palindrome—spelled the same forwards and backwards.

   c. It has some strange behavior. This is better than terminating immediately, since an attacker might not be sure what caused the termination.

   d. You can simply replace the calls to `IsDebuggerPresent()` with `NOP`s.

   e. TBD

   f. Programming problem.

8. a. Programming problem.

   b. TBD

9. a. Yes, but it would be very difficult. It would be necessary to analyze assembly code and determine what it is doing.

   b. Yes, but it would be somewhat more difficult, since you would lose the "big picture" view that the assembly code provides.

10. It might be possible to jump into the middle of an "instruction," which would yield a similar effect as the false disassembly discussed previously.

11. You would have to decrypt to check the hash, or hash the encrypted code, both of which are problematic.

12. a. Opaque predicates might force an attacker to analyze much more code.

    b. The triangle inequality would be a good one, that is, the sum of the distances from a to b and b to c must be greater than or equal to the distance from a to c. This would be a complex expression to unravel in assembly code.

    c. Input three points in the pane, then build an opaque predicate based on the triangle inequality, as mentioned in part b.

13. a. Change `if(a < b)` to `if(a < b || a > b || a == b)`. Of course, this is not very stealthy.

    b. Obvious.

    c. It's not stealthy, but it could be made more stealthy by basing it on some identity.

14. a. One simple approach would be to hide a watermark in the code that never executes.

    b. You could execute (or emulate) the code for various inputs, and if some branch never executes, change it. This would be risky, unless you knew which inputs were of interest.

15. You could "pack" the code, that is compress it. The effect is very similar to encrypted code. Some debuggers can automatically unpack standard packers, but if you customize the packer, it would be a challenge to disassemble.

16. Use multiple threads and create deadlock situations.

17. a. It's protection that stays with content after it has been delivered.

    b. The key must be given to the potential attacker, so it is not sufficient. It is necessary to use encryption to securely deliver the content.

18. a. They could sell things over the Internet without fear of piracy.

    b. Medical records could be posted online, which would enhance access by doctors, but access could be controlled.

19. Not necessarily, since important information (e.g., a key) could be determined by debugging the code, without patching it.

20. a. Closed systems are relatively good at protecting their secrets.

    b. Open systems are more flexible—they can do more things.

21. a. You could re-allocate memory each time the data is cached and use a randomly generated key to encrypt the data before storing it. Of course, the key also needs to be available to recover the cached data.

    b. It would likely be more difficult to analyze and harder to reverse engineer.

22. a. Without such protection, one successful attack can be applied to all instances of the software.

    b. For another approach, see Microsoft's discussion of "individualization" at
    `msdn.microsoft.com`
    `/library/default.asp?url=/library/en-us/dnwmt/html/wmrm71security.asp`

23. a. The key should be made to look like structured data, such as text. One way to accomplish this is to disburse the key in a large amount of ordinary text. This is, perhaps, somewhat counterintuitive.

    b. The software could be used to reconstruct the key from structured data in some non-obvious way.

24. a. Buffer overflow attacks are delicate—things must align properly. By changing the internal structure, one successful attack is unlikely to work on other instances of the code, even if the buffer overflow flaw is present in all instance. That is, the attack must be customized for each instance of the software, and some instances will likely be immune to attack.

    b. Possibly race conditions.

    c. Software development would be a nightmare if many copies of the software must be maintained. However, the metamorphism could be applied to a base version of the code, so only that base version needs to be maintained. However, there would still be significant issues with bug tracking and such.

25. a. It's essentially a reputation system, which can enable users to detect websites that do not follow good practices in dealing with private info.

    b. For one thing, there is no authentication, so malicious users can skew the results without being detected.

26. a. The good guys only found 1/1000th of the bugs, so the chance that they did not find Trudy's bug is 99.9%.

    b. Let $P(k)$ be the probability that the good guys found exactly k of Trudy's bugs. Then $k \in \{0, 1, 2, 3\}$ and

$$P(k) = \binom{3}{k} \left(\frac{1}{1000}\right)^k \left(\frac{999}{1000}\right)^{3-k}.$$

    The probability that at least one of Trudys bugs was not found by the good guys is

$$1 - P(3) = 0.999999999$$

    which is as close as you'll ever come to metaphysical certitude.

27. a. Trudy should find 4 bugs and the good guys should find about 2,000.

    b. Let $P(k)$, for $k = 0, 1, 2, 3, 4$, be the probability that exactly k of Trudy's bugs are found by the good guys. Then

$$P(k) = \binom{4}{k} \left(\frac{1}{5}\right)^k \left(\frac{4}{5}\right)^{4-k}.$$

    The probability that at least one of Trudy's bugs is not found by the good guys is

$$1 - P(4) = 0.9984.$$

28. The bugs are also harder for the good guys to remove, so there are more of them than if it were open source.

29. a. A system where the user has easy access to the internal workings of the system.

    b. A system where the user does not have easy access to the internal workings of the system.

    c. Open systems are more flexible—they can be made to do many different things. In other words, they are extensible.

    d. Closed systems tend to be able to protect their secrets better—it is much more difficult to break the security of such a system.

30. The closed source project also has an MTBF of t/K. The argument is the same as given in the text for the case where bugs are twice as hard to find.

31. At first glance, it appears that the MTBF in the closed source case is better. That is, it seems that the MTBF should be $(2t)^2/K = 4t^2/K$. However, the testing is only half as effective so that we have

$$\text{MTBF} = 4(t/2)^2/K = t^2/K$$

    in the closed source case, which is the same as in the open source case.

32. Under this assumption,

$$\text{MTBF} = \frac{1}{100} \sum_{k=1}^{100} k = \frac{(101)(100)}{2 \cdot 100} = 50.5 \text{ hours}.$$

33. a. Secret plans would make it harder for someone else to build their own death star, and secret plans would probably make it more difficult to attack the Good Death Star.

    b. If the plans are open, errors might be easier to find, and at less cost. An open design might also convince enemies of the potential strength of the Good Death Star.

c. Six of one, half-dozen of another.

34. a. Since 1/4 of the inserted typos were found, it is reasonable to assume that the 800 represents about 1/4 of the typos. Therefore, about 3,200 typos were in the manuscript, of which about 2,400 remain. This is assuming that the inserted typos are somewhat similar to the "wild" typos.

b. Replace "typos" with "bugs" and "manuscript" with "software" and the same technique could apply to software. However, inserting representative bugs into code will likely be far more challenging than inserting representative typos.

35. a. Since 40% of the known bugs were found, you might reasonably assume that 40% of the unknown bugs were found, which implies that there were 300 bugs when you started. Since 120 of these were found there are now about 180 bugs remaining.

b. The inserted bugs might not be representative of the unknown bugs.

36. No alpha testing is the optimal choice, since it costs far more to pay developers to find each bug ($1000) than it costs the company in lost sales when they let customers find the bugs ($20).

37. Let n be the number of security flaws that developers find before Doors is released. Then $1{,}000{,}000 - n$ flaws remain, and the revenue to SM is $R_n = 2{,}000{,}000(500) - 20(1{,}000{,}000 - n)$ dollars. Number the bugs from 0 to 999,999 in the order they are (or would be) found. Then the time (in hours) that it takes a developer to find the kth bug is $100{,}000/(1{,}000{,}000 - k)$, and hence the cost to find the kth bug is $C_k = 100 \cdot 100{,}000/(1{,}000{,}000 - k)$, and the total cost of finding n flaws is

$$\begin{aligned} C_n &= \sum_{k=0}^{n-1} C_k \approx 10{,}000{,}000(\ln 1{,}000{,}000 - \ln(1{,}000{,}001 - n)) \\ &= 138{,}155{,}105 - 10{,}000{,}000 \cdot \ln(1{,}000{,}001 - n) \end{aligned}$$

So the problem is to maximize $R_n - C_n$, where n ranges from 0 to 1,000,000. That is, we want to maximize $f(n) = c + 20n + 10{,}000{,}000 \cdot \ln(1{,}000{,}001 - n)$, where n in the range of 0 to 1,000,000. We find $n = 500{,}001$, that is, SM should find about half of the flaws before release.

Chapter 13

1. TCG = Trusted Computing Group
   TCB = Trusted Computing Base
   PITA = Pain in the A$$ (OK, that one is not in the book, but I couldn't resist)
   MAC = Mandatory Access Control
   DAC = Discretionary Access Control
   NGSCB = Next Generation Secure Computing Base

2. a. Trust implies reliance, that is, a trusted system is one that you rely on for your security.

   b. Yes. If you don't trust it, you don't rely on it for security, so if it's compromised, it won't affect your security. On the other hand, if you rely on it, and it's compromised, then your security will likely be affected.

3. a. Segmentation divides memory into logical units while paging uses fixed size pages. Paging is simpler and more efficient than segmentation, but segmentation is logically "better."

   b. Segmentation provides for more fine-grained control.

   c. Efficiency or simplicity.

4. One approach would be to use paging, but restrict a page to one logical unit. This would be somewhat more wasteful of memory than standard paging, but it would assure that each page required only one level of protection.

5. a. Mandatory means that access is not controlled by the owner of the object, whereas discretionary implies that access is controlled by the owner of the object.

   b. The definition pretty much says it all.

   c. Mandatory includes such things as MLS or any type of system-wide protection such as a firewall that prevents users from accessing certain data. Discretionary would include ACLs and capabilities.

6. If Trudy "owns" the OS, she can break core security processes which would enable her to break the security of virtually any application.

7. a. A blacklist is a list of things (or actions) that are explicitly not allowed. By default, anything not on the blacklist would be allowed.

   b. A list of things (or actions) that are explicitly allowed. By default, anything not on the whitelist would be disallowed.

   c. Whitelisting is definitely more secure since whenever some new security risk appears it would have to be put on the blacklist.

d. Blacklisting is likely to be far more user-friendly, since with whitelisting, everything is considered "guilty until proven innocent," so lots of things that users want to do will probably not be on the whitelist (or will be added to it only slowly).

8. The TCB in NBSCG consists of the Nexus and NCAs. Without a TCB, there would be no point to NGSCB. The purpose of NGSCB is to protect the secret/key that is hidden in the protected memory, and the TCB is the part of NGSCB that we rely on to accomplish this.

9. Malware protection is another potential benefit to the NGSCB approach. Since malware could not pass the attestation check, it would be possible to virtually eliminate many types of malware, at least on the trusted RHS of an NGSCB-ed system.

10. Perhaps the biggest issue in DRM is protecting a key in software. With NGSCB, you could make sure that only trusted software is allowed to access the key, which would makes attacks to recover the key very challenging.

11. In MLS, users only have access to certain information, and trusted software could be used to enforce such restrictions.

12. Businesses have the same basic problem—how to restrict access to information when resources are shared.

13. It's definitely a possibility. However, in recent times, vendors seem to have lost interest in this approach. Since such a system would have some obvious benefits in the business world, it will be interesting to see if the ideas are revived in the future.

14. a. Both are forms of access control, etc. Both have proven to be difficult—if not impossible—to actually implement securely in any meaningful way in practice. But perhaps the deepest link is that, at some level, both deal with the "remote control" of data, that is, attempting to place restrictions on data that is outside of your immediate control.

    b. For one thing, MLS is an almost completely theoretical topic, while DRM tends to be an extremely "applied" topic, with little or no concern for any possible theoretical underpinnings. In this sense, the topics are at opposite extremes.

15. Probably. This is not quite as straightforward as the other way around, but with compartments, you could enforce just about any restrictions on the use of data imaginable.

16. Yes. Each document would carry a classification and only authenticated users with the appropriate clearances could obtain access.

17. a. Attestation is the authentication of "things", such as software and devices.

    b. The purpose of an NCA is to enable communication between the untrusted LHS and the trusted RHS. NCAs also make NGSCB extensible.

18. a. Anderson's main beef seems to be that the owner of the data does not have complete control over its use.

    b. Thomborson views NGSCB as a "security guard" that might be spying on you.

    c. I find Thomborson's arguments to be much more persuasive and sensible. But, without a doubt, Anderson was much more influential.

19. While it's possible to reverse engineer hardware, it's far more difficult than reverse engineering software—expensive equipment is often needed to attack hardware. While SRE attacks are open to any 16 year-old Norwegian living in his parent's basement, hardware-breaking attacks generally require a higher level of investment, making them more the domain of government and industry.

20. Game systems such as the Game Cube, Playstation and Xbox are all good at protecting their secrets—at least they are good enough to make them commercially successful. Another example comes from the European cable TV system discussed in Ross Anderson's *Security Engineering* book. The set-top box that is used to decrypt the signal is intended to be a closed system. This has proven somewhat less successful at protecting its secrets than the gaming systems.

21. The PC is the archetype of an open system. Of course, PCs are not really able to defend themselves against attacks. The Android cell phone is another example of a reasonably open platform, which is somewhat better at protecting itself from attack.

22. a. Open — pick your favorite malware attack.

    b. Mostly closed — there have been a few malware attacks, but nothing too major so far. Attacks on the iPhone which enable the phone to be used with any carrier have occurred.

    c. Closed — attacks don't seem to be much of an issue.

    d. Closed — People have broken the security on the Xbox as a way to obtain cheap computing hardware.

    e. Closed — attacks don't seem to have been widely publicized (yet).

23. Richard Stallman's article "Can you trust your computer"
`www.gnu.org/philosophy/can-you-trust.html`
contains some insightful criticisms of NGSCB.

24. An excellent balanced discussion of trusted computing can be found in Peter Ericson's dissertation: `pericson.com/writings/tcpa-tcg_ngscb/`
While this is not strictly a pro-NGSCB article, it does provide a fairly positive picture. For example, the concluding remark is, "the author is looking forward to a future with trusted computing with great interest and optimism." This is virtually the opposite of the dark vision of "treacherous computing" that many NGSCB opponents promote.

25. Anderson's primary concern seems to be that users will lose some control over their computers. He paints a very dark picture of trusted computing, particularly if left in the hands of Microsoft.

26. It would be difficult for a Kerberos-like system to scale to the necessary size. See Kaufman, Perlman and Speciner, *Network Security*, for a discussion of how Kerberos KDCs communicate with each other. This would be necessary on a huge scale if a similar system were used in NGSCB.

27. The NCA handles the integrity check and the NCA has already passed the attestation check. Therefore, we can trust the NCA to do the sealed storage integrity check with a symmetric key.

28. If an HMAC were used, then a symmetric key would have to be known to all participants. If one user were able to break the key from the software, the entire system would be compromised. By using public keys, the signer only needs to protect its own private key.

29. a. Process isolation prevents malicious software from attacking NGSCB by accessing the memory where NGSCB lives.

    b. Sealed storage protects a key that can be used for attestation, which is critical to detect software-based attacks.

    c. The secure path prevents keystroke logging and similar attacks.

    d. Attestation is used to detect unauthorized software and devices.

30. Attestation prevents unauthorized software from running on a system. Attestation can be used to detect malware and to prevent many software-based attacks. Attestation is also needed to be certain that devices are authenticated and thereby prevent low-level attacks.

31. a. You could have one process spy on another, and possibly recover keys, etc.

    b. The keys could not be stored securely—in effect, you'd be back in the realm of software-based protection.

    c. You could snoop info as it's sent to and from devices. This would be a serious issue in, say, DRM.

    d. If you could break attestation, then the whole thing collapses, since you'd be able to get untrusted code into the system.

32. The reference is to set-top boxes that are sometimes used to decode cable TV signals. These boxes are provided by cable companies as a way to ensure that they get paid by users who view their programs. This is an excellent analogy to the TCG/NGSCB approach, which embeds special hardware into a system that the user cannot directly access. Music or video could be packaged so that it can only be played if you process it using this special hardware.

33. a. Use Google, instant message to other students in the class (or anybody else), email questions to a friend, etc.

    b. Have a particular NCA that must be authorized on students' computers. This NCA could monitor for email, Web activity, etc. The NCA could also require users to sign their results.

    c. This would be much more difficult. You would need to find a loophole in the NCA-enforced protection, or you could try to attack the NGSCB architecture. If you could break this, you'd deserve an A+ in any security class.

34. a. The paper "Native Client: A Sandbox for Portable, Untrusted x86 Native Code" by Yee, et al, may be the best place to look for such info.

    b. This is described in detail in the paper cited in part a.

    c. Good question. The paper cited in part a says that Xax is the most similar to NaCl. According to the same source, CFI consumes far more resources.