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

$$k_0k_1k_2\ldots k_n = 1000001101110000011110000001100 1$$

   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:

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

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 $C_i = P_i \oplus h(h(K))$. 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, recomputing 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.

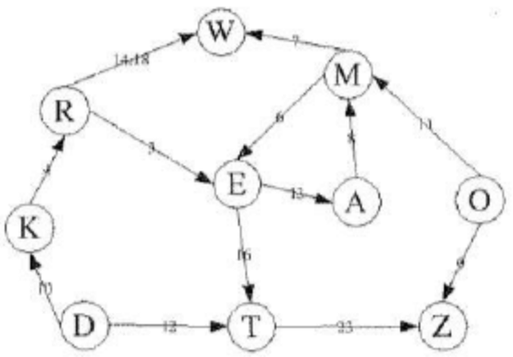
Chapter 6

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.

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

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 $T(C_j) - \tilde{t}_{0\ldots 2}$.

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.