CHAPTER 2 CRYPTOGRAPHIC TOOLS

ANSWERS TO QUESTIONS

2.1 Plaintext, encryption algorithm, secret key, ciphertext, decryption algorithm.

2.2 One secret key.

2.3 (1) a strong encryption algorithm; (2) Sender and receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure.

2.4 Message encryption, message authentication code, hash function.

2.5 An authenticator that is a cryptographic function of both the data to be authenticated and a secret key.

2.6 **(a)** A hash code is computed from the source message, encrypted using symmetric encryption and a secret key, and appended to the message. At the receiver, the same hash code is computed. The incoming code is decrypted using the same key and compared with the computed hash code. **(b)** This is the same procedure as in (a) except that public-key encryption is used; the sender encrypts the hash code with the sender's private key, and the receiver decrypts the hash code with the sender's public key. **(c)** A secret value is appended to a message and then a hash code is calculated using the message plus secret value as input. Then the message (without the secret value) and the hash code are transmitted. The receiver appends the same secret value to the message and computes the hash value over the message plus secret value. This is then compared to the received hash code.

2.7
1. H can be applied to a block of data of any size.
2. H produces a fixed-length output.
3. H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical.
4. For any given value h, it is computationally infeasible to find x such that H(x) = h.
5. For any given block x, it is computationally infeasible to find $y \neq x$ with H(y) = H(x).

6. It is computationally infeasible to find any pair (x, y) such that H(x) = H(y).

2.8 **Plaintext:** This is the readable message or data that is fed into the algorithm as input. **Encryption algorithm:** The encryption algorithm performs various transformations on the plaintext. **Public and private keys:** This is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption. The exact transformations performed by the encryption algorithm depend on the public or private key that is provided as input. **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different ciphertexts. **Decryption algorithm:** This algorithm accepts the ciphertext and the matching key and produces the original plaintext.

2.9 **Encryption/decryption:** The sender encrypts a message with the recipient's public key. **Digital signature:** The sender "signs" a message with its private key. Signing is achieved by a cryptographic algorithm applied to the message or to a small block of data that is a function of the message. **Key exchange:** Two sides cooperate to exchange a session key. Several different approaches are possible, involving the private key(s) of one or both parties.

2.10 The key used in conventional encryption is typically referred to as a **secret key**. The two keys used for public-key encryption are referred to as the **public key** and the **private key**.

2.11 A **digital signature** is an authentication mechanism that enables the creator of a message to attach a code that acts as a signature. The signature is formed by taking the hash of the message and encrypting the message with the creator's private key. The signature guarantees the source and integrity of the message.

2.12 A **pubic-key certificate** consists of a public key plus a User ID of the key owner, with the whole block signed by a trusted third party. Typically, the third party is a certificate authority (CA) that is trusted by the user community, such as a government agency or a financial institution.

2.13 Several different approaches are possible, involving the private key(s) of one or both parties. One approach is Diffie-Hellman key exchange. Another approach is for the sender to encrypt a secret key with the recipient's public key.

ANSWERS TO PROBLEMS

2.1 Yes. The eavesdropper is left with two strings, one sent in each direction, and their XOR is the secret key.

2.2 a.

2	8	10	7	9	6	3	1	4	5
C	R	Y	P	T	O	G	A	H	I
B	E	A	T	T	H	E	T	H	I
R	D	P	I	L	L	A	R	F	R
O	M	T	H	E	L	E	F	T	O
U	T	S	I	D	E	T	H	E	L
Y	C	E	U	M	T	H	E	A	T
R	E	T	O	N	I	G	H	T	A
T	S	E	V	E	N	I	F	Y	O
U	A	R	E	D	I	S	T	R	U
S	T	F	U	L	B	R	I	N	G
T	W	O	F	R	I	E	N	D	S

4	2	8	10	5	6	3	7	1	9
N	E	T	W	O	R	K	S	C	U
T	R	F	H	E	H	F	T	I	N
B	R	O	U	Y	R	T	U	S	T
E	A	E	T	H	G	I	S	R	E
H	F	T	E	A	T	Y	R	N	D
I	R	O	L	T	A	O	U	G	S
H	L	L	E	T	I	N	I	B	I
T	I	H	I	U	O	V	E	U	F
E	D	M	T	C	E	S	A	T	W
T	L	E	D	M	N	E	D	L	R
A	P	T	S	E	T	E	R	F	O

```
ISRNG  BUTLF  RRAFR  LIDLP  FTIYO  NVSEE  TBEHI  HTETA
EYHAT  TUCME  HRGTA  IOENT  TUSRU  IEADR  FOETO  LHMET
NTEDS  IFWRO  HUTEL  EITDS
```

b. The two matrices are used in reverse order. First, the ciphertext is laid out in columns in the second matrix, taking into account the order dictated by the second memory word. Then, the contents of the second matrix are read left to right, top to bottom and laid out in columns in the first matrix, taking into account the order dictated by the first memory word. The plaintext is then read left to right, top to bottom.

c. Although this is a weak method, it may have use with time-sensitive information and an adversary without immediate access to good cryptanalysis t(e.g., tactical use). Plus it doesn't require anything more than paper and pencil, and can be easily remembered.

2.3 a. Let -X be the additive inverse of X. That is -X $\boxed{+}$ X = 0. Then:

$$P = (C \boxed{+} -K_1) \oplus K_0$$

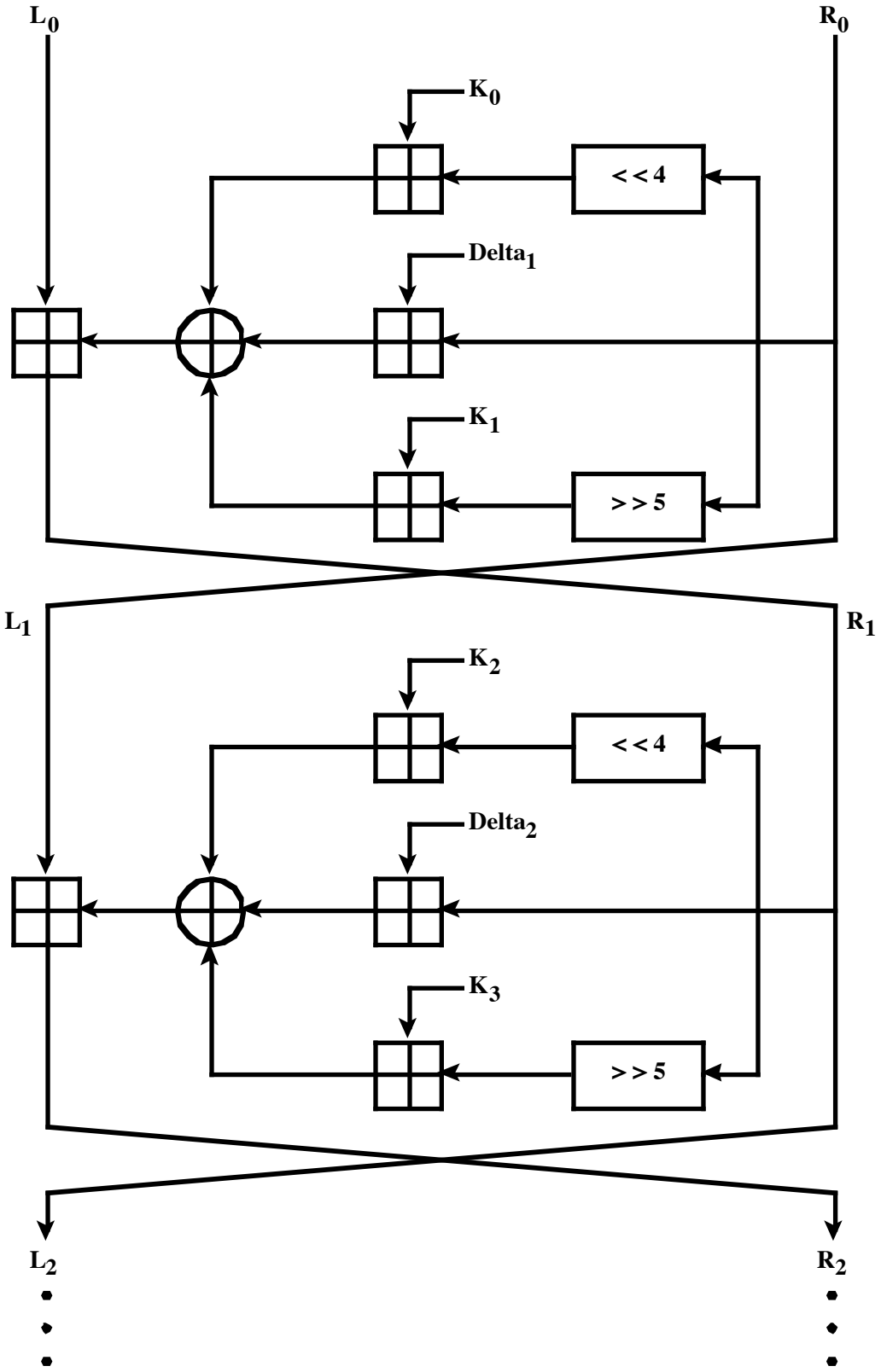
b. First, calculate –C'. Then –C' = (P' $\oplus$ K_0) $\boxed{+}$ (– K_1). We then have:

C $\boxed{+}$ –C' = (P $\oplus$ K_0) $\boxed{+}$ (P' $\oplus$ K_0)

However, the operations $\boxed{+}$ and $\oplus$ are not associative or distributive with one another, so it is not possible to solve this equation for K_0.

2.4 a. The constants ensure that encryption/decryption in each round is different.

b. First two rounds:



c. First, let's define the encryption process:

$L_2 = L_0 \boxed{+} [(R_0 << 4) \boxed{+} K_0] \oplus [R_0 \boxed{+} \delta_1] \oplus [(R_0 >> 5) \boxed{+} K_1]$

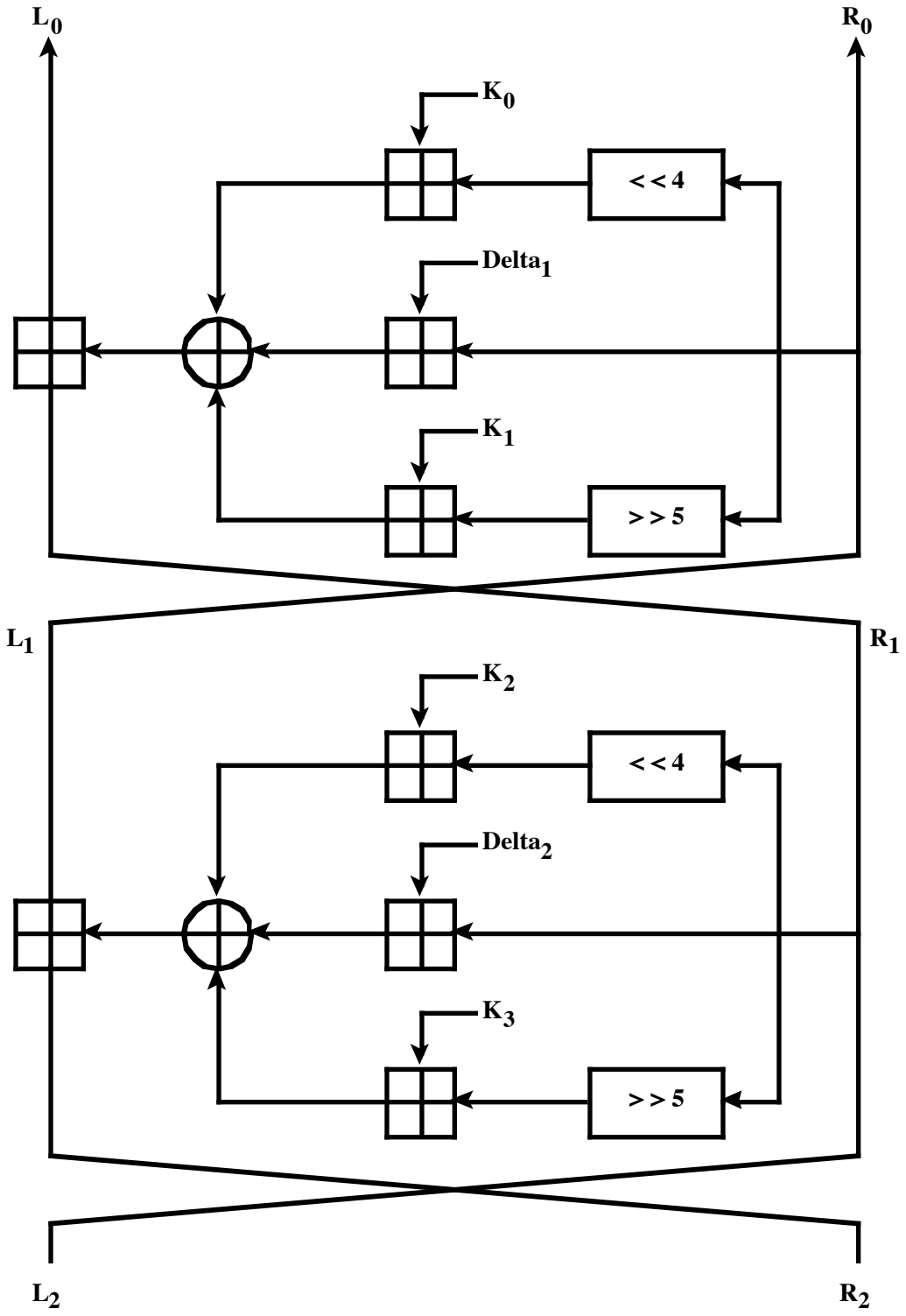
$R_2 = R_0 \boxed{+} [(L_2 << 4) \boxed{+} K_2] \oplus [L_2 \boxed{+} \delta_2] \oplus [(L_2 >> 5) \boxed{+} K_3]$

Now the decryption process. The input is the ciphertext (L_2, R_2), and the output is the plaintext (L_0, R_0). Decryption is essentially the same as encryption, with the subkeys and delta values applied in reverse order. Also note that it is not necessary to use subtraction because there is an even number of additions in each equation.

$R_0 = R_2 \boxed{+} [(L_2 << 4) \boxed{+} K_2] \oplus [L_2 \boxed{+} \delta_2] \oplus [(L_2 >> 5) \boxed{+} K_3]$

$L_0 = L_2 \boxed{+} [(R_0 << 4) \boxed{+} K_0] \oplus [R_0 \boxed{+} \delta_1] \oplus [(R_0 >> 5) \boxed{+} K_1]$

d.



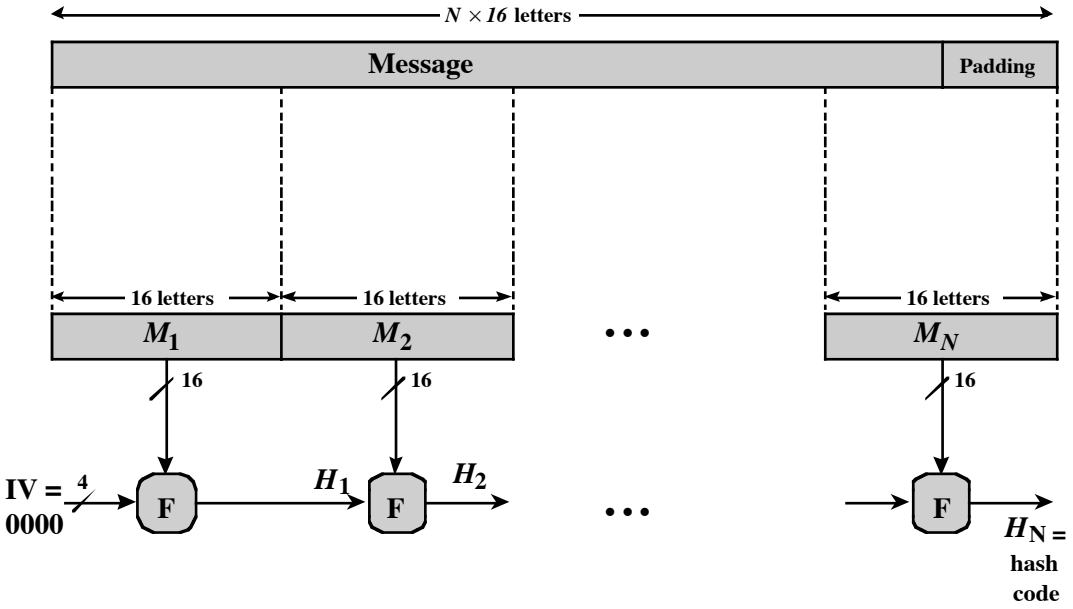
2.5 **a.** Will be detected with both (i) DS and (ii) MAC.
b. Won't be detected by either (Remark: use timestamps).
c. (i) DS: Bob simply has to verify the message with the public key from both. Obviously, only Alice's public key results in a successful verification.

(ii) MAC: Bob has to challenge both, Oscar and Bob, to reveal their secret key to him (which he knows anyway). Only Bob can do that.

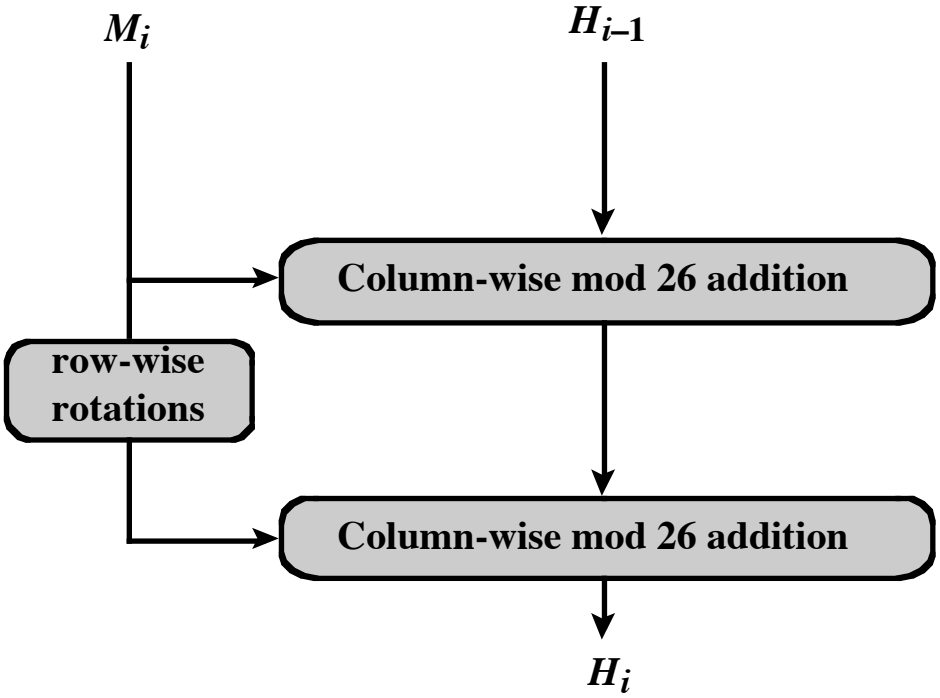
d. (i) DS: Alice has to force Bob to prove his claim by sending her a copy of the message in question with the signature. Then Alice can show that message and signature can be verified with Bob's public key) Bob must have generated the message.
(ii) MAC: No, Bob can claim that Alice generated this message.

2.6 The statement is false. Such a function cannot be one-to-one because the number of inputs to the function is of arbitrary, but the number of unique outputs is 2^n. Thus, there are multiple inputs that map into the same output.

2.7 a. Overall structure:



Compression function F:



b. BFQG

c. Simple algebra is all you need to generate a result:

AYHGDAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAA

2.8

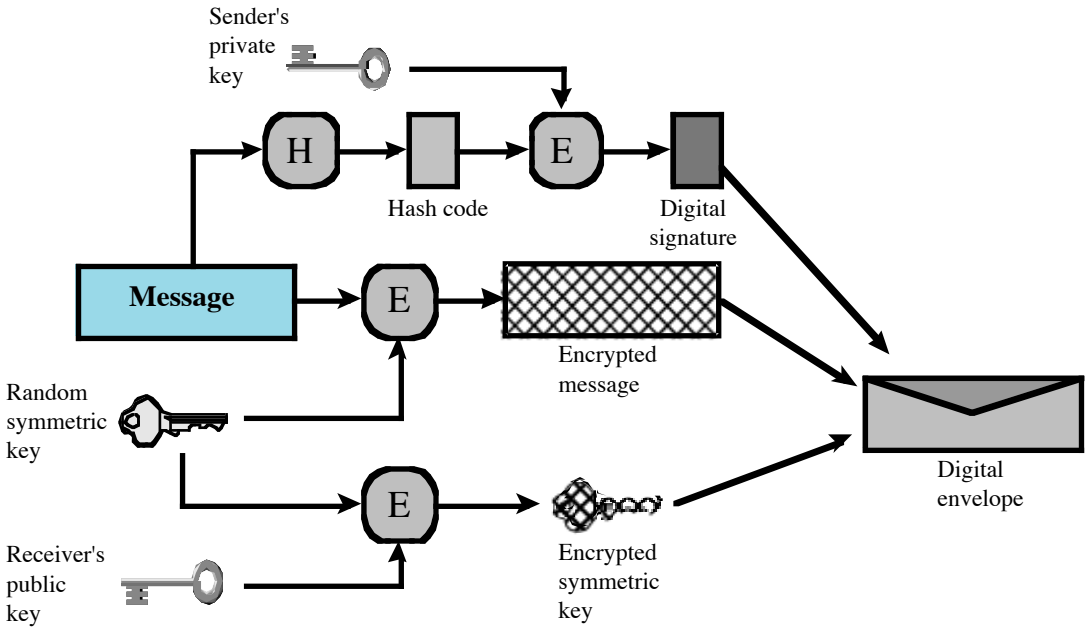
a. $M3 =$

5	2	1	4	5
1	4	3	2	2
3	1	2	5	3
4	3	4	1	4
2	5	5	3	1

b. Assume a plaintext message p is to be encrypted by Alice and sent to Bob. Bob makes use of M1 and M3, and Alice makes use of M2. Bob chooses a random number, k, as his private key, and maps k by M1 to get x, which he sends as his public key to Alice. Alice uses x to encrypt p with M2 to get z, the ciphertext, which she sends to Bob. Bob uses k to decrypt z by means of M3, yielding the plaintext message p.

c. If the numbers are large enough, and M1 and M2 are sufficiently random to make it impractical to work backwards, p cannot be found without knowing k.

2.9 We show the creation of a digital envelope:



PART 2 PRACTICAL SECURITY ASSESSMENTS

Examining the current infrastructure and practices of an existing organization is one of the best ways of developing skills in assessing its security posture. Students, working either individually or in small groups, select a suitable small- to medium-sized organization. They then interview some key personnel in that organization in order to conduct a suitable selection of security risk assessment and review tasks as it relates to the organization's IT infrastructure and practices. As a result, they can then recommend suitable changes, which can improve the organization's IT security. These activities help students develop an appreciation of current security practices, and the skills needed to review these and recommend changes.

The document **PracticalAssessments** included at the IRC provides detailed guidance.

A PROFESSIONAL PRACTICE COMPONENT IN WRITING:

A SIMPLE WAY TO ENHANCE AN EXISTING COURSE*

Karen Anewalt
Department of Computer Science
Mary Washington College
Fredericksburg, VA 22401
anewalt@mwc.edu

ABSTRACT

The annual survey conducted in 2001 by the National Association of Colleges and Employers showed that employers rank good communication skills (both written and oral) as the most desirable quality in applicants seeking employment [7]. The recently published 2001 ACM/IEEE Computing Curriculum guidelines respond to industry demands by stressing the importance of incorporating "professional practice" components, including coursework focusing on written communication skills, in the undergraduate curriculum [1]. Despite evidence that communication skills are highly valued and professional recommendations to include writing in the computer science curriculum, many computer science faculty members are reluctant to add written components to their courses. This paper describes simple, practical ways of incorporating writing into existing computer science courses. Examples of writing assignments used in a sophomore-level data structures course are provided.

INTRODUCTION

The importance of developing good communication skills as part of the undergraduate curriculum has received recent attention in the ACM/IEEE Computing Curriculum 2001 guidelines [1]. The guidelines stress the importance of developing professional skills, including effective written communication skills, as part of the undergraduate curriculum. Motivation to enhance student communications skills also comes from the corporate world. In the annual

---

survey conducted in 2001 by the National Association of Colleges and Employers, employers ranked good communication skills (both written and oral) as the most desirable quality in applicants seeking employment [7]. It is not surprising that employers and professional organizations stress the importance of good communication skills. Regardless of post-graduate choices, students will undoubtedly use their written communication skills in their future lives.

In spite of the obvious need for students to develop writing skills, many faculty members are reluctant to add writing assignments to their existing computer science courses. A common attitude is that instruction in writing is better left to humanities and social science instructors. This opinion is not a wise one. In [8], Pensente states that "isolated attempts to teach writing hinder the transfer of learning to other courses and, eventually, to the work place". The Computing Curriculum states that educators have a responsibility to provide exposure to professional practice and ease the transition from academia to the business world [1]. In addition to providing exposure to professional practice, teaching writing in a computer science context can have many benefits that would not be realized if students were only to receive writing practice in non-computer science courses.

- **When writing is taught in a computer science setting, the assignments and feedback can be relevant to the computer science discipline.**

  Writing experiences in computer science courses can provide students with valuable practice communicating with assorted audiences in various formats that will be required in the corporate or postgraduate world. Computer scientists do not operate in a vacuum. In the modern world, computer scientists and software engineers need to be able to effectively communicate, both orally and on paper. In industry, programmers are expected to communicate with groups of other programmers in order to design and complete software projects. Programmers may also be required to communicate with non-technical individuals about project goals and requirements. Thus it is important to teach computer science students how to communicate with a range of audiences. Because computer scientists are much more familiar than non-computer scientists with the types of documents typically required of computer scientists, it is appropriate that instruction in creating such written works come from computer scientists.

- **Parallels can be drawn between the software design process and the writing process.**

  The similarity between the writing process and the software design process is something that few traditional English literature courses recognize, but can make computer science students feel more connected to the writing process. The software design process is iterative. Most programmers do not receive a project assignment and immediately sit down to write all of the code perfectly. The software design process should involve thought and planning prior to the implementation phase. Following the implementation phase, software design involves a cycle of testing and modifying the code. Finally,

software enters the maintenance phase where it is modified to meet changing needs and goals. The writing process follows a very similar sequence of phases [9]. When a written assignment is given, the writer should think about the goals of the final document and organize his or her thoughts about the assignment. After a clear plan is constructed, the process of selecting language to communicate the ideas begins and the document is created. There is a stage of reading and modifying the document to ensure that the final product has the desired qualities. Finally, many computer science-related documents, like user's manuals, enter a maintenance stage in which they are modified as the software that they refer to is modified. Even beginning computer science students are familiar with the software design process and can see the parallels. By presenting this type of analogy to students, the relationship between the writing process and the software design process is made clear, and computer science students will feel more familiar with the writing process.

- **Writing promotes active learning and can increase student understanding of course material.**

  In my experience, students often believe that they understand complex concepts presented in lecture without any additional exposure or hands-on practice with the concepts. However, when these students are asked to write a detailed explanation of the concept, they realize that their understanding is far from complete. Written assignments are one way of encouraging students to interact with course content on a personal level, which can in turn increase student understanding [2]. As faculty, we continually engage in active learning; we read textbooks, compare ideas from different sources, prepare lecture notes and create assignments. Students are often deprived of this type of active learning experience [5]. Using targeted written assignments as part of a course encourages students to participate in the same type of rich learning experience that we engage in as professionals. Writing activities can teach students to pose questions, develop hypotheses, collect and analyze data, and organize arguments. These critical thinking skills are important to develop in order to be successful in the computer science field.

INCORPORATING WRITING INTO AN EXISTING COURSE

Committing to including writing assignments in a course does not necessarily imply that the course format must be restructured. Assignments that could easily be modified to include a written component are already used in most computer science courses. Typical computer science courses do use programming projects and/or homework problem sets to reinforce lecture topics. Written components can be easily added to programming projects by requiring a short summary of the software's purpose, a design summary describing design decisions made by the student, or a document describing the efficiency of an algorithm being used in the software. This type of assignment can be particularly useful because it provides practice in written communication and does so in a professional context.

Another method of incorporating additional writing into an existing course is by replacing traditional problem sets with writing assignments that achieve the same learning goals. Alternately, a traditional homework assignment can be used along with a question to be answered in paragraph form. The focus of the assignment remains the same, but by adding the additional writing component to the assignment, students receive additional practice and exposure to writing. Writing can also be used to reinforce topics covered in lecture or reading assignments. For example, when reading is assigned, the instructor can provide a question to which students should write a short one-page response. The question will focus students on the key points of the reading, engage the students in actively thinking about issues discussed in the reading, and forces the students to be accountable for the reading by creating a tangible deliverable. This type of writing assignment can actually reduce the amount of time that is spent covering the basics of a concept during lecture, because the students will have already absorbed the basics by completing the reading and writing assignment.

The key to enhancing a course through the use of writing assignments is to evaluate the learning objectives of the course and the existing assignments. Assignments can then be added or modified to increase the amount of writing required while maintaining the original learning goal. In this way, writing assignments can be designed to enhance course content rather than detract from course content.

WRITING ASSIGNMENT SUGGESTIONS

I was first exposed to teaching writing within a computer science context after being asked to teach a data structures course designated to fulfill part of the writing-across-the-curriculum requirement at my institution. The primary computer science goals of the course are to teach data structures and to introduce students to Java as a second programming language. The course is the third semester programming course for computer science majors and covers heaps, trees, and graphs while emphasizing efficiency analysis and software reusability. In the course, students learn to use Java as their second programming language, having already acquired skills in C++. Like most computer science courses, there are many opportunities to introduce writing assignments into this type of course without sacrificing course content.

I designed the writing component of the course with two main goals. The first goal is to provide students with significant practice in writing and communicating ideas. The second goal is to expose students to a variety of types of technical writing appropriate to the computer science discipline. These goals were achieved using a variety of writing assignments. I have found that using a variety of types of writing assignments benefits the students. The variety increases the chance that a student will be successful in at least one paper. Some students are better at writing for peers; others excel in creating documents for non-technical audiences. Some students enjoy creative papers; others feel more confident when the assignment is very structured. When a broad spectrum of writing styles is explored, students are likely to find something with which they feel comfortable. In addition, the students benefit by familiarizing themselves with the goals and challenges of a larger variety of document styles.

My course uses three or four formal writing assignments (3-5 page papers) during the semester. Examples of some writing assignments that I have used in this course are included in the Appendix. Other types of formal writing assignments that I've successfully used in the course include software design documents, descriptive essays about course topics and short research papers.

As discussed earlier, adding writing to a course does not necessarily mean that formal papers must be assigned. Shorter written assignments can be equally effective in providing additional practice in writing and focusing students' attention on key concepts discussed in a course. In addition to the formal papers described above, I assign short, informal writing assignments to encourage active learning. I typically assign an informal writing assignment once a week and grade the papers on a credit/no credit basis. Examples of informal writing assignments that I have used include assignments to summarize the reading assignment in a few paragraphs (depending on the length of the reading), answer a focus question (provided at the time the assignment is given) related to the reading, to find Web resources that contradict one another about a definition or concept described in class and argue that one definition is correct, to make connections between lecture topics discussed at different times in the course, and to compare and contrast two data structures. These informal writing assignments are designed to stimulate independent thought about course topics and to promote active learning. I have found that the informal assignments were very useful to me as an instructor because the student responses occasionally indicated weaknesses in student understanding and could be used to generate discussion in future class meetings.

STUDENT RESPONSE TO WRITING IN COMPUTER SCIENCE

Each semester that I have offered the data structures course with a strong writing component, several students have approached me at the end of the semester and mentioned that the writing assignments were very beneficial and helped them to draw connections about course material that they would have otherwise missed. They have said that by writing about computer science topics, they began to recognize their own weaknesses in understanding and were better able to formulate questions about the course topics. After having completed a writing assignment, they felt that they had a more complete understanding of the related course material.

In the most recent offering of the data structures course, I requested that students complete a short questionnaire about the writing portion of the course. I prepared a list of statements and the students were allowed to anonymously respond that they agreed or disagreed with each statement. Because the enrollment in the course was small, statistical analysis of the data is not meaningful, however I felt the information collected would be useful to get a feel for the student reactions to the effectiveness of the writing assignments used in the course. The responses show that most of the students felt that the writing assignments were beneficial both because they exposed them to the types of writing that might be required of them after graduation and because they felt that the emphasis on writing enhanced their written communication skills.

The responses collected from the students enrolled in the course along with the questions asked are indicated below:

		Agree	Disagree
1)	In general, I feel that my writing has improved during. the semester.	5	2
2)	After completing a computer science course with writing assignments, I feel that I have a clearer picture of how writing is used in the computer science field.	6	1
3)	I feel that the documents assigned as formal writing assignments in class were representative of the types of documents that I may be required to write as a professional in the computer science field.	6	1
4)	I think that I will benefit from being exposed to these types of documents even if I will not be required to write any documents in my future career.	7	0
5)	In general, I felt that the writing in the course helped me to develop understanding of course material.	7	0
6)	I felt that I would have learned as much in the course if it had not included writing assignments.	2	5

CONCLUSION

The ACM/IEEE Computing Curriculum guidelines emphasize the importance of good communication skills and encouraged institutions to include communication skill development in their curriculum. Including writing assignments in a computer science context exposes students to the types of writing that will be expected of them in the future and can also encourage students to be more personally involved in learning course material. I have found that including writing assignments in a computer science course can improve student communication skills, understanding of the professional practice and can enhance student understanding of course topics.

REFERENCES

[1] Computing Curricula 2001: Report of the ACM/IEEE-CS joint curriculum tast force, Association for Computing Machinery, Dec 2001, http://www.acm.org/sigs/sigcse/cc2001

[2] Bean, John C. Engaging Ideas. Jossey-Bass, 1996.

[3] Gersting, J. and Young, F. Shall We Write? SIGCSE Bulletin, vol. 33, no 2, June 2001, pp. 18-19.

[4] Kay, David G. Computer Scientists Can Teach Writing: An Upper Division Course for Computer Science. In Proceedings of the 29th SIGCSE Technical Symposium, 1998, pp. 52-54.

[5] McConnell, J. Active learning and its use in Computer Science. In Proceedings of the Conference on Integrating Technology into Computer Science, 1996, pp. 52-54.

[6] Michael, M. Fostering and Assessing Communication Skills in the Computer Science Context. In Proceedings of the 31st SIGCSE Technical Symposium, 2000, pp. 119-123.

[7] National Association of Colleges and Employers. Job outlook '01 (online version). http://www.jobweb.com/joboutlook/

[8] Pesante, Linda H. Integrating Writing Into Computer Science Courses. In Proceedings of the 22nd SIGCSE Technical Symposium, 1991, pp. 205-209.

[9] Zaidman, M. Integrating Writing Into the Computer Science Curriculum, In Proceedings of the 5th Annual Easetern Small College Computing Conference, 1989, pp. 113-116.

APPENDIX

SAMPLE 1:

This assignment is given on the first day of class, immediately after discussing the writing goals and guidelines for the course. The assignment is designed with both course content goals and writing skill goals in mind.

Course Goals:

- One purpose of the course is to introduce Java as the students' second programming language. This assignment motivates students to begin to make the transition immediately.

- The IDE referenced in the paper is most of the students' first exposure to an IDE. All of their prior course work is done in a UNIX environment. The paper encourages students to explore various aspects of the IDE and become familiar with this alternate tool for program development.

Writing Goals:

- Students are introduced to the concept of "audience" for the paper. In this assignment students develop skills in communicating technical ideas to a non-technical audience.
- In class, students discuss how the format for this type of paper may differ from traditional papers that they have written in high school or in earlier courses. Methods of appropriately differentiating commands and code from standard text within sentences is emphasized.

Because this is the first writing assignment that the students have in the course, I provide them with many leading questions that they should consider while writing their paper. In future papers, the students do not require as much guidance.

CPSC 321: Data Structures -- Writing Assignment 1

Assignment Goals:

- Learn to verbalize technical concepts for a less-technical audience
- Familiarize yourself with Forte for Java and the Java program lifecycle

Part I

Task: In order to familiarize yourself with the Forte for Java Integrated Development Environment (IDE), create two programs that display a simple message (such as "Hello, world!"). You will create an applet as well as an application program with this functionality. (The message printed by the applet should be displayed in the applet viewer; the application should print the message to standard output.) Very basic instructions on how to create, compile, and run Java programs within the Forte IDE are provided at the end of this document.

Deliverables: Turn in printouts of your source files (staple to the end of your paper for Part II).

Part II

Task: You are a member of the programming team responsible for developing Forte for Java at Sun Microsystems. Your team has recently completed the development and testing of the Forte software package. At the weekly team meeting, your supervisor assigns you the task of writing a user's guide for the IDE. She tells you that the guide must explain how to create a Java source file, compile it, and execute it using Forte for Java. This guide will be distributed along with the software when it is sold commercially. She also tells you that your guide should focus on writing one type of program. In other words, your guide should either explain to the reader the steps necessary to create an applet OR to create an application (BUT NOT BOTH). Someone else on the team will be responsible for the instructions for developing the other type of program.

Because the user's guide will be included in the commercial release package, the guide should be understandable to a novice computer scientist (think of someone taking computer science for the first time) who has never used Forte, but who is familiar with the concept of programming (in C++ or Ada), windows, menus, point and click, etc. As you write your user's guide, anticipate questions and problems that the reader might have as they use the IDE and provide the reader with guidance and explanations in your text.

The paper should contain a separate section for each of the following:

- An introduction section introducing the Forte application. (What is the purpose of this guide? What is Forte used for? Why will Forte make it easy/easier for the reader to develop Java code?)
- Directions on how to launch the Forte application. Assume that the reader already has the Forte program installed on their computer. (What does the environment look like when it comes up? What should the reader expect?)
- Separate sections instructing the user in each of the following tasks: opening a new project, creating a source (.java) file, saving a project, building a project, finding and correcting syntax errors, and executing a program. Use an example program (like the "Hello, world!" program that you wrote in Part I) to walk the user through the steps required to complete these tasks. Including an example within the text of a user's guide makes the guide more interesting and understandable to the reader.
- Because the guide should enlighten the reader about the nature of syntax errors and how a programmer identifies and corrects such errors, you should include at least one intentional syntax error in your example code. Explain to the user the nature of this error and how to correct the error. (Where do the compilation errors appear in the development environment? What does the error message given to the user mean? Is there a debugger that might help the user to correct errors? How is the debugger used?)
- The guide should explain how to identify correct output for the program. (Where is the output displayed? How should it appear?)
- A conclusion summarizing the guide and providing other appropriate information (such as where the reader can find more information about the topics covered).

The guide should serve several purposes for the reader. It should familiarize the reader with the product, be useful as a hands-on exercise that illustrates the Java program development cycle from file creation to execution, and it should serve as a reference for the preparation of future programs.

Deliverables: The paper should be 3-4 typed double-spaced pages. Standard margins (1 inch at the top, 1 inch at the bottom, 1 ¼ inches on either side) and 12-point Times Roman font should be used. Any example Java code provided with in the text of the guide should be obviously delimited from the rest of the text and spaced appropriately (as it would appear in the editor, not necessarily double spaced).

SAMPLE 2:

As this second example assignment is given, students have just "reviewed" object-oriented concepts. Because the department is in a transitional period, many of the students in the course had not had significant exposure to object-oriented design principles. In a previous homework assignment, I had asked the students to do some research on the Web to find definitions for various object-oriented terms including "encapsulation" and "information hiding". Several

students found resources that claimed that these terms have identical meanings. After reviewing the informal writing assignment, I corrected the misconception during lecture and created this formal writing assignment to reinforce the concepts.

Course Goals:

- This assignment encourages students to review object-oriented terminology.
- This assignment emphasizes that information hiding and encapsulation are in fact distinct concepts.

Writing Goals:

- The assignment emphasizes that all information found on the Web (or even in a textbook) is not factually correct and demonstrates the need to verify sources.
- The importance of appropriate documentation is discussed in class and students are expected to use documentation in the paper.
- In this assignment, fewer guidelines were given (no leading questions) and the students wrote to an audience of peer computer scientists.

CS 321 – Data Structures -- Writing Assignment #2

Project Goals:

- To explore some ideas of importance in object-oriented design and how the ideas relate to one another
- To apply object-oriented design to a real-world object
- To practice verbalizing technical topics to peer computer scientists

Paper Description:

Object-oriented programming (OOP) differs from procedural programming. The principles of encapsulation and information hiding are cornerstones of object-oriented program design. Many people incorrectly believe that the terms "encapsulation" and "information hiding" refer to the same concept. Read at least three references on the topic of OOP. At least one of your references must be a source other than those given in the "Possible References" list. After you have consulted your sources, write an essay in which you:

- Describe the concepts of a class, an object, encapsulation and information hiding
- Discuss the importance of encapsulation and information hiding to OOP
- Describe the difference between the concept of information hiding and the concept of encapsulation

To demonstrate the usefulness of the object-oriented design philosophy, apply the concepts of a class, an object, information hiding, and encapsulation to an example item. You may choose a radio, microwave oven, or television and describe that real-world object using object-oriented design. Include a discussion of the public interface, data members, and

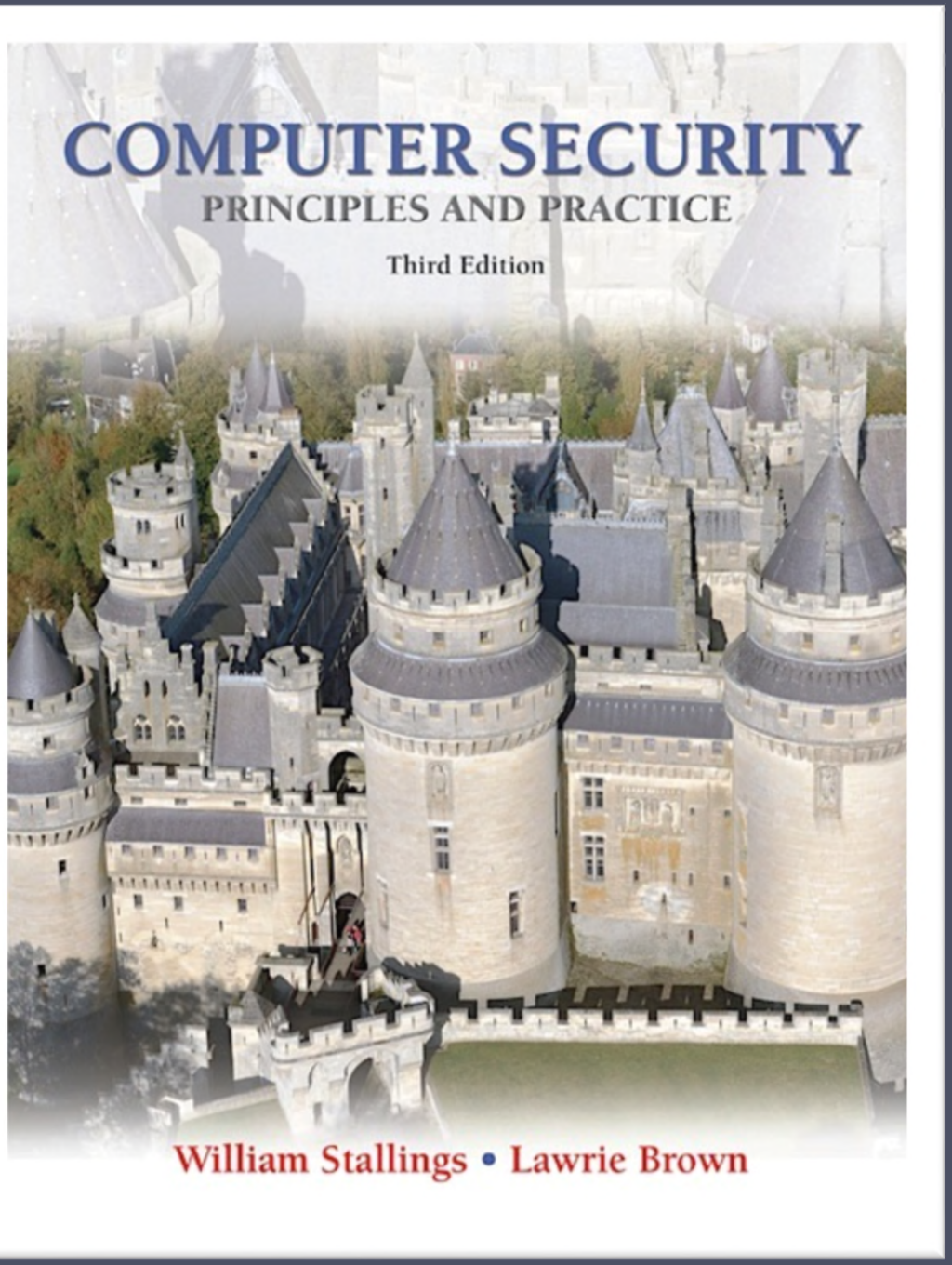
methods of each object. You should NOT write any Java code that would be associated with implementing your chosen object.

Example:
Suppose that we are describing a watch using object-oriented design.
A watch could be an object of the class Clock.
The public interface of the watch could include:
the nob to set the correct time
a button that controls the alarm
a knob to set the alarm time
The data members of the watch object might include:
hours, minutes, seconds (for the current time)
hours, minutes and seconds (for the alarm time)
a flag that indicates whether the alarm is set or not set
Some methods could include:
changeAlarmStatus(), changeTime(), changeAlarmTime()

There is no correct or incorrect design for each item. Your design will depend on the radios, microwaves or TVs that you are familiar with. Unlike the example above, your paper should contain a narrative response and NOT a list of information.

Possible References:
http://www.javaworld.com/javaworld/jw-05-2001/jw-0518-encapsulation.html
http://www.toa.com/pub/abstraction.txt
Data Structures and Algorithms in Java by Goodrich and Tamassia
C++ Plus Data Structures by Dale and Teague

Specifications:
- This paper should be 3-4 typed, double-spaced pages.
- Audience: Other computer science students who are studying the concepts of object-oriented programming (for example someone who has taken computer science 2 or data structures).
- Your paper must have a title, introduction and conclusion!
- Your paper should include a list of references that you consulted and contain appropriate citations.


COMPUTER SECURITY
PRINCIPLES AND PRACTICE
Third Edition
William Stallings • Lawrie Brown

Chapter 2

Cryptographic Tools

Symmetric Encryption

- The universal technique for providing confidentiality for transmitted or stored data
- Also referred to as conventional encryption or single-key encryption
- Two requirements for secure use:
  - Need a strong encryption algorithm
  - Sender and receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure

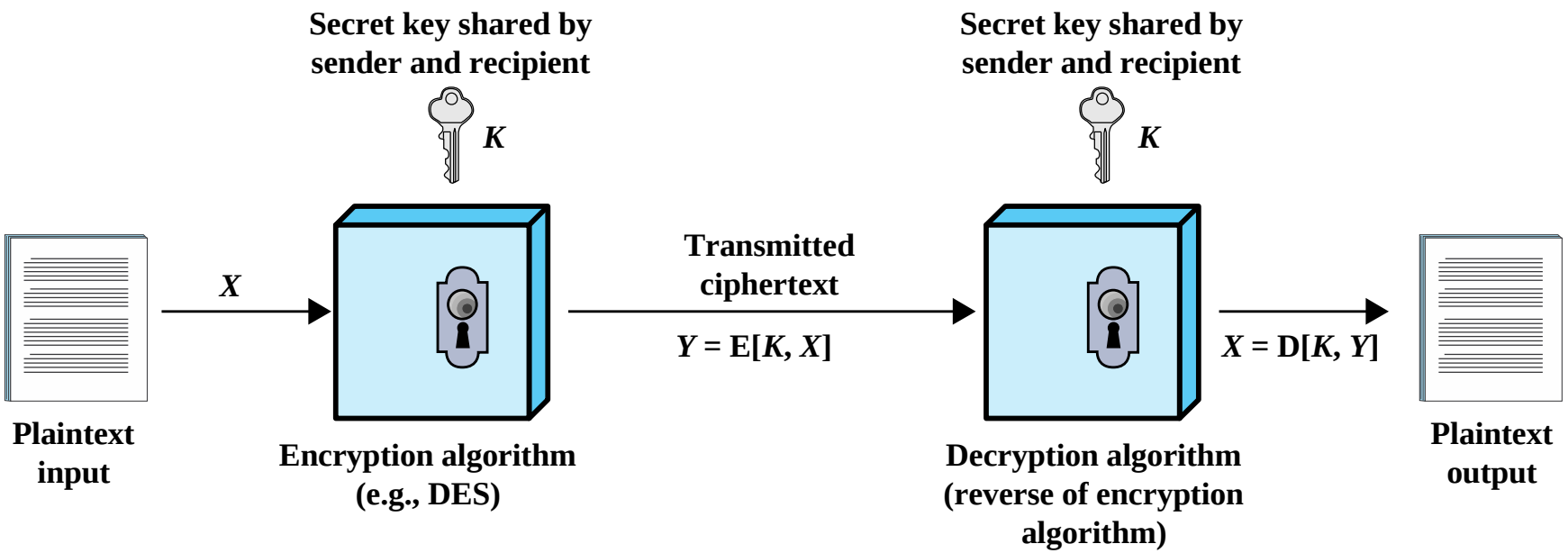


Figure 2.1 Simplified Model of Symmetric Encryption

Attacking Symmetric Encryption

Cryptanalytic Attacks

- Rely on:
  - Nature of the algorithm
  - Some knowledge of the general characteristics of the plaintext
  - Some sample plaintext-ciphertext pairs
- Exploits the characteristics of the algorithm to attempt to deduce a specific plaintext or the key being used
  - If successful all future and past messages encrypted with that key are compromised

Brute-Force Attack

- Try all possible keys on some ciphertext until an intelligible translation into plaintext is obtained
  - On average half of all possible keys must be tried to achieve success

Table 2.1

	DES	Triple DES	AES
Plaintext block size (bits)	64	64	128
Ciphertext block size (bits)	64	64	128
Key size (bits)	56	112 or 168	128, 192, or 256

DES = Data Encryption Standard
AES = Advanced Encryption Standard

Comparison of Three Popular Symmetric Encryption Algorithms

Data Encryption Standard (DES)

- The most widely used encryption scheme
  - FIPS PUB 46
  - Referred to as the Data Encryption Algorithm (DEA)
  - Uses 64 bit plaintext block and 56 bit key to produce a 64 bit ciphertext block

- Strength concerns:
  - Concerns about algorithm
    - DES is the most studied encryption algorithm in existence
  - Use of 56-bit key
    - Electronic Frontier Foundation (EFF) announced in July 1998 that it had broken a DES encryption

Table 2.2

Key size (bits)	Cipher	Number of Alternative Keys	Time Required at 10^9 decryptions/s	Time Required at 10^{13} decryptions/s
56	DES	$2^{56} \approx 7.2 \times 10^{16}$	2^{55} ns = 1.125 years	1 hour
128	AES	$2^{128} \approx 3.4 \times 10^{38}$	2^{127} ns = 5.3×10^{21} years	5.3×10^{17} years
168	Triple DES	$2^{168} \approx 3.7 \times 10^{50}$	2^{167} ns = 5.8×10^{33} years	5.8×10^{29} years
192	AES	$2^{192} \approx 6.3 \times 10^{57}$	2^{191} ns = 9.8×10^{40} years	9.8×10^{36} years
256	AES	$2^{256} \approx 1.2 \times 10^{77}$	2^{255} ns = 1.8×10^{60} years	1.8×10^{56} years

Average Time Required for Exhaustive Key Search

Triple DES (3DES)

- Repeats basic DES algorithm three times using either two or three unique keys
- First standardized for use in financial applications in ANSI standard X9.17 in 1985
- Attractions:
  - 168-bit key length overcomes the vulnerability to brute-force attack of DES
  - Underlying encryption algorithm is the same as in DES
- Drawbacks:
  - Algorithm is sluggish in software
  - Uses a 64-bit block size

Advanced Encryption Standard (AES)

Needed a replacement for 3DES

3DES was not reasonable for long term use

NIST called for proposals for a new AES in 1997

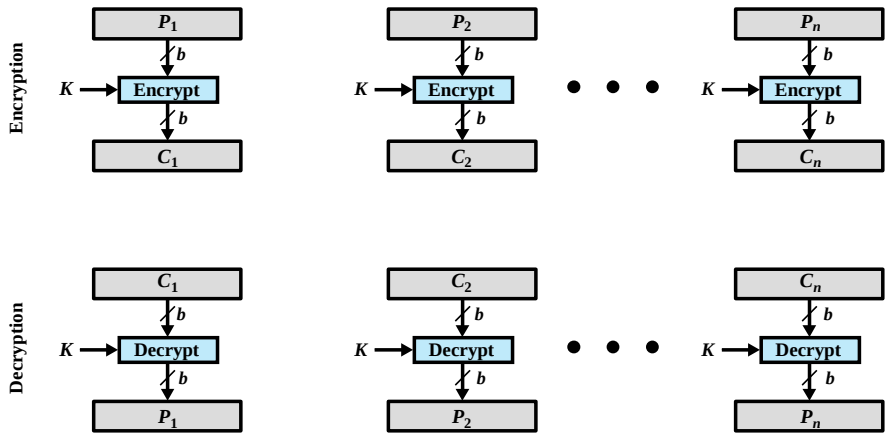
- Should have a security strength equal to or better than 3DES
- Significantly improved efficiency
- Symmetric block cipher
- 128 bit data and 128/192/256 bit keys

Selected Rijndael in November 2001

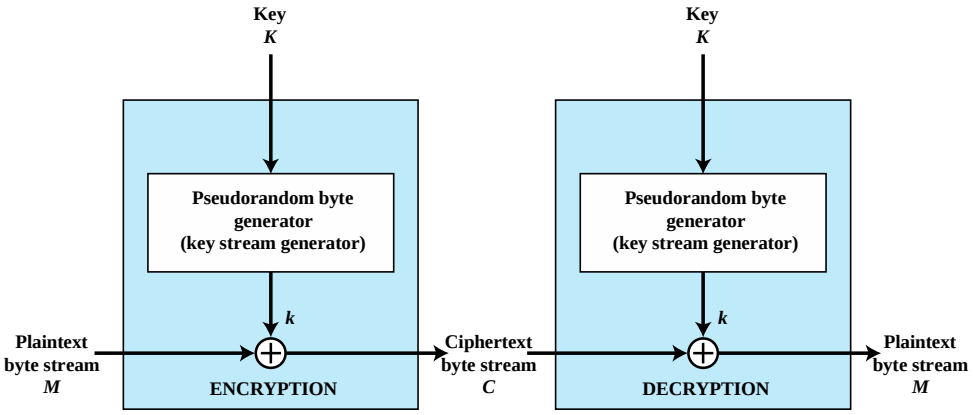
Published as FIPS 197

Practical Security Issues

- Typically symmetric encryption is applied to a unit of data larger than a single 64-bit or 128-bit block
- Electronic codebook (ECB) mode is the simplest approach to multiple-block encryption
  - Each block of plaintext is encrypted using the same key
  - Cryptanalysts may be able to exploit regularities in the plaintext
- Modes of operation
  - Alternative techniques developed to increase the security of symmetric block encryption for large sequences
  - Overcomes the weaknesses of ECB



(a) Block cipher encryption (electronic codebook mode)



(b) Stream encryption

Figure 2.2 Types of Symmetric Encryption

Block & Stream Ciphers

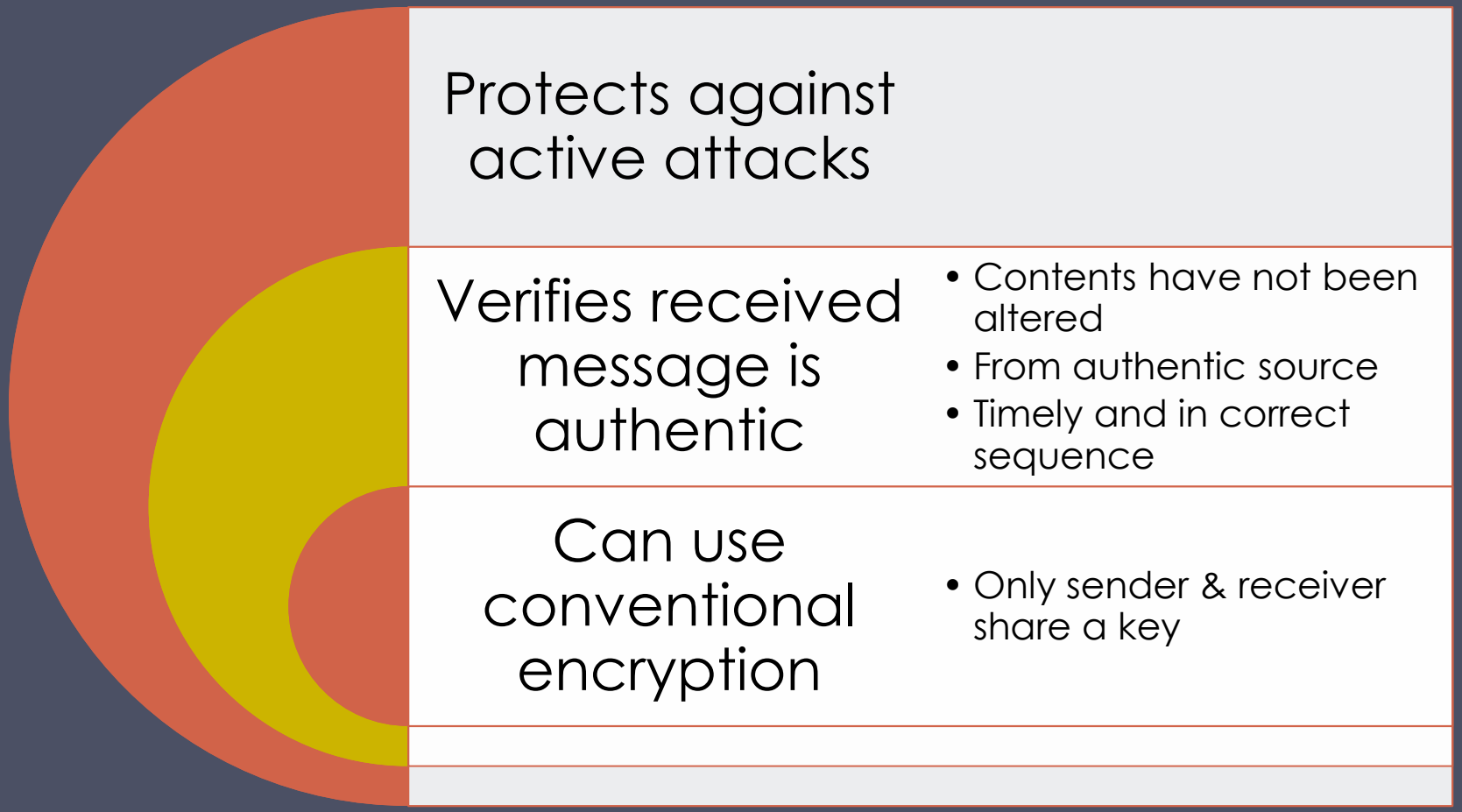
Block Cipher

- Processes the input one block of elements at a time
- Produces an output block for each input block
- Can reuse keys
- More common

Stream Cipher

- Processes the input elements continuously
- Produces output one element at a time
- Primary advantage is that they are almost always faster and use far less code
- Encrypts plaintext one byte at a time
- Pseudorandom stream is one that is unpredictable without knowledge of the input key

Message Authentication



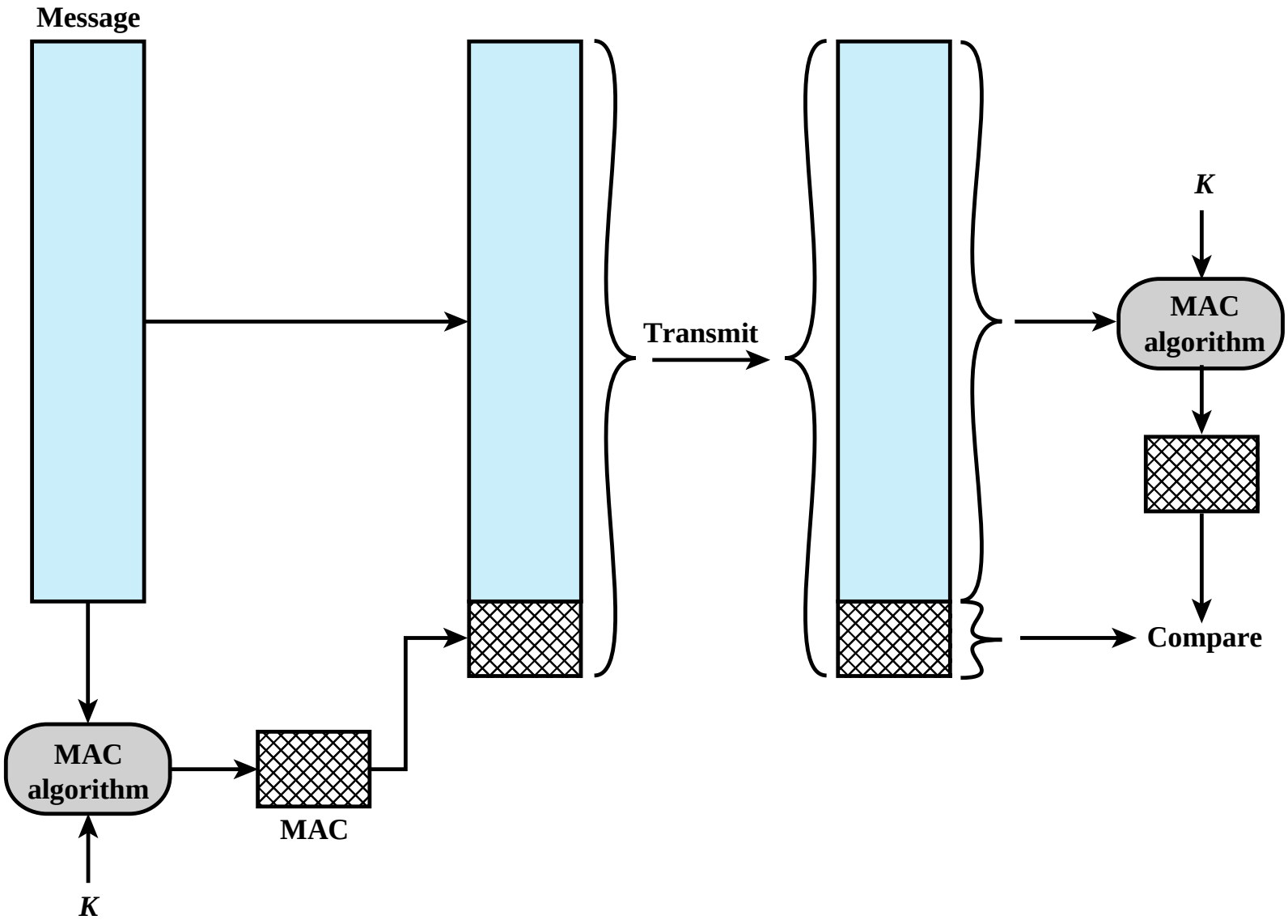


Figure 2.3 Message Authentication Using a Message Authentication Code (MAC).

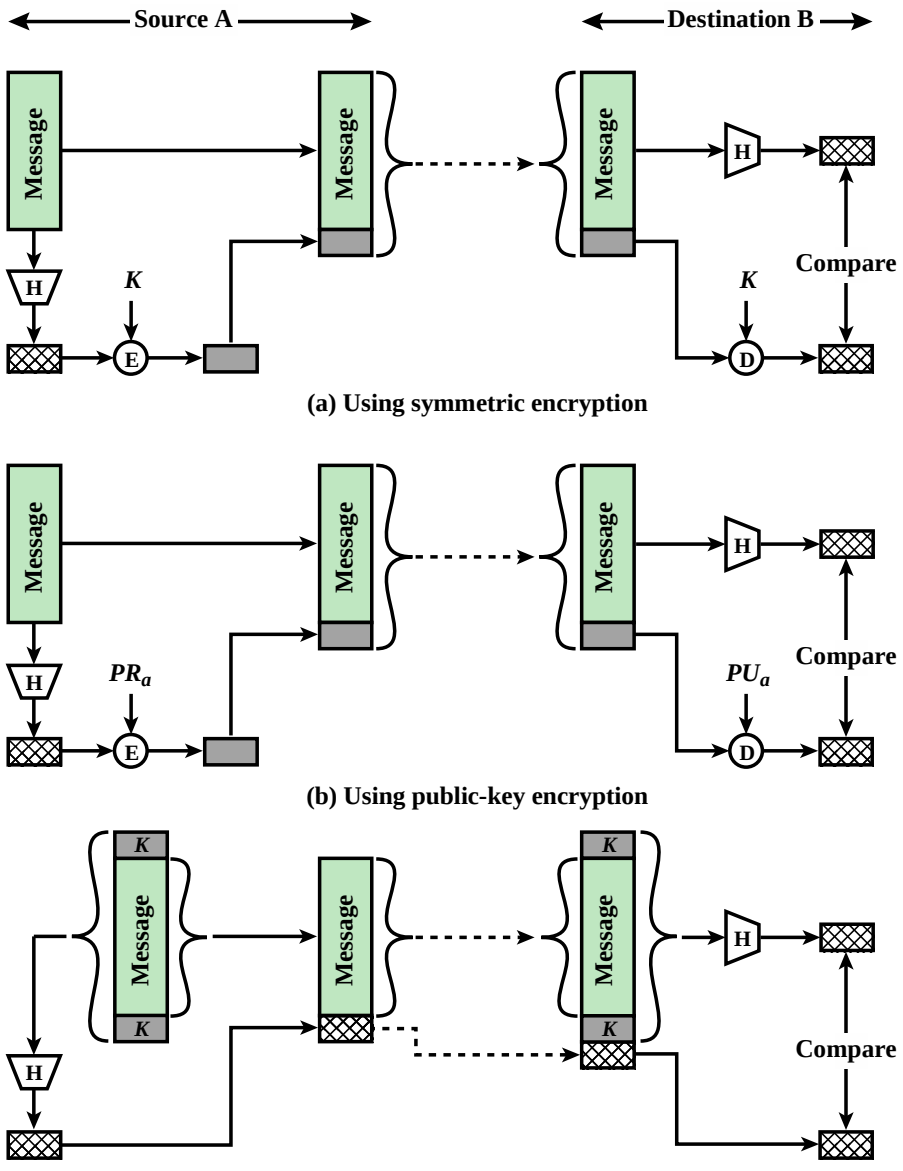


Figure 2.5 Message Authentication Using a One-Way Hash Function.

Hash Function Requirements

- Can be applied to a block of data of any size
- Produces a fixed-length output
- H(x) is relatively easy to compute for any given x
- One-way or pre-image resistant
  - Computationally infeasible to find x such that H(x) = h
- Computationally infeasible to find $y \neq x$ such that H(y) = H(x)
- Collision resistant or strong collision resistance
  - Computationally infeasible to find any pair (x,y) such that H(x) = H(y)

Security of Hash Functions

There are two approaches to attacking a secure hash function:

- **Cryptanalysis**
  - Exploit logical weaknesses in the algorithm
- **Brute-force attack**
  - Strength of hash function depends solely on the length of the hash code produced by the algorithm

SHA most widely used hash algorithm

Additional secure hash function applications:

- **Passwords**
  - Hash of a password is stored by an operating system
- **Intrusion detection**
  - Store H(F) for each file on a system and secure the hash values

Public-Key Encryption Structure

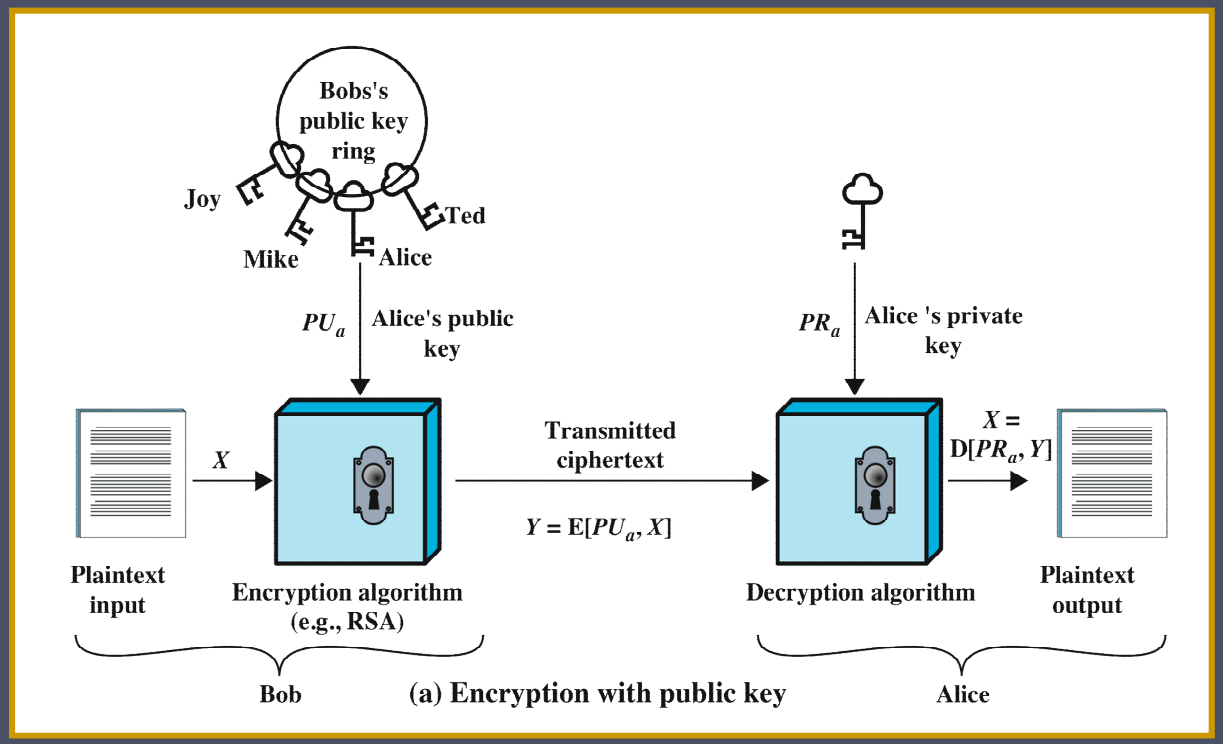
Publicly proposed by Diffie and Hellman in 1976

Based on mathematical functions

Asymmetric

- Uses two separate keys
- Public key and private key
- Public key is made public for others to use

Some form of protocol is needed for distribution



- **Plaintext**
  - Readable message or data that is fed into the algorithm as input
- **Encryption algorithm**
  - Performs transformations on the plaintext
- **Public and private key**
  - Pair of keys, one for encryption, one for decryption
- **Ciphertext**
  - Scrambled message produced as output
- **Decryption key**
  - Produces the original plaintext

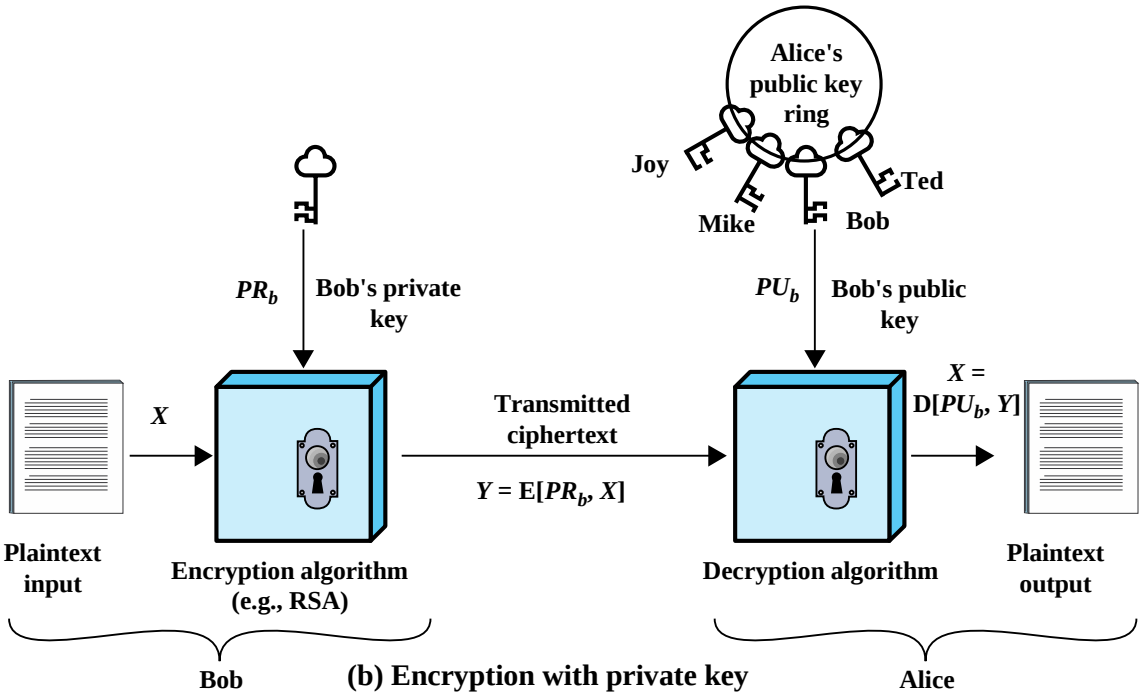


Figure 2.6 Public-Key Cryptography

- User encrypts data using his or her own private key
- Anyone who knows the corresponding public key will be able to decrypt the message

Table 2.3

Applications for Public-Key Cryptosystems

Algorithm	Digital Signature	Symmetric Key Distribution	Encryption of Secret Keys
RSA	Yes	Yes	Yes
Diffie-Hellman	No	Yes	No
DSS	Yes	No	No
Elliptic Curve	Yes	Yes	Yes

Requirements for Public-Key Cryptosystems

Computationally easy to create key pairs

Computationally easy for sender knowing public key to encrypt messages

Computationally easy for receiver knowing private key to decrypt ciphertext

Computationally infeasible for opponent to determine private key from public key

Computationally infeasible for opponent to otherwise recover original message

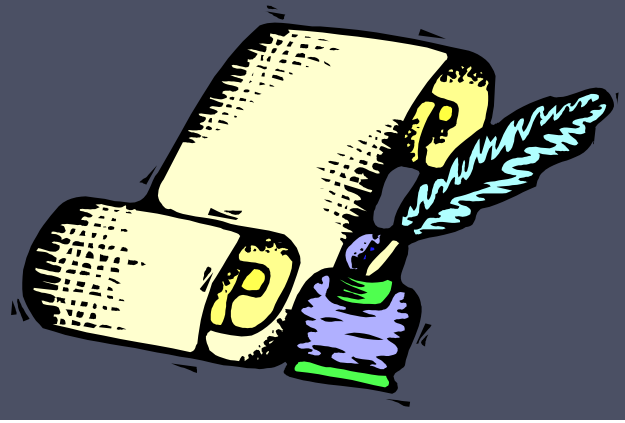
Useful if either key can be used for each role

Asymmetric Encryption Algorithms

RSA (Rivest, Shamir, Adleman)	Developed in 1977	Most widely accepted and implemented approach to public-key encryption	Block cipher in which the plaintext and ciphertext are integers between 0 and *n*-1 for some *n*.
Diffie-Hellman key exchange algorithm	Enables two users to securely reach agreement about a shared secret that can be used as a secret key for subsequent symmetric encryption of messages	Limited to the exchange of the keys	
Digital Signature Standard (DSS)	Provides only a digital signature function with SHA-1	Cannot be used for encryption or key exchange	
Elliptic curve cryptography (ECC)	Security like RSA, but with much smaller keys		

Digital Signatures

- Used for authenticating both source and data integrity
- Created by encrypting hash code with private key
- Does not provide confidentiality
  - Even in the case of complete encryption
  - Message is safe from alteration but not eavesdropping



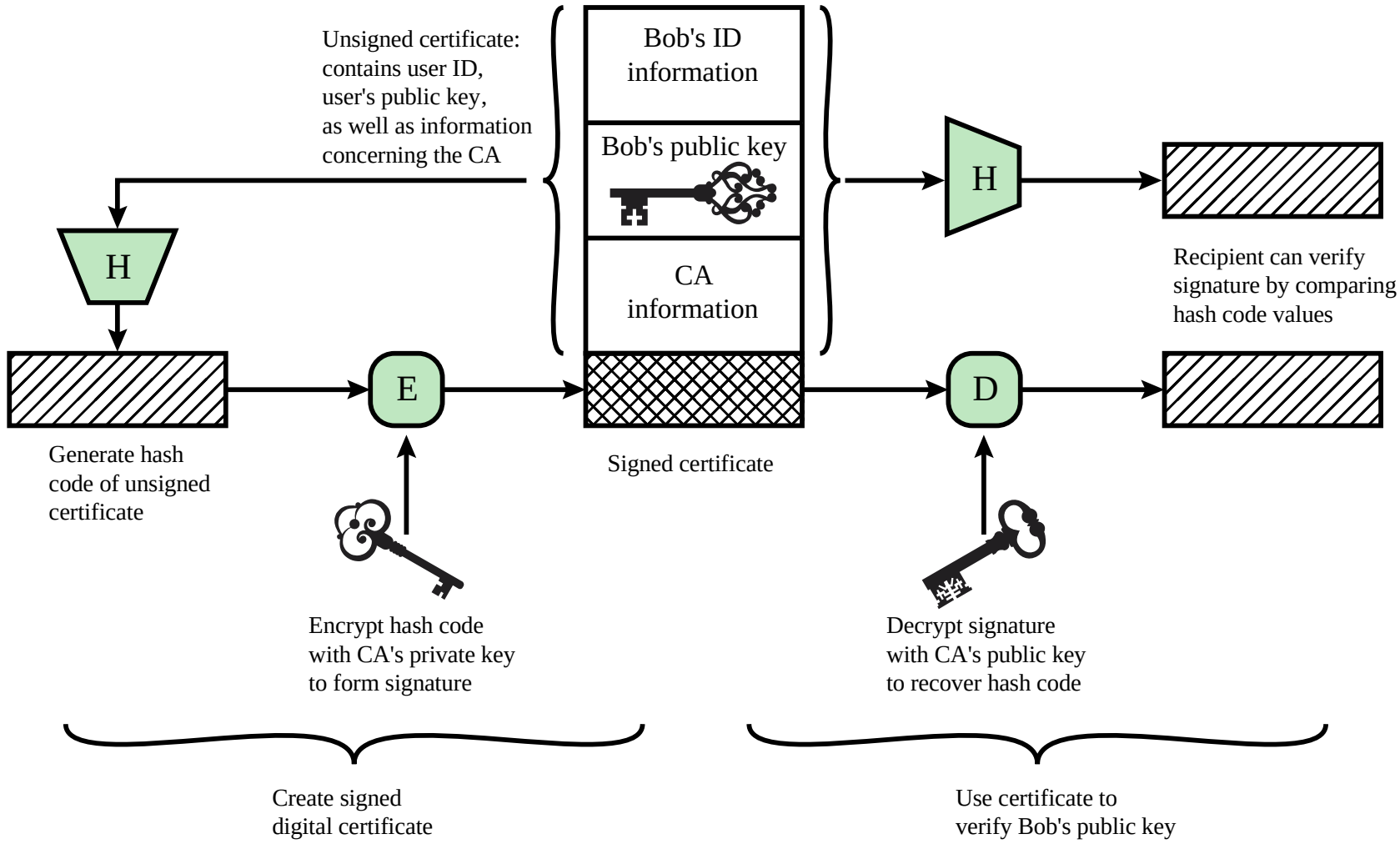
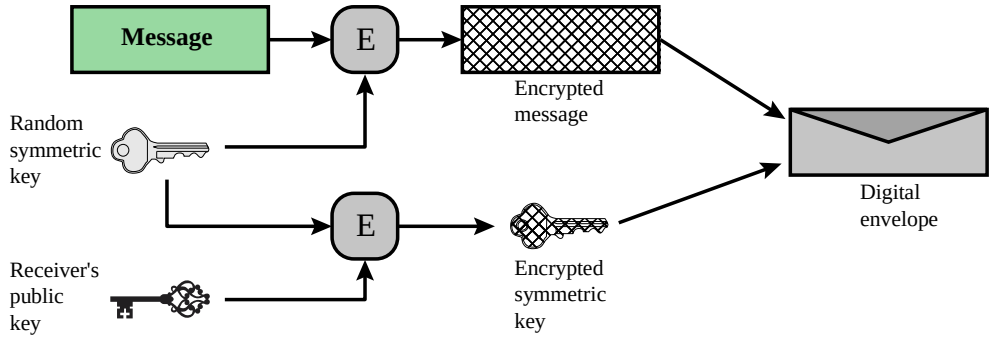


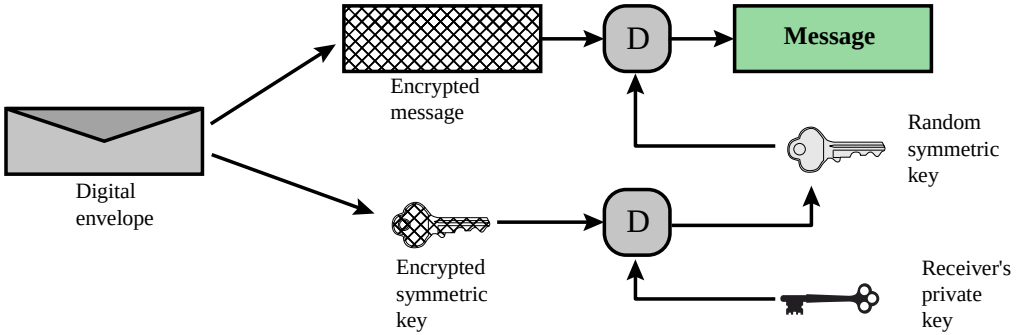
Figure 2.7 Public-Key Certificate Use

Digital Envelopes

- Protects a message without needing to first arrange for sender and receiver to have the same secret key
- Equates to the same thing as a sealed envelope containing an unsigned letter



(a) Creation of a digital envelope



(b) Opening a digital envelope

Figure 2.8 Digital Envelopes

Random Numbers

Uses include generation of:

- Keys for public-key algorithms
- Stream key for symmetric stream cipher
- Symmetric key for use as a temporary session key or in creating a digital envelope
- Handshaking to prevent replay attacks
- Session key

Random Number Requirements

Randomness

- Criteria:
  - Uniform distribution
    - Frequency of occurrence of each of the numbers should be approximately the same
  - Independence
    - No one value in the sequence can be inferred from the others

Unpredictability

- Each number is statistically independent of other numbers in the sequence
- Opponent should not be able to predict future elements of the sequence on the basis of earlier elements

Random versus Pseudorandom

Cryptographic applications typically make use of algorithmic techniques for random number generation

- Algorithms are deterministic and therefore produce sequences of numbers that are not statistically random

Pseudorandom numbers are:

- Sequences produced that satisfy statistical randomness tests
- Likely to be predictable

True random number generator (TRNG):

- Uses a nondeterministic source to produce randomness
- Most operate by measuring unpredictable natural processes
  - e.g. radiation, gas discharge, leaky capacitors
- Increasingly provided on modern processors

Practical Application: Encryption of Stored Data

Common to encrypt transmitted data

Much less common for stored data

- There is often little protection beyond domain authentication and operating system access controls
- Data are archived for indefinite periods
- Even though erased, until disk sectors are reused data are recoverable

Approaches to encrypt stored data:

- Use a commercially available encryption package
- Back-end appliance
- Library based tape encryption
- Background laptop/PC data encryption

Summary

- Confidentiality with symmetric encryption
  - Symmetric encryption
  - Symmetric block encryption algorithms
  - Stream ciphers
- Message authentication and hash functions
  - Authentication using symmetric encryption
  - Message authentication without message encryption
  - Secure hash functions
  - Other applications of hash functions
- Random and pseudorandom numbers
  - The use of random numbers
  - Random versus pseudorandom

- Public-key encryption
  - Structure
  - Applications for public-key cryptosystems
  - Requirements for public-key cryptography
  - Asymmetric encryption algorithms
- Digital signatures and key management
  - Digital signature
  - Public-key certificates
  - Symmetric key exchange using public-key encryption
  - Digital envelopes